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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
6 March 2014 (06.03.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/035410 A1

(51) International Patent Classification:
G06F 15/16 (2006.01) *G06F 9/30* (2006.01)

(21) International Application Number:
PCT/US2012/053212

(22) International Filing Date:
30 August 2012 (30.08.2012)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant *(for all designated States except US)*: **HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P.** [US/US]; 11445 Compaq Center Drive W., Houston, Texas 77070 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **SHANI, Inbar** [IL/IL]; Altalef St., Altalef St. No. 5, 56100 Yehud (IL). **NITSAN, Amichai** [IL/IL]; Altalef St., Altalef St. No. 5, 56100 Yehud (IL). **MANOR, Lior** [IL/IL]; Shabazi 19, 56100 Yehud (IL).

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

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, 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, 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,

[Continued on next page]

(54) Title: GLOBAL FEATURE LIBRARY USEABLE WITH CONTINUOUS DELIVERY

(57) Abstract: A method to manage a global feature library is provided herein. The method provides a global feature library with a plurality of features defined therein. The global feature library includes a feature switch for each of the plurality of features. The feature switch is useable with an application code. The feature switch is linked to an application code. The feature switch includes a feature value that turns a feature associated with the feature switch on and off based on a global value rule. The global rule is transmitted to a client device capable of storing the feature value in-memory.

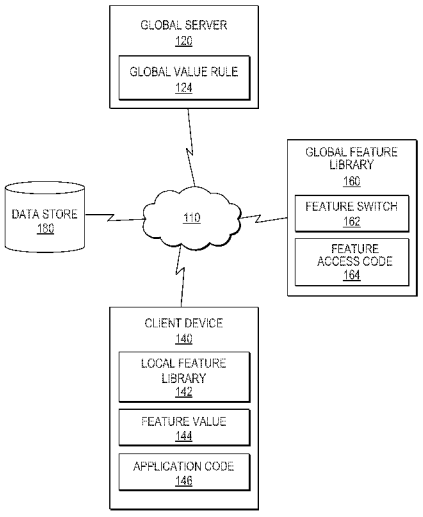

Fig. 1

WO 2014/035410 A1

TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to the identity of the inventor (Rule 4.17(i))*

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

Published:

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

GLOBAL FEATURE LIBRARY USEABLE WITH CONTINUOUS DELIVERY

BACKGROUND

[0001] Software development life cycles use continuous delivery to reduce the time code changes spend in a production line. Continuous delivery includes continuous integration (CI) and continuous deployment (CD) Continuous integration automates the process of receiving code changes from a specific source configuration management (SCM) tool, constructing deliverable assemblies with the code changes, and testing the assemblies.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] Non-limiting examples of the present disclosure are described in the following description, read with reference to the figures attached hereto and do not limit the scope of the claims. In the figures, identical and similar structures, elements or parts thereof that appear in more than one figure are generally labeled with the same or similar references in the figures in which they appear. Dimensions of components and features illustrated in the figures are chosen primarily for convenience and clarity of presentation and are not necessarily to scale. Referring to the attached figures:

[0003] FIG. 1 illustrates a network environment to manage a global feature library in a continuous delivery software lifecycle according to an example;

[0004] FIG. 2 illustrates a block diagram of an apparatus and system to manage the global feature library according to an example;

[0005] FIG. 3 illustrates block diagram of a system useable with the network environment of FIG. 1 to manage the global feature library according to an example;

[0006] FIG. 4 illustrates a block diagram of a computer readable medium useable with the apparatus of FIG. 2 according to an example; and

[0007] FIG. 5 illustrates a flow chart of a method to manage the global feature library according to an example.

DETAILED DESCRIPTION

[0008] In the following detailed description, reference is made to the accompanying drawings which form a part hereof, and in which is illustrated by way of specific examples in which the present disclosure may be practiced. It is to be understood that other examples may be utilized and structural or logical changes may be made without departing from the scope of the present disclosure.

[0009] Continuous integration (CI) and continuous deployment (CD) automate the construction, testing, and deployment of code assemblies with a code change. Continuous integration automates the process of retrieving code changes from a source configuration management (SCM) tool, constructing deliverable assemblies, such as executing a build and unit testing the assemblies. The automation begins after a code change is committed to the SCM tool. When the code change is committed to the SCM tool, the code change is assigned to a particular continuous deployment pipeline (CD pipeline or deployment pipeline). The code change moves through the continuous deployment pipeline as the code change is tested as part of a code base or an assembly of code.

[0010] Continuous deployment extends continuous integration by automatically deploying the assemblies into a test environment and executing testing on the assemblies. A core practice of continuous deployment is to develop all the code on one repository, such as a SCM tool, without branching the code repository. While this practice reduces code mergers and ensures the code is “in production” from the time of deployment, at times the code is “in production” prematurely. For example, the code may pass the tests in the test

environment but does not have full functionality or stability in production. Feature switches have been developed and used to turn a feature on and/or off in production. Feature switches place source code that performs a feature within an “if then” clause and uses the "if statement" to determine when the feature should be active. A feature value is used to determine when the feature is active is typically a local value that must be defined within the application code on each local device. For example, if a company runs a website on one thousand local or distributed client devices across the country and wants to remove a feature, the feature value must be updated on each of the devices since there is not a global on or off switch for the feature that can be distributed to all of the client devices.

[0011] In examples, a method to manage a global feature library is provided. The method provides a global feature library with a plurality of features defined in the library. The global feature library includes a feature switch for each of the plurality of features. The feature switch is useable with an application code. The feature switch is linked to an application code. The feature switch includes a feature value that turns a feature associated with the feature switch on and off based on a global value rule. The global rule is transmitted to a client device capable of storing the feature value in-memory. By storing the feature value in-memory on the client device, the feature provides a global and scalable solution yet be efficiently accessed at runtime.

[0012] The phrase “application code” refers to source code for a software application.

[0013] The phrase “continuous deployment pipeline” or “deployment pipeline” refers to a set of actions executed serially and/or in parallel on a queue to approve source code (or code) for the software application. For example, the continuous deployment pipeline may include building the code, executing unit tests, deploying the code, running automated tests, staging the code, running end-to-end tests, and deploying the code to production.

[0014] The phrase “feature library” refers to a feature switch or feature name that identifies the “function” of the feature or flag. The feature library provides the ability to define feature values to each feature switch. A global feature library provides global access to a feature library for client devices and

applications. A client device may store a copy of the feature library locally on the client device or a data store connected thereto. The local copy of the feature library is synchronized with the global feature library and contains local information for quick access

[0015] The term “feature or flag” refers to code that performs a specific function. The term feature is used hereinafter to refer to a feature or a flag. The feature is stored within the application code (or a local library associated with the application code) and is in the same programming language as the application code.

[0016] The phrase “feature switch” refers to the term used in the global feature library to identify the feature.

[0017] The phrase “feature value” refers to the actual value assigned to the feature switch that turns the feature on or off in each application code. The actual value is stored locally, i.e., on the client device.

[0018] The phrase “feature access code” refers to a generic computer code that is used to access the global feature library and return access to the value of the feature. The feature access code may access the global version of the global feature library or a local or in-memory version of the global feature library. The features in the feature library may be accessed in multiple programming languages depending on the programming language of the application code. For example, the feature access code may be a generic code that accesses the feature in multiple programming languages such as, Ruby, PHP, SQL, Javascript, Java, C++, and C#.

[0019] FIG. 1 illustrates a network environment 100 to manage a global feature library in a continuous delivery software lifecycle according to an example. The network environment 100 includes a link 110 that connects a global server 120, a client device 140, a global feature library 160, and a data store 180. The global server 120 represents generally any computing device or combination of computing devices that manages a global feature library 160. The global server 120 provides a centralized location to update client devices 140 and manage features from a global feature library 160.

[0020] The global feature library 160 is a central repository for features for an application code 146. The global feature library 160 represents generally any database that stores a feature switch 162. Each feature switch 162 defines a feature in the application code, such as code, application configuration, or database that is turned on and off based on conditions. The global feature library 160 also includes a feature access code 164 to access that feature in the application code. The global features defined by the features switch 162 may be incorporated into an application code 146 deployed in a continuous delivery environment using, for example, a continuous deployment pipeline.

[0021] For example, a developer may access, add, and/or remove feature switches 162 to the global feature library 160 using, for example, a macro. The macro that can quickly embed a feature switch 162 into an application code 146 from a list of features existing and/or in development, such as an application lifecycle management tool. The feature may be selected after the macro is activated and an "if then" clause is generated in the application code 146. The macro adds the feature switch 162 to the global feature library 160 and available to others via interfaces, such as through the global server 120. Similarly when the feature switch 162 is removed from an application code 146, the developer may use a macro to automatically remove the feature switch 162 from the global feature library and provide a notification that the feature switch 162 is no longer available.

[0022] The developer sets defaults and initializes the features and the feature switches 162 using, for example an interface on the global server 120 that is integrated into a developer's integrated development environment. The developer may use the interface to view definitions of the feature and the feature switches 162. The developer may also use the interface to initialize the conditions and defaults of the feature switch 162. To make the initialization process more efficient, the developer can use a configuration file, a database, an application programming interface call such as, REpresentational State Transfer (REST), or custom applications to make modifications in real-time.

[0023] The developer can also restrict access to the features. For example, specific access right or user credentials may be needed to add,

remove, update or change a feature switch 162. The access rights may be determined by user and/or machines based on pre-defined criteria.

[0024] The client device 140 represents a computing device and/or a combination of computing devices configured to interact with the global server 120 and via the link 110. The interaction may include transmitting and/or receiving data related to an application code 146 and/or a feature switch 162. The client device 140 may be, for example, a local server or a personal computer which includes a software application managed by the global server 120. The client device 140 stores the application code 146 for the software application and run the software application for the user. The client device 140 also includes local data relating to the global feature library 160.

[0025] For example, the client device 140 receives local feature library 142 from the global server 120. The local feature library 142 includes data from the global feature library 160, such as a copy of the feature switches 162, the feature access code 164, and/or the global value rule 124 distributed through the global server 120. The client device 140 stores a portion of the local feature library 142 in-memory including the feature value 144 that is used to turn the feature switch 162 on or off based on a global value rule 124. The global rule value 124 is received with the local feature library 142 from the global server 120.

[0026] Use of the local feature library 142 to locally or in-memory store features values 144, for example, optimizes the performance of the client device 140 at runtime because the client device 140 does not need to query the global server 120 during execution of the application code. Instead a distributed and scalable system of client devices 140 with local or in-memory storage are provided to efficiently obtain feature data and avoid loss of efficiency due to a centralized server that is slow. On start-up or initialization, the client device 140 take advantage of the local feature library 142 and efficiently query the global server 120 and update the local feature library 142 to be synchronized with the global feature library 160, while the application programs are loading to ensure quick access at runtime.

[0027] Moreover, the use of a global feature library 160 that centrally stores the data and distributes it to the local feature library 142 provides a scalable solution that enables usage of the feature switches 162 across servers, domains, networks, and/or data centers, including usage in cloud computing. For example, the local feature library 142 may be used to efficiently distribute updates or access to the features to all the client devices 140 and/or a subset of the client devices 140. When only a subset of the client devices 140 receive an update or access to a feature, the global server 120 may strategically limit or control the number of users of a feature based on, for example, domains, machine addresses, and/or characteristics of the machine to test the feature under specific conditions in production.

[0028] The data store 180 represents generally any memory, such as data repository, configured to store data that can be accessed by the global server 120, the client device 140, and/or the global feature library 160 in the performance of its function. The global server 120 functionalities may be accomplished via the link 110 that connects the global server 120 to the client device 140, the global feature library 160, and the data store 180.

[0029] The link 110 represents generally one or more of a cable, wireless, fiber optic, or remote connections via a telecommunication link, an infrared link, a radio frequency link, or any other connectors or systems that provide electronic communication. The link 110 may include, at least in part, an intranet, the Internet, or a combination of both. The link 110 may also include intermediate proxies, routers, switches, load balancers, and the like.

[0030] FIG. 2 illustrates a block diagram of an apparatus 200 to manage a global feature library 160 according to examples. The apparatus 200 is useable with a continuous deployment pipeline and includes the global server 120, a memory 222, and a processor 224. The global server 120 manages a global feature library 160. The memory 222 stores a set of instructions. The processor 224 is coupled to the memory 222 to execute the set of instructions.

[0031] The set of instructions provide the global feature library 160 with a plurality of features defined in the library. The global feature library 160

includes a feature switch 162 for each of the plurality of features. The feature switch 162 is useable with an application code 146 on a client device 140.

[0032] The set of instructions link the feature switch 162 to an application code 146. The feature switch 162 includes a feature value 144 associated therewith. The feature value 144 turns a feature associated with the feature switch 162 on and off based on a global value rule 124.

[0033] The set of instructions transmit the global value rule 124 to a client device 140 capable of storing the feature value 144 in-memory thereon. The client device 140 creates a distributed system of local devices that enable quick access to the feature values 144 without having to access the global server 120 each time a value is needed.

[0034] The apparatus 200, when combined with the global feature library 160, provides a system 250 for management of the global feature library 160. The system 250 is usable with a client device 140, that for example, has application code 146 deployed through a deployment pipeline. The system 250 distributes data from the global feature library 160 to the client devices 140 via a local feature library 142 that contains copies of a portion of the global feature library 160. The local feature library 142 corresponds to the client device 140 and is stored locally or in-memory on the client device 140.

[0035] As illustrated in FIG. 2, the global server 120 receives the global feature library 160 data from the global feature library 160. The global server 120 then transmits a local feature library 142 to a plurality of client devices 140. FIG. 2 illustrate the global server 120 connected to three client devices 140, e.g., client devices 1-3, and three local feature libraries 142, e.g., local feature library 1-3. The local feature library 142 includes a portion of the global feature library 160 stored locally or in-memory, such that the local feature library 142 is actually on the client device 140 and/or accessible via a local database connected thereto. Data stored in the local feature library 142 may include the, the feature value 144, feature switch 162, and/or the feature access code 164.

[0036] FIG. 3 illustrates a block diagram of a system 250 useable with the environment 100 of FIG. 1 according to an example. The system 250 manages a global feature library. The system 250 includes the global server

120, the global feature library 160, and the data store 180. As illustrated in FIG. 1, the global server 120, the global feature library 160, and the data store 180 are connected via the link 110.

[0037] The global feature library 160 has a plurality of features defined therein. The global feature library 160 includes a feature switch 162 for each of the plurality of features. The feature switch 162 is useable with an application code 146. The feature switch 162 is linkable to an application code 146. The global feature library calls the feature access code 164 associated with the feature switch 162, and the feature access code 164 calls the feature in the programming language of the application code 146.

[0038] The global feature library 160 supports multiple programming languages and provides similar user experiences relating to the feature switches 162 and code macros across the programming languages. For example, developers are able to use the global feature library 160 in a preferred development setting and streamline the use of the feature switches 162 across server several technologies in a tiered and/or composite application. Examples of programming languages include Ruby, PHP, SQL, Javascript, Java, C++, and C#.

[0039] The feature switch 162 includes a feature value 144 associated therewith. The feature value 144 turns a feature associated with the feature switch 162 on and off based on a global value rule 124. The global value rule 124 is transmittable to a client device 140 capable of storing the feature value 144 in-memory thereon.

[0040] The global server 120 is connected to the global feature library 160 and manages the global feature library 160. The global server 120 is connected to the client device 140 and provides the client device 140 with the feature value to store in-memory. The client device 140 stores the feature value in-memory to enable quick access via a distributed system instead of requiring the client device 140 to query the global server 120 each time a feature value is needed. However, when the feature value changes, the global server 120 transmits an update to the local feature library 142, which updates the feature value 144 in-memory on the client device 140. The update transmission is

triggered when, for example, a request is made to change the global value rule 124. The update may occur due to scheduled or periodical polling of the global server 120 for updates. The local code library may also be robust and identify the most recent data from the global server 120 and give that data priority.

[0041] The global server 120 is illustrated as including a library engine 320. The library engine 320 represents generally a combination of hardware and/or programming that provides the global feature library to the client device(s) 140 and manages the global feature library for the client device(s) 140. For example, the library engine will manage the feature definitions, initialization, dynamic settings, efficiency issues, synchronization of the client devices 140, formulating conditions and execution, and collection of usage data 382. The library engine 320 also links feature switches of the global feature library 160 to an application code 146 and transmits the global value rule 124 to the client device(s) 140.

[0042] The client device 140 is illustrated as including a feature engine 342, a value engine 344, and a code engine 346. The feature engine 342 represents generally a combination of hardware and/or programming that manages the features from the global server 120 for the application code 146 on the client device 140. The data used by the feature engine 342 is provided by the global server 120 and used to update the local feature library 142. The feature engine 342 may also collect usage data 382 corresponding to the features and feature switches 162 utilized. The usage data 382 may be provided to or obtained by the global server 120, evaluated, and/or saved in a data store 180, such as a global repository or the global feature library 160.

[0043] The value engine 344 represents generally a combination of hardware and/or programming that stores and updates the feature value 144 in-memory for the client device 140. The feature value 144 may be determined based on the global value rule 124. The code engine 346 represents generally a combination of hardware and/or programming that manages the application code 146. The code engine 346 may use the feature access code 164 to access the code from the application code 146 library associated with the feature.

[0044] The data store 180 may store data accessible by the global server 120 and/or the client device 140. The data store 180 is, for example, a database or a combination of databases that stores at least one of the following a global feature library 160 including the global value rule 124, the local feature library, the feature value 144, the application code 146, the feature switch 162, the feature access code 164, the usage data 382, and an instruction 384, such as an instruction to be performed by a processor 224. For example, the data store 180 may include a global data repository to collect and store the usage data 382 for the feature switch 162, the initialization data, and/or configuration file.

[0045] FIG. 4 illustrates a block diagram of a computer readable medium 400 useable with the apparatus 200 of FIG. 2 according to an example. In FIG. 4, the global server 120 is illustrated to include a memory 222, a processor 224, and an interface 410. The processor 224 represents generally any processor configured to execute program instructions stored in memory 222 to perform various specified functions. The interface 410 represents generally any interface enabling the global server 120 to communicate with the client device 140 and/or the global feature library 160 via the link 110, as illustrated in FIGS. 1 and 3.

[0046] The memory 222 is illustrated to include an operating system 430 and applications 450. The operating system 430 represents a collection of programs that when executed by the processor 224 serve as a platform on which applications 450 may run. Examples of operating systems 430 include various versions of Microsoft's Windows® and Linux®. Applications 450 represent program instructions that when executed by the processor 224 function as an application that manages a global feature library usable with continuous delivery. For example, FIG. 4 illustrates a library module 420 as executable program instructions stored in memory 222 of the global server 120.

[0047] Referring back to FIG. 3, the library engine 320 of global server 120 and the feature engine 342, the value engine 344, and the code engine 346 of the client device 140 are described as combinations of hardware and/or programming. As illustrated in FIG. 4, the hardware portions may include the

processor 224. The programming portions may include the operating system 430, applications 450, and/or combinations thereof. For example, the library module 420 represents program instructions that when executed by a processor 224 cause the implementation of the of the library engine 320 of FIG. 3. The feature module 440 represents program instructions that when executed by a processor 224 cause the implementation of the of the feature engine 342 of FIG. 3. The value module 460 represents program instructions that when executed by a processor 224 cause the implementation of the of the value engine 344 of FIG. 3. The code module 480 represents program instructions that when executed by a processor 224 cause the implementation of the of the code engine 346 of FIG. 3. Additional functionality may be performed by the library module 420, the feature module 440, the value module 460, and/or the code module 480 or by additional module(s).

[0048] The programming of the library module 420, the feature module 440, the value module 460, and/or the code module 480 may be processor executable instructions stored on a memory 222 that includes a tangible memory media and the hardware may include a processor 224 to execute the instructions. The memory 222 may store program instructions that when executed by the processor 224 cause the processor 224 to perform the program instructions. The memory 222 may be integrated in the same device as the processor 224 or it may be separate but accessible to that device and processor 224.

[0049] In some examples, the program instructions may be part of an installation package that can be executed by the processor 224 to perform a method using the system 250. The memory 222 may be a portable medium such as a CD, DVD, or flash drive or a memory maintained by a server from which the installation package can be downloaded and installed. In some examples, the program instructions may be part of an application or applications already installed on the server. In further examples, the memory 222 may include integrated memory, such as a hard drive.

[0050] FIG. 5 illustrates a flow diagram 500 of a method, such as a processor implemented method, to manage a global feature library usable with a

continuous delivery environment. In block 520, a global feature library is provided with a plurality of features defined therein. The global feature library including a feature switch for each of the plurality of features. The feature switch is useable with an application code. The global feature library stores the feature switch in a plurality of programming languages such that the feature switch causes the global feature library to call the feature access code associated with the feature switch, and the feature access code calls the feature in the programming language of the application code. Examples of programming languages include Ruby, PHP, SQL, Javascript, Java, C++, and C#.

[0051] The feature switch is linked to an application code in block 540. The feature switch may be linked and/or unlinked from the application code using at least one of a macro and a user interface. The feature switch includes a feature value associated with it. The feature value may be initialized by the global server and/or the global feature library. In block 560, the feature value turns a feature associated with the feature switch on and off based on a global value rule. The global value rule is transmitted to a client device capable of storing the feature value in-memory or locally.

[0052] By storing the feature value in-memory of the client device, the client device avoids delays due to communications with the global server. The feature value on the client device is updated by the global server, which transmits updates relating to the feature value to the client device. One way an update is triggered is when the global server makes a request to change the global value rule. At that time the global server transmits an update to the feature value on the client device. The feature value may be modified in real-time using for example, a configuration file, an application programming interface call such as REpresentational State Transfer (REST), or custom applications.

[0053] Access to each of the plurality of feature switches may also be based on a user credential that determines the feature value. For example, the user credentials may be used to change the feature value, and/or limit access to the feature switch to, for example, a specific machine or group of machines for testing and/or management or based on condition such as location of the user, as defined in a repository or the global feature library.

[0054] The method may also collect usage data for the feature switch. The usage data includes at least one of where the feature switch is embedded in the application code, when the feature switch is embedded into the application code, who embedding the feature switch, when the feature is on, when the feature switch is off, conditions of when the feature switch is accessed, and the performance effect of the feature switch on the application code's runtime. The usage data may be collected asynchronously and stored in a data store, such as a global data repository accessible by the global server and/or the client device(s).

[0055] FIGS. 1-5 aid in illustrating the architecture, functionality, and operation according to examples. The examples illustrate various physical and logical components. The various components illustrated are defined at least in part as programs, programming, or program instructions. Each such component, portion thereof, or various combinations thereof may represent in whole or in part a module, segment, or portion of code that comprises one or more executable instructions to implement any specified logical function(s). Each component or various combinations thereof may represent a circuit or a number of interconnected circuits to implement the specified logical function(s).

[0056] Examples can be realized in any computer-readable media for use by or in connection with an instruction execution system such as a computer/processor based system or an ASIC (Application Specific Integrated Circuit) or other system that can fetch or obtain the logic from computer-readable media and execute the instructions contained therein. "Computer-readable media" can be any media that can contain, store, or maintain programs and data for use by or in connection with the instruction execution system. Computer readable media can comprise any one of many physical media such as, for example, electronic, magnetic, optical, electromagnetic, or semiconductor media. More specific examples of suitable computer-readable media include, but are not limited to, a portable magnetic computer diskette such as floppy diskettes or hard drives, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory, or a portable compact disc.

[0057] Although the flow diagram of FIG. 5 illustrates specific orders of execution, the order of execution may differ from that which is illustrated. For example, the order of execution of the blocks may be scrambled relative to the order shown. Also, the blocks shown in succession may be executed concurrently or with partial concurrence. All such variations are within the scope of the present invention.

[0058] The present disclosure has been described using non-limiting detailed descriptions of examples thereof and is not intended to limit the scope of the present disclosure. It should be understood that features and/or operations described with respect to one example may be used with other examples and that not all examples of the present disclosure have all of the features and/or operations illustrated in a particular figure or described with respect to one of the examples. Variations of examples described will occur to persons of the art. Furthermore, the terms "comprise," "include," "have" and their conjugates, shall mean, when used in the present disclosure and/or claims, "including but not necessarily limited to."

[0059] It is noted that some of the above described examples may include structure, acts or details of structures and acts that may not be essential to the present disclosure and are intended to be exemplary. Structure and acts described herein are replaceable by equivalents, which perform the same function, even if the structure or acts are different, as known in the art. Therefore, the scope of the present disclosure is limited only by the elements and limitations as used in the claims.

CLAIMS

WHAT IS CLAIMED IS:

1. A processor implemented method to manage a global feature library usable with a continuous delivery environment, the method comprising:

providing a global feature library with a plurality of features defined therein, the global feature library including a feature switch for each of the plurality of features, the feature switch is useable with an application code;

linking the feature switch to an application code, the feature switch includes a feature value associated therewith, the feature value turns a feature associated with the feature switch on and off based on a global value rule; and

transmitting the global value rule to a client device capable of storing the feature value in-memory thereon.

2. The method of claim 1, further comprising updating the feature value on the client device by transmitting updates relating to the feature value from a global server to the client device.

3. The method of claim 1, further comprising configuring the feature switch to be useable with a plurality of programming languages, the global feature library calls the feature access code associated with the feature switch, and the feature access code calls the feature in the programming language of the application code.

4. The method of claim 1, further comprising collecting a usage data for the feature switch,

the usage data includes at least one of where the feature switch is embedded in the application code, when the feature switch is embedded into the application code, who embedding the feature switch, when the feature is turned on, when the feature

switch is turned off, conditions of when the feature switch is accessed, the performance effect of the feature switch on the application code's runtime.

5. The method of claim 4, further comprising:
collecting the usage data asynchronously; and
storing the usage data in a global data store.

6. The method of claim 1, further comprising linking and unlinking the feature switch from the application code using at least one of a macro and a user interface.

7. The method of claim 1, further comprising limiting access to each of the plurality of features based on a user credential.

8. The method of claim 1, further comprising modifying the feature value in real-time.

9. The method of claim 1, further comprising providing a portion of the global feature library to a local feature library in-memory on a client device.

10. A system to manage a global feature library usable with a continuous delivery environment, the system comprising:
a global feature library with a plurality of features defined therein, the global feature library including:
a feature switch for each of the plurality of features, the feature switch is useable with an application code, the feature switch is linkable to an application code,
the feature switch includes a feature value associated therewith, the feature value turns a feature associated with the feature switch on and off based on a global value rule,

the global value rule is transmittable to a client device capable of storing the feature value in-memory thereon; and

a global server connected to the global feature library to:

manage the global feature library; and

provide a client device with the feature value to store in-memory.

11. The system of claim 11, further comprising a global data store to collect and store a usage data for the feature switch.

12. The system of claim 11, wherein the global feature library stores the feature switch such that the feature switch is useable with a plurality of programming languages, the global feature library calls the feature access code associated with the feature switch, and the feature access code calls the feature in the programming language of the application code.

13. The system of claim 11, wherein the global server transmits an update to the feature value in-memory on the client device when a request is made to change the global value rule.

14. The system of claim 11, wherein the global server provides a portion of the global feature library to a local feature library in-memory on a client device.

15. An apparatus usable with a continuous delivery environment, the apparatus comprising:

a global server to manage a global feature library;

a memory to store a set of instructions; and

a processor coupled to the memory to execute the set of instructions to:

provide the global feature library with a plurality of features defined therein, the global feature library including a feature switch for each of the plurality of features, the feature switch is useable with an application code;

link the feature switch to an application code, the feature switch includes a feature value associated therewith, the feature value turns a feature associated with the feature switch on and off based on a global value rule; and

transmit the global value rule to a client device capable of storing the feature value in-memory thereon.

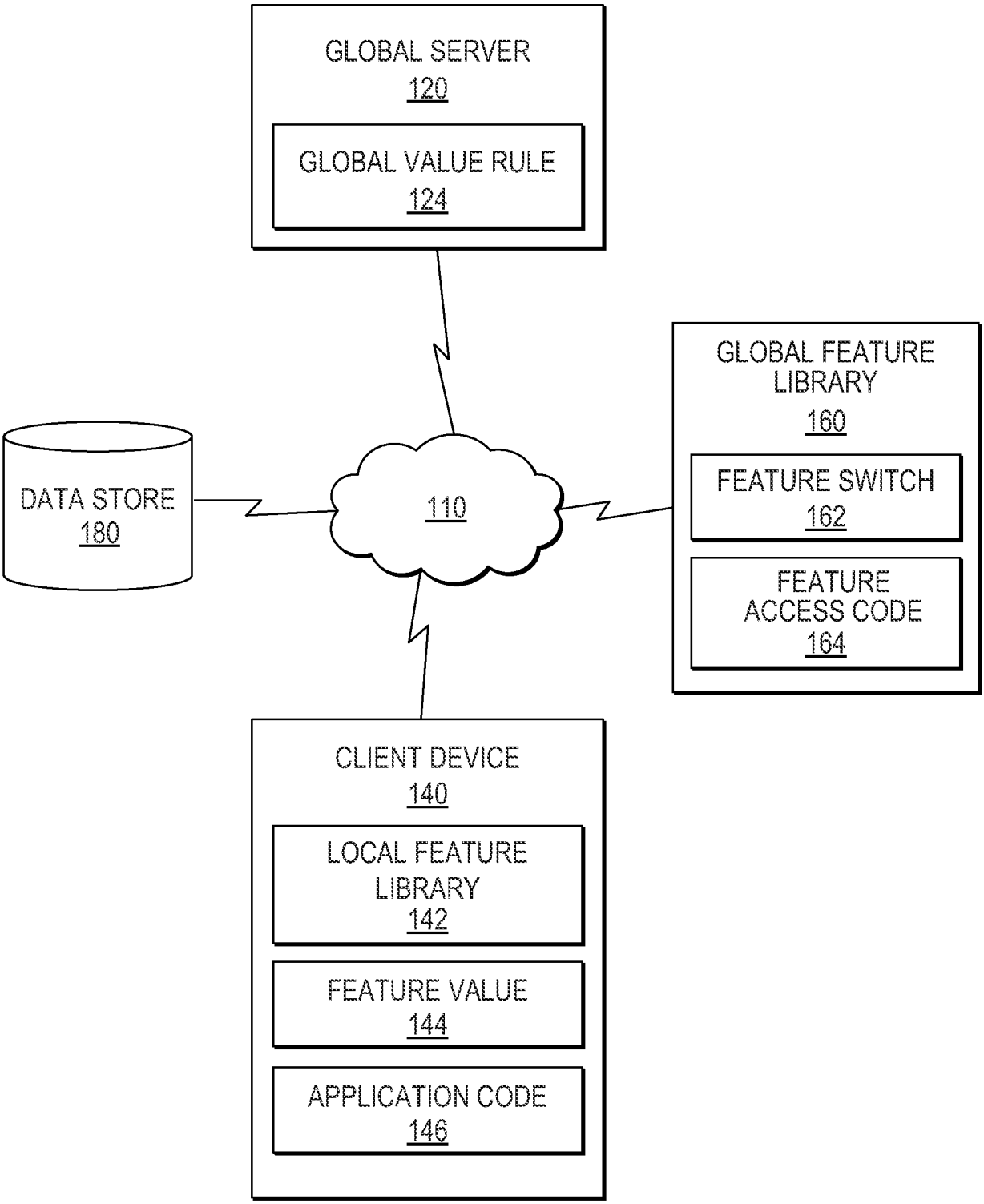
GLOBAL SERVER
120
GLOBAL VALUE RULE
124
GLOBAL FEATURE LIBRARY
160
FEATURE SWITCH
162
FEATURE ACCESS CODE
164
DATA STORE
180
110
CLIENT DEVICE
140
LOCAL FEATURE LIBRARY
142
FEATURE VALUE
144
APPLICATION CODE
146

Fig. 1

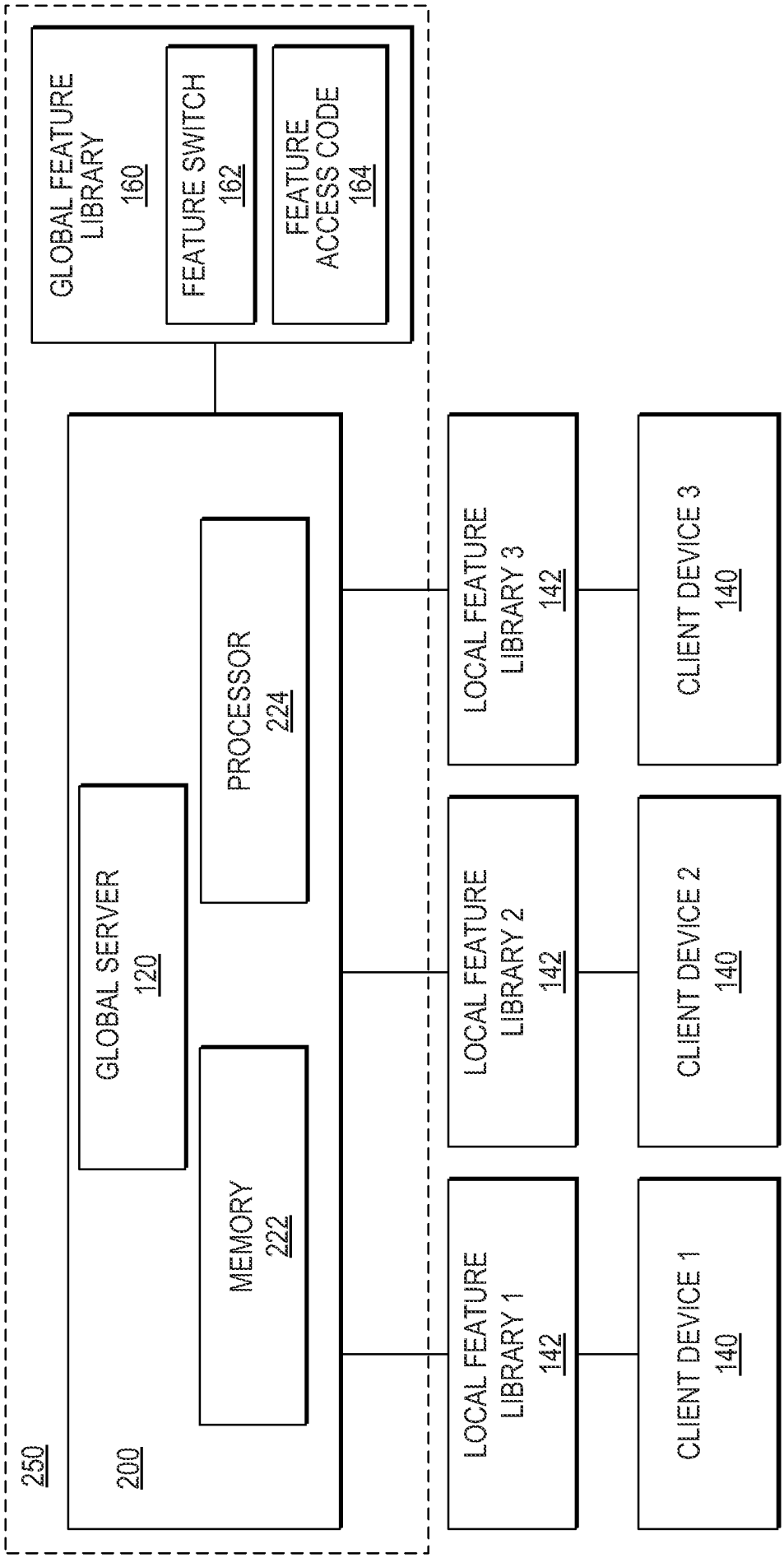
250
200
GLOBAL FEATURE LIBRARY 160
FEATURE SWITCH 162
FEATURE ACCESS CODE 164
GLOBAL SERVER 120
PROCESSOR 224
MEMORY 222
LOCAL FEATURE LIBRARY 1 142
LOCAL FEATURE LIBRARY 2 142
LOCAL FEATURE LIBRARY 3 142
CLIENT DEVICE 1 140
CLIENT DEVICE 2 140
CLIENT DEVICE 3 140

Fig. 2

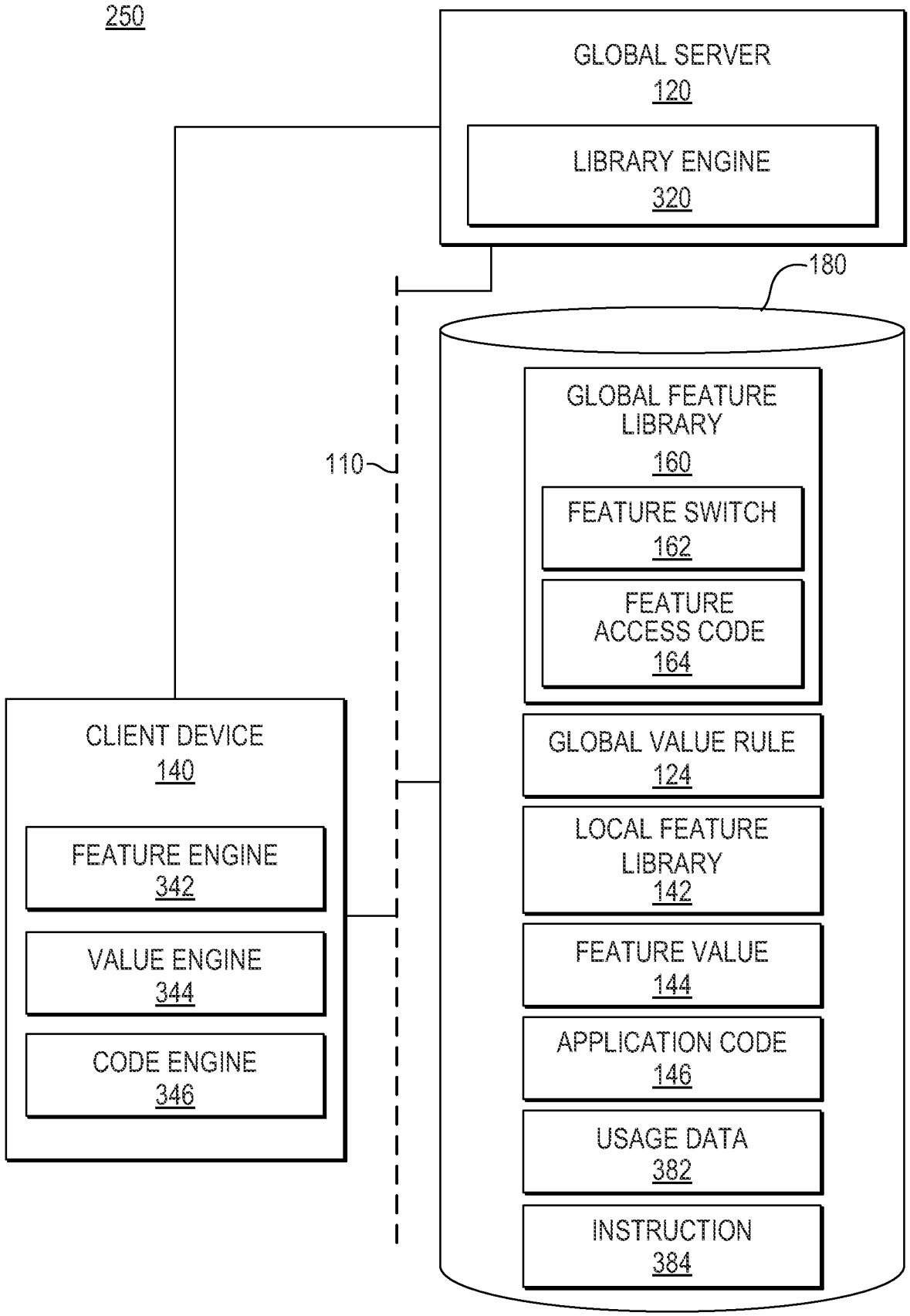
250
GLOBAL SERVER
120
LIBRARY ENGINE
320
180
110
GLOBAL FEATURE LIBRARY
160
FEATURE SWITCH
162
FEATURE ACCESS CODE
164
GLOBAL VALUE RULE
124
LOCAL FEATURE LIBRARY
142
FEATURE VALUE
144
APPLICATION CODE
146
USAGE DATA
382
INSTRUCTION
384
CLIENT DEVICE
140
FEATURE ENGINE
342
VALUE ENGINE
344
CODE ENGINE
346

Fig. 3

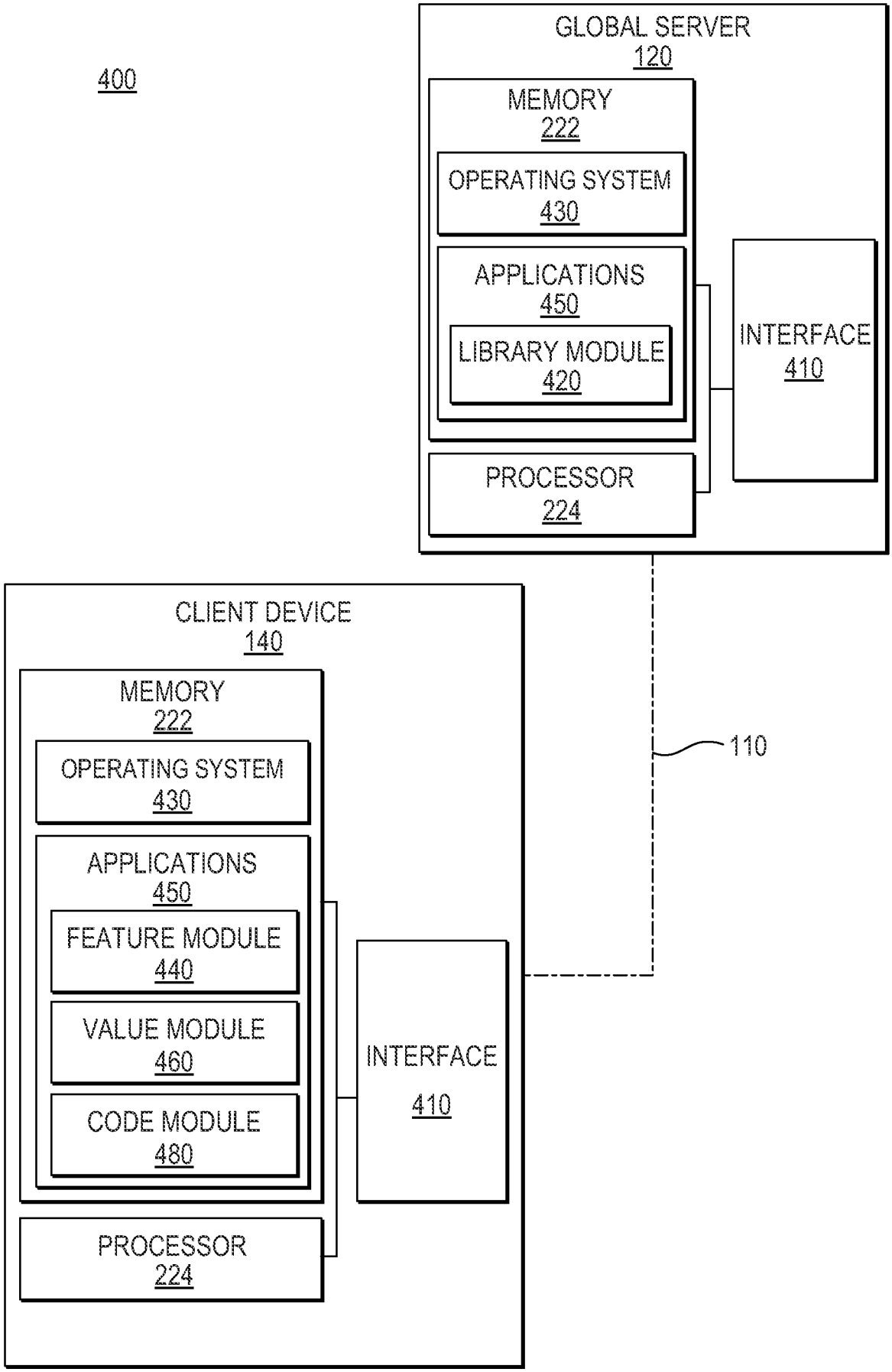
400
GLOBAL SERVER
120
MEMORY
222
OPERATING SYSTEM
430
APPLICATIONS
450
LIBRARY MODULE
420
INTERFACE
410
PROCESSOR
224
110
CLIENT DEVICE
140
MEMORY
222
OPERATING SYSTEM
430
APPLICATIONS
450
FEATURE MODULE
440
VALUE MODULE
460
CODE MODULE
480
INTERFACE
410
PROCESSOR
224

Fig. 4

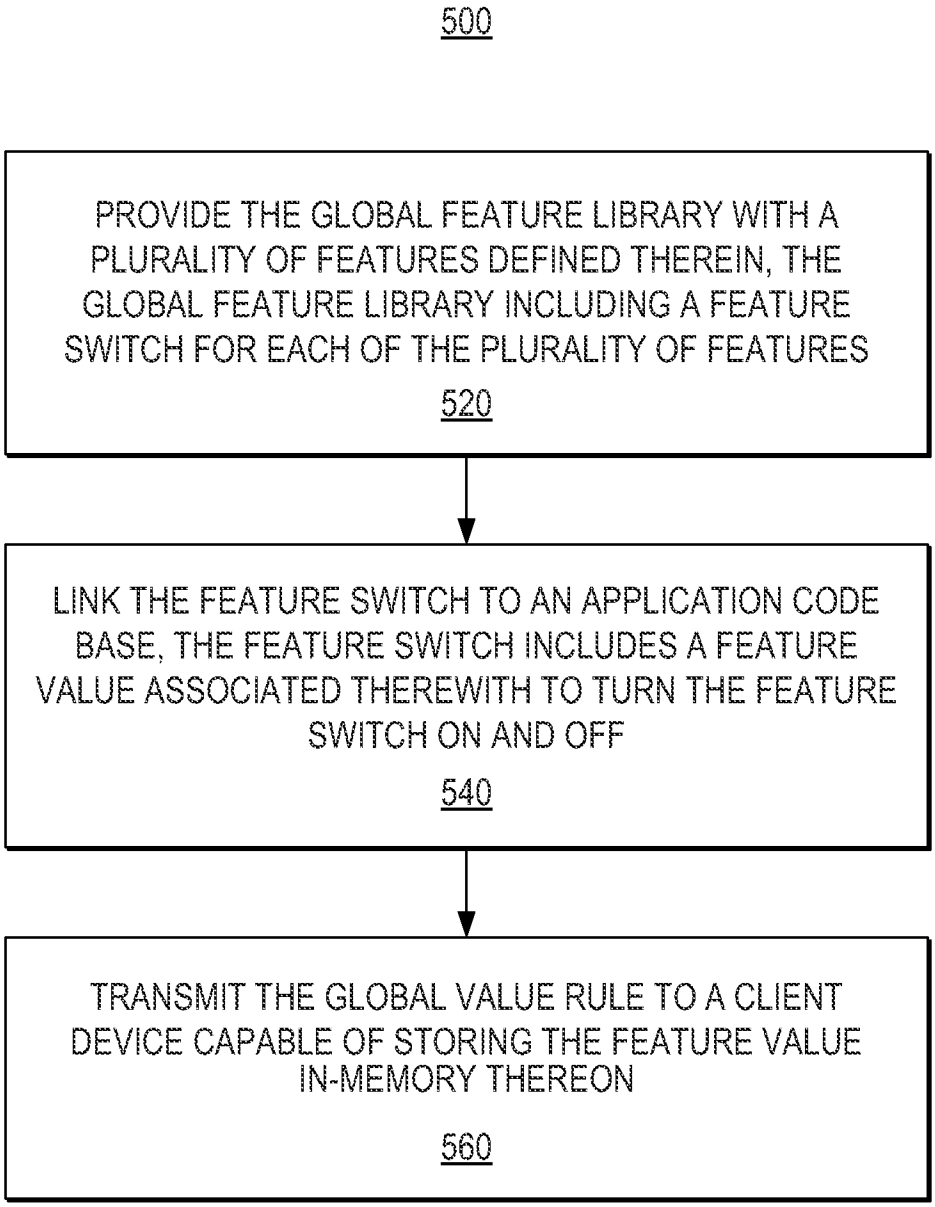
500
PROVIDE THE GLOBAL FEATURE LIBRARY WITH A PLURALITY OF FEATURES DEFINED THEREIN, THE GLOBAL FEATURE LIBRARY INCLUDING A FEATURE SWITCH FOR EACH OF THE PLURALITY OF FEATURES
520
LINK THE FEATURE SWITCH TO AN APPLICATION CODE BASE, THE FEATURE SWITCH INCLUDES A FEATURE VALUE ASSOCIATED THEREWITH TO TURN THE FEATURE SWITCH ON AND OFF
540
TRANSMIT THE GLOBAL VALUE RULE TO A CLIENT DEVICE CAPABLE OF STORING THE FEATURE VALUE IN-MEMORY THEREON
560

Fig. 5

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2012/053212

A. CLASSIFICATION OF SUBJECT MATTER

G06F 15/16(2006.01)i, G06F 9/30(2006.01)i

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F 15/16; G06F 17/50; G06F 17/30; G06F 9/46; G06F 15/163

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

Korean utility models and applications for utility models

Japanese utility models and applications for utility models

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

eKOMPASS(KIPO internal) & Keywords: "LIBRARY", "APPLICATION", "SWITCH", "CLIENT"

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2012-0191760 A1 (KAUFMAN MICHAEL PHILIP et al.) 26 July 2012 See abstract, paragraphs [0026] - [0239] and figure 5.	1-15
A	US 2008-0295045 A1 (AKTOUF CHOUKI) 27 November 2008 See abstract, paragraphs [0075] - [0116] and figure 10.	1-15
A	US 2009-0319472 A1 (JAIN RAMESH et al.) 24 December 2009 See abstract, paragraphs [0028] - [0094] and figure 9.	1-15
A	WO 99-60487 A1 (TRIDIUM, INC.) 25 November 1999 See abstract, page 6 line 13 – page 15 line 4 and figure 1.	1-15

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

☒ See patent family annex.

* Special categories of cited documents:

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

"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 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

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
29 March 2013 (29.03.2013)	**29 March 2013 (29.03.2013)**
Name and mailing address of the ISA/KR Korean Intellectual Property Office 189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City, 302-701, Republic of Korea Facsimile No. 82-42-472-7140	Authorized officer YUN, Byeong Soo Telephone No. 82-42-481-8530

Form PCT/ISA/210 (second sheet) (July 2009)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2012/053212

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2012-0191760 A1	26.07.2012	None	
US 2008-0295045 A1	27.11.2008	AT 440291 T	15.09.2009
		DE 602005016079 D1	01.10.2009
		EP 1716425 A1	02.11.2006
		EP 1716425 B1	19.08.2009
		FR 2866435 A1	19.08.2005
		FR 2866435 B1	04.04.2008
		JP 04-654203 B2	24.12.2010
		JP 2007-522574 A	09.08.2007
		JP 2007-522574 T	09.08.2007
		JP 4654203 B2	16.03.2011
		US 8010918 B2	30.08.2011
		WO 2005-083454 A1	09.09.2005
US 2009-0319472 A1	24.12.2009	None	
WO 99-60487 A1	25.11.1999	AU 1999-39931 A1	06.12.1999
		AU 1999-39931 B2	20.03.2003
		BR 9910512 A	02.01.2001
		CA 2332009 A1	25.11.1999
		CA 2332009 C	07.08.2007
		CN 1305611 A0	25.07.2001
		CN 1305611 B	02.06.2010
		EP 1082669 A1	14.03.2001
		EP 1082669 A4	03.05.2006
		HK 1038970 A1	04.03.2011
		JP 04-330799 B2	26.06.2009
		JP 2002-516432 A	04.06.2002
		JP 2002-516432 T	04.06.2002
		JP 2002-516432 T	04.06.2002
		JP 4330799 B2	16.09.2009
		KR 10-0563291 B1	27.03.2006
		WO 99-60487A1	25.11.1999
		WO 99-60487A9	01.03.2001