

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
5 February 2009 (05.02.2009)

PCT

(10) International Publication Number
WO 2009/018489 A2

(51) International Patent Classification:
G06F 17/00 (2006.01) **G06F 15/16** (2006.01)

(21) International Application Number:
PCT/US2008/071834

(22) International Filing Date: 31 July 2008 (31.07.2008)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/962,877 31 July 2007 (31.07.2007) US
12/102,848 14 April 2008 (14.04.2008) US
12/102,854 14 April 2008 (14.04.2008) US
12/102,842 14 April 2008 (14.04.2008) US
12/182,706 30 July 2008 (30.07.2008) US
12/182,800 30 July 2008 (30.07.2008) US
12/182,557 30 July 2008 (30.07.2008) US
12/182,740 30 July 2008 (30.07.2008) US

(71) Applicant (for all designated States except US): **ETELOS, INC.** [US/US]; 1900 O'Farrell Street, Suite 320, San Mateo, CA 94403 (US).

(72) Inventors; and

(75) Inventors/Applicants (for US only): **BAITALMAL, Ahmad** [US/US]; 825 Harrington PL SE, Unit 2146, Renton, WA 98058 (US). **KOLKE, Daniel, J.** [US/US]; 45908 SE Edgewick Road, North Bend, WA 98045 (US). **COLLETTE, Jon, K.** [US/US]; 255 SW 184th Street, Normandy Park, WA 98166 (US). **STRID, Jerrod** [US/US]; 19659 104th Ave SE, Renton, WA 98055 (US). **TOMPKINS, Casey** [US/US]; 19435 NE 159th St, Woodinville, WA 98077 (US).

(74) Agents: **GRAY, Andrew, J. IV** et al.; Morgan Lewis & Bockius LLP, 2 Palo Alto Square, 3000 El Camino Real, Suite 700, Palo Alto, CA 94306 (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, BR, BW, BY, BZ, CA, CH, 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,

[Continued on next page]

(54) Title: SOFTWARE MARKETPLACE AND DISTRIBUTION SYSTEM

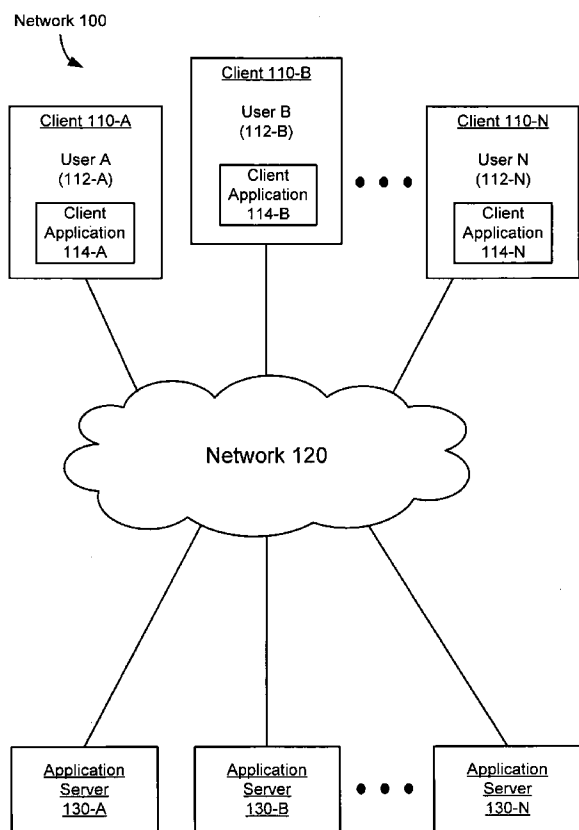


Figure 1

(57) Abstract: In an embodiment, a computer implemented method is described. The method is performed at one or more servers, hosting a marketplace application. A software application is received from a software vendor for distribution. License terms are associated with the software application. The software application is made available for distribution through the marketplace application. The software application is deployed to one or more user accounts on one or more hosting servers, in accordance with the license terms.



MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

(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),

Published:

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

Software Marketplace and Distribution System

TECHNICAL FIELD

[0001] The disclosed embodiments relate generally to synchronizing data for applications.

5 [0002] The disclosed embodiments also relate generally to an application framework for synchronizing data for applications.

[0003] The disclosed embodiments also relate generally to marketing and distribution of software applications.

10 [0004] The disclosed embodiments also relate generally to licensing and enforcement of software applications.

BACKGROUND

[0005] Two different applications may share common data. For example, an email application may include a list of contacts and/or a list of tasks, and a customer relationship management (CRM) application may also include a list of contacts and/or a list of tasks. A
15 user of the email application and the CRM application may desire to synchronize data between these two applications. However, unless the email application and the CRM application share a common data format or unless there is a customized synchronization application that can synchronize data between the two applications, the data cannot be shared easily. Furthermore, for web-based applications, there is presently no way to synchronize
20 data between web-based applications that were not previously designed to synchronize data with each other. Thus, a web-based email application that includes a list of contacts and/or a list of tasks cannot easily synchronize the list of contacts and/or the list of tasks with a web-based CRM application.

[0006] It is often difficult for developers or vendors of software applications,
25 particularly small software applications, to economically market and license their applications. Furthermore, it is often difficult to enforce a license for a small software application, because it may not be cost effective to pursue a violator of the license. It would be desirable to have a system for marketing, licensing and enforcing of software applications

that permits a software developer or vendor to concentrate on creating software applications without the overhead of managing the business aspects of marketing and licensing.

SUMMARY

[0007] An embodiment of the present application relates to a marketplace for software applications where, once licensed, the software applications are hosted at a user account.

[0008] The present application describes some embodiments of a software marketplace whereby software vendors can easily upload and license software applications and receive revenue in return. Among other advantages, this frees software vendors from the need to manage financial and legal issues associated with licensing software applications to large numbers of users. In one embodiment, the software marketplace is associated with a software platform provider (in one example, Etelos) and the software vendors develop software applications for this software platform. This arrangement benefits both the software vendor (who can concentrate on writing applications and receiving revenue for them) and the software platform provider (who has a large number of developers supporting their software platform).

[0009] This arrangement is particularly attractive to vendors of small software applications, where the revenue per licensed application is small, and the number of licensees is high. It may not be cost effective or time effective for the software vendor to engage with large numbers of small payments and licensees, particularly when the licensees may be spread around geographically, in different time zones, use different currencies, etc. By combining ease of use, tight integration, and transparent billing and licensing for the vendors' software applications, the software platform provider can provide an attractive service for its customers

[0010] As the number of software vendors supporting the software platform increases, the software vendors may provide custom application development to customers of the software platform. In some embodiments, the software platform provider can monitor this process and ensure quality. In some embodiments, customers of the software platform may place jobs (i.e., custom software specifications) out for bid, where developers bid on the work. A software customer may specify a bid based on a combination of cost, quality, delivery time, and other factors.

[0011] Software vendors are commonly concerned about the overhead of licensing their software applications, and about enforcing their software licenses. Some embodiments enable software vendors to specify a set of license terms (e.g., commonly used license types such as open source, proprietary, executable only, source code license, etc.) for a software application, and prevent licensees of the software application from misusing the software application outside the terms of the license.

[0012] Some embodiments provide a method for detecting changes made to a first data set in a plurality of data sets, and synchronizing at least a first subset of the changes to a data framework that facilitates data synchronization between the plurality of data sets.

[0013] In some embodiments, at least a second subset of the synchronized changes from the data framework to a second data set in the plurality of data sets is synchronized.

[0014] In some embodiments, at least a third subset of the synchronized changes from the data framework to a second data framework is synchronized.

[0015] Some embodiments provide a computer readable storage medium storing one or more programs configured for execution by a computer, the one or more programs including instructions for detecting changes made to a first data set in a plurality of data sets, and synchronizing at least a first subset of the changes to a data framework that facilitates data synchronization between the plurality of data sets.

[0016] Some embodiments provide a system including one or more processors, memory, and one or more programs stored in the memory, the one or more programs comprising instructions to: detect changes made to a first data set in a plurality of data sets; and synchronize at least a first subset of the changes to a data framework that facilitates data synchronization between the plurality of data sets.

[0017] Some embodiments provide a method for identifying a first data set in a plurality of data sets that is to be synchronized with a data framework, determining a mapping between one or more data fields in a data structure for the first data set and one or more data fields in a data structure for the data framework, and generating synchronization rules for the first data set based on the determined mapping.

[0018] Some embodiments provide a computer readable storage medium storing one or more programs configured for execution by a computer, the one or more programs including instructions for identifying a first data set in a plurality of data sets that is to be

synchronized with a data framework, determining a mapping between one or more data fields in a data structure for the first data set and one or more data fields in a data structure for the data framework, and generating synchronization rules for the first data set based on the determined mapping.

5 **[0019]** Some embodiments provide a system including one or more processors, memory, and one or more programs stored in the memory, the one or more programs comprising instructions to: identify a first data set in a plurality of data sets that is to be synchronized with a data framework, determine a mapping between one or more data fields in a data structure for the first data set and one or more data fields in a data structure for the data framework, and generate synchronization rules for the first data set based on the
10 determined mapping.

[0020] Some embodiments provide a method for detecting changes made to a first data set for a first web-based application in an account, identifying at least a second data set for a second web-based application in the account, wherein the second data set includes at
15 least a subset of the data included in the first data set, and applying synchronization rules to synchronize the second data set with the first data set.

[0021] Some embodiments provide a computer readable storage medium storing one or more programs configured for execution by a computer, the one or more programs including instructions for detecting changes made to a first data set for a first web-based
20 application in an account, identifying at least a second data set for a second web-based application in the account, wherein the second data set includes at least a subset of the data included in the first data set, and applying synchronization rules to synchronize the second data set with the first data set.

[0022] Some embodiments provide a system including one or more processors,
25 memory, and one or more programs stored in the memory, the one or more programs comprising instructions to: detect changes made to a first data set for a first web-based application in an account, identify at least a second data set for a second web-based application in the account, wherein the second data set includes at least a subset of the data included in the first data set, and apply synchronization rules to synchronize the second data
30 set with the first data set.

[0023] Some embodiments provide a computer system including one or more processors, memory, and one or more programs and data structures stored in the memory, the

one or more programs and data structures including: an application data structure configured to store data and program files for a web-based application, an account data structure configured to store one or more instances of the application data structure for an account, wherein an instance of the application data structure corresponds to an instance of a web-based application, and a synchronization module configured to synchronize data between web-based applications within an account based on synchronization rules.

[0024] Some embodiments provide a computer readable storage medium storing one or more programs and data structures configured for execution by a computer, the one or more data structures including: an application data structure configured to store data and program files for a web-based application, and an account data structure configured to store one or more instances of the application data structure for an account, wherein an instance of the application data structure corresponds to an instance of a web-based application. The one or more programs include instructions for synchronizing data between web-based applications within an account based on synchronization rules.

[0025] In accordance with some embodiments, a computer-implemented method is performed at a system. The computer-implemented method includes at one or more servers hosting a marketplace application: receiving from a vendor a software application for distribution; associating license terms with the software application; making the software application available for distribution through the marketplace application; and deploying the software application to one or more user accounts on one or more hosting servers, in accordance with the license terms.

[0026] In accordance with some embodiments, a computer-implemented method is performed at a system. The computer-implemented method includes at one or more servers hosting a marketplace application: receiving from a vendor a software application for distribution; generating license terms in response to a selection by the vendor from options provided by the marketplace application; associating the license terms with the software application; and making the software application available for distribution through the marketplace application, in accordance with the license terms.

[0027] In accordance with some embodiments, a computer-implemented method is performed at a system. The computer-implemented method includes at one or more servers hosting a marketplace application, in response to a request from a syndicated server to distribute a software application from the marketplace: identifying one or more user accounts

associated with the request; verifying that the one or more user accounts has permission to use the software application; and deploying the software application to the one or more user accounts, in accordance with license terms associated with the software application.

[0028] In accordance with some embodiments, a computer-implemented method is performed at a system. The computer-implemented method includes at one or more servers hosting a marketplace application: making a software application available for distribution through the marketplace application; receiving a user request to license the software application; and providing the software application for deployment to one or more user accounts, hosted on one or more servers.

[0029] In accordance with some embodiments, a system for distributing software applications is described. The system comprises one or more processors, memory, and one or more programs stored in the memory. The one or more programs comprise instructions for implementing: a program module configured to provide a software application for distribution in response to an access request from a user; a program module configured to receive and deploy the software application from the marketplace module to an account on one or more servers; and a program module configured to provide at least one or more user accounts, from which the user accesses the software application.

[0030] In accordance with some embodiments, a server system comprises one or more processors, memory, and one or more programs stored in the memory. The one or more programs comprise instructions for at one or more servers hosting a marketplace application: receiving from a vendor a software application for distribution; associating license terms with the software application; making the software application available for distribution through the marketplace application; and deploying the software application to one or more user accounts on one or more hosting servers, in accordance with the license terms.

[0031] In accordance with some embodiments, a computer readable storage medium stores one or more programs configured for execution by a computer. The one or more programs comprise instructions to, at one or more servers hosting a marketplace application: receive from a vendor a software application for distribution; associate license terms with the software application; make the software application available for distribution through the marketplace application; and deploy the software application to one or more user accounts on one or more hosting servers, in accordance with the license terms.

[0032] In accordance with some embodiments, a server system comprises one or more processors, memory, and one or more programs stored in the memory. The one or more programs comprise instructions for, at one or more servers hosting a marketplace application: receiving from a vendor a software application for distribution; generating license terms in response to a selection by the vendor from options provided by the marketplace application; associating the license terms with the software application; and making the software application available for distribution through the marketplace application, in accordance with the license terms.

[0033] In accordance with some embodiments, a computer readable storage medium stores one or more programs configured for execution by a computer. The one or more programs comprise instructions for, at one or more servers hosting a marketplace application: receiving from a vendor a software application for distribution; generating license terms in response to a selection by the vendor from options provided by the marketplace application; associating the license terms with the software application; and making the software application available for distribution through the marketplace application, in accordance with the license terms.

[0034] In accordance with some embodiments, a server system comprises one or more processors, memory, and one or more programs stored in the memory. The one or more programs comprise instructions for, at one or more marketplace servers hosting a marketplace application, in response to a request from a syndicated server to distribute a software application from the marketplace: identifying one or more user accounts associated with the request; verifying that the one or more user accounts has permission to use the software application; and deploying the software application to the one or more user accounts, in accordance with license terms associated with the software application.

[0035] In accordance with some embodiments, a computer readable storage medium stores one or more programs configured for execution by a computer. The one or more programs comprise instructions for, at one or more marketplace servers hosting a marketplace application, in response to a request from a syndicated server to distribute a software application from the marketplace: identifying one or more user accounts associated with the request; verifying that the one or more user accounts has permission to use the software application; and deploying the software application to the one or more user accounts, in accordance with license terms associated with the software application.

[0036] In accordance with some embodiments, a server system comprises one or more processors, memory, and one or more programs stored in the memory. The one or more programs comprise instructions for at one or more servers hosting a marketplace application: making a software application available for distribution through the marketplace application; receiving a user request to license the software application; and providing the software application for deployment to one or more user accounts, hosted on one or more servers.

[0037] In accordance with some embodiments, a computer readable storage medium stores one or more programs configured for execution by a computer. The one or more programs comprise instructions for at one or more servers hosting a marketplace application: making a software application available for distribution through the marketplace application; receiving a user request to license the software application; and providing the software application for deployment to one or more user accounts, hosted on one or more servers.

BRIEF DESCRIPTION OF THE DRAWINGS

[0038] Figure 1 presents a block diagram of a network, according to embodiments of the present invention.

[0039] Figure 2 presents a block diagram of a network, according to embodiments of the present invention.

[0040] Figure 3 presents a block diagram of an application database, according to embodiments of the present invention.

[0041] Figure 4 presents a block diagram of a user database, according to embodiments of the present invention.

[0042] Figure 5 presents a block diagram of a synchronization engine on an application server, according to embodiments of the present invention.

[0043] Figure 6 presents a block diagram of a synchronization engine on a client computer system, according to embodiments of the present invention.

[0044] Figure 7 presents a block diagram of an application server, according to embodiments of the present invention.

[0045] Figure 8 presents a block diagram of a client, according to embodiments of the present invention.

[0046] Figure 9 presents a block diagram illustrating exemplary group memberships for users for a given web-based application, according to embodiments of the present invention.

[0047] Figure 10 presents a block diagram of an exemplary AOP framework that provides offline access to a web-based application, according to embodiments of the present invention.

[0048] Figure 11 illustrates an exemplary process of using an AOP framework to run a web-based application while offline, according to embodiments of the present invention.

[0049] Figure 12 presents a block diagram of an exemplary process of installing an instance of an AOP framework on a client computer system, according to embodiments of the present invention.

[0050] Figure 13 illustrates an exemplary process of installing an instance of an AOP framework on a client computer system, according to embodiments of the present invention.

[0051] Figure 14 presents a block diagram of an exemplary process of initializing an AOP account on a client computer system, according to embodiments of the present invention.

[0052] Figure 15 illustrates an exemplary process of initializing an AOP account on a client computer system, according to embodiments of the present invention.

[0053] Figure 16 presents a block diagram of an exemplary process of installing a web-based application on a client computer system, according to embodiments of the present invention.

[0054] Figure 17 illustrates an exemplary process of installing a web-based application on a client computer system, according to embodiments of the present invention.

[0055] Figure 18 presents a block diagram of an exemplary process of performing an initial data synchronization for a web-based application on a client computer system, according to embodiments of the present invention.

[0056] Figure 19 illustrates an exemplary process of performing an initial data synchronization for a web-based application on a client computer system, according to embodiments of the present invention.

[0057] Figure 20 presents a block diagram of an exemplary process of using the AOP framework on a client computer system, according to embodiments of the present invention.

[0058] Figure 21 illustrates an exemplary process of using the AOP framework on a client computer system, according to embodiments of the present invention.

5 [0059] Figure 22 presents a block diagram of an exemplary process for a client computer system determining how to access a web-based application, according to embodiments of the present invention.

[0060] Figure 23 illustrates an exemplary process for a client computer system determining how to access a web-based application, according to embodiments of the present
10 invention.

[0061] Figure 24 presents a block diagram of an exemplary process for synchronizing a web-based application on a client computer system with a web-based application on an application server, according to embodiments of the present invention.

[0062] Figure 25 illustrates an exemplary process for synchronizing a web-based
15 application on a client computer system with a web-based application on an application server, according to embodiments of the present invention.

[0063] Figure 26 presents a block diagram of an exemplary process for synchronizing a web-based application on a client computer system with a web-based application on an application server, according to embodiments of the present invention.

20 [0064] Figure 27A illustrates an exemplary process for synchronizing a web-based application on a client computer system with a web-based application on an application server, according to embodiments of the present invention.

[0065] Figure 27B continues the process illustrated in Figure 27A, according to embodiments of the present invention.

25 [0066] Figure 27C continues the process illustrated in Figure 27B, according to embodiments of the present invention.

[0067] Figure 27D continues the process illustrated in Figure 27C, according to embodiments of the present invention.

[0068] Figure 27E continues the process illustrated in Figure 27D, according to
30 embodiments of the present invention.

[0069] Figure 27F continues the process illustrated in Figure 27E, according to embodiments of the present invention.

[0070] Figure 28 presents a block diagram of an exemplary process for resolving conflicts for web-based applications that use auto-incrementing identifiers, according to
5 embodiments of the present invention.

[0071] Figure 29 illustrates an exemplary process for resolving conflicts for web-based applications that use auto-incrementing identifiers, according to embodiments of the present invention.

[0072] Figure 30 presents a flowchart of an exemplary process for providing access to
10 a web-based application while offline, according to embodiments of the present invention.

[0073] Figure 31 presents a flowchart of an exemplary process for synchronizing a web-based application on a client computer system with a web-based application on an application server, according to embodiments of the present invention.

[0074] Figure 32 presents a flowchart of an exemplary process for synchronizing a
15 web-based application on a client computer system with a web-based application on an application server, according to embodiments of the present invention.

[0075] Figure 33 presents a flowchart of an exemplary process for resolving conflicts between a web-based application on a client computer system and a web-based application on an application server, according to embodiments of the present invention.

[0076] Figure 34 presents a flowchart of an exemplary process for resolving conflicts
20 between a web-based application on a client computer system and a web-based application on an application server which uses automatically incrementing identifiers for database records, according to embodiments of the present invention.

[0077] Figure 35 presents a flowchart of an exemplary process for providing access to
25 a web-based application while offline, according to embodiments of the present invention.

[0078] Figure 36 presents a block diagram illustrating an exemplary user interface for creating synchronization rules, according to embodiments of the present invention.

[0079] Figure 37 presents a block diagram illustrating an exemplary user interface for generating auto-incrementing identifiers, according to embodiments of the present invention.

[0080] Figure 38 presents a block diagram illustrating an exemplary process for creating synchronization rules, according to embodiments of the present invention.

[0081] Figure 39 presents a flowchart of an exemplary process for creating synchronization rules, according to embodiments of the present invention.

5 [0082] Figure 40 presents a block diagram of an exemplary foreign key mapping, according to embodiments of the present invention.

[0083] Figure 41 presents a flowchart of an exemplary process for creating a foreign key mapping, according to embodiments of the present invention.

10 [0084] Figure 44 is a block diagram illustrating deploying applications to a user account, according to some embodiments.

[0085] Figure 45 is a block diagram illustrating managing licenses in a multi-tenancy environment, according to some embodiments.

[0086] Figure 46 is a block diagram illustrating a web application marketplace and hosting infrastructure, according to some embodiments.

15 [0087] Figure 47 is a flow diagram illustrating a process for selecting and deploying a software application in a user account, according to some embodiments.

[0088] Figure 48 is a block diagram illustrating a marketplace server and hosting infrastructure, according to some embodiments.

20 [0089] Figure 49 is a block diagram illustrating licensing options in a software distribution marketplace, according to some embodiments.

[0090] Figure 50 is a block diagram illustrating dynamic billing in a software distribution marketplace, according to some embodiments.

[0091] Figure 51 is a block diagram illustrating an application manager and repository in a software distribution marketplace, according to some embodiments.

25 [0092] Figure 52 is a block diagram illustrating syndicated deployment across a network in a software distribution marketplace, according to some embodiments.

[0093] Figure 53 is a block diagram illustrating a packager in a software distribution marketplace, according to some embodiments.

[0094] Figure 54 is a block diagram illustrating an alternative embodiment of a packager in a software distribution marketplace, according to some embodiments.

[0095] Figure 55 is a block diagram illustrating an alternative embodiment of a packager in a software distribution marketplace, according to some embodiments.

5 [0096] Figure 56 is a block diagram illustrating licensing a software application in a software distribution marketplace, according to some embodiments.

[0097] Figure 57 is a block diagram illustrating providing security for software applications, deployed to a user account, in a software distribution marketplace, according to some embodiments.

10 [0098] Figure 58 is a block diagram illustrating multiple logons to a software application in a software distribution marketplace, according to some embodiments.

[0099] Figure 59 is a block diagram illustrating a user access control interface, in a software distribution marketplace, according to some embodiments.

[00100] Figure 60 is a block diagram illustrating a user access control interface, in a
15 software distribution marketplace, according to some embodiments.

[00101] Figure 61 is a system block diagram illustrating a server hosting a software marketplace and licensing system, according to some embodiments.

[00102] Figure 62 is a system block diagram illustrating a client interfacing with a software marketplace and licensing system, according to some embodiments.

20 [00103] Figure 63 is a block diagram illustrating an exemplary application framework, according to some embodiments.

[00104] Figure 64A is a block diagram illustrating exemplary components of an account, according to some embodiments.

[00105] Figure 64B is a flow diagram of an exemplary process for creating an account
25 and installing applications into the account, according to some embodiments.

[00106] Figure 65 is a block diagram of a server and a client, according to some embodiments.

[00107] Figure 66 is a flow diagram of an exemplary process for synchronizing applications, according to some embodiments.

[00108] Figure 67 is a block diagram illustrating an exemplary process for synchronizing applications, according to some embodiments.

[00109] Figure 68 is a block diagram illustrating an exemplary process for handling parent-child relationships during a synchronization process, according to some embodiments.

5 [00110] Figure 69 is a flow diagram of an exemplary process for translating owner IDs, according to some embodiments.

[00111] Figure 70 is a block diagram illustrating an exemplary process for merging users between applications, according to some embodiments.

10 [00112] Figure 71 is a flow diagram of an exemplary process for merging users between applications, according to some embodiments.

[00113] Figure 72 presents a block diagram of an exemplary server, according to some embodiments.

[00114] Figure 73 presents a block diagram of an exemplary client computer system, according to some embodiments.

15 [00115] Figure 74 presents a block diagram illustrating an exemplary application synchronization module, according to some embodiments.

[00116] Figure 75 presents exemplary synchronization data structures, according to some embodiments.

20 [00117] Figure 76 is a flow diagram of an exemplary process for synchronizing applications, according to some embodiments.

[00118] Figure 77 is a flow diagram of an exemplary process for generating synchronization rules that are used to synchronize applications, according to some embodiments.

25 [00119] Figure 78 is a flow diagram of an exemplary process for synchronizing applications, according to some embodiments.

[00120] Figure 79 presents exemplary synchronization rules data structures, according to some embodiments.

[00121] Figure 80 is a flow diagram of a process for distributing a software application, according to some embodiments.

[00122] Figure 81 is a flow diagram of a process for distributing a software application, according to some embodiments.

[00123] Figure 82 is a flow diagram of a process for licensing a software application, according to some embodiments.

5 [00124] Figure 83 is a flow diagram of a process for distributing a software application, according to some embodiments.

[00125] Figure 84 is a flow diagram of a process for syndicated deployment of a software application, according to some embodiments.

10 [00126] Figure 85 is a flow diagram of a process for licensing and receiving payment for a software application, according to some embodiments.

[00127] Figure 101 is an exemplary screenshot 10100 of an account user management interface.

[00128] Figure 102 is an exemplary screenshot 10200 of an application packager.

[00129] Figure 103 is an exemplary screenshot 10300 of an application packager.

15 [00130] Figure 104 is an exemplary screenshot 10400 of an application packager.

[00131] Figure 105 is an exemplary screenshot 10500 of an application packager.

[00132] Figure 106 is an exemplary screenshot 10600 of an application packager.

[00133] Figure 107 is exemplary screenshot 10700 of an application packager.

[00134] Figure 108 is an exemplary screenshot 10800 of an application packager.

20 [00135] Figure 109 is an exemplary screenshot 10900 of an application packager.

[00136] Figure 110 is an exemplary screenshot 11000 of an application packager.

[00137] Figure 111 is an exemplary screenshot 11100 of an application packager.

[00138] Figure 112 is an exemplary screenshot 11200 of an application packager.

[00139] Figure 113 is an exemplary screenshot 11300 of an application packager.

25 [00140] Figure 114 is an exemplary screenshot 11400 of an account information details screen.

[00141] Figure 115 is an exemplary screenshot 11500 of a development environment.

- [00142] Figure 116 is an exemplary screenshot 11600 of an application packager.
- [00143] Figure 117 is an exemplary screenshot 11700 of an AOP sync rules manager.
- [00144] Figure 118 is an exemplary screenshot 11800 of an AOP sync rules manager.
- [00145] Figure 119 is an exemplary screenshot 11900 of an AOP sync rules manager.
- 5 [00146] Figure 120 is an exemplary screenshot 12000 of an AOP sync rules manager.
- [00147] Figure 121 is an exemplary screenshot 12100 of an AOP sync rules manager.
- [00148] Figure 122 is an exemplary screenshot 12200 of an integration application group manager.
- [00149] Figure 123 is an exemplary screenshot 12300 of a sync rules manager.
- 10 [00150] Figure 124 is an exemplary screenshot 12400 of a sync rules manager.
- [00151] Figure 125 is an exemplary screenshot 12500 of a sync rules manager.
- [00152] Figure 126 is an exemplary screenshot 12600 of a sync rules manager.
- [00153] Figure 127 is an exemplary screenshot 12700 of a sync rules manager.
- [00154] Figure 128 is an exemplary screenshot 12800 of a sync rules manager.
- 15 [00155] Figure 129 is an exemplary screenshot 12900 of a support page.
- [00156] Figure 130 is an exemplary screenshot 13000 of a product/service basic information page.
- [00157] Figure 131 is an exemplary screenshot 13100 of a staging application page.
- [00158] Figure 132 is an exemplary screenshot 13200 of an edit licensing page.
- 20 [00159] Figure 133 is an exemplary screenshot 13300 of a setup marketing information page.
- [00160] Figure 134 is an exemplary screenshot 13400 of a setup marketing full page.
- [00161] Figure 135 is an exemplary screenshot 13500 of a setup features full page.
- [00162] Figure 136 is an exemplary screenshot 13600 of a product demo page.
- 25 [00163] Figure 137 is an exemplary screenshot 13700 of an “Add a Blog” feed page.
- [00164] Figure 138 is an exemplary screenshot 13800 of a setup frequently asked questions (FAQs) page.

- [00165] Figure 139 is an exemplary screenshot 13900 of a setup getting started application page.
- [00166] Figure 140 is an exemplary screenshot 14000 of a support page.
- [00167] Figure 141 is an exemplary screenshot 14100 of an “About us” page.
- 5 [00168] Figure 142 is an exemplary screenshot 14200 of a product upgrade page.
- [00169] Figure 143 is an exemplary screenshot 14300 of a “my store style” page.
- [00170] Figure 144 is an exemplary screenshot 14400 of a pricing grid and purchase/license link setup page.
- 10 [00171] Figure 145 is an exemplary screenshot 14500 of a web services “Post Data” setup page.
- [00172] Figure 146 is an exemplary screenshot 14600 of a support page.
- [00173] Figure 147 is an exemplary screenshot 14700 of a support page.
- [00174] Figure 148 is an exemplary screenshot 14800 of a support page.
- [00175] Figure 149 is an exemplary screenshot 14900 of a support page.
- 15 [00176] Figure 150 is an exemplary screenshot 15000 of a store product list.
- [00177] Figure 151 is an exemplary screenshot 15100 of a store listing report.
- [00178] Figure 152 is an exemplary screenshot 15200 of a store listing report.
- [00179] Figure 153 is a screenshot of an exemplary screenshot 15300 of a transactions report.
- 20 [00180] Figure 154 is an exemplary screenshot 15400 of a support page.
- [00181] Figure 155 is an exemplary screenshot 15500 of a marketplace page.
- [00182] Figure 156 is an exemplary screenshot 15600 of a hosting and development environment.
- [00183] Figure 157 is an exemplary screenshot 15700 of a marketplace page.
- 25 [00184] Figure 158 is an exemplary screenshot 15800 of a licensing page.
- [00185] Figure 159 is an exemplary screenshot 15900 of a support page.
- [00186] Figure 160 is an exemplary screenshot 16000 of a shopping cart page.

- [00187] Figure 161 is an exemplary screenshot 16100 of a CRM test listing page.
- [00188] Figure 162 is an exemplary screenshot 16200 of a developer toolkit page.
- [00189] Figure 163 is an exemplary screenshot 16300 of a support page.
- [00190] Figure 164 is an exemplary screen shot 16400 from the bottom portion of the
5 screenshot 16300.
- [00191] Figure 165 is an exemplary screenshot 16500 of a support page.
- [00192] Figure 166 is an exemplary screenshot 16600 of an installation page.
- [00193] Figure 167 is an exemplary screenshot 16700 of an installation processing
page.
- 10 [00194] Figure 168 is an exemplary screenshot 16800 of an installation processing
page.
- [00195] Figure 169 is an exemplary screenshot 16900 of a support page.
- [00196] Figure 170 is an exemplary screenshot 17000 of a support page.
- [00197] Figure 171 is an exemplary screenshot 17100 of a marketplace page.
- 15 [00198] Figure 172 is an exemplary screenshot 17200 of a marketplace page.
- [00199] Figure 173 is an exemplary screenshot 17300 of a support page.
- [00200] Figure 174 is an exemplary screenshot 17400 of a support page.
- [00201] Figure 175 is an exemplary screenshot 17500 of a support page.
- [00202] Figure 176 is an exemplary screenshot 17600 of a support page.
- 20 [00203] Figure 177 is an exemplary screenshot 17700 of a support page.
- [00204] Figure 178 is an exemplary screenshot 17800 of a support page.
- [00205] Figure 179 is an exemplary screenshot 17900 of a support page.
- [00206] Figure 180 is an exemplary screenshot 18000 of a support page.
- [00207] Figure 181 is an exemplary screenshot 18100 of a support page.
- 25 [00208] Figure 182 is an exemplary screenshot 18200 of a marketplace homepage.
- [00209] Figure 183 is an exemplary screenshot 18300 of a marketplace page.
- [00210] Figure 184 is an exemplary screenshot 18400 of a marketplace page.

[00211] Figure 185 is an exemplary screenshot 18500 of a marketplace page.

[00212] Figure 186 is an exemplary screenshot 18600 of a marketplace page.

[00213] Figure 187 is an exemplary screenshot 18700 of a marketplace page.

[00214] Figure 188 is an exemplary screenshot 18800 of a marketplace page.

5 [00215] Figure 189 is an exemplary screenshot 18900 of a marketplace page.

[00216] Like reference numerals refer to corresponding parts throughout the drawings.

DESCRIPTION OF EMBODIMENTS

Definitions

10 [00217] A Remote Procedure Call (RPC) is a programming interface that allows one program to use the services of another program in a remote machine. The calling program sends a message and data to the remote program, which is executed, and results are passed back to the calling program. Note that RPC refers to XML RPC.

15 [00218] A virtual host (vhost) is a server that includes multiple web sites, each with its own domain name. A <virtualhost> </virtualhost> is an Apache HTTP server directive (instruction) that maps domain names to different directories (and other instructions) on the filesystem. vhosts can be used to define the boundaries of an application. Each web-based application in an account on the client computer system must have at least one virtual host in Apache. Note that there can be more than one vhost, all pointing to the same shared directory/config. Also note that other web servers have similar functions as the Apache
20 virtualhost directive.

[00219] Vmap is a virtual database query map. A vmap is a variable map, or a field map. It maps variables (columns, fields) in one database to variables (columns, fields) in another DB.

25 [00220] LAMP is a solution stack of software that is used to run dynamic web sites. The LAMP solution stack typically comprises open source software. The LAMP stack can refer to a suite of software that includes LINUX, Apache HTTP server, MySQL and/or PostgreSQL, Perl, Python, Ruby, Ruby on Rails, Apache Tomcat, and/or PHP.

[00221] A solution stack is a set of software subsystems or components that are required to deliver a specified solution (e.g., product or service).

[00222] A web application or a web-based application is an application that can be accessed through the web over a network (e.g., the Internet, etc.). Web-based applications typically generate dynamic content. Dynamic content is content that can be generated when a request is received from a user. For example, a user may request scores for one or more sporting events. These scores can be retrieved and a page listing the scores can be displayed to the user. Alternatively, dynamic content can be periodically generated and cached. When a user requests the dynamic content, the cached version is displayed to the user. Web-based applications do not require distribution because they are typically hosted on an application server. Users who desire to use a web-based application can use a browser to access the application server hosting the web-based application. A client-server model requires a specialized client program that serves as a user interface to the server and that must be installed on each computer system that is to access the server application. Any upgrades to the server application typically require an update to the client application. In contrast, web-based applications use a browser engine, such as found in a web browser engine, as an interface to the application server. In general, each page delivered to the browser is a static document; however, the static document is typically composed of a number of dynamic elements that are embedded into the document prior to being delivered to the browser. Web-based applications are typically structured as a three-tier application with a browser engine at the first tier, an engine that can process dynamic content (e.g., scripting languages, etc.) in the second tier, and a database in the third tier.

Overview

[00223] Presently, in order to receive the full functionality of a web-based application, a client computer system must be able to communicate with an application server hosting the web-based application. Although some features of a web-based application may be available while the client computer system is not connected to an application server hosting the web-based application, other features of the application server may not function properly or may not function at all if the client computer system is not connected to the application server. For example, consider a user who wants to update a contact in a web-based address book. Updating a contact in a web-based address book typically requires updating data records in a database or some other mechanism for storing and managing data (e.g., files). Typically, the database is part of the application server or accessible to the application server through a

network connection. Thus, in order to update a contact in the web-based address book, the client computer system for the user must be connected to the application server hosting the web-based address book so that the update to the contact can be made to in the database.

[00224] Thus, in some embodiments, a client computer system is loaded with a framework that allows the client computer system to access web-based applications locally without requiring a network connection to an application server hosting the web-based application. This framework is referred to as the “Applications on a Plane” (AOP) framework in this specification. This framework is useful for at least traveling salespeople, drivers, business travelers (e.g., on airplanes), etc. In these embodiments, while working on the application locally, data can be added, modified, and changed without the need to be connected to the application server hosting the web-based application. The AOP framework tracks the changes and when the network connection between the client computer system and the application server is reestablished, the changes can be synchronized between the client computer system and the application server.

[00225] In some embodiments, the AOP framework can include an application server and an account management system. In some embodiments, the application server is the Etelos Application Server (EAS). In some embodiments, the account management system is the Etelos Management System (EMS). This specification may use the terms EAS and EMS in the generic sense to refer to an application server and an account management system, respectively.

[00226] In some embodiments, AOP framework installations can be split into two steps: installation of an open source stack (e.g., Apache HTTP server, PHP, PostgreSQL, MySQL, Python, etc.) and installation of the AOP framework (e.g., the EAS and EMS installation). Other software stacks can be used. For example, WAMP (Windows, Apache HTTP server, MySQL, PHP) and/or LAPP (Linux, Apache HTTP server, PostgreSQL, PHP).

[00227] In some embodiments, AOP synchronization is managed at two levels. A user-based synchronization enables the user to synchronize information that only the user is allowed to see based upon a set of user-based synchronization rules. A group administrative synchronization synchronizes all changes in the web-based application for a user that a user is allowed to see because of group administrative permissions.

[00228] In some embodiments, the AOP framework allows existing web-based applications to support offline use on a client computer system (e.g., without a connection to

an application server that hosts the web-based application) without changing the architecture and code for the web-based application. For example, if a developer builds a web-based application based on the Linux, Apache HTTP server, MySQL, and PHP (LAMP) web development environment, the web-based application can be used in the AOP framework with little or no code changes. A developer can then use an administrative tool to create rules for how the web-based application is to be synchronized with client computer systems that have made changes to the web-based application while offline. Note that in prior art systems, a developer must re-architect (e.g., changing the data model) and/or recode the web-based application to be able to run on a client computer system without a network connection to an application server.

[00229] In some embodiments, the AOP framework assigns a local domain extension to a universal resource locator (URL) for a web-based application so that a user can access the web-based application on the client computer system instead of on the application server. In doing so, a user can choose when to access the web-based application locally and when to access the web-based application on the application server. For example, the local domain extension can be an “.aop” suffix that is added to the end of the URL. If the URL is `http://app1.com`, the modified URL is `http://app1.com.aop`. If a user wants to run the web-based application on the AOP framework on a client computer system, the user can access the web-based application by entering the local URL (e.g., `http://app1.com.aop`). Note that the “.aop” suffix is one example of a suffix that can be appended to a URL. Any other suffix can be appended to a URL. Alternatively, the URL can be modified in other ways (e.g., completely rewriting the URL, etc.) that can indicate local access is required. In some embodiments, an entry in a hosts file is added to map the URL with the appended suffix to the local client computer system. In other embodiments, an entry in a domain naming service is added to map the URL with the appended suffix to the local client computer system.

[00230] In some embodiments, a suffix is not appended to the URL. In these embodiments, the URL itself is used to direct the browser to the local web server on the client computer system. In these embodiments, an entry in the hosts file or in a DNS server on the client computer system associates the URL with the client computer system. Note that since a client computer system starts its search for an IP address associated with the URL on the client computer system, if an IP address is found in the hosts file or a local DNS server on the client computer system, the client computer system uses this IP address regardless as to whether or not another IP address (e.g., the real IP address exists).

[00231] Allowing users to make changes to web-based applications while disconnected from an application server hosting the web-based application creates several problems including synchronization of data between the client computer system and the application server. As long as the client computer system is connected to the application server, the two systems can remain synchronized with each other by periodically communicating with each other (e.g., through polling or through triggers). However, offline use of web-based applications can cause data conflicts because users can be creating, modifying, and deleting records on different instances of the web-based application. This problem is compounded by the fact that most web-based applications use automatically incrementing database identifiers to provide a unique identifier for a given database record. For example, for a user table, the user ID column may use an automatically incrementing user ID generator. When a new user is added to the user table, the database determines the next unique user ID from the automatically incrementing user ID generator. Since, each instance of the web-based application includes the automatically incrementing user ID generator that increments the next user ID independently of the other instances of the web-based application, it is possible that the user IDs between all of the instances of the web-based application are not unique. Thus, in some embodiments, the AOP framework provides a synchronization technique that can solve the above-described problems. The synchronization technique can be separate from the web-based application so that the code for the web-based application does not need to be modified. Not modifying the code is advantageous for at least the reason that rearchitecting and recoding web-based applications can be a burdensome and time-consuming process.

[00232] Most applications that are designed for the web are designed to run on a single large piece of infrastructure with shared resources. This technique allows a “software as a service” (SaaS) provider to scale the infrastructure as needed. Since many users (e.g., companies) may be using the same database separated only by differences in primary keys, control to applications and databases are set so that security is not compromised. These security models are difficult to port to existing systems that allow use of web-based applications while disconnected from an application server. However, the AOP framework solves these problems as described below. In some embodiments, the AOP framework enables users to see only their own data and not see data from other users. However, if a user is a part of an administrative group, the administrative user can see data for other users over which the administrative user has administrative rights.

[00233] In some embodiments, the AOP platform provides mechanisms for distribution of web-based applications through a marketplace. These mechanism can facilitate billing services (e.g., for purchases, subscriptions, and other licenses) and user management.

5

AOP

[00234] Figure 1 presents a block diagram of network 100, according to embodiments of the present invention. Network 100 includes clients 110-A to 110-N and application servers 130-A to 130-N. Clients 110-A to clients 110-N and application servers 130-A to 130-N are connected to each other through network 120. Network 120 can include, but is not limited to, a local area network (LAN), a wide area network (WAN), the Internet, an intranet, a wireless network, a mobile network, a combination of networks, or any type of network now known or later developed. Clients 110-A to 110-N can not only communicate with application servers 130-A to 130-N through network 120, but can also communicate with each other through network 120. Similarly, application servers 130-A to 130-N can communicate with each other through network 120.

10

15

[00235] In some embodiments, application servers 130-A to 130-N include one or more web-based applications. In some embodiments, a web-based application is an application that is hosted on an application server and that can be accessed by clients that are connected to the application server through a network. Although a web-based application may have a set of functionality that can be used without a network connection to the application server hosting the web-based application, a web-based application typically has another set of functionality that cannot be used by a client computer system unless the client computer system is connected to the application server hosting the web-based application. For example, the set of functionality that requires a network connection to the application server can include functionality that requires access to data stored in a database for the web-based application.

20

25

[00236] Figure 2 presents a block diagram of network 200, according to embodiments of the present invention. Network 200 includes clients 210-A and 210-B, network 120, and application server 130. Clients 210-A and 210-B may correspond to any one of clients 110-A to 110-N illustrated in Figure 1. Application server 130 may correspond to any one of the application servers 130-A to 130-N illustrated in Figure 1.

30

[00237] Clients 210-A and 210-B include browser 212, TCP/IP communications module 220, local application 214, local database 216, stack 218, synchronization application 220, traffic management module 222, license authentication module 224, copy protection and data security module 226. Browser 212 can include any application that can access data and/or services on a remote computer system through a network (e.g., network 120). In some embodiments, browser 212 is a web browser. TCP/IP communications module 220 provides procedures and routines that allow clients 210-A and 210-B to communicate with other computer system through network 120 using the TCP/IP protocol.

[00238] Local application 214 can include any application that can be run on a client. In some embodiments, local application 214 is an instance of a web-based application that is hosted on an application server (e.g., application server 130).

[00239] Local database 216 can include a database that can be used by local application 214. For example, local database 216 can include, but is not limited to, MySQL, PostgreSQL, ORACLE, etc. In some embodiments, local database 216 includes a user database and an application database. The user database is described in more detail below with reference to Figure 4. The application database is described in more detail below with reference to Figure 3.

[00240] Stack 218 can include a number of software packages that enable a web-based application to run on a client. In some embodiments, the software packages can include, but are not limited to, a web server (e.g., Apache HTTP Server), a database (e.g., MySQL, PostgreSQL, ORACLE), application servers (e.g., Apache Tomcat), scripting languages (e.g., Python, Ruby, Ruby on Rails, etc.), and libraries. In some embodiments, stack 218 includes open source software (OSS) packages. In some embodiments, the OSS packages include Apache HTTP server, MySQL, PostgreSQL, Apache Tomcat, Python, and libraries. Stack 218 is described in more detail below with reference to Figure 12.

[00241] Synchronization application 220 can synchronize a web-based application that is running on a client (e.g., clients 210-B) with a corresponding web-based application running on an application server (e.g., application server 130). Synchronization application 220 is described in more detail below with reference to Figures 6 and 23-28.

[00242] Traffic management module 222 can manage network traffic between the client and other computer systems. For example, if a user using client 210-A enters a universal resource locator (URL) into browser 212, traffic management module determines

an Internet protocol (IP) address for the URL. In some embodiments, traffic management module 222 first looks at a hosts file for client 210-A. The hosts file can map URLs to IP addresses. In some embodiments, for web-based application that are installed on client 210-A, the hosts file can include an entry that associates an IP address for client 210-A (e.g., 127.0.0.1) with the URL. If an entry for the URL exists in the hosts file, traffic management module returns the IP address associated with the URL. If an entry for the URL does not exist in the hosts file, traffic management module 222 can query a dynamic naming service (DNS) server to determine an IP address for the URL. The DNS server may be on client 210-A or may be on a remote DNS server. Note that if a network connection to a remote DNS server does not exist, the client computer system may resort to using the local hosts file and/or the local DNS server to resolve IP addresses. If an entry for the URL is found in a DNS server, traffic management module 222 returns the IP address associated with the URL. If an IP address associated with the URL is not found, an error message can be returned.

[00243] License authentication module 224 can be used to verify whether a given user or a given client can use a web-based application loaded on the given client. Copy protection and data security module 226 can protect data that is located in local database 216 and/or local application 214 from being accessed by unauthorized users. For example, copy protection and data security module 226 can protect data by using access control mechanisms to restrict access to data. Alternatively, copy protection and data security module 226 can protect data by encrypting the data. Copy protection and data security module 226 can also work with license authentication module 224 to prevent users that have expired licenses from accessing data stored in local database 216.

[00244] As illustrated in Figure 2, client 210-B is operating in an AOP mode whereas client 210-A is operating in a networked mode. In a networked mode, a user on client 210-A can access a web-based application on application server 130 through network 120. In contrast, a user on client 210-B does not have a connection to application server 130, and thus cannot access functionality for a web-based application that requires a network connection to application server 130. However, since client 210-B has the AOP framework installed, client 210-B can operate in an AOP mode where the full functionality of web-based application (including functionality that normally requires a network connection to application server 130) is available to the user. In some embodiments, a local web server 228 runs web-based application locally. In some embodiments, local web server 228 can be part of stack 218. Note that client 210-A can also include local web server 228. If client 210-A loses a

connection to network 120, client 210-A can load the AOP framework and use a local web server to run the web-based application locally until the network connection is restored.

[00245] Application server 130 can include one or more of: lightweight directory access protocol (LDAP) gateway 242, web server engine 246, auxiliary services 250, synchronization, access and query engine 260, applications database 262, user database 270, and ecommerce services 280. LDAP gateway 242 can provide directory services to clients through network 120. Note that LDAP gateway 242 is optional in some embodiments.

[00246] Web server engine 246 can respond to requests for static web pages, dynamically generated web pages, and web-based applications. These requests can come from clients (e.g., 210-A and 210-B) or from other application servers. In some embodiments, web server engine 246 is an open source web server (e.g., Apache HTTP server). Web server engine 246 can access LDAP gateway 242, auxiliary services 250, user database 270, synchronization, access, and query engine 260, and e-commerce services 280.

[00247] Auxiliary services 250 can include, but are not limited to: famfamfam (icon images), phpMyAdmin (php web-based MySQL database management interface), phpPGAdmin (php web-based PostgreSQL database management interface), WebSVN (php web-based Subversion interface), ImageMagick (image manipulation library), ZendFramework (php utility framework), IconCube Loaders (encrypted-php decryption library), libpng (PNG manipulation library), libjpeg (JPEG manipulation library), Neon (WebDAV client library), mcrypt (encryption library), and FreeType (font utilities library).

[00248] Synchronization, access, and query engine 260 can synchronize data and files between a web-based application hosted on application server 130 and a corresponding web-based application hosted on a client (e.g., clients 210-A and 210-B). Synchronization, access, and query engine 260 can include rules for conflict management and techniques for handling automatically incrementing record identifiers in a database for web-based applications. Synchronization, access, and query engine 260 can access applications database 262, which can include information about web-based applications available on the application server and can include data for the web-based applications. Synchronization, access, and query engine 260 is described in more detail below with reference to Figures 5 and 23-28 below. Applications database 262 is described in more detail below with reference to Figure 3 below.

[00249] User database 270 can include information about users. In some embodiments, user database 270 can be used to track users that are allowed to access web-based applications on application server 130. User database 270 is described in more detail below with reference to Figure 4.

5 [00250] E-commerce services 280 can provide payment and order fulfillment services. In some embodiments, e-commerce services 280 includes process payment module 282, authenticate license module 284, and serve application module 286. Process payment module 282 can process payments for goods and services. For example, process payment module can authorize payments by credit card, debit cards, electronic funds transfers, or other
10 payment mechanisms (e.g., credits, prepaid tokens, etc.). Authenticate license module 284 can verify that a user has a valid license for a specified service and/or web-based application. E-commerce services 280 can access user database 270 to determine license information and/or payment information. Serve application module 286 can serve applications to a user after a product or service has been purchased or after a license has been verified.

15 [00251] Figure 3 presents a block diagram of application database 300, according to embodiments of the present invention. Application database 300 includes records 302 to 306 for application 1 to application M, respectively. In some embodiments, applications 1 to application M are web-based applications hosted on an application server. Note that in general there can be any number of applications stored in applications database 300. Each
20 application can have a number of records associated with the application.

[00252] Record 302 for application 1 can include information about an application hosted on an application server. For example, record 302 can include, but is not limited to, application name 330, application title 332, application host 334, application user 336, and application password 338. Application name 330 can be a name for the application.

25 Application title 332 can be a title for the application. Application host 334 can be a URL and/or an IP address that is associated with the application on the application server. Application user 336 can be the user or group of users who can manage the application. Application password 338 can be a password for accessing the management features for the application on the application server. Records 304 to 306 for application 2 to application M,
30 respectively, are similar to record 302 for application 1.

[00253] Record 1 312 for application 2 can include metadata and content for application 2. For example, record 1 312 can include, but is not limited to, record

metadata 350, record log 352, and record content 1 354-1 to record content N 354-N. Record metadata 350 can include information about a given record. For example, the metadata can include, but is not limited to, the date and time the record was created, the user that created the record, and/or the IP address of the user who created the record. Record log 352 can include a log for record 312. For example, record log 352 can be used when synchronizing application 2 between the application server and a client computer system. Record content 1 354-1 to record content N 354-N can include content for record 1 312. In some embodiments, record 1 312 to record N 314 can be in the same table within application database 300. In other embodiments, record 1 312 to record N 316 are in different tables within application database 300 or in other databases linked to application database 300. The other records 308-310 and 314-318 are similar to record 312.

[00254] Application index 360 is an index to the applications available in application database 300 (e.g., applications 1, 2, ... M). For example, applications index can include exemplary applications such as Etelos CRM (ECRM) 370, Media Wiki 372, phpBB3 374, Projects 376, Sugar CRM 378, and/or Wordpress 380.

[00255] In some embodiments, each application hosted on an application server is stored in a separate database. These separate databases can be located on the application server or on remote database servers.

[00256] Note that the discussion above is one example of an application database. The information included in the application database can include more or fewer records than what are illustrated in Figure 3.

[00257] Figure 4 presents a block diagram of user database 400, according to embodiments of the present invention. User database 400 includes a number of user records 402-1 to 402-N. User database 400 also includes map 470 that can map user IDs to user records. For example, as illustrated in Figure 4, map 470 maps user ID 472 to user record 402-2.

[00258] User records 402-1 and 402-N are similar to user record 402-2. Thus, the description of user record 402-2 below applies to user records 402-1 and 402-N. User record 402-2 includes, but is not limited to, user ID 410, user metadata 412, query/contact list 414, user client device ID/type 416, user preferences 418, user authentication information 420, user personal information 422, and user enabled features 424. In some embodiments, user ID 410 can be a unique identifier for a user within user database 400. In other embodiments,

user ID 410 can be a unique identifier for a user across a specified set of databases (e.g., all or a subset of the database) in the AOP framework. User metadata 412 can include metadata information about the user record. For example, the metadata information can include, but is not limited to, a date and time when the user record was created, a user who created the user record, the IP address of the user who created the user record, etc.

[00259] Query/contact list 414 can include queries that are used to retrieve contact records for a user (e.g., user-rule 1 460-1 to user-rule 460-N). The user rules can also be used to retrieve contact information, sales opportunities, filter database information, rules, tasks, appointments, and other contact-related information. For a given contact within query/contact list 414, information related to the contact can be stored in the contact records for the contact. For example, the information can include contact information for the contact (e.g., name, phone number, fax number, address, email address, etc.), action items due to the contact, a last interaction with the contact, etc.

[00260] User client device ID/type 416 can store information about the type or IDs of computer devices that the user has used to access applications on the application server. For example, user client device ID/type 416 can indicate that a user used a laptop and a PDA to access applications on the application server. This information can then be used to generate a response that is substantially optimized for the computer device that the user is using to access the application.

[00261] User preferences 418 can include preferences for the user. These preferences can be used when generating responses for the user. For example, the preferences can include, font types and sizes, color schemes, etc. User authentication information 420 can be used to authenticate a user. For example, the authentication information can be a username/password combination for the user or can be a digital certificate for the user.

[00262] User personal information 422 includes, but is not limited to, name 430, phone number 432 (e.g., home, work, mobile, pager, fax, etc.), email addresses 434, office information 436 (e.g., company name, office address, phone number, fax number, etc.), and/or department 438.

[00263] User enabled features 424 includes a list of features on the application server that have been enabled for the user. In some embodiments, user enabled features 424 are determined by the license (e.g., subscription, free, purchased, etc.) granted to the user. In some embodiments, user enabled features 424 are determined by the features that were

purchased by the user. In some embodiments, user enabled features 424 are determined by the web-based applications that are available on an application server. Exemplary user enabled features 424 can include, but are not limited to, Application 1 440, Application N 442, EDE 444, Devkit 446, Schedule events 448, user management 450, AOP framework 452, distribute 454, and integrate 456. User management 450 includes a list of rights and privileges for a user. For example, user management 462-1 can include, but is not limited to, administrative rights and access privileges for the user associated with user record 402-2. In some embodiments, these rights and privileges can be determined based on the applications available for the user and/or user enabled features 424. Note that user admin rights and access privileges are described in more detail below with reference to Figure 9.

[00264] Note that the information in the user records can be located within one or more tables of user database 400 and/or in other associated databases. Also note that the discussion above is one example of a user database. The information included in the user database can include more or fewer records than what are illustrated in Figure 4.

[00265] In some embodiments, user database 400 is a distributed database. In some embodiments, user databases 400 can be located on the application server or on remote database servers.

[00266] Figure 5 presents a block diagram of synchronization engine 500 on an application server, according to embodiments of the present invention. Synchronization engine 500 includes a synchronization data module 502, a conflict management module 504, synchronization rules 506, and synchronization operations 508.

[00267] Synchronization data module 502 includes data used by synchronization engine 500 to synchronize a web-based application on a client computer system with a web-based application on the application server. As illustrated in Figure 5, synchronization data module 502 includes files 562, application 510, application table 512, application column 514, and primary key 516.

[00268] Conflict management module 504 resolves conflicts between the application server and client computer systems. These conflicts can arise when client computer systems operate web-based applications while not connected to the application server. For example, if a web-based application uses an automatically incrementing identifier for database records, client computer systems that are not connected to the application server can unknowingly use the same identifiers as the application server for different data resulting in data conflicts.

Thus, in some embodiments, a record increment module 520 is provided to resolve conflicts for web-based application that use automatically incrementing identifiers for database records. Conflict management module 504 and record increment module 520 are described in more detail below with reference to Figures 25-28 below.

5 **[00269]** Synchronization rules 506 are rules used by synchronization engine 500 to determine how to synchronize records between a client computer system and an application server. These rules can include developer specified rules 522 and default rules 524.

10 **[00270]** Synchronization operations module 508 includes, but is not limited to, create accounts module 540, create databases module 542, create account directories module 544, install EAS module 546, setup scheduler module 548, and connect to EMS client 550. Create accounts module 540 can be used to create accounts for web-based applications. Create databases module 542 can be used to create the databases for web-based applications and the AOP framework. Create account directories 544 can be used to create the directory structure of web-based applications. Install EAS module 546 can be used to install an application
15 server that can be used to serve web-based applications to clients. In some embodiments, the application server module is the Etelos Application Server (EAS) module. Setup schedule module 548 can setup a synchronization schedule between the application server and client computer systems.

20 **[00271]** Connect to EMS client module 550 can be used to connect an application server with the EMS module on a client computer system. Connect to EMS client module 550 can send data to an EMS module on the client computer system to synchronize the web-based application between the application server and the client computer system. The data can include, but is not limited to, setup data 552, schema 554, files 556, rules 558, and/or data 560.

25 **[00272]** Figure 6 presents a block diagram of synchronization engine 600 on a client computer system, according to embodiments of the present invention. The modules in synchronization engine 600 perform similar functions as the modules in synchronization engine 500. For example, synchronization data module 602 generally corresponds to synchronization data module 502 and synchronization operations 608 generally corresponds
30 to synchronization operations 508. As a result, the discussion above for synchronization engine 600 is not repeated.

[00273] In some embodiments, the file structures for a web-based application in the application server and the client computer system are similar.

[00274] Figure 7 presents a block diagram of application server 700, according to embodiments of the present invention. The application server 700 generally includes one or more processing units (CPU's) 702, one or more network or other communications interfaces 704, memory 710, and one or more communication buses 708 for interconnecting these components. The communication buses 708 may include circuitry (sometimes called a chipset) that interconnects and controls communications between system components. The application server 700 may optionally include a display 706 and one or more input devices 705 (e.g., keyboard, mouse, trackpoint, etc.). Memory 710 includes high-speed random access memory, such as DRAM, SRAM, DDR RAM or other random access solid state memory devices; and may include non-volatile memory, such as one or more magnetic disk storage devices, optical disk storage devices, flash memory devices, or other non-volatile solid state storage devices. Memory 710 may optionally include one or more storage devices remotely located from the CPU(s) 702. Memory 710, or alternately the non-volatile memory device(s) within memory 710, comprises a computer readable storage medium. In some embodiments, memory 710 stores the following programs, modules and data structures, or a subset thereof: operating system 711, network communication module 712, LDAP gateway module 714 (optional), synchronization, access and query module 716, user database 728, e-commerce services module 730, web server engine module 738, application database management module 744, user database management module 754, application database 764, and/or auxiliary services modules 766.

[00275] Operating system 711 includes procedures for handling various basic system services and for performing hardware dependent tasks. Network communication module 712 can be used for connecting the application server 700 to other computers via the one or more communication network interfaces 704 (wired or wireless) and one or more communication networks, such as the Internet, other wide area networks, local area networks, metropolitan area networks, and so on. LDAP gateway module 714 can provides directory services for application server 700 and other computer systems. In some embodiments, LDAP gateway module 714 is optional.

[00276] Synchronization, access, and query module 716 can provides synchronization procedures to synchronize a web-based application on a client computer system with a

web-based application on an application server. Synchronization, access, and query module 716 includes one or more of synchronization module 718, synchronization files module 719, access database module 720, query database module 722, and/or conflict management module 724. Synchronization module 718 can perform synchronization operations between a client computer system and application server 700. Synchronization files module 719 synchronizes files between the client computer system and application server 700. Access database module 720 performs read (e.g., select operations) and write operations (e.g., insert, update, and delete operations) on databases. Query database module 722 performs queries in the databases and returns results of the query. Conflict management module 724 resolves conflicts between application server 700 and client computer systems. Conflict management module 724 can include record increment module 726, which can handle conflicts that arise from automatically incrementing identifiers for database records.

[00277] User database 728 includes user information as described above with reference to Figures 2 and 4. Application database 764 includes application data as described above with reference to Figures 2 and 3.

[00278] E-commerce services module 730 can provide electronic commerce services. E-commerce services module 730 includes one or more of process payment module 732, authenticate license module 734, serve application module 736. E-commerce services module 730 is described in more detail above with reference to Figure 2.

[00279] Web server engine module 738 can serve web pages to client computer system or other application servers. In some embodiments, web server engine module 738 includes web development environment module 740, which provides software and tools to build and serve web-based applications. In some embodiments, web development environment module 740 is LAMP module 741, which includes the LINUX operating system, Apache HTTP server, the MySQL database management system, and the PHP scripting language.

[00280] The components of LAMP module 741 can be substituted for other compatible technologies. In general, LAMP module 741 includes an operating system, a web server, a database, and support for a scripting language. For example, LAMP module 741 can include WAMP (Windows, Apache HTTP, MySQL, PHP) and/or LAPP (Linux, Apache HTTP, PostgreSQL, PHP). LAMP module 741 can also include Apache Tomcat. In some embodiments, web server engine module 734 includes a Python module 742 that supports the

Python scripting language. Optionally, Python module 742 can also support Ruby, Ruby on rails, and/or any other languages.

[00281] Application database management module 744 provides an interface to manage and access an applications database for web-based applications that are hosted on application server 700. Application database management module 744 can include an add/delete application module 746, a synchronize application module 748, and an application access module 750. Add/delete application module 746 allows for the addition or deletion of web-based application records in application database 764 on application server 700.

Synchronize application module 748 synchronizes applications database 764 with an application database on a client computer system. Application access module 750 determines whether a user is allowed to access a given application within application server 700.

[00282] User database management module 754 provides an interface to manage and access a user database for web-based applications that are hosted on application server 700. Users database management module 754 can include an add/delete user module 756, an edit user information module 758, a user authentication module 760, and/or a user permissions module 762. Add/delete user module 756 allows for the addition or deletion of users in user database 728 on application server 700. Edit user information module 758 allows for the editing of user information in user database 728. User authentication module 760 authenticates users by checking user credentials stored in user database 728 (e.g., by checking a username and password for a user, or a digital certificate). User permissions module 762 determines whether a user is allowed to access specified resources and/or applications within application server 700.

[00283] Auxiliary services module(s) 766 includes, but is not limited to: famfamfam (icon images), phpMyAdmin (php web-based MySQL database management interface), phpPGAdmin (php web-based PostgreSQL database management interface), WebSVN (php web-based Subversion interface), ImageMagick (image manipulation library), ZendFramework (php utility framework), IconCube Loaders (encrypted-php decryption library), libpng (PNG manipulation library), libjpeg (JPEG manipulation library), Neon (WebDAV client library), mcrypt (encryption library), and FreeType (font utilities library).

[00284] Each of the above identified elements may be stored in one or more of the previously mentioned memory devices, and corresponds to a set of instructions for performing a function described above. The above identified modules or programs (i.e., sets

of instructions) need not be implemented as separate software programs, procedures or modules, and thus various subsets of these modules may be combined or otherwise re-arranged in various embodiments. In some embodiments, memory 710 may store a subset of the modules and data structures identified above. Furthermore, memory 710 may store
5 additional modules and data structures not described above.

[00285] Although Figure 7 shows an “application server,” Figure 7 is intended more as functional description of the various features that may be present in a set of servers than as a structural schematic of the embodiments described herein. In practice, and as recognized by those of ordinary skill in the art, items shown separately could be combined and some items
10 could be separated. For example, some items shown separately in Figure 7 could be implemented on single servers and single items could be implemented by one or more servers. The actual number of servers used to implement an application server and how features are allocated among them will vary from one implementation to another, and may depend in part on the amount of data traffic that the system must handle during peak usage
15 periods as well as during average usage periods.

[00286] Figure 8 presents a block diagram of client 800, according to embodiments of the present invention. The client 800 generally includes one or more processing units (CPU's) 802, one or more network or other communications interfaces 804, memory 810, and one or more communication buses 808 for interconnecting these components. The
20 communication buses 808 may include circuitry (sometimes called a chipset) that interconnects and controls communications between system components. The client 800 includes a display 806 and one or more input devices 805 (e.g., keyboard, mouse, trackpoint, etc.). Memory 810 includes high-speed random access memory, such as DRAM, SRAM, DDR RAM or other random access solid state memory devices; and may include non-volatile
25 memory, such as one or more magnetic disk storage devices, optical disk storage devices, flash memory devices, or other non-volatile solid state storage devices. Memory 810 may optionally include one or more storage devices remotely located from the CPU(s) 802. Memory 810, or alternately the non-volatile memory device(s) within memory 810, comprises a computer readable storage medium. In some embodiments, memory 810 stores
30 the following programs, modules and data structures, or a subset thereof: operating system 811, network communication module 812, DNS support module 813, receive and process user input module 814, display module 815, browser engine module 816, synchronization, access and query module 818, e-commerce client module 826, local web server engine

module 834, local application database module 848, local user database management module 858, application database 868, and/or user database 870.

[00287] Operating system 811 includes procedures for handling various basic system services and for performing hardware dependent tasks. Network communication module 812 can be used for connecting the client 800 to other computers via the one or more communication network interfaces 804 (wired or wireless) and one or more communication networks, such as the Internet, other wide area networks, local area networks, metropolitan area networks, and so on. DNS support module 813 provides DNS services for client 800. Receive and process user input module 814 receives and processes user inputs received from input devices 805. Display module 815 displays a user interfaces for applications running on client 800. Browser engine module 816 can include any application with a rendering engine that can access data and/or services on a local or a remote computer system and render the results so that a user can view the data and/or interact with the services. In some embodiments, browser engine module 816 is a web browser.

[00288] Synchronization, access, and query module 818 can provides synchronization procedures to synchronize a web-based application on a client computer system with a web-based application on an application server. Synchronization, access, and query module 818 includes one or more of synchronize files module 819, synchronization module 820, local copy of database module 822, and edit local database module 824.

Synchronization module 820 can perform synchronization operations between client 800 and an application server. Local copy of database module 822 makes a local copy of databases located on an application server. Edit local database module 824 provides an interface to edit the local copy of the databases.

[00289] E-commerce client module 826 can provide electronic commerce services that enable use of web-based applications on client 800. E-commerce client module 826 includes one or more of download and enable application module 828, authenticate license module 830, and/or copy protection and data security module 832. Download and enable application module downloads and enables web-based applications from an application server. Authenticate license module 830 determines whether the license for a web-based application is valid. If so, a user is allowed to access the web-based application. Otherwise, a user is prevented from accessing the web-based application. Copy protection and data security module 832 can protect data that is located in local databases (e.g., an application

database or a user database) from being accessed by unauthorized users. Copy protection and data security module 832 can work with authenticate license module 830 to prevent users that have expired licenses from accessing data stored in local databases.

[00290] Local web server engine module 834 can serve web pages to client 800 or to other computer systems. In some embodiments, local web server engine module 834 includes web development environment module 836, framework language module 840, AOP framework module 844, and/or traffic management module 846. Web development environment module 836 provides software and tools to build and serve web-based applications. In some embodiments, web development environment module 840 includes a number of open source software (OSS) packages. Thus, web development environment module is sometimes referred to as a software stack. In some embodiments, web development environment module 836 includes LAMP module 838, which includes the LINUX operating system, Apache HTTP server, MySQL database management system, and support for the PHP scripting language and Apache Tomcat. The components of LAMP module 838 can be substituted for other compatible technologies (e.g., WAMP, LAPP, etc.). In general, LAMP module 838 includes an operating system, a web server, a database, and support for a scripting language. Framework language module 840 provides support for one or more programming languages used to implement the AOP framework. In some embodiments, framework language module 840 includes python module 842 which supports the Python, Ruby, and/or Ruby on Rails scripting language. AOP framework module 844 provides data structures and procedures for running web-based applications on a client computer system (e.g., client 800). AOP framework module 844 is described in more detail below with reference to Figures 10-28. Traffic management module 846 manages network traffic between client 800 and other computer systems. Traffic management module 846 is described in more detail above with reference to Figure 2.

[00291] Local application database management module 848 provides an interface to manage and access applications database 868 for web-based applications that are hosted on client 800. Local application database management module 848 can include an add/delete record module 850, an application database access module 852, and/or an application database query module 854. Add/delete record module 850 allows for the addition or deletion of web-based application records in application database 868. Application database access module 852 performs reads and writes into application database 868. For example, application database access module 852 can process requests for retrieving data, adding

records, deleting records, and editing records. Application database query module 854 performs queries on and returns results from application database 868.

[00292] Local user database management module 858 provides an interface to manage and access user database 870 for web-based applications that are hosted on client 800. Local user database management module 858 can include an add/delete user module 860, an edit user information module 862, a user authentication module 864, and/or a user permissions module 866. Add/delete user module 860 allows for the addition or deletion of users in user database 870. Edit user information module 862 allows for the editing of user information in users database 868. User authentication module 864 authenticates users by checking user credentials stored in user database 870 (e.g., by checking a username and password for a user, or a digital certificate). User permissions module 866 determines whether a user is allowed to access specified resources and/or applications within client 800.

[00293] Auxiliary services module(s) 864 includes, but is not limited to: famfamfam (icon images), phpMyAdmin (php web-based MySQL database management interface), phpPGAdmin (php web-based PostgreSQL database management interface), WebSVN (php web-based Subversion interface), ImageMagick (image manipulation library), ZendFramework (php utility framework), IconCube Loaders (encrypted-php decryption library), libpng (PNG manipulation library), libjpeg (JPEG manipulation library), Neon (WebDAV client library), mcrypt (encryption library), and/or FreeType (font utilities library).

[00294] Application database 868 is described in more detail above with reference to Figures 2 and 3 above. User database 870 is described in more detail above with reference to Figures 2 and 4 above.

[00295] Each of the above identified elements may be stored in one or more of the previously mentioned memory devices, and corresponds to a set of instructions for performing a function described above. The above identified modules or programs (i.e., sets of instructions) need not be implemented as separate software programs, procedures or modules, and thus various subsets of these modules may be combined or otherwise rearranged in various embodiments. In some embodiments, memory 810 may store a subset of the modules and data structures identified above. Furthermore, memory 810 may store additional modules and data structures not described above.

AOP Framework

[00296] Figure 10 presents a block diagram 1000 of exemplary AOP framework 100 which provides offline access to web-based application 1006, according to embodiments of the present invention. In some embodiments, client computer system 1001 is coupled to application server 1005 through network 120. Network 120 can include, but is not limited to, a local area network (LAN), a wide area network (WAN), the Internet, a mobile network, and/or a wireless network. Client computer system 1001 includes operating system 1002, browser 1003, and AOP framework 1010. As illustrated in Figure 1, AOP framework 1010 includes software stack 1004 and web-based application 1007. Software stack 1004 may be comprised of open source software, commercial software, or a combination of both. Note that the details of AOP framework 1010 are described in more detail below with reference to Figures 10B-28 below. Application server 1005 includes web-based application 1006.

[00297] Consider a user who wants to access web-based application 1006 located on application server 1005. For example, the URL for the web-based application can be <http://app1.com>. The user can use browser engine 1003 on client computer system 1001 to access the URL <http://app1.com>. When client computer system 1001 has a network connection with application server 1005, application server 1005 can activate web-based application 1006 to respond to the request from browser engine 1003.

[00298] On a client computer system without AOP framework 1010, the user can only access the full functionality of web-based application 1006 when there is a network connection between the client computer system and application server 1005. However, since client computer system 1001 includes AOP framework 100, the user can access the full or substantially the full functionality of web-based application 1006 when there is no network connection between client computer system 1001 and the application server 1005. As illustrated in Figure 1, AOP framework 1010 includes web-based application 1007, which is a local instance of web-based application 1006. Software stack 1004 provides the software required to execute web-based application 1006 on client computer system 1001. In some embodiments, Software stack 1004 is the same set of software used on application server 1005 to run web-based application 1006.

[00299] When client computer system 1001 does not have a network connection with application server 1005, AOP framework 1010 can allow the user to continue having access to the functionality of web-based application 1006 by activating web-based application 1007

on client computer system 1001. Note that the loss of a network connection with application server can result from the user manually disconnecting the network connection (e.g., through software or hardware) or can be a result of external factors (e.g., power outages, network outages, etc.). In some embodiments, AOP framework 1010 automatically activates web-based application 1007 when the network connection to application server 1005 no longer exists. In other embodiments, after the network connection to application server 1005 no longer exists, AOP framework 1010 waits for the user to indicate that web-based application 1007 should be activated.

[00300] Figure 11 illustrates an exemplary process 1100 of using AOP framework 1010 to run web-based application 1006 while offline, according to embodiments of the present invention. Note that Figure 11 corresponds to the block diagram in Figure 10. The process begins when client computer system 1001 receives an input from a user (1102). Client computer system 1001 starts a browser application (1104). Client 1001 then receives a user request to visit a web page or an application (1106). The user request can include a URL. Optionally, client computer system 1001 then detects if a connection cannot be made to the web page or the application (1108). For example, the user may be offline and may not have an Internet connection. If the connection cannot be made, then a web server on client computer system 1001 responds to the request (1110). Client computer system 1001 can then perform one or more of the optional steps 1112-1118. Client computer system 1001 can run the application locally using the client's operating system (1112). Client computer system 1001 can run the software in conjunction with the AOP framework (1114). Client computer system 1001 can receive a user request to navigate the browser to a web address (e.g., a URL). For example, the URL can be <http://appl.com.aop>. Note that the extension ".aop" can be replaced with any other extension. AOP framework 1010 can then respond to the request for the specified URL and activates web-based application 1007 to respond to the request. Client computer system 1001 can then detect when a connection can be made to a remote web page or a server corresponding to the local virtual instance (e.g., application server 1005) (1118). When the network connection between client computer system 1001 and application server 1005 is reestablished, any changes made in web-based application 1007 on client computer system 1001 can be synchronized with web-based application 1006 on application server 1005. In some embodiments, client computer system 1001 is synchronized with application server 1005 using Network 120.

[00301] In some embodiments, the AOP framework modifies the operating system's network hosts file so that application domains ending with a specified domain suffix are treated as local domains served by a local web server. For example, if the specified domain suffix is ".aop", an application may have the universal resource locator (URL)

5 "app1.com.aop". In some embodiments, the local web server is on the same client computer system as the client computer system running the AOP framework. For example, if the AOP framework is running on a laptop, the laptop may also include a local web server.

[00302] In some embodiments, when a user attempts to use a browser to access an application that has the specified domain suffix, the request is handled by a local web server and control is delegated to the application associated with the domain name on the client
10 computer system.

[00303] In some embodiments, the AOP framework supports web application development languages including, but not limited to, PHP, Python, Ruby, and/or Ruby on Rails.

15 [00304] In some embodiments, the AOP framework determines all local changes made to applications on the client computer system and synchronizes these changes with the application server. In some embodiments, the synchronization is performed on a predetermined frequency. In other embodiments, the synchronization is performed on demand.

AOP Installation

[00305] The AOP framework enables web-based applications to run locally without requiring a network connection to an application server hosting the web-based application. As a result, some embodiments install software packages on a client computer system. These
25 software packages can include, but are not limited to a web server (e.g., Apache HTTP Server), a database (e.g., MySQL, PostgreSQL), application servers (e.g., Apache Tomcat), scripting languages (e.g., Python), and libraries. In some embodiments, the software package includes Apache HTTP server, MySQL, PostgreSQL, Apache Tomcat, Python, and libraries. In other embodiments, the software packages are included in the package for the AOP
30 framework.

[00306] Figure 12 presents a block diagram 1200 of an exemplary process of installing an instance of an AOP framework on a client computer system, according to embodiments of the present invention. The installation process includes downloading OSS packages (1201) and an AOP package (1202), installing the software packages (1203), installing the AOP package (1204), creating an account (1205), initializing an account (1206), connecting to the application server (1207), downloading application files from the application server (e.g., using RPC and SVN through Network 120), and setting up an AOP application on the client computer system (1209).

[00307] Figure 13 illustrates an exemplary process 1300 of installing an instance of an AOP framework on a client computer system, according to embodiments of the present invention. Note that Figure 13 corresponds to the block diagram in Figure 12. The process begins when a user downloads software packages (1301). The software packages can be included in a single compressed file. The user also downloads the AOP framework package (1302). The AOP framework package can be included in a compressed file. In some embodiments, the AOP framework package includes an installation utility.

[00308] The user then installs the software packages on the client computer system (1303). If the software packages are already installed on the client computer system, the software packages are not reinstalled. In some embodiments, if the software packages are already installed on the client computer system, the versions of the software packages installed on the client computer system are determined. The software packages are either updated or not updated based on the determined versions. For example, if a minor software version update occurred for a software package installed on the client computer system, the minor software version update may not be applied. In some embodiments, if the software packages are already installed on the client computer system, the software packages may periodically check for updates and apply the updates according to a set of rules. For example, the set of rules can specify that certain software packages can be updated without user intervention whereas others require specified instructions from users and/or the developers of the web-based application. After the software packages are installed, the user starts the AOP package installation process (1304). The AOP package installation process installs the AOP framework. During the installation process, the user enters account information about web-based applications that the user desires to use (1305). For example, the account information can include one or more of universal resource locator (URL) for the web-based application, a username, and a password.

[00309] The AOP framework then initializes the account (1306). The account initialization operation is described in more detail with reference to Figures 14-15 below.

[00310] After the account is initialized, the client computer system connects to the application server so that the application server can authenticate the user and verify AOP permissions (1307). After the user has been verified, the application server transfers information required to complete the installation of the web-based application on the client computer system (1308). For example, the application server can transfer files and data for the web-based application to the client computer system. The client computer system then installs up the web-based application (1309). Step 1309 is described in more detail below with reference to Figures 16-17.

AOP Initialize Account

[00311] Figure 14 presents a block diagram 1400 of an exemplary process of initializing an AOP account on a client computer system, according to embodiments of the present invention. Figure 14 includes AOP application 1401, vhost file 1405, AOP initialization file 1406, database 1408-1409, packages 1414, account 1404, EMS 1402, software stack 1410, operating system 1412, host file 1407, download 1408, connect to server 1403, and Network 120.

[00312] Figure 15 illustrates an exemplary process 1500 of initializing an AOP account on a client computer system, according to embodiments of the present invention. Note that Figure 15 corresponds to the block diagram in Figure 14. The AOP framework is installed on the client computer system (1501). After the AOP framework has been installed on the client computer system, a database for an account management system is installed (1502). In some embodiments, the account management system is the Etelos Management System (EMS). The term EMS is used below to describe a generic account management system. The AOP framework connects to the application server and authenticates user permissions (1503). If the user is authenticated, an account is created in the account management system (1504).

[00313] The local web server instance within the AOP framework is configured to enable navigation to the web-based application that is installed on the client computer system (1505). In some embodiments, a virtual hosts (vhost) file is configured to map a specified URL to a location for the web-based application on client computer system. For example, the

vhost file can map http://app1.com.aop to /web/app1. As noted above, the “.aop” suffix can be any suffix.

[00314] An AOP initialization file that includes additional configuration information is saved (1506). The specified URL is then associated with the client computer system (1507).

5 In some embodiments, an entry is added to a hosts file on the client computer system. The entry can be used to redirect a request for the specified URL to the client computer system. In other embodiments, an entry is added to a dynamic naming service (DNS) server on the client computer system. The entry can be used to redirect a request for the specified URL to the client computer system.

10 [00315] Packages that are required by the web-based application are then downloaded from a package repository onto the client computer system (1508). These packages can include packages such as EAS, Zend, fam fam fam, phpMyAdmin, etc (1509). The packages are then installed (1510).

[00316] The account is initialized and the system will move onto application
15 installation.

Web-based Application Installation

[00317] Figure 16 presents a block diagram 1600 of an exemplary process of installing a web-based application on a client computer system, according to embodiments of the
20 present invention. Figure 16 includes AOP framework 1601, software stack 1613, operating system 1614, files 1615, schema 1616, data 1617, rules 1618, Network 120, and connect to server 1620. AOP Framework 1601 includes application 1602, packages 1607, account 1608, EMS 1609, and databases 1610-1611. Application 1602 includes vhost 1603, web files 1604, file transfer 1605 (e.g., SVN, rsync, etc.), and database 1606.

25 [00318] Figure 17 illustrates an exemplary process 1700 of installing a web-based application on a client computer system, according to embodiments of the present invention. Note that Figure 17 corresponds to the block diagram in Figure 16. The client computer system downloads and installs the schema of the database for the web-based application from the application server (1701). The client computer system also downloads the files required
30 by the web-based application (1702). For example, the files can be downloaded using Subversion (SVN), which is a version control system, and/or rsync. The client computer

system then downloads synchronization rules and inserts the rules into the database for the account management system (1703). Next, the client computer system downloads application data as part of the initial synchronization process (1704). Step 1704 is described in more detail below with reference to Figures 18-19.

5

Initial Data Synchronization

[00319] Figure 18 presents a block diagram 1800 of an exemplary process of performing an initial data synchronization for a web-based application on a client computer system, according to embodiments of the present invention. Figure 18 includes AOP framework 1801, software stack 1813, operating system 1814, files 1817, data 1818, Network 120, and connect to server 1816. AOP Framework 1801 includes application 1802, packages 1807, account 1808, EMS 1809, synchronizer 1810, and databases 1811-1812. Application 1802 includes vhost 1803, web files 1804, file transfer 1805 (e.g., SVN, rsync, and other file transfer tools), and database 1806.

10

[00320] Figure 19 illustrates an exemplary process 1900 of performing an initial data synchronization for a web-based application on a client computer system, according to embodiments of the present invention. Note that Figure 19 corresponds to the block diagram in Figure 18. The initial data sync can be a large and time consuming process because all data records for the web-based application on the application server are synchronized with the web-based application on the client computer system. The synchronization rules processing can be more simple because a new installation typically does not have conflicting data between the application server and the client computer system.

15

20

[00321] The process begins when a synchronization engine within the AOP framework on the client computer system starts an initial data synchronization process (1901). The synchronizer then retrieves initial data synchronization rules from the database (1902). The synchronizer sends a request to the application server to synchronize with the application server (1903). This request can include user authentication information. Once the user is authenticated, the client computer system synchronizes files for the web-based application using a file transfer utility (1904). For example, the file transfer utility can include Subversion (SNV), rsync, etc. The application server then sends data records to the client computer system (1905). In some embodiments, the data records include all data records for the web-based application.

25

30

Using an the AOP Framework

[00322] Figure 20 presents a block diagram 2000 of an exemplary process of using the AOP framework on a client computer system, according to embodiments of the present invention. Figure 20 includes AOP framework 2001, software stack 2012, operating system 2013, Network 120, connect to server 2015, AOP launch interface 2016, launch browser with application running locally 2017. AOP Framework 2001 includes application 2002, packages 2007, account 2008, EMS 2009, and databases 2010-2011. Application 2002 includes vhost 2003, web files 2004, file transfer 2005 (e.g., SVN, rsync, and other file transfer tools), and database 2006.

[00323] Figure 21 illustrates an exemplary process 2100 of using the AOP framework on a client computer system, according to embodiments of the present invention. Note that Figure 21 corresponds to the block diagram in Figure 20. The AOP framework allows a user to use web-based applications on a client computer system regardless of whether the client computer system is connected to an application server that hosts the web-based application. In order to use web-based applications on the client computer system without a connection to the application server, the AOP framework must be running. A user on a client computer system executes the AOP framework on the client computer system (2101). The AOP framework loads the software stack (2102) and the EMS (2103). The AOP framework then loads a user interface (2104) and web-based applications that are available on the client computer system (2105). In some embodiments, the user interface can include a desktop interface (2109). A user can then select a web-based application to use. This selection is detected and causes the browser to navigate to the URL associated with the selected application (2106). The AOP framework causes the client computer system to be directed to a web server running on the client computer system (2107). For example, an entry in a hosts file or an entry in a DNS server on the client computer system can be used to direct the client computer system to the web server running on the client computer system. The web server on the client computer system listens for connections and responds to a connection request by serving the requested web-based application (2108). In some embodiments, the web server includes Apache HTTP server (2110).

[00324] A user that is using a web-based application on the application server can later lose a network connection to the application server. In some embodiments, if the client

computer system is running the AOP framework, the user can continue using the web-based application without the network connection to the application server. In these embodiments, the AOP framework can seamlessly continue providing the services required by the web-based application so that the user does not know that the network connection to the application server is lost. After the network connection to the application server is restored, any changes made on the client computer system and the application server can be synchronized with each other.

[00325] In some cases, a network connection may be slow or unreliable. Thus, in some embodiments, a user can use a web-based application from the client computer system regardless of whether a network connection to the application server exists or not. In these embodiments, the AOP framework periodically synchronizes changes made on the application server and the client computer system with each other. In doing so, a poor user experience that could result from a low-quality or intermittent network connection can be reduced or eliminated.

[00326] In some embodiments, a web browser is used to access the web-based application.

Client Computer System

[00327] When a user requests a web-based application by entering a URL into a browser, the client computer system determines how to access the web-based application. Figure 22 presents a block diagram 2200 of an exemplary process for a client computer system determining how to access a web-based application, according to embodiments of the present invention. Figure 22 includes browser 2201, host file 2202, web server 2203, vhost file 2204, application 2205, and packages 2209. Application 2005 includes vhost 2206, web files 2207, file transfer 2208 (e.g., SVN, rsync, and other file transfer tools), and database 2209.

[00328] Figure 23 illustrates an exemplary process 2300 for a client computer system determining how to access a web-based application, according to embodiments of the present invention. Figure 23 corresponds to the block diagram in Figure 22. The client computer system first determines an IP address associated with the URL (2301). In some embodiments, the URL includes a domain suffix “.aop”. The client computer system first checks a local hosts file to determine whether the URL is included in an entry in the hosts file

(2302). If the URL is not included in an entry in the hosts file, the client computer system checks one or more DNS servers to locate the IP address associated with the URL. If the URL is included in an entry in the hosts file, the client computer system retrieves the IP address for the URL (2303). The IP address for the web-based application is set to an address associated with the client computer system (e.g., 127.0.0.1) (2310). In some embodiments, the IP address associated with the URL for web-based applications is set to an address associated with the client computer system each time the AOP framework is started on the client computer system. In other embodiments, the IP address associated with the URL for web-based applications is set to an address associated with the client computer system when the web-based application is installed on the client computer system.

[00329] The web browser is then directed to the IP address associated with the client computer system on which the web server on the client computer system listens (2304). The local web server receives the request (2305) and compares the requested URL to a virtual host (vhost) configuration file (2306) to determine where the files for the web-based application are located on the client computer system (2307) (e.g., the local application web root). The web-based application and tools required to complete the request are loaded (2308). In some embodiments, the tools include the AOP framework, database, etc (2311). The web server then returns a web page with the results of the request (2309).

AOP Synchronization

[00330] Figure 24 presents a block diagram 2400 of an exemplary process for synchronizing a web-based application 2403 on a client computer system 2401 with a web-based application 2406 on an application server 2402, according to embodiments of the present invention. As illustrated in Figure 24, client computer system 2401 includes web-based application 2403, database 2404, and changes log 2405. Application server 2402 includes web-based application 2406, database 2407, and conflicts log 2408. Web-based application 2403 corresponds to web-based application 2406. Similarly, database 2404 corresponds to database 2407.

[00331] Figure 25 illustrates an exemplary process 2500 for synchronizing a web-based application on a client computer system with a web-based application on an application server, according to embodiments of the present invention. Figure 25 corresponds to the block diagram in Figure 24. A synchronization process (or a synchronizer module) on the

client computer system detects changes in a database 2404 for the client computer system 2401 (2501). In some embodiments, the synchronization process periodically polls database 2404 to determine whether changes have been made (2508). In other embodiments, the synchronization process listens for notification messages which are sent when database 2404 has been changed (2508).

[00332] The synchronization process then collects all the changes from database 2404 into a changes log 2405 (2502). The changes are sent to application server 2402 (2503). A synchronization process (or synchronization module) on the application server applies the changes received from the client computer system 2401 to database 2407 (2504). In some embodiments, the connection between client computer system 2401 and application server 2402 is kept alive until the synchronization process is completed.

[00333] If conflicts exist between the changes made on client computer system 2401 and application server 2402, the data on the server “wins” (2505). The conflicting data is resolved at application server 2402 and conflict resolution data is collected in a conflict resolution log 2408.

[00334] The synchronization process on application server 2402 then sends the conflict resolution log 2408 to client computer system 2401 (2506). The conflict resolution log is applied to database 2407 on client computer system 2401 (2507). Both client computer system 2401 and application server 2402 are synchronized at this point.

[00335] Figure 26 presents a block diagram 2600 of an exemplary process for synchronizing a web-based application on a client computer system 2601 with a web-based application on an application server 2602, according to embodiments of the present invention. Figure 26 includes client 2601 and server 2602. Client 2601 includes client in process 2606, client applications database 2607, client logs 2608, and client out process 2609. Application server 2602 includes server applications database 2603, server logs 2604, server out process 2605, and server in process 2610.

[00336] Figures 27A-27F illustrate an exemplary process 2700 for synchronizing a web-based application on a client computer system 2601 with a web-based application on an application server 2602, according to embodiments of the present invention. Figure 27 corresponds to the block diagram in Figure 26. A user makes changes to the web-based application on application server 2602 (2701). For example, the user can make changes to a

database for a web-based application by adding new records or modifying existing records in the database.

[00337] Database triggers watch for changes in database 2603 and note these changes in an internal log (2702). In some embodiments, if a rule type is “auto-increment,” then a flag is added to the log (2730). A synchronization process on application server 2602 then pulls data from the internal log and sends the data to an account log (2703). The data in the account log is then sent to an outbound log (2704). The outbound log is used to speed up the synchronization process because the outbound log can become large and slow to process. The synchronization process then waits for a user to initialize a synchronization operation with the application server (2705).

[00338] Once a synchronization operation is started, the data is sent to server out process 2605 for outbound processing (2706). In some embodiments, a transaction id called “sync block ID” is added to the data. The server out process performs several operations (2707), which can include, but are not limited to: applying filters, mapping data to users, sending user data to user tables, running group management rule checks, identifying administrative group memberships relationships for users, and/or mapping additional data to admin users based on a group admin of user A. The data is then sorted by users (2708). In some embodiments, the data is separated into separate “user out” tables, wherein each user has a “user out” table. In doing so, the speed of the synchronization can be optimized.

[00339] Client computer system 2601 receives the data and sends the data to client in process 2606 for processing (2709). Client in process 2606 performs a number of operations (2710), including, but not limited to, filtering Auto Increment Rules (flag), checking auto increment (AI) flagged column for conflict, marking data as “insert” if there are no conflicts, marking data in the sync log as “leave” (which are processed in the next session) if there are conflicts (see 2724), setting an auto-increment counter to max + 1, filtering data into inserts, updates and deletes, and/or sorting priorities (e.g., inserts before updates).

[00340] The synchronization process then inserts data that is marked as “insert” into client applications database 2607 for the web-based application (2711). The synchronization process then sends data that is marked as “update” or “delete” to a sync account log (2712). The updates and deletes in the sync account log are then processed by client applications database 2607 (2713). In some embodiments, when the synchronization process inserts data, the synchronization process disables triggers within a synchronization session to prevent

doubling back into an infinite loop (2713a). The web-based application on client computer system 2601 is now synchronized with the web-based application on application server 2602 (2714).

[00341] A user using web-based application on client computer system 2601 changes data in client applications database 2607 for the web-based application (2715). Database triggers watch for changes in client applications database 2607 and notes these changes in an internal log (2716). A synchronization process on client computer system 2601 then pulls data from the internal log and sends the data to an account log (2717). The data in the account log is then sent to an outbound log (2718). The outbound log is used to speed up the synchronization process because the outbound log can become large and slow to process. The synchronization process then waits for a user to initialize a synchronization operation with the application server (2719).

[00342] Once a synchronization operation is started, the data is sent to client out process 2609 for outbound processing (2720). In some embodiments, a transaction id called “sync block ID” is added to the data. Client out process 2609 performs several operations (2721), including, but not limited to, mapping data to sync account (user info is tied to sync acct) and/or pushing data to the sync account out log. Sync account out log is broadcast to application server 2601 (2722).

[00343] In some embodiments, every user has a table associated with the user (e.g., “in-table”), which is used for inbound processing. User in-tables (e.g., user A in-table, user B in-table) are pushed to server in process 2610 (2723). Server in process 2610 performs a number of operations (2724), including, but not limited to, comparing data – user out and user in – duplicate transactions, based on __eas_syncmap_id (added by trigger) and “sync_block_id” added by sync session, server wins – so server deletes conflicts, filtering inserts from updates and deletes, filtering auto increment flags, inserting records with no conflict, and/or inserting records with conflicts at the next unused record. Several examples of conflicting records include: __eas_sync_map_id = __eas_sync_map_id, Contact10 = contact10, and Email = john@etelos vs. email = john.smith@etelos.com.

[00344] Data records that are marked as “insert” are inserted (2725). Data records that are marked as “update” or “delete” are sent to a sync account log (2726). The updates and deletes in the sync account log are then processed by server applications database 2603 (2727). In some embodiments, when the synchronization process inserts data, the

synchronization process disables triggers this synchronization session to prevent doubling back into an infinite loop (2732). The web-based application on application server 2602 is now synchronized with the web-based application on client computer system 2601 (2728).

[00345] If the automatic incrementing rules detect a conflict between application server 2602 and client computer system 2601, another pass is taken through the process to update conflicted records (2729). During this process, records that were previously left out because of conflicts are synchronized.

[00346] In some embodiments, the synchronization process can be used to synchronize between different types of database management systems (DBMSs). In some embodiments, the synchronization process can be used to synchronize between the same type of DBMSs. In some embodiments, the changes are synchronized directly with a DBMS associated with the web-based application. In all of these embodiments, database drivers are used to perform the synchronization between the DBMSs. The database drivers can handle the translations of data between different types of DBMSs as well as determining what changes occurred since the last synchronization process.

[00347] In some embodiments, the synchronization process synchronizes directly with the file structures associated with the web-based applications. In these embodiments, a file synchronization utility such as SVN or rsync can be used to synchronize the file structures.

Auto Increment Conflict Resolution

[00348] Some web-based applications use automatically incrementing identifiers in a database when a new record is added to the database. In these web-based applications, conflicts can arise if multiple instances of the web-based application (e.g., on the application server, on multiple client computer systems, etc.) are all inserting new data records into a database for that instance of the web-based application. Thus, some embodiments provide a technique to resolve conflicts for web-based application that use auto-increment identifiers.

[00349] Figure 28 presents a block diagram 2800 of an exemplary process for resolving conflicts for web-based applications that use auto-incrementing identifiers, according to embodiments of the present invention. Figure 28 illustrates the state of data on an application server and a client computer system at several points in time (e.g., before

synchronization, after a first pass of synchronization, after a second pass of synchronization, and after synchronization is complete).

[00350] Figure 29 illustrates an exemplary process 2900 for resolving conflicts for web-based applications that use auto-incrementing identifiers, according to embodiments of the present invention. Figure 29 corresponds to the block diagram in Figure 28. As illustrated in Figure 29, both the application server and the client computer system have added new records into their respective databases for the web-based application. The application server and the client computer system also have records that are already synchronized. Note that in Figure 29, the identifiers increase from left to right. Thus, the new records on the application server and the client computer system include a number of records that use the same identifiers, resulting in conflicts.

[00351] At the start of a synchronization operation, the data that is already synchronized on both the application server and the client computer system (e.g., the AOP instance) is identified (2901). The application server then determines that new records have been added to the application server since the last synchronization with the client computer system (2902). The client computer system determines that new records have been added to the client computer system since the last synchronization with the application server (2903).

[00352] The conflict resolution process begins when the application server sends its new records to the client computer system (2904). Since there is a conflict between the identifiers used in the application server and the client computer system, the conflict resolution process waits for a second pass before inserting the new records received from the application server. On the second pass, the new records from the application server are treated as an update.

[00353] The client computer system sends its new records to the application server (2905). The application server assigns new identifiers for these records and inserts these records into the database for the web-based application (2906). The application server can assign the new identifiers for these records by using the automatically incrementing identifiers in the database. The application server then generates an update log that includes the changes for these records (2907). The application server sends the update log to the client computer system which then processes updates from that result from these records and from the new records received from the application server at step 2804 (2908). The records in both databases are now synchronized (2909).

Users and Groups

[00354] Figure 9 presents a block diagram illustrating exemplary group memberships for users for a given web-based application, according to embodiments of the present invention. As illustrated in Figure 9, user A is a member of group A. This membership can be managed through a memberships table. User B is a member of a group of administrators that are administrators of group A. This administrator membership can be managed through an admin table. In some embodiments, changes to user A's information are also added to a user out log table for user B. Thus, user B can see changes that user A has made. For example, a sales manager can view contacts for sales representatives that report to the sales manager.

[00355] In some embodiments, a group sync app group filter is used to indicate whether the web-based application supports group memberships. These rules can be setup in the account database for the web-based application. The filter can include information such as a data table (e.g., a contacts table), a membership table (e.g., the membership table), and a group table (e.g., groups). This can be a one time setup that a developer of a web-based application does to configure the web-based application to synchronize with the AOP framework. In some embodiments, the filter is setup when the developer ports the application to the AOP framework environment.

[00356] In some embodiments, the group filter sets flags in database triggers to send memberships data with a log entry so that the AOP framework knows what to use to filter in the server out process. In other words, the group filter is used to indicate which table includes user information, whether there is a membership table for linking users to groups, and the table that keeps track of groups.

Rules Setup

[00357] Figure 36 presents a block diagram 3600 illustrating an exemplary user interface 3600 for creating synchronization rules, according to embodiments of the present invention. User interface 3600 includes one or more of: new synchronization rule interface 3602, auto increment function 3622, and/or create new sync rule function 3624.

New synchronization rule interface 3602 includes one or more of: application interface 3604, mass create interface 3608, save function 3626, and/or close function 3628.

[00358] Application interface 3604 includes one or more of database function 3610, table function 3612, column function 3618, and/or primary key function 3620. Database function 3610 allows a user to select available databases. Table function 3612 allows a user to select tables within the available databases. Column function 3618 allows a user to select columns within the tables. Primary key function 3620 allows a user to select primary keys for the tables. Using these functions, a user can specify synchronization rules on one or more of the database level, the table level, the column level, and the primary key level.

[00359] Mass create interface 3608 includes a mass create function which allows a user to automatically create sync rules for the AOP framework at every match for a specified level. For example, a user can use the mass create function to create sync rules for all database tables.

[00360] Auto increment function 3622 allows a user to specify which columns on a given table use automatically incrementing identifiers. Create new sync rule 3624 creates the new synchronization rule specified by the user in user interface 3600. Save function 3626 saves the new rule whereas close function 3628 closes user interface 3600.

[00361] Figure 37 presents a block diagram 3700 illustrating an exemplary user interface 3700 for generating auto-incrementing identifiers, according to embodiments of the present invention. User interface 3700 includes auto increment definition interface 3702, which includes one or more of application interface 3704, relationships interface 3706, and/or close function 3724. Application interface 3704 includes one or more of application 1 3708, application 2 3710, and/or primary key 3712. Primary key 3712 includes one or more of a table selection function, a column selection function, save function 3720, and/or cancel function 3722. Relationships interface 3706 includes one or more of create primary key reference function 3714, contact ID function 3716, groups description function 3718, and an edit function.

[00362] Figure 38 presents a block diagram 3800 illustrating an exemplary process for creating synchronization rules, according to embodiments of the present invention. Figure 38 includes one or more of account admin interface 3802, select application and database 3804, select table 3806, select column 3808, auto increment flag 3810, rule create 3812, application 3814, and/or account 3822. Application 3814 include one or more of database 3816, trigger

3818, and/or sync map ID 3820. Account 3822 includes one or more of database 3824, vmap sync rule 3826, and AI flag 3828.

[00363] Figure 39 presents a flowchart of an exemplary process 3900 for creating synchronization rules, according to embodiments of the present invention. Figure 39 corresponds to the block diagram in Figure 38. From the account admin interface (3902), a user can perform the following operations: selecting an application (which inherently selects the associated databases) (3904), selecting a table (3906), selecting a column (which can involve selecting a primary key in a table row for proper synchronization at this specific level) (3908), and/or optionally setting an auto increment flag (e.g., if relevant to the database configuration) (3910). After the user performs one or more of the previous options, the user can select a create rules function (3912), which calls functions to perform one or more of steps 3914-3922. Alternatively, the user can refine the rules by using the account admin interface to specify more rules.

[00364] After selecting the create rules function, a rules generator creates a sync vmap rule in the account database (3914). The rules generator sets an auto increment flag if appropriate for rule type (e.g., if the database table uses an auto incrementing primary key column) (3916). The rules generator adds a column for a sync_map_id, which tracks sync transactions and ensures data integrity (3918). The rules generator creates a trigger in the database (3920). The synchronization rules are now setup at step 3922. Note that a user can also select a “mass create” at one or more levels (e.g., all tables, specific tables, specific columns, etc.) in the process from now on that can automatically create sync rules for the AOP framework at every match at that level.

Foreign Key Mapping

[00365] Foreign key mappings specify the relationships between a parent table and a child table. These mappings are used during the synchronization and conflict resolution processes. Figure 40 presents a block diagram 4000 of an exemplary foreign key mapping, according to embodiments of the present invention. The foreign key mapping includes parent 4002 and child 4020. Parent 4002 includes one or more of identifier 4004, application database identifier (e.g., App_DB_ID) 4006, application table 4008, application column 4010, and auto increment flag 4012. Child 4020 includes one or more of identifier 4022, parent identifier 4024, application database identifier (e.g., App_DB_ID) 4026, application

table 4028, application column 4030, and auto increment flag 4032. Parent identifier 4024 for child 4020 corresponds to identifier 4004 for parent 4002.

[00366] Figure 41 presents a flowchart of an exemplary process 4100 for creating a foreign key mapping, according to embodiments of the present invention. In an account, rules for auto increment and relationships with foreign keys are created (4102). A parent is created (e.g., table Contact, ID, Name, etc.) (4104). A child is created (e.g., Groups ID, Contact.ID, Description, etc.) (4106). The parent-child relationship is updated (auto-inc = true) (4108). Auto increment any conflicts that may arise and updated parent-child table relationships in the data structure (4110).

Summary of AOP

[00367] Figure 30 presents a flowchart of an exemplary process 3000 for providing access to a web-based application while offline, according to embodiments of the present invention. The process begins by providing on a computer system a local software stack configured to provide local web services for dynamic, web-based applications that are executed on the computer system when it is offline (3002). When the computer system is offline, the computer system executes a first dynamic, web-based application using the web services provided by the local software stack, such that functionality of the first dynamic, web-based application when the computer system is offline is substantially similar to functionality of the first dynamic, web-based application when the computer system is online (3004). In some embodiments, the first web-based application is a database-driven web-based application (3006). In some embodiments, the architecture of the first dynamic, web-based application is not modified to provide the functionality when the computer system is offline (3008).

[00368] In some embodiments, in response to detecting a network connection with an application server, the computer system synchronizes with the application server changes in information associated with the first dynamic, web-based application due to its offline execution (3010).

[00369] In some embodiments, a user of the computer system initiates execution of the first, dynamic web-based application when the computer system is offline by directing a web browser on the first computer system to a specified universal resource locator (URL) that is associated with the computer system instead of a remote application server (3012). The

specified URL can be associated with the computer system through a modified hosts file on the first computer system (3014). Moreover, the specified URL can be associated with the computer system through a modified hosts file on the first computer system (3016).

Furthermore, the specified URL can be associated with the first computer system through a
5 dynamic naming server (DNS) record on the first computer system (3018).

[00370] In some embodiments, the local software stack comprises: a web server responsive to browser-issued commands, a database management system, and a dynamic scripting language (3020). In some embodiments, the web server is Apache web server, the database management system is MySQL, and the dynamic scripting language is at least one
10 of PHP, Python, Perl, or Ruby (3022). In some embodiments, the first dynamic, web-based application communicates with the local software stack using TCP/IP (3024).

[00371] Figure 31 presents a flowchart of an exemplary process 3100 for synchronizing a web-based application on a client computer system with a web-based application on an application server, according to embodiments of the present invention. For
15 a web-based application that is not designed to be used while a first computer system on which the web-based application executes does not have a network connection to an application server (3102), changes are made to the web-based application while the first computer system does not have a network connection to the application server (3104). The changes made to the web-based application while the first computer system is disconnected
20 from the application server are tracked (3105). When the network connection between the first computer system and the application server is reestablished (3106), the changes are synchronized for the web-based application made on the first computer system with the web-based application on the application server (3107). The changes are synchronized for the web-based application on the application server with the web-based application on the
25 first computer system (3108).

[00372] In some embodiments, the date and time when the changes to the web-based application were made are tracked (3109). In some embodiments, the changes are synchronized when a network connection between the first computer system and the application server is reestablished (3110). In some embodiments, the changes are
30 synchronized between different types of database management systems (DBMS) (3112). In some embodiments, the changes are synchronized between the same types of database management system (DBMS) (3114). In some embodiments, the changes are synchronized

directly with a database management system (DBMS) associated with the web-based application (3116). In some embodiments, the changes are synchronized directly with file structures associated with the web-based application (3118). In some embodiments, for a newly-installed instance of the web-based application on the first computer system, the method further comprises performing a complete synchronization of all records in the web-based application on the application server with the web-based application on the first computer system (3120). In some embodiments, the changes include one or more of: changes to data in the web-based application; changes to data structures in the web-based application; and/or changes to files for the web-based application (3122).

[00373] Figure 32 presents a flowchart of an exemplary process 3200 for providing synchronizing a web-based application on a client computer system with a web-based application on an application server, according to embodiments of the present invention. For a web-based application that is being used on a first computer system while the first computer system is disconnected from an application server, in response to detecting a network connection to the application server (3202), changes made for the web-based application on the first computer system are synchronized with the web-based application on the application server (3204). The changes made for the web-based application on the application server are also synchronized with the web-based application on the first computer system (3206).

[00374] In some embodiments, for a newly-installed instance of the web-based application on the first computer system, a synchronization of records is performed in the web-based application on the application server with the web-based application on the first computer system (3208).

[00375] In some embodiments, the changes include one or more of: changes to data in the web-based application; changes to data structures in the web-based application; and/or changes to files for the web-based application (3210).

[00376] Figure 33 presents a flowchart of an exemplary process 3300 for resolving conflicts between a web-based application on a client computer system and a web-based application on an application server, according to embodiments of the present invention. On a first computer system that is disconnected from an application server, a web-based application that is configured to interact over a network connection with the application server to provide specified functionality is used (3302). When the network connection is reestablished, changes made to the web-based application while the first computer system

was disconnected from the application server are synchronized (3304). If a conflict between the first computer system and the application server exists, the conflicts are resolved so that both the first computer system and the application server are synchronized with each other (3306).

5 **[00377]** In some embodiments, a conflict exists if the first computer system includes a first set of records that are not included on the application server, and wherein resolving the conflict so that both the first computer system and the application server are synchronized with each other involves: sending the first set of records to the application server; receiving
10 new identifiers for the first set of records, wherein the new identifiers are assigned by the application server when the first set of records are added to the application server; and updating all references to the old identifiers for the first set of records with the new identifiers for the first set of records (3308).

[00378] In some embodiments, a conflict exists if the application server includes a second set of records that are not included on the first computer system, and wherein
15 resolving the conflict so that both the first computer system and the application server are synchronized with each other involves: receiving the second set of records from the application server; determining whether identifiers for the second set of records are already being used by the first computer system; and/or if the identifiers for the second set of records are not being used by the first computer system, inserting the second set of records (3310).

20 **[00379]** In some embodiments, if the identifiers for the second set of records are already being used by the first computer system, the following operations are performed: determining a third set of records that include identifiers that conflict with the identifiers for the second set of records; sending the third set of records to the application server; receiving
25 new identifiers for the third set of records, wherein the new identifiers are assigned by the application server when the third set of records are added to the application server; updating all references to the old identifiers for the third set of records with the new identifiers for the third set of records; and inserting the second set of records (3312).

[00380] In some embodiments, if a conflict between the first computer system and the application server does not exist, records are updated on the first computer system so that the
30 first computer system is synchronized with the application server (3314).

[00381] Figure 34 presents a flowchart of an exemplary process 3400 for resolving conflicts between a web-based application on a client computer system and a web-based

application on an application server that use automatically incrementing identifiers for database records, according to embodiments of the present invention. A first set of records are created with a corresponding first set of identifiers in a first database (3402). The first database is synchronized with a second database (3404). If the corresponding first set of
5 identifiers already exists in the second database, a new set of identifiers is received for the first set of records from the second database, wherein the new set of identifiers is assigned to the first set of records when the first set of records is added to the second database, and all references to the corresponding first set of identifiers is updated for the first set of records with the new set of identifiers for the first set of records (3406).

10 **[00382]** In some embodiments, if the corresponding first set of identifiers does not exist in the second database, the first set of records is inserted into the second database using the corresponding first set of identifiers (3408).

[00383] In some embodiments, if the second database includes a second set of records that does not exist in the first database, the following operations are performed: receiving the
15 second set of records from the second database; sending the first set of records to the second database; receiving new identifiers for the first set of records, wherein the new identifiers are assigned by the second database when the first set of records are added to the second database; updating all references to the old identifiers for the first set of records with the new identifiers for the first set of records; and inserting the second set of records (3410).

20 **[00384]** In some embodiments, the first database and the second database are used for a web-based application that requires a network connection with an application server to provide specified functionality (3412). In some embodiments, the web-based application was not designed to be used without a network connection to the application server (3414). In some embodiments, the first database is on a client computer system (3416). In some
25 embodiments, the second database is on an application server (3418).

[00385] Figure 35 presents a flowchart of an exemplary process 3500 for providing access to a web-based application while offline, according to embodiments of the present invention. A dynamic, web-based application configured to interact over a network connection with an application server to provide desired functionality is used (3502). The
30 network connection is disconnected from the application server (3504). The web-based application continues to be used while still providing the desired functionality (3506).

[00386] In some embodiments, the dynamic, web-based application is a database-driven web-based application. In some embodiments, the architecture of the dynamic, web-based application is not modified to provide functionality when the computer system is offline. In some embodiments, in response to detecting a network connection with an application server, changes in information associated with the dynamic, web-based application due to its offline execution are synchronized with the application server. In some embodiments, a user of the computer system initiates execution of the dynamic, web-based application when the computer system is offline by directing a browser on the first computer system to a specified universal resource locator (URL) that is associated with the computer system instead of a remote application server.

[00387] Marketplace and Licensing

[00388] Figure 44 is a high-level block diagram 4400 of a software marketplace, illustrating deployment of applications to a user account. A marketplace server 4410 makes available for distribution a plurality of software applications 4412, 4414, 4416. A licensing engine 4420 associates licensing terms with the software applications. In some embodiments, the licensing engine 4420 is part of the marketplace 4410. In some embodiments, the marketplace server is executed by a marketplace module 6122 (Figure 61).

[00389] One or more software applications (e.g., application 4412 and application 4414) are deployed to an account 4432 hosted at a hosting infrastructure server 4430. An account is a data structure associated with a user, maintained on a server (e.g., the hosting infrastructure server 4430), where the account data structure includes information regarding which software applications are licensed to the user. In some embodiments the account may include license files, keys, etc. associated with one or more software applications. In some embodiments, applications licensed by a user are configurable to cross-synchronize, i.e. to synchronize data between the applications associated with the user. In some embodiments, the user associated with a user account could be an individual or an organization.

[00390] In some embodiments, the hosting infrastructure server is executed by a hosting module 6154 (Figure 61). In some embodiments, the deployment is across a network 4402, e.g., where the hosting infrastructure server is located separately from the marketplace server 4410. Deployment means the software application is associated with the user account, such that the user associated with the user account is permitted to access the software

application. In some embodiments, one or more files relating to the software application are provided to the user account as part of the deployment.

[00391] In some embodiments, the hosting infrastructure server 4430 and the marketplace server 4410 are logical servers on one physical system, for example application sever 6100 (Figure 61). The deployment is performed in accordance with the licensing terms, provided by licensing engine 4420, in some embodiments executed by licensing module 6126 (Figure 61). In some embodiments, the licensing engine controls the deployment of and access to software applications, and enforces license terms provided by the software vendor. In some embodiments, the licensing module provides the software code or executables to operate the licensing engine. The hosting infrastructure server 4430 includes an account management system (EMS) 4438 for managing the accounts 4432, executed by user accounts module 6156 (Figure 61). A user account allows a user (person licensed to use software applications provided by the marketplace) to authenticate to system services. The user account also provides a user with the opportunity to be authorized to access deployed software applications. Users may need to identify themselves for the purposes of accounting, security, logging and resource management. In some embodiments, a user identifies him or herself using an account identifier and a username, and in some embodiments the user also has a password.

[00392] A software stack 4440 and operating system 4442 provide the software architecture to support the account framework and application execution. In some embodiments, the network 4402 is the Internet or an intranet or local connection between the marketplace 4410 and server 4340.

[00393] One or more client systems 4450, 4452, 4454 access the hosting infrastructure server 4430 to execute the applications. In some embodiments, the client systems correspond to application client 6200. In some embodiments, one or more of applications 4434, 4436 are executed at the server 4430, and results are displayed to the client through a client application or browser, executed by client/browser display module 6215. In some embodiments, applications are run in a local mode (e.g., AOP mode as described earlier) where a client hosts and executes applications 4462 and 4464 associated with an account 4460, corresponding to applications 4434 and 4436 hosted at hosting infrastructure server 4430. An account management system (EMS) 4470 manages the accounts at the client, in some embodiments executed by user account access module 6272. A client-side software stack

4472 and operating system 4474 provide the software architecture to support the account framework and application execution on the client.

[00394] Figure 45 illustrates a block diagram 4500 of a software marketplace illustrating management of licenses in a multi-tenancy environment. A multi-tenancy environment is one which allows multiple customers or users to access a shared data model. In some embodiments, the software marketplace includes a licensing engine 4510, executed by licensing module 6126, to control licensing for hosted infrastructure servers (e.g., 4520, 4530). A hosted infrastructure server is one that hosts user accounts, and in some embodiments software applications associated with those user accounts. In some embodiments, the hosted infrastructure server may exist as a logical server on the same physical hardware as the marketplace server. In some embodiments, the hosted infrastructure server may be physically separate from the marketplace server. A multi-tenant license management engine 4512 generates and manages licenses (e.g., 4514, 4516) for deployment and execution of software applications on hosted infrastructure servers. The licenses are generated in response to a selection by a software vendor from options provided by the marketplace application. The license terms are associated with a software application. Software applications are deployed to the hosting infrastructure servers in accordance with the licenses (4514, 4516).

[00395] In some embodiments, hosted infrastructure servers 4520, 4530 are distributed tenants in a multi-tenant licensing system. In a distributed tenant architecture, each user runs its applications in a distributed-computing environment. In some embodiments, the hosted infrastructure servers 4520, 4530 are logical servers on one or more physical servers. The hosted servers include an account (4522, 4532) into which applications are installed, and account management system (EMS) (4524, 4534) for managing the accounts, and databases (4526, 4536) for supporting the hosted infrastructure.

[00396] In some embodiments, the databases 4536 can include relational database management systems such as PostgreSQL, MySQL, and optionally Oracle, Microsoft SQL, or any other Open Database Connectivity (ODBC) interfacing database may be used.

[00397] Figure 46 illustrates a server system 4600 for a web application marketplace and hosting infrastructure. One or more servers 4602 hosts the marketplace application. The one or more servers include one or more processors 4604. The processors include memory

4610 storing instructions for implementing the marketplace application. In some embodiments, the circled numbers of Figure 46 correspond to steps shown in the flow diagram of Figure 47.

[00398] A marketplace application 4612 (executed in some embodiments by marketplace module 6122, Figure 61) receives from a vendor a software application for distribution. The vendor can include a developer, coder, broker, or licensee with right to resell software. In some embodiments, the marketplace receives the application via an Internet (including in some embodiments a world wide web) connection 4618. License terms 4622 are associated with the software application by a licensing module 6126 (Figure 61).

The associating may include storing a license selection in a license manager, with a link/pointer to the account storing the respective software application. The software application is made available for distribution through the marketplace application, executed in some embodiments by the distribution module 6120 (Figure 61). Making the software application available for distribution can include displaying (via a listing manager) a summary of the software application, price, license terms, and a platform(s) or framework (s) on which the software application operates, in a catalog of software applications hosted by the marketplace application. The distribution may be initiated through the marketplace app, but the distribution and deployment process do not have to take place through the marketplace application.

[00399] Throughout this application, displaying means that the server sends data for display to a client associated with the user. Prior to display, the data for display may be formatted by the server prior to sending to the client associated with the user, or may be formatted by the client after receiving the data, or may include a combination of these operations.

[00400] In some embodiments, the marketplace application presents a description of the license terms to a user, via the user's web application or client 4616. This presenting may be performed by the listing module 6122 (Figure 61). The description can be a full or brief description (e.g., open source, proprietary, source code, object file, repackaging, etc.) or an icon indicating a license type associated with the software application.

[00401] In response to a user selection of one or more software applications from the marketplace 4612, the software application is deployed by deployer 4624 to one or more user

accounts on one or more hosting servers 4630. In some embodiments, a deployment confirmation is sent back to the marketplace 4612. The hosting servers 4630 may be provided by the marketplace operator, or may be provided by the user. The hosting servers 4630 may be physical servers, or virtual servers. The deployment is performed in accordance
5 with the license terms, executed by application deployment module 6150 (Figure 61). In some embodiments, the deploying is associated with a billing operation 4620. In some embodiments, the one or more user accounts are accounts associated with the user who made the request to deploy the software application. Deploying includes at least one of the following: downloading the software application to the one or more user accounts, and/or
10 enabling a flag associated with the software application in the one or more user accounts, enabling a license for the software application to the one or more user accounts, where the application itself stays stored at the marketplace server.

[00402] In some embodiments, a software vendor provides a license key to the marketplace application, whereby the marketplace application or a related application
15 deploys/provisions the license key to a user account, and makes an API call to the software vendor at or after the time that the software application is licensed by the user. An application programming interface (API) is a set of declarations of the functions (or procedures) that an operating system, library or service provides to support requests made by computer programs. In some embodiments, the purpose of the API call is to inform the
20 software vendor that a particular user or account has licensed the software application, which version is licensed, etc. In some embodiments, this information is beneficial to the software vendor when responding to technical support requests from the user.

[00403] In some embodiments, a software vendor may provide a license key associated with the software application to the marketplace application, and the license key is deployed
25 to an account associated with a licensee. In some embodiments, the deployed license key enables the account to access or execute the software application. In some embodiments, the purpose of the license key is to enforce licensing terms.

[00404] In some embodiments, a software vendor may provide a license key update to the marketplace application, and the license key update is deployed to an account associated
30 with a licensee, where the license key update modifies an already installed license key. In some embodiments, this modification includes extending or shortening a period of time during which the account may access the software application. In some embodiments, this

modification includes adding or removing features that the account may access in the software application. When the application is deployed to an account associated with a user, in some embodiments confirmation is provided to the user, and then the user can access and run the application deployed at hosting infrastructure 4630.

5 **[00405]** In some embodiments, the hosting infrastructure servers are physically separate from the marketplace servers hosting the marketplace application. In some embodiments, the deploying of a software application is performed in response to a user request to deploy the software application. In some embodiments, the marketplace application determines a user account type, and based on the user account type, prepares to
10 deploy (via the deployer) a software application to the user account.

[00406] In some embodiments, distribution to a user through the marketplace application 4612 is performed through a website (e.g., 4618) associated with the marketplace application. In some embodiments, distribution to a user through the marketplace application includes distribution through a client associated with the marketplace application 4612, where
15 the client accesses the marketplace through a secure connection.

[00407] In some embodiments, the software application is packaged, executed in some embodiments by packager module 6136, Figure 61, for distribution via the marketplace application 4612 hosted by the one or more servers 4602. The packaging can include one or more of parsing code for the software application, performing quality control for the software
20 application, testing the software, packaging the software application for distribution, and packaging an update to the software application or packaging an update to the licensing of an application. In some embodiments, the packaged software application is stored in an application repository 4614.

[00408] In some embodiments, software applications are presented, described and
25 made available at the marketplace application by a listing manager, executed by listing module 6124, Figure 61. The listing manager performs one or more of storing listings (sale details for software applications), storing technical articles and information, and storing support resources (help, customer support, etc.). The listing manager may also provide one or more of a search function, a list of most popular software applications, a package deal for
30 licensing a suite of software applications, a customer review, and other related functions. In some embodiments, the listing manager includes a store listing for licensing the software

application. In some embodiments, the marketplace application stores, or stores links to, technical support for the software application. This technical support includes application notes, instructional articles, datasheets, frequently asked questions (FAQs), etc.

[00409] FIG. 47 illustrates a flowchart 4700 of operations for selecting and deploying a software application in a user account. A user navigates (1) the web (or accesses via a client), and visits the application marketplace (4702). The user selects an application from the marketplace (2), and checks out (4704). The application is provisioned (3) (or deployed) on demand to the hosting infrastructure (4706). In some embodiments, the hosting infrastructure installs the application and sends confirmation (4) back to the marketplace (4708). In some embodiments, a link is sent (5) from the marketplace to the user's application (4710). The user accesses the application (6) and the application validates the license with the licensing server (hosted at the hosting infrastructure) via their client or browser (4712).

[00410] Figure 48 illustrates a system 4800 including a marketplace server 4802 and a hosting infrastructure 4810. The hosting infrastructure includes one or more hosting servers 4820, 4830, 4840. The one or more hosting servers include one or more deployed software applications 4822, 4824, 4826.

[00411] A user accesses a marketplace 4806 via a web connection or client 4804. The user selects and checks out a software application, as described. The software application is hosted at the one or more hosting servers, e.g., deployed software application 4822, at server 4820. The user accesses the deployed software through a client or web connection 4808, as described. Each server may host one or more software applications associated with one or more user accounts. The servers may be physical servers or logical servers operating in one or more groups on a physical server. User accounts are managed by user accounts module 6156 (Figure 61).

[00412] In some embodiments, deploying an application (in some embodiments executed by or under the control of a deployer module 6152, Figure 61) includes downloading the software application to the one or more user accounts. User account access is managed by user account access module 6172 (Figure 61). In some embodiments, deploying an application includes activating a flag associated with the software application in the one or more user accounts that indicates when the application is available to a user

associated with the account. In some embodiments, the flag enables the software application for the user. In some embodiments, deploying an application includes activating a license for the software application in the one or more user accounts, which activation renders the software application available to a user associated with the corresponding account. In some
5 embodiments, deploying an application includes providing the software application for hosting by a user on hosting servers associated with the user, the hosting executed by user hosting module 6174 (Figure 61).

[00413] Figure 49 illustrates a server system 4900 for selecting license options in a software distribution marketplace. This figure shows operations at the server, under control
10 of the licensing manager, where a vendor may associate one or more license terms with its software application. The server system includes one or more servers 4902, hosting a marketplace application. The server system is configured to receive from a vendor 4910 a software application for distribution 4920. The server system includes a licensing manager 4930 and is configured to perform under control of the license manager 4930 a number of
15 operations 4920-4928 related to software vendor selection of license terms associated with the software application.

[00414] License manager 4930 is configured to generate license terms in response to a selection by the vendor from options provided by the marketplace application. The license terms are associated 4922 with the software application. In some embodiments, the license
20 terms are stored in a licensing manager.

[00415] The server is configured to make the software application available for distribution (4926) through the marketplace application, in accordance with the license terms.

[00416] In some embodiments, the software application is deployed 4928 to one or more user accounts on one or more hosting servers, in accordance with the license terms.
25 Deploying an application in accordance with the license terms means that conditions specified by the software vendor in the license terms (e.g., source code is to be hidden from the licensee) are enforced when deploying and when granting access to the software application. In some embodiments, a deployer enables or disables functions or options within the deployed software application, where this enabling or disabling is determined by the
30 license terms provided by the software vendor.

[00417] In some embodiments, the deploying is in response to a payment associated with the one or more user accounts.

[00418] In some embodiments, the license terms are selected by a vendor 4910 of the software application. In some embodiments, the license terms are selected from a grid 4912 or set of options provided by the marketplace application. In some embodiments, the license terms include at least one of one or more open source licenses 4932, closed licenses 4934, object file licenses 4936, source code licenses 4938, and repacking licenses 4940. An open source license 4932 may provide typical license terms for open source software, e.g., a GNU license or BSD license, where the application source code is available under terms that allow for modification and redistribution without having to pay the original author. A closed license 4934 (also known as a proprietary license) is one that does not allow access to source code. An object file license 4936 is one that allows a user to execute the software application, but not to access any source code or object files. The source code license 4938 is one that allows a user to access the source code of the licensed software application, including making changes to the source code, but only allows distribution of binary code. In some embodiments, a repacking license 4940 determines whether a user of the software application is permitted to repackage and redistribute the software application, in accordance with repacking terms 4952. In some embodiments, the repacking license has an associated royalty. In some embodiments, the royalty is one selected from a wholesale royalty, a retail royalty, or a flat fee. In some embodiments, the vendor may upload its own term sheet(s) for its software applications, and this term sheet is provided with the software application.

[00419] In some embodiments, the license terms may address (limit or permit) redeployment of software to a user within the same organization as the licensing user. This might be the case where a system administrator licenses a software application, customizes it for her organization, and then redistributes the customized software to users within her organization. In some embodiments, the license terms may allow redeployment to the whole marketplace. This might be the case where a developer can license a first software application, write more code to add value to it, and resell the enhanced software application to the marketplace. In this situation, a licensee could be required to remit a royalty to the vendor of the first software application or to the marketplace administrator, or to some combination thereof.

[00420] In some embodiments, the license manager 4930 determines user permissions for installation, activation, and access to features of applications. In some embodiments, the license options provided by the marketplace application include an option to use license terms 4914 supplied by the vendor. These vendor-supplied terms 4914 may be considered as

‘hard’ terms (i.e., a fixed termsheet provided by the software vendor, rather than terms generated by the marketplace application in response to selections of terms by the user), since they are not generated by the license manager in response to a selection from the standard license options.

5 **[00421]** In some embodiments, the license manager 4930 includes a transaction reporter 4950 (executed in some embodiments by transactions report module 6132, Figure 61) to display licensing events for a respective software application made available for distribution through the marketplace application. In some embodiments, this transaction reporter shows transactions going through the marketplace for a respective software
10 application, including installation status, user check-in status, payment status etc. In some embodiments, the transaction reporter displays the licensing events to a respective vendor associated with the respective software application.

15 **[00422]** In some embodiments, a price associated with the software application is stored at a licensing manager separately from the software application. In some embodiments, the price is dynamically adjusted by the licensing manager in response to a selection by the software vendor, or by the marketplace administrator, or by usage metrics triggers embedded in the software application itself. In some embodiments, an adjustment to the price is provided to the marketplace, which updates pricing information to charge the adjusted price for licensed software applications.

20 **[00423]** In some embodiments, at least one selected from the group consisting of access duration, features, and price, is dynamically adjusted by the licensing manager in response to a selection by the software vendor, or by the marketplace administrator. The access duration includes a trial period for the software application, or other promotional offers. The features includes access to features (including new or premium features) within
25 the software application. The price is the price associated with the licensed software application of features thereof. Vendors can also choose, bandwidth allowed, disk space, users, record table max limits and/or if ads are displayed in the application, among other things.

30 **[00424]** Figure 50 illustrates a server system 5000 for performing dynamic billing in a software distribution marketplace. This figure shows operations at the server, under control of the billing manager, where a vendor may associate one or more prices with its software application. The server system includes a billing manager 5030, and is configured to perform

under control of the billing manager 5030 a number of operations (including 5010-5016, indicated on Figure 50 as circled numbers 1 to 4) related to user selection of license terms associated with the software application.

[00425] The server system includes one or more servers 5002, hosting a marketplace application 4806 (Figure 48) and a billing manager 5030. The server system is configured to receive from a vendor a software application for distribution 5010, and store it in an application repository, executed by application repository module 6134 (Figure 61). The system is configured to associate a licensing price (stored in the billing manager 5030) with the software application 5012, where the license price is controlled by billing manager 5030, executed by billing module 6130 (Figure 61). The licensing management is executed by licensing module 6126 (Figure 61). The vendor of the software application provides a price per license 5020, which may be stored by the server at the billing manager 5030. This price per license may be changed dynamically (i.e. changed 'on the fly') by the vendor 5020 or by an administrator 5022 of the software distribution marketplace. The billing manager 5030 updates the prices associated with each license type (if more than one) for a software application.

[00426] In some embodiments, a first price 5042 is associated with an open source license 5032, a second price 5044 is associated with a closed license 5034, a third price 5046 is associated with an object file 5036, a fourth price 5048 is associated with a source code license 5038, and a fifth price 5050 is associated with a repacking license 5040. In some embodiments, there are more or fewer prices associated with one or more of the license types. In some embodiments, a subscription or 'flat fee' price 5052 is supported by the billing manager. In some embodiments, the server 5002 receives a payment 5014 for licensing the software application. In some embodiments, the billing manager supports payment processing (for paid licenses). In some embodiments, the billing manager supports registration (where no payment required for a license) 5054.

[00427] During the licensing process, (e.g., before a user pays to license a software application), license terms associated with the software application, or a link to those license terms, are displayed 5018 to the user. In some embodiments, the license terms are provided to the user before executing the software application. In some embodiments, the license terms are provided as click-wrap license terms. The server is configured to deploy 5016 the licensed software application to an account associated with the user.

[00428] Figure 51 illustrates a server system 5100 for managing an application repository and deployer in a software distribution marketplace. This figure shows operations at the server, under control of the application repository and/or deployer, where software applications provided by a software vendor may be deployed to a user account associated with a user. Server 5102 is configured to receive a software application from a software vendor for distribution 5110. An application packager 5120 (executed in some embodiments by a packager module 6136, Figure 61) packages 5112 the software application for distribution. An application deployer 5150 deploys the software application to a user account 5116. The packaged application is stored 5114 in an application repository. In some embodiments, the application deployer serves as a gateway between the user account and the application repository 5140.

[00429] In some embodiments, the server 5102 hosts the deployed software application for one or more user accounts. The hosting includes executing the deployed software on behalf of the associated users. In some embodiments, the software application is hosted at a server separate from the server 5102.

[00430] In some embodiments, the application packager 5120 prepares updates 5122 to previously deployed software applications 5214. In some embodiments, this preparation included one or more of testing the software application for functionality and/or performance to ensure quality, providing technical support material (e.g., a manual, product application notes, etc.), and specifying a language and framework associated with the software application.

[00431] In some embodiments, the previously deployed function 5124 is required for the update 5122 to operate. Such an update is sometimes referred to as a 'mini application' and may include plugins or add-on features that can be bought from the store. In some embodiments, the 'mini application' may include updates to the applications.

[00432] In some embodiments, the application packager 5120 prepares updates to standalone distributions 5126. A standalone distribution is one that does not require a previously deployed function to operate. In some embodiments, the standalone distribution includes a software application and one or more patches to the software application 5128.

[00433] In some embodiments, the application packager 5120 prepares applications or updates for distribution 5130 by performing a quality assurance test 5132. This test may be automatic or may involve some human input. The application packager 5120 includes

technical support material 5134 with the packaged application (e.g., a readme file). The application packager 5120 may include instructions to specify what language, framework, database, supporting packages and libraries, or operating system the software application is compatible with 5136.

5 **[00434]** In some embodiments, the application repository 5140 stores a plurality of packaged applications 5142, 5144, to 5146. These packaged applications are those applications that are ready for distribution and deployment to a user account.

10 **[00435]** In some embodiments, the application deployer 5150 retrieves a packaged application from the repository 5140 if it does not exist in a local cache and then deploys it to a user account (5152, 5116). The deployer can deploy an application by a push method 5154 (where a user does not have any choice about whether or not to receive the deployed update or application), by a pull method 5156 (where the user chooses to retrieve the deployed update), or by a hybrid method 5158 (which may be a combination of the push and pull methods, or another method). The application deployer performs deployment in accordance
15 with license terms 5160.

[00436] In some embodiments, the application deployer 5150 generates a new user account compatible with the software application and deploys the software application to the new user account. The account type can include one or more of operating system, language, or framework associated with the user account.

20 **[00437]** Figure 52 illustrates a server system 5200 for performing syndicated deployment of a software application across a network, executed by syndicated deployment module 6160 (Figure 61). Server 5202 is configured to receive a request from syndicated server 5230, across a network 120, as described. In some embodiments, the request is received at marketplace 5210, including at deployer 5214. The server identifies one or more
25 user accounts associated with the request. The server (including deployer 5214) checks licensing terms and permissions 5212, to determine if the syndicated server, including syndicated server software distribution 5232 and syndicated server user accounts 5234, have permission to receive the deployed software application, and if the syndicated deployment is in accordance with the license terms.

30 **[00438]** In some embodiments, deploying an application includes providing through an application deployer, across a network to one or more user accounts, a software application stored at the application repository. The deployer accesses repository 5216 and retrieves an

application ready to deploy 5218. This application is deployed to syndicated server user account 5234, in accordance with the license terms.

[00439] In some embodiments, deploying an application includes presenting the software application for deployment to a user or to a syndicated server associated with the user. In these embodiments, the syndicated server deploys the presented software application into a user account associated with the syndicated server. A syndicated server is a server that is logically separate from the marketplace server. In some embodiments, the syndicated server is also physically separate from the marketplace server and is controlled by someone other than the controller of the marketplace and hosting servers. In some embodiments, the syndicated server primarily provides services other than the licensing of software applications in a marketplace. In some embodiments, the syndicated server offers to its users the ability to license software applications from the marketplace by providing its users with an interface (in some embodiments, implemented using an API) by which to browse and/or license software applications from the marketplace. In some embodiments, a licensed application is deployed to an account associated with the user, hosted by or under the control of the syndicated server.

[00440] In some embodiments, the software application is made available for distribution through the syndicated server. In this embodiment, the syndicated server acts as a marketplace front for the software application, and the application is licensed by a user through the syndicated server.

[00441] In some embodiments, deploying an application includes selecting one of a plurality of software applications, compatible with the one or more user accounts, from the application repository. The selected software application is deployed to an account associated with the user, hosted by the hosting infrastructure server.

[00442] In some embodiments, the application repository stores software applications in any one or more of a number of states, including at least one state of a ready to deploy state, an undergoing quality assurance state (undergoing testing 5220), a ready to submit for quality assurance state, and an unfinished state. The ready to deploy state means that the software application is fully tested and is ready for licensing to a user. Only software applications in the ready to deploy state can be listed on the marketplace and licensed to a user, and thus only software applications in this state can generate revenue. The undergoing quality assurance state means that the software application is being tested and verified for

quality, and is not yet ready for licensing to a user. The ready to submit for quality assurance state means that the software application code is written, but testing and quality assurance has not yet been performed on it. The unfinished state means that the software application is still in development.

5 **[00443]** Figure 53 illustrates an embodiment of a server system 5300 for packaging software applications for distribution to a user account. The system 5300 includes a packager 5320, executed in some embodiments by a packager module 6136 (Figure 61).

10 **[00444]** The packager receives as input software application files 5310, and prepares them for deployment. In some embodiments, the packager also receives data files, local
install scripts 5312 and auxiliary software packages 5314 associated with the software application files 5310. The packager prepares one or more of the files 5310, 5312 and 5314, and generates a packaged application 5330, which is stored in application repository 5342 associated with marketplace application 5340. Upon licensing by one or more users 1 to N, the packaged application 5344 is deployed to one or more accounts 1 to N associated with the
15 users 1 to N, on hosting application servers 5350-1 to 5350-N. This deploying is performed by the application deployer.

20 **[00445]** Figure 54 illustrates an alternate embodiment of a server system 5400 for packaging updates to software applications for distribution to a user account. The system 5400 includes a packager update tool 5420, which in some embodiments is executed by the packager module 6136 (Figure 61).

25 **[00446]** The packager receives as input updated software application files 5410, updated data files, local install scripts 5412, and updated auxiliary software packages 5414 prepares them for deployment, and generates an updated (patched) application 5430. In some embodiments, the updated application (patch) 5446 is stored in the application repository 5442 associated with marketplace application 5440, for deployment to one or more servers 5450-1 to 5450-N. The updated application (patch) is deployed to servers that already have the original packaged application (e.g., version 1) 5444 installed.

30 **[00447]** Figure 55 illustrates an alternate embodiment of a server system 5500 for packaging updates together with software applications for distribution to a user account. System 5500 includes packager 5520, executed by packager module 6136 (Figure 61). Packager 5520 packages both software applications and updates files for the software applications separately or together for distribution.

[00448] The packager receives as input software application files 5510, updated data files 5512, and updated auxiliary software packages 5514; prepares them for deployment either together or separately; and generates a packaged new version of an application 5530. In some embodiments, a packaged new version of an application is stored in the application repository 5542 associated with the marketplace application 5440, for deployment to one or more servers 5550-1 to 5550-N. The packaged new version of the application 5549 is deployed by the application deployer to one or more user accounts associated with hosting infrastructure servers. In some embodiments, the application repository 5542 stores the original packaged application 5544, and updates 5546 to 5548 to the packaged application. In some embodiments, when the packaged new version of the application 5549 is placed in the application repository 5542, the older version 5544, and updates 5546 to 5548, may be removed from the application repository.

[00449] Figure 56 illustrates a method 5600 of licensing a software application from a marketplace, for deployment to a user account associated with a user, hosted at a hosting infrastructure server. In this scenario, the licensing process includes receiving a selection from the user for a software application from the marketplace and adding the application to a cart associated with the user, receiving the user's request to check out the cart and thereby the user's request to license the software application, providing the user with terms of use associated with the software application, and receiving the user's agreement to the terms of use (in some embodiments, receiving a payment from the user). In some embodiments, upon licensing of the software application by the user, the software application is deployed to an account associated with the user.

[00450] At one or more servers hosting a marketplace application 5620, one or more software applications 5622 to 5624 are made available for distribution through the marketplace application. The marketplace server receives a user request to license the software application. The marketplace server provides the software application for deployment 5648 to one or more user accounts, hosted on the one or more servers, or hosted at a hosting infrastructure server, as described regarding Figure 44.

[00451] In some embodiments, providing the software application for deployment includes providing the software application across a network for deployment at one or more servers associated with the user, hosting the one or more user accounts.

[00452] In some embodiments, the user request includes a selection by the user to add the software application to a cart 5640 associated with the user, and to check out 5642 the cart.

[00453] In some embodiments, prior to deploying or prior to receiving the user request, the marketplace server presents a description of license terms of use associated with the software application to the user 5644.

[00454] In some embodiments, the license terms are specified by a vendor 5614 associated with the software application. The license terms may be specified before the software application is presented for deployment, or the license terms may be specified following deployment. In some embodiments, the license terms may evolve after deployment. In some embodiments, specifying the license terms includes selecting (by the vendor) the license terms 5614 from a plurality of options provided by a licensing engine 5630 associated with the one or more servers hosting a marketplace application. In some embodiments, the licenses are stored 5632, 5634 at the licensing engine 5630. In some embodiments, the licensing engine 5630 validates that the user request to license the software application complies with the license terms.

[00455] In some embodiments, the request to license the software application includes a payment 5646 (e.g., credit card or debit card payment, or bank transfer). In some embodiments, the request may be associated with a prospective future payment (trial period). The prospective future payment may include receiving a credit card number, but not billing the credit card until some time in the future. e.g., at the expiration of the trial period.

[00456] In some embodiments, prior to presenting an application for deployment, the vendor builds 5610 and packages 5612 the software application 5616. In some embodiments, the software application is a web-based software application, executed at one or more servers.

[00457] Figure 57 illustrates a system 5700 for providing security for software applications, deployed to a user account. One or more servers 5702 hosts a marketplace application 5710. The marketplace application includes a billing manager 5712. The marketplace application is accessed by a web connection or client 5714. The deployed application is accessed by a user's web application 5720, having an associated user identifier 5722.

[00458] The server 5702 includes a security application 5730, executed by deployment security module 6158 (Figure 61). In some embodiments, the security application 5730

verifies that only a licensed set of features and/or number of copies of a deployed software application is running at a time 5732. These requirements may be provided by the software vendor via the licensing manager (described in Figure 49).

[00459] In some embodiments, the security application 5730 includes a payment verifier 5736 to compare a user identifier 5722 associated with the one or more user accounts and an application id (e.g. 5750, 5760) associated with the deployed application against the billing manager 5712 to verify that a valid payment has been recorded. In some embodiments, this verification is performed prior to executing the deployed application. The security application verifies that both the user and application are associated with a valid transaction before allowing a user to execute the software application.

[00460] In some embodiments, the security application 5730 includes a permission verifier 5738 to verify that the one or more user accounts have permission to execute the deployed application, and in the event of a verification failure, to warn the user. The user account permissions are executed by permissions module 6128 (Figure 61). In some embodiments, the security application 5730 includes periodic verifier 5740 to periodically (e.g., daily, weekly, monthly etc.) repeat one or more of the verification steps described. Following one or more verification failures, a shutdown application 5742 warns the user 5744, and/or prevents 5746 (in the event of multiple failures) the one or more user accounts from executing the deployed application. The plurality of failures includes factors such as the number of failures, the period of time over which failures occur, IP addresses from which attempted logons to the application are made, etc. In some embodiments, the shutdown application 5742 prevents access by the user to data stored at the one or more servers, upon determining that the one or more user accounts has been disabled. The user access is executed by access module 6170 (Figure 61). The account enablement verifier 5734 verifies that a user account has permission to access the deployed software applications (e.g., 5752, 5762). In some embodiments, the user permission/access is made available to the application vendor through the licensing API to enable custom messaging and actions by the vendor. Making available means that a list of privileges or permissions associated with a the user is provided to the software application vendor so the vendor can understand what terms the user has agreed to when licensing the software. This can assist the software vendor in resolving a user's problems. In some embodiments, when a user is denied access, the user is redirected to the vendors marketplace storefront listing to resolve the issue.

[00461] In some embodiments, the security application 5730 includes instance verifier 5732 to check for multiple instances of the software application being simultaneously executed by the one or more user accounts. In some embodiments, for some software applications (e.g., AOP) multiple logons to a user account are permitted, but only one logon
5 may use (execute) a software application associated with the user account at a time. As an example, a user can stay logged into her account at work, but go home and access the account from her home PC. Since she is using her work PC to execute applications while she is at home, she may execute a software application while logged into her account from her home PC.

10 [00462] If the security application 5730 determines that the user account is authorized to access a deployed software application, a confirmation 5772 is provided to a hosted application execution engine 5770 for running deployed software. This engine executes one or more of the hosted deployed software applications 5752, 5762, hosted on one or more servers 5754, 5764.

15 [00463] Figure 58 illustrates a system 5800 for allowing multiple logons by a user 5860 to a software application, deployed to a user account. In an example, a user may require multiple logons if he logged onto the software application from his work computer, and left work and traveled home forgetting to log out. If the user wants to access the software application from his home computer (while still logged on from work, having forgotten to log
20 out) then the user may log in from home (thus creating a simultaneous logon situation). In another example, a user may be logged on to the software application from his desk computer, and while in a meeting at another location, may want to access the software application using a handheld device (thus creating a simultaneous logon situation). One or more servers 5805 hosts a licensing engine 5180. The licensing engine includes a
25 simultaneous logon manager 5811, executed by user account access module 6172 (Figure 61). In some embodiments, the one or more servers 5805 also host a first 5814 and a second 5186 deployed software applications, where a user has licenses to the first and second deployed applications.

[00464] A first client system 5820 (e.g., an office PC) executes a browser 5822 in
30 which a first instance 5824 of first software application 5814 is open and logged in. The first client system also includes an account management system (EMS) 5828, a software stack 5830, and an operating system (OS) stack 5832. A second client system 5840 (e.g., a home

PC) executes a browser 5842, in which a second instance 5844 of first software application 5814 is open and logged in. The second client system also includes an account management system (EMS) 5848, a software stack 5850, and an OS stack 5852.

[00465] When a client 5820 attempts to access a deployed application 5814 (associated with a user account), the licensing engine 5810 and logon manager 5811 determine if the client 5820 has permission to access the application, and also determine if another client is accessing that application, associated with the user account. In some embodiments, the logon manager 5811 will only permit (depending on the license terms specified by the software vendor via the licensing manager, described in Figure 49) a single logon to an application associated with a user account at a time. In some embodiments, the logon manager 5811 will permit a plurality of logons (e.g., by clients 5820 and 5840) to an application associated with a user account, if the licensing terms so permit. In some embodiments, the login manager 5811 will permit multiple logons, as long as only one client system is operating the application at a time. Operating means providing input to the application and/or receiving information from the application.

[00466] Figure 59 illustrates an exemplary user access control interface 5900 for controlling user access to a software application, deployed to a user account. This figure shows operations at the marketplace server, under control of the licensing manager, where a system administrator (or in some embodiments, a vendor) may view and/or change license terms associated with the software application and associated with users of those applications. This is useful because it gives the administrator a 'birds eye' view of licensing permissions and activity within the system.

[00467] A plurality of applications 5902-1 to 5932-N are shown on the left hand column. Associated with each application is one or more users 5920-1 to 5920-N, and 5934-1. For each user, a status indicator 5904 determines if the user is active (i.e., using) that particular application. An administrator status 5906 determines if the user has administrator privileges. An AOP status 5908 indicates if a user has permission to run an application offline. A source code access status 5910 determines whether a user is allowed to view or edit source code associated with an application. By changing the status buttons or indicators, an administrator or vendor can change the permissions associated with an application or with a user. Other control 5912 determines if a user can access other features, in accordance with a particular embodiment.

[00468] Figure 60 illustrates an exemplary user access control interface 6000 for controlling user access to a software application, deployed to a user account. One or more software applications 6002 through 6004 are shown. For each software application, one or more users 6010-1, 6010-2 through 6010-N may access the software application, where this access is controlled by this interface.

[00469] Figure 61 is a block diagram illustrating a server system 6100 configured to host a software marketplace and licensing system. The system 6100 generally includes one or more processing units (CPU's) 6102, one or more network or other communications interfaces 6104, memory 6110, and one or more communication buses 6108 for interconnecting these components. The communication buses 6108 may include circuitry (sometimes called a chipset) that interconnects and controls communications between system components. The system 6100 may optionally include a user interface, for instance a display 6106 and input device (e.g., a keyboard and/or mouse) 6105. Memory 6110 may include high speed random access memory such as DRAM, SRAM, DDR RAM or other random access solid state memory devices; and may include non-volatile memory, such as one or more magnetic disk storage devices, optical disk storage devices, flash memory devices, or other non-volatile solid state storage devices. Memory 6110 may include mass storage that is remotely located from the central processing unit(s) 6102.

[00470] Memory 6110, or alternately the non-volatile memory device(s) within memory 6110, comprises a computer readable storage medium. In some embodiments, memory 6110 stores the following programs, modules and data structures, or a subset thereof:

- an operating system 6111 that includes procedures for handling various basic system services and for performing hardware dependent tasks;
- a network communication module 6112 that is used for connecting the server 6100 to other computers via the one or more communication network interfaces 6104 (wired or wireless) and one or more communication networks, such as the Internet, other wide area networks, local area networks, metropolitan area networks, and so on;
- a distribution module 6120 that is used to market and distribute software applications to users of the software marketplace and licensing system;
- a marketplace module 6122 that provides an electronic marketplace where software vendors can display, and users can select, software applications for licensing;

- a listing module 6124 that is used to generate and display software application products within an electronic marketplace;
- a licensing module 6126 that controls the distribution of software applications, with license terms selected or provided by a software vendor;
- 5 • In some embodiments, a permissions module 6128 that enforces license terms during deployment;
- a billing module 6130 that receives and processes payment from a user associated with a licensed software application;
- a transaction report module 6132 for reporting on distribution and licensing activity
10 taking place within the marketplace;
- an application repository module 6134 for storing and managing software applications ready for distribution;
- a packaging module 6136 for preparing software applications and updates for distribution;
- 15 • a deployment module 6150, for managing deployment of software applications to a user account, and managing hosting of the software application;
- a deployer module 6152 (in some embodiments part of the deployment module 6150) for deploying a software application to a hosted user account, or to a user-controlled server for hosting the account;
- 20 • a hosting module 6154 for hosting the software application in a hosted user account and for executing the software application;
- a user account module 6156 for managing user accounts, permissions and access;
- a security module 6158 for managing deployment and ensuring that a software application is deployed in accordance with license terms, to authorized users, at
25 authorized servers;
- a syndicated deployment module 6160 for performing syndicated deployment (in some embodiments, across a network) in response to a request from a syndicated server;
- an access module 6170 for controlling access to deployed applications and to the
30 marketplace;

- a user account access module 6172 (in some embodiments, part of user account module 6156 or access module 6170) for controlling user access to hosted applications;
- a user hosting module 6174 (in some embodiments, part of a hosting module 6154) for hosting the deployed software application at a server controlled by the user;
- a marketplace database management module 6180 for managing a marketplace database;
- a licensing database management module 6182 for managing a licensing database;
- a deployment database management module 6184 for managing a deployment database;
- a billing database management module 6186 for managing a billing database; and
- a user account database management module 6188 for managing a user account database.

[00471] In some embodiments, the database management modules 6180, 6182, 6184, 6186, 6188 can be separate or combined into one or more groups.

[00472] Each of the above identified elements may be stored in one or more of the previously mentioned memory devices, and corresponds to a set of instructions for performing a function described above. The above identified modules or programs (i.e., sets of instructions) need not be implemented as separate software programs, procedures or modules, and thus various subsets of these modules may be combined or otherwise re-arranged in various embodiments. In some embodiments, memory 6110 may store a subset of the modules and data structures identified above. Furthermore, memory 6110 stores additional modules and data structures not described above.

[00473] Although Figure 61 shows a server system for software application distribution, Figure 61 is intended more as functional description of the various features that may be present in a set of servers than as a structural schematic of the embodiments described herein. In practice, and as recognized by those of ordinary skill in the art, items shown separately could be combined and some items could be separated. For example, some items shown separately in Figure 61 could be implemented on single servers or single items could be implemented by one or more servers. The actual number of servers used to implement a

software application distribution system and the way in which features are allocated among them will vary from one implementation to another, and may depend in part on the amount of data traffic that the system must handle during peak usage periods as well as during average usage periods.

5 **[00474]** Figure 62 illustrates a block diagram of a client system 6200 for accessing a software marketplace and licensing system, hosted at one or more servers. The client system 6200 generally includes one or more processing units (CPU's) 6202, one or more network or other communications interfaces 6204, memory 6210, and one or more communication buses 6208 for interconnecting these components. The communication buses 6208 may include
10 circuitry (sometimes called a chipset) that interconnects and controls communications between system components. The system 6200 also includes a user interface, for instance a display 6206 and input device (e.g., a keyboard and mouse) 6205. Memory 6210 may include high speed random access memory such as DRAM, SRAM, DDR RAM or other random access solid state memory devices; and may include non-volatile memory, such as
15 one or more magnetic disk storage devices, optical disk storage devices, flash memory devices, or other non-volatile solid state storage devices. Memory 6210 may include mass storage that is remotely located from the central processing unit(s) 6202.

[00475] Memory 6210, or alternately the non-volatile memory device(s) within memory 6210, comprises a computer readable storage medium. In some embodiments,
20 memory 6210 stores the following programs, modules and data structures, or a subset thereof:

- an operating system 6211 that includes procedures for handling various basic system services and for performing hardware dependent tasks;
- a network communication module 6212 that is used for connecting the client 6200 to a server hosting software applications, via the one or more communication network
25 interfaces 6104 (wired or wireless) and one or more communication networks, such as the Internet, other wide area networks, local area networks, metropolitan area networks, and so on;
- a receive and process user input module 6214 for receiving input from input device(s) 6205, such as a keyboard and mouse, and interpreting that input to control a
30 client/browser;

- a client/browser module 6215 (including browser engine module 6216) for providing a window by which a user accesses the hosted software application, and through which the user (or in some embodiments an administrator) manages the hosted software application;
- 5 • accessible through the client/browser module 6215: a distribution module 6220, application deployment module 6250 and access module 6270, each with constituent modules, and database management modules 6280 through 6288, corresponding to modules in Figure 61 as described earlier;
- accessible through the client/browser module 6215: a cart checkout/module 6290 for
10 allowing a user to place a software application in the cart, and checking out the cart to license the software application;
- accessible through the client/browser module 6215: a process payment module 6292 for processing a payment associated with the licensed software application;
- accessible through the client/browser module 6215: a select license module 6294 for
15 selecting a license associated with a software application, prior to licensing the application;
- accessible through the client/browser module 6215: a submit license terms module 6296 for a vendor to select license terms associated with the software application, or to submit custom license terms provided by the vendor (e.g., a text file); and
- 20 • auxiliary services module(s) 6298 for providing other features or services associated with the client.

[00476] Flowcharts for Marketplace and Licensing

[00477] Figure 80 is a flowchart representing a server method 8000 for distributing a software application, according to certain embodiments of the invention. Server method
25 8000 may be governed by instructions that are stored in a computer readable storage medium and that are executed by one or more processors of one or more servers. Each of the operations shown in Figure 80 may correspond to instructions stored in a computer memory or computer readable storage medium. The computer readable storage medium may include a magnetic or optical disk storage device, solid state storage devices such as Flash memory, or
30 other non-volatile memory device or devices. The computer readable instructions stored on

the computer readable storage medium are in source code, assembly language code, object code, or other instruction format that is interpreted by one or more processors.

[00478] At one or more servers hosting a marketplace application (e.g., Figure 129, marketplace storefront; Figure 155, user marketplace), a software application is received from a vendor for distribution (8002). License terms (e.g., Figure 133, license terms) are associated with the software application (8004). In some embodiments, the license terms are received from the vendor of the software application, and are stored in a database (e.g., by licensing database management module 6182, Figure 61) with a pointer to the vendor's software application. Associated means that when the software application is selected for licensing by a user, the vendor license terms corresponding to that vendor's software application are used during licensing and distribution of the software application. In some embodiments, a vendor may have different sets of license terms per software application.

[00479] The software application is made available for distribution (e.g., Figures 129, 155) through the marketplace application (8006). The software application is deployed (e.g., Figure 175, "My Installations") to one or more user accounts on one or more hosting servers, in accordance with the license terms.

[00480] In some embodiments, the one or more hosting servers (e.g., Figure 148, developer hosting accounts; Figure 149 hosting sub-domains) are physically separate from the one or more servers hosting the marketplace application (8026).

[00481] In some embodiments, the deploying is performed after a user request (e.g., Figures 166, 167, 168, download and install) to deploy the software application (8101). In some embodiments, the deploying includes downloading the software to one or more user accounts (8012). In some embodiments, deploying includes activating a flag associated with the software application in the one or more user accounts (8014). In some embodiments, the flag enables the software application for the user account (8016). In some embodiments, the deploying includes activating a license for the software application in the one or more user accounts (8018).

[00482] In some embodiments, the deploying includes providing the software application for hosting by a user on hosting servers (e.g., Figure 148 developer hosting account, Figure 149 sub-domains associated with an account) associated with the user (8020).

[00483] In some embodiments, distribution to a user through the marketplace application includes through a website (e.g., Figure 155, marketplace storefront) associated

with the marketplace application (8022). In some embodiments, distribution to a user through the marketplace application includes distribution through a client associated with the marketplace application (8024).

[00484] In some embodiments, the deploying is performed (e.g., Figure 175, “My Installations”) by an application deployer (8028), such as the application deployer described in Figure 51, and/or the application deployer module 6152 (Figure 61).

[00485] In some embodiments, the deployed software application is hosted for the one or more user accounts (8030).

[00486] Figure 81 is a flowchart 8100 continuing the server method 8000 of Figure 180.

[00487] In some embodiments, a description of the license terms (e.g., Figure 158, accept terms) is presented to a user (Figure 81, 8114).

[00488] In some embodiments, the software application is packaged (e.g., Figure 154, 173, My Packaged Applications) for distribution via the marketplace application hosted by the one or more servers (8102). In some embodiments, a packaged software application is stored in an application repository (8104). In some embodiments, the packaging includes preparing an update (e.g., Figure 103, update 10304) to a previously deployed software application, where the update requires the previously deployed software application to function (8106). This scenario might be where a patch update (set of fixes to a previously deployed application) is deployed. A patch generally requires the earlier deployed application to function, since the patch only changes a small portion of the deployed application, and thus the patch is not a standalone application.

[00489] In some embodiments, the packaging includes preparing a standalone distribution (e.g., Figure 103, update 10302) for a software application (8110). In some embodiments, the standalone distribution includes a software application and one or more patches to the application (8112). In some embodiments, the update is deployed to the one or more user accounts selected from the group consisting of (e.g., Figure 142, Product Upgrade) a push method, a subscription (pull) method, and a hybrid method, in accordance with the license terms (8108).

[00490] In some embodiments, the making available (e.g., Figure 130, Product/Service Basic Information) is performed by a listing manager (8116), executed by listing module

6124 (Figure 61). In some embodiments, the listing manager includes a store listing (e.g., Figures 130-153, store listing details/setup, Figure 155 storefront) for licensing the software application (8118). In some embodiments, the marketplace application includes technical support (e.g., Figures 174, 177-181, support and knowledge base) for the software application (8122). By making available, the listing manager performs one or more of storing and presenting software application listings (sale details for software applications), storing technical articles and information, and storing support resources (help, customer support, etc.). The listing manager may also provide one or more of a search function, a list of most popular software applications, a package deal for licensing a suite of software applications, a customer review, and other related functions.

[00491] In some embodiments, making the software application available for distribution can be done by determining a user account type, and based on the user account type, preparing to deploy a software application to the user account, or generating a new user account (e.g., Figure 169, developer accounts) compatible with the software application and preparing to deploy the software application to the new user account (8120).

[00492] Figure 82 is a flowchart representing a server method 8200 for licensing a software application, according to certain embodiments of the invention. The server method 8200 may be executed by licensing module 6126 (Figure 61) and as described earlier with regard to Figure 49.

[00493] At one or more servers, hosting a marketplace application, a software application is received from a vendor for distribution (8202). License terms (e.g., Figure 132, license selection) are generated (8204) in response to a selection by the vendor from options provided by the marketplace application. The license terms are associated with the software application (8206). The software application is made available for distribution through the marketplace application (e.g. Figure 155, storefront), in accordance with the license terms (8208).

[00494] In some embodiments, the software application is deployed (e.g., Figure 175, My Installations) to one or more user accounts on one or more hosting servers, in accordance with the license terms (8210). For example, the license terms can influence an application in the following way. Deploying an application in accordance with the license terms means that conditions specified by the software vendor in the license terms (e.g., source code is to be hidden from the licensee) are enforced when deploying and when granting access to the

software application. In some embodiments, a deployer enables or disables functions or options within the deployed software application, where this enabling or disabling is determined by the license terms provided by the software vendor.

[00495] In some embodiments, the deploying is in response to a payment (e.g., Figure 159, billing history) associated with the one or more user accounts (8212).

[00496] In some embodiments, license terms include at least one of (e.g., Figure 132, part four) an open source license, a closed license, a source code license, an executable (object file) license, and a repacking license (8214). In some embodiments, the repacking license (e.g., Figure 132, part three) determines whether a user of the software application is permitted to repackage and redistribute the software application (8216). In some embodiments, the repacking license has an associated royalty (8218). In some embodiments, the associated royalty is one selected from the group consisting of a wholesale royalty, a retail royalty, and a flat fee (8220). In some embodiments, the license manager determines user permissions (e.g., Figure 132, part four) for installation, activation, and access to features of applications (8222). In some embodiments, the options provided by the marketplace application includes an option to use license terms supplied by the vendor (8224).

[00497] In some embodiments, licensing events (e.g., Figure 153, transaction report) for a respective software application are made available for distribution through the marketplace application (8226). In some embodiments, the license events are displayed (e.g., Figure 151, store report) to a respective vendor associated with the respective software application (8228). In some embodiments, the license terms are stored in a licensing manager (8230).

[00498] Figure 83 is a flowchart 8300 continuing the server method 8200 of Figure 82.

[00499] In some embodiments, a price (e.g., Figure 132, part two; Figure 144 pricing grid) associated with the software application is stored at a licensing manager separately from the software application (8302). In some embodiments, the price is dynamically adjusted by the licensing manager in response to a selection by the software vendor (8304). In some embodiments the vendor selects a price by entering a price (e.g., price in US dollars, Euros, Yen, or any other currency) in the licensing manager 4930 (Figure 49) or billing manger 5030 (Figure 50). In some embodiments, the vendor selects a price by choosing a price from a set

of options provided by the marketplace. In some embodiments, at least one selected from the group consisting of access duration, features, and price, is dynamically adjusted by the licensing manager in response to a selection by the software vendor (8306).

[00500] In some embodiments, the one or more user accounts are stored separately
5 from the marketplace application (8308).

[00501] In some embodiments, a payment (e.g., Figure 159, billing history) associated with the software application is processed (8310). In some embodiments, prior to processing a payment, a promotional code is received from the user, and the payment is processed based on the promotional code (8312). In some embodiments, a record of the processed payment is
10 stored in a billing record (8314).

[00502] In some embodiments, prior to executing the deployed application, a user identifier associated with the one or more user accounts and an application id associated with the deployed application are compared against a billing manager to verify that a valid payment has been recorded (8316). In some embodiments, the system verifies that the one or
15 more user accounts has permission to execute the deployed application, and in the event of a verification failure, warns the user (8318). In some embodiments, the verifying is performed periodically, and following a plurality of verification failures, the one or more user accounts are prevented from executing the deployed application (8320). In some embodiments, the verifying includes checking for multiple instances of the software application being
20 simultaneously executed by the one or more user accounts (8322). In some embodiments, the system prevents access by the user to data stored at the one or more servers, upon determining that the one or more user accounts has been disabled (8324).

[00503] Figure 84 is a flowchart representing a server method 8400 for syndicated deployment of a software application, according to certain embodiments of the invention.
25 Server method 8400 may be governed by instructions, as described earlier.

[00504] At one or more marketplace servers hosting a marketplace application, in response to a request from a syndicated server to distribute a software application from the marketplace, one or more user accounts associated with the request are identified (8402). The system verifies that the one or more user accounts has permission to use the software
30 application (8404). The software application is deployed to the one or more user accounts, in accordance with license terms associated with the software application.

[00505] In some embodiments, the software application is presented for deployment to a user (8408). In some embodiments, the software application is made available for distribution through the syndicated server. In some embodiments, deploying includes providing through an application deployer, across a network to the one or more user accounts, a software application stored at the application repository (8412).

[00506] In some embodiments, deploying includes selecting one of a plurality of software applications, compatible with the one or more user accounts, from the application repository (8414). In some embodiments, the application repository stores software applications in a plurality of states, including at least one selected from the group consisting of a ready to deploy state, an undergoing quality assurance state, a ready to submit for quality assurance state, and an unfinished state (8416).

[00507] Figure 85 is a flowchart representing a server method 8500 for licensing and receiving payment for a software application, according to certain embodiments of the invention. Server method 8500 may be governed by instructions, as described earlier.

[00508] At one or more servers hosting a marketplace application (e.g., Figures 129, 155), a software application is made available for distribution through the marketplace application (8502). A user request to license the software application is received (8504). The software application is provided for deployment to one or more user accounts, hosted on one or more servers (8506). In some embodiments, the software application is provided across a network for deployment at one or more servers associated with the user, hosting the one or more user accounts (8508).

[00509] In some embodiments, a selection by the user is processed that includes adding the software application to a cart (e.g., Figure 160, 172, cart and checkout) associated with the user, and checking out the cart (8510). In some embodiments, prior to deploying, a description of license terms associated with the software application is presented (e.g., Figure 158, license) to the user (8512). In some embodiments, the license terms are specified by a vendor (e.g., Figure 132) associated with the software application (8514). In some embodiments, the license terms are selected from a plurality of options provided by a licensing engine associated with the one or more servers hosting a marketplace application (8516). In some embodiments, the system validates that the request to license the software application complies with the license terms (8518). In some embodiments, the request to license the software application includes a payment (e.g., Figure 170, payment info) selected

from the group consisting of a cash payment, a credit payment, and a prospective future payment (8520).

[00510] In some embodiments, the software application is a web-based software application, executed at one or more servers (8522).

5

Application Framework

[00511] Some embodiments provide an application framework that facilitates synchronizing data between applications. The application framework can include data structures and a set of rules for synchronizing data between two applications. For example, consider a customer relationship management (CRM) application and an email application, both of which include tasks for a user. The fields (or columns, etc.) to be synchronized between the CRM application and the email application are first mapped to application framework data structure. For example, the task identifier fields for these two applications can be mapped to the “identifier” field in the application framework data structure. During a synchronization operation between the CRM application and the email application, the CRM application can first synchronize itself with the application framework data structure. The email application can then synchronize itself with the application framework data structure. Synchronization rules can be used to map the fields from the CRM application to the application framework data structure and to map the fields of the application framework data structure to fields for the email application. Note that that the synchronization process can be performed in both directions (e.g., from the CRM application to the email application and from the email application to the CRM application. Also note that in these embodiments, the code and/or the architecture of the applications do not need to be modified in order to synchronize data between the applications. In some embodiments, the data is synchronized between the databases for the applications without using explicit commands executed by the applications. In other words, the synchronization of data occurs at the database level, and not the application level.

[00512] Figure 63 is a block diagram illustrating an exemplary application framework 6300, according to some embodiments. In some embodiments, the application framework 6300 is located on one or more servers. For example, the application framework 6300 can be located on a server such as the servers 4420-4840 in Figure 44, any one of the servers 4520-4530 in Figure 45, the hosting infrastructure 4630 and/or the server 4602 in Figure 46, any

one of the servers 4802 and/or 4830 in Figure 48, the server 4902 in Figure 49, the server 5002 in Figure 50, the server 5102 in Figure 51, any one of the server 5202 and/or the syndicated server 5230 in Figure 52, any one of the servers 5350, 5450, and 5550 in Figures 53-55, respectively, and/or any one of the servers 5702, 5754, and 5764 in Figure 57. The one or more servers can include an end user server maintained by an end user (e.g., a customer-provisioned server, etc.) and a hosted server maintained by a service provider (e.g., a software vendor, a web hosting company, etc.). In some embodiments, the application framework 6300 is located on one or more client computer systems. For example, the application framework 6300 can be located on a client computer system such as the client systems 4450-4454 in Figure 44, and/or any one of the client systems 5820 and 5840 in Figure 58.

[00513] In some embodiments, the application framework 6300 includes one or more of an account management system (EMS) 6302 and/or a software stack 6340. Recall that “EMS” is used to describe a generic account management system. The EMS 6302 can create and manage accounts within the application framework 6300, provide user authentication, and/or synchronize applications within an account and/or between account frameworks. The EMS 6302 can include one or more of a database 6304 and/or an application synchronization module 6306. The database 6304 is described in more detail with reference to Figures 64 and 72-73 below and the application synchronization module 6306 is described in more detail below with reference to Figures 65-79 below. Note that although the discussion below refers to a single database, multiple databases can be used. The software stack 6340 can include one or more of: an email module 6342, a web server module 6344, an authoring module 6346, a browser module 6348, and/or auxiliary modules 6350. The email module 6342 can include any electronic mail program. For example, the email module 6342 can include email servers (e.g., Sendmail, Postfix, and qmail, etc.), mail filtering programs (e.g., procmail, SpamAssassin, etc.), client email programs (e.g., elm, pine, Microsoft Outlook, etc.), etc. The web server module 6344 can include any application that can respond to client requests for pages, data, graphics, videos, documents, etc, hosted on a server coupled to clients and/or other servers via a network. For example, the web server module 6344 can include Apache HTTP Server, Microsoft Internet Information Server, etc. The authoring module 6346 can include any application that allows a developer or a user to generate web-based applications. For example, the authoring module 6346 can include a text editor, a word processor, a web development environment (e.g., Microsoft Expression Web, Adobe Dreamweaver, etc.), etc.

The browser module 6348 can include any application with a rendering engine that can access data and/or services on a local and/or a remote computer system and render the results so that a user can view the data and/or interact with the services. In some embodiments, browser module 6348 is a web browser. The auxiliary modules 6350 can include other applications such as a Lightweight Data Access Protocol (LDAP) interface module and/or database, an Internet Message Access Protocol (IMAP) module, a Post Office Protocol (POP) module, a Dovecot module (e.g., an IMAP and POP server), a web server (e.g., Apache HTTP server, Microsoft IIS Server, etc.), one or more database management systems (e.g., MySQL, PostgreSQL, etc.), logging applications (e.g., Syslog-ng, etc.), authentication services (e.g., saslauthd, etc.). Note that although Figure 63 illustrates a single module for each type of service, the software stack 6340 can include any number of modules for a given type of service. For example, two different email modules can be included within software stack 6340.

[00514] In some embodiments, the application framework 6300 includes one or more accounts. For example, as illustrated in Figure 63, the application framework 6300 includes an account 6310. The account 6310 can include one or more of: an account directory 6312, and a database 6314. The account directory 6312 can include a directory structure configured to store one or more applications. The account database 6314 can include meta data for the account 6310. The meta data can include descriptions for the software packages 6316, users associated with the account 6310, and/or default synchronization hooks/rules (e.g., for user management, licensing, etc.).

[00515] In some embodiments, the account 6310 includes software packages 6316. For example, the software packages 6316 can include utilities (e.g., phpmyadmin, phppgadmin, websvn, EDK, etc.), tools, and/or shared libraries (e.g., Zend, PHPunit, famfamfam, the Etelos Application Server Framework, etc.) used by applications within an account.

[00516] In some embodiments, the account 6310 includes one or more applications 6320-1 to 6320-N. For example, the applications 6320 can include applications purchased through the marketplace 4410 in Figure 44. In some embodiments, one or more of the applications 6320 are web-based applications. In these embodiments, a user can interact with one or more of the applications 6320 using a browser engine (e.g., the browser module 6348). In some embodiments, the applications 6320 are located on a remote server separate and

distinct from a computer system and/or a server that includes the browser module 6348. In these embodiments, the browser module 6348 can access the applications 6320 that are executed on the remote server via a network connection. In some embodiments, the applications 6320 are located on the same computer system that includes the browser module 6348. In these embodiments, the browser module 6348 can access the applications 6320 that are executed on the local computer system. In some embodiments, the applications 6320 are located both on a remote server separate and the local computer system. In these embodiments, the browser module 6348 can access the applications 6320 that are executed on the remote server via a network connection. If a connection to the remote server is not available (e.g., the network connection is not available, the remote server is unavailable, etc.), the browser module 6348 can access the applications 6320 on the local computer system. These embodiments are discussed above in reference to the AOP mode of operation. In some embodiments, the browser module 6348 can access the applications 6320 on the local computer system and the remote server.

[00517] In some embodiments, a respective application 6320 includes one or more of: a database 6322, files 6326, and/or a version control repository 6328. The databases 6322 can include data for the respective application 6320. The files 6326 can include application files for the respective application 6320. For example, the files 6326 can include static HTML files, scripts, graphics files, video files, XML files, access control files, documents, etc. The version control repository 6328 can include one or more versions of the files 6326. For example, the version control repository 6328 can include repositories managed by version control systems such as Subversion (SVN), Concurrent Version System (CVS), Revision Control System (RCS), etc. In some embodiments, the files 6326 can be modified by developers through a version control interface (e.g., WebDAV). When making changes to the files 6326, the version control system maintains a history of the changes, which can be used to generate patches, updates, and/or new versions of the files 6326 that can be distributed and deployed to other instances of the respective application (e.g., the application can be running on different account on a different server, etc.). In some embodiments, the files 6326 and the version control repository 6328 for a respective application are associated with one or more vhosts 6324. A virtual host is a mechanism that allows a web server to host one or more domains. For example, www.example.com and www.example2.com can both be hosted on the same web server using a virtual host mechanism. Each virtual host may include a separate directory hierarchy within the web server and/or a separate database (e.g.,

either on the web server or on a separate database server or servers). A virtual host (e.g., vhost) for a respective application can point to a different set of files that are owned and/or used by the respective application.

[00518] In some embodiments, one or more of the applications 6320 are configured to use a specified email domain. In these embodiments, the email module 6342 can be configured by the applications 6320 to send and/or receive email on the specified email domain.

[00519] Note that a newly-created account may not include any applications. Also note that although Figure 63 illustrates a single account, the application framework 6300 can support any number of accounts (e.g., 0, 1, 2, ... , N). Moreover, although components are grouped together in Figure 63, these groupings can be changed. For example, the software stack 6340 can include all or a subset of the software packages 6316.

[00520] Figure 64A is a block diagram illustrating exemplary components of an account 6420, according to some embodiments. The account 6420 can be the account 6310 in Figure 63. The account 6420 can include one or more of an account directory 6422, a database 6424, software packages 6426, and applications 6430 (including one or more of databases 6432, files 6436, version control repository 6438, and vhost 6434), which correspond in general to the account components described above with reference to Figure 63. In some embodiments, the account 6420 includes domains 6428, which include one or more Internet domain names associated with applications within the account 6420.

[00521] In some embodiments, an EMS database 6402 includes one or more of: a users table 6404, an accounts table 6406, a databases table 6408, a domains table 6410, an applications table 6412, and/or a vhosts table 6414. Recall that “EMS” is used to describe a generic account management system. The users table 6404 can include user records for one or more users that can be associated with one or more accounts. The accounts table 6406 can include account records that can include information about accounts, locations of account directories, users associated with the accounts, etc. The databases table 6408 can include “database” records that can include information about databases for the account 6420 (e.g., the database 6424, the database 6432, etc.). The domains table 6410 can include domain records for one or more domains associated with the account 6420 and/or the applications 6430. The applications table 6412 can include application records that can include information about the applications 6430, the files 6436 and directories within the account

directory 6422, the version control repositories 6438, and/or the vhosts 6434 associated with the account 6420. The vhosts table 6414 can include virtual host records that can include information associated with a vhost directory, an account domain, and/or the version control repository 6438.

5 [00522] Figure 64A also includes a software stack 6450, which can correspond to the software stack 6340 in Figure 63.

[00523] Figure 64B is a flow diagram of an exemplary process 6460 for creating an account and installing applications into the account, according to some embodiments. Note that the circled numbers in Figure 64B correspond to the circled numbers in Figure 64A. In
10 some embodiments, an EMS (e.g., the EMS 6302) performs these operations. The discussion of Figure 64B below describes the operations as being performed by the EMS for the sake of clarity. However, it should be noted that other modules can perform all or a subset of these operations. The process 6460 may be governed by instructions that are stored in a computer readable storage medium and that are executed by one or more processors of one or more
15 servers. Each of the operations shown in Figure 64B may correspond to instructions stored in a computer memory or computer readable storage medium. The computer readable storage medium may include a magnetic or optical disk storage device, solid state storage devices such as Flash memory, or other non-volatile memory device or devices. The computer readable instructions stored on the computer readable storage medium are in source code,
20 assembly language code, object code, or other instruction format that is interpreted by one or more processors.

[00524] The EMS creates an admin user record for an admin user in the admin users table 6404 (6462). In some embodiments, an application synchronization operation and/or an AOP synchronization operation cannot be performed for admin users. Note that non-admin
25 users associated with an account can also be created in a separate process after the account has been created. These non-admin users can be stored in a “users” table (not shown). In some embodiments, an application synchronization operation and/or an AOP synchronization operation can be performed for non-admin users.

[00525] Returning to Figure 64B, the EMS creates an account record in the accounts table 6406 and an account directory that are associated with the admin user record (6464).
30 The EMS then creates an account database and an associated “database” record in the databases table 6408 for the newly created account database (6466). Next, the EMS creates a

domain record in the domains table 6410 (6468). The EMS then creates an application directory and an application record in the applications table 6412 associated with the application directory (6470). The EMS then creates an application database and a “database” record in the databases table 6408 associated with the application database (6472). Next, the EMS creates a vhost directory, a version control repository for the vhost directory, and/or a vhosts record in the vhosts table 6414 (6474). Note that a given application (e.g., the application 6430) can have many virtual hosts (e.g., the vhost 6434, etc.) associated with it. Thus, multiple vhost directories, version control repositories, and/or vhosts records can be associated with the given application. In some embodiments, a virtual host (e.g., the vhost 6434) for an application (e.g., the application 6430) can be associated with a database. In these embodiments, a database connection string, which indicates how the virtual host can connect to the database, can be stored in the databases table 6408 and/or the vhosts table 6414. The EMS then downloads and installs the software packages 6426 (6476). The EMS then creates configuration files for the modules in the software stack 6450 (6478).

[00526] Although Figure 64B describes an exemplary process for creating a new account and installing applications into the new account, only a subset of the operations in Figure 64B may need to be performed. For example, if a user record for a given user and an account record associated with the user already exist in the EMS database 6402, then steps 6462-6466 can be omitted when adding a new application to the existing account.

Synchronizing Applications

[00527] Some embodiments provide techniques for synchronizing data between applications. In these embodiments, the code and/or the architecture of the applications do not need to be modified in order to synchronize data between the applications. In some embodiments, the data is synchronized between the databases for the applications without using explicit commands executed by the applications. In other words, the synchronization of data occurs at the database level, and not the application level. In some embodiments, the application framework described above is used to facilitate synchronizing data between applications. For example, the data that can be synchronized between applications can include tasks, contacts, calendar items, etc. The data is typically stored in tables within a database. The tables in which the data is stored can be different for each application. For example, a first application may store contact data in a “contacts” table in a database for the

first application, whereas a second application may store contact data in an “email” table in a database for the second application. In some embodiments, one or more of the applications is a web-based application.

[00528] Figure 65 is a block diagram 6500 of a server 6510 and a client 6530, according to some embodiments. In some embodiments, the server 6510 includes an application framework 6512. The application framework 6512 can include one or more accounts. For example, the application framework 6512 can include an account 6514. The account 6514 can include one or more applications 6516. The applications 6516 can be associated with application databases 6518, which can store, update, delete, retrieve, query, and search for data in application databases 6518. As illustrated in Figure 65, the account 6514 includes the applications 6516-1 and 6516-2 and the associated application databases 6518-1 and 6518-2, respectively.

[00529] In some embodiments, a subset of the data stored in the application databases 6518-1 and 6518-2 can be synchronized with each other. For example, the subset of the data can be data that has changed (e.g., new data, updated data, deleted data, etc.). In some embodiments, the subset of the data stored in the application databases 6518-1 and 6518-2 are first synchronized with a data framework 6519 based on synchronization rules 6520. The synchronization rules 6520 are then applied to the data that was synchronized to the data framework 6519 to synchronize this data with the respective application database.

Alternatively, the synchronization rules can be applied directly to synchronize the subset of the data stored in the application databases 6518-1 and 6518-2 without using the data framework 6519. The synchronization rules 6520 are described in more detail below. In some embodiments, the applications 6516-1 and 6516-2 can be applications that do not have any built-in capabilities to synchronize data with each other. These applications can include, but are not limited to, email applications, CRM applications, calendar applications, etc. If these applications are standalone applications, they may not be able to synchronize data with each other. For example, GOOGLETM calendar may not be able to synchronize tasks with SUGAR CRM. In some embodiments, the application databases 6518-1 and 6518-2 are synchronized with each other directly. For example, a database that includes contact information for GOOGLETM calendar can synchronize directly with a database that includes contact information for SUGAR CRM. This synchronization can be performed “outside” of the applications. In other words, the applications do not need to be executing or active to perform the synchronization.

[00530] In some embodiments, the client 6530 includes an application framework 6532. The application framework 6532 can include one or more accounts. For example, the application framework 6532 can include an account 6534. The account 6534 can include one or more applications 6536. The applications 6536 can be associated with application
5 databases 6538, which can store, update, delete, retrieve, query, and search for data in application databases 6538. As illustrated in Figure 65, the account 6534 includes the applications 6536-1 and 6536-2 and the associated application databases 6538-1 and 6538-2, respectively. For example, the applications 6536 can be GOOGLETM calendar and SUGAR CRM, respectively, and the databases 6538 can be databases for GOOGLETM calendar and
10 SUGAR CRM, respectively.

[00531] In some embodiments, a subset of the data stored in the application databases 6538-1 and 6538-2 can be synchronized with each other. For example, the subset of the data can be data that has changed (e.g., new data, updated data, deleted data, etc.). In some
15 embodiments, the subset of the data stored in the application databases 6538-1 and 6538-2 are first synchronized with a data framework 6539 based on synchronization rules 6540. The synchronization rules 6540 are then applied to the data that was synchronized to the data framework 6539 to synchronize this data with the respective application database. For
20 example, if changes have occurred in GOOGLETM calendar, fields (or columns, etc.) in the database for GOOGLETM calendar first synchronized with the data framework 6539, then the synchronization rules 6540 are applied to the data framework 6539 to propagate/synchronize the data from the data framework 6539 to the database for SUGAR CRM. Alternatively, the
25 synchronization rules can be applied directly to synchronize the subset of the data stored in the application databases 6538-1 and 6538-2 without using the data framework 6539. The synchronization rules 6530 are described in more detail below. In some embodiments, the applications 6536-1 and 6536-2 can be applications that do not have any built-in capabilities to synchronize data with each other. In some embodiments, the application databases 6538-1 and 6538-2 are synchronized with each other directly.

[00532] In some embodiments, the applications 6516 in the account 6514 on the server 6510 can be synchronized with the applications 6536 in the account 6534 on the client 6530.
30 In some embodiments, the data frameworks 6519 and 6539 are first synchronized with each other, and then the synchronization rules 6520 and 6540, respectively, are applied to the synchronized data to synchronize the application databases 6518 and 6538, respectively. In some embodiments, the AOP synchronization process described above is used to synchronize

the application databases 6518 on the server 6510 with the application databases 6538 on the client 6530.

[00533] Note that although Figure 65 illustrates a single account in the server 6510 and a single account in the client 6530, more than one account can be included on the server 6510 and/or the client 6530. Furthermore, each account can include any number of applications.

[00534] In some embodiments, the data frameworks 6519 and 6539 and the sync rules 6520 and 6540 are included in an application synchronization module (e.g., the application synchronization module 6306 in Figure 63).

[00535] Although Figure 65 illustrates that both the server 6510 and the client 6530 include application frameworks, in some embodiments, either the server 6510 or the client 6530 may not include an application framework.

[00536] Figure 66 is a flow diagram of an exemplary process 6600 for synchronizing applications, according to some embodiments. Note that the circled numbers in Figure 66 correspond to the circled numbers in Figure 65. The process 6600 may be governed by instructions that are stored in a computer readable storage medium and that are executed by one or more processors of one or more servers. Each of the operations shown in Figure 66 may correspond to instructions stored in a computer memory or computer readable storage medium. The computer readable storage medium may include a magnetic or optical disk storage device, solid state storage devices such as Flash memory, or other non-volatile memory device or devices. The computer readable instructions stored on the computer readable storage medium are in source code, assembly language code, object code, or other instruction format that is interpreted by one or more processors.

[00537] As illustrated in Figure 66, changes are detected in an application database (e.g., the application database 6518-1 in Figure 65) for a first application (e.g., the application 6516-1) when a user uses the first application (6602). The changes are then pushed to a data framework (e.g., the data framework 6519 in Figure 65) (6604) and the changes are then synchronized with a second application (e.g., the application 6516-2 in Figure 65) (6606). The second application has synchronized the changes (6614). In some embodiments, the changes are synchronized between the server (e.g., the server 6510 in Figure 65) and the client (e.g., the client 6530 in Figure 65) (6616). In these embodiments, the AOP synchronization technique described above can be used.

[00538] In some embodiments, synchronizing the changes with the second application includes pulling changes from the data framework (6608), applying synchronization rules to the changes (6610), and applying the changes to an application database (e.g., the application database 6518-2 in Figure 65) for the second application (6612).

5 [00539] Note that the process 6600 can be performed at the client 6530 and/or the server 6510 in Figure 65. Moreover, changes from any applications within an application framework can be synchronized with other applications within the application framework. For example, changes made in the application 6516-2 can be synchronized with the application 6516-1 and changes made in the application 6516-1 can be synchronized with the
10 application 6516-2. Furthermore, changes made in applications on a server/client can be synchronized with applications on a client/server (e.g., using the AOP synchronization technique described above).

[00540] Figure 67 is a block diagram illustrating an exemplary process 6700 for synchronizing applications, according to some embodiments. The process 6700 may be
15 governed by instructions that are stored in a computer readable storage medium and that are executed by one or more processors of one or more servers. Each of the operations shown in Figure 67 may correspond to instructions stored in a computer memory or computer readable storage medium. The computer readable storage medium may include a magnetic or optical disk storage device, solid state storage devices such as Flash memory, or other non-volatile
20 memory device or devices. The computer readable instructions stored on the computer readable storage medium are in source code, assembly language code, object code, or other instruction format that is interpreted by one or more processors. In some embodiments, the process is performed on a computer system that is to be synchronized. For example, the computer system can be a server or a client computer system.

25 [00541] An outbound process module 6702 receives an outbound order 6740, which includes changes to data for one or more applications and an order that changes to data are applied. Note that the changes to the data can be made to one or more databases, one or more structured files, and/or one or more documents. In some embodiments, the ordering of changes to data in the outbound order 6740 can be: all updates 6741, all inserts 6742, all
30 deletes 6743, parent deletes 6744, child deletes 6745, parent inserts 6746, and child inserts 6747. The all updates 6741 includes all updates to existing data. The all inserts 6742 includes all new data records that are to be added. The all deletes 6743 includes all existing

data records that are to be deleted. The parent deletes 6744 include parent data records that are to be deleted. The child deletes 6745 include child data records that are to be deleted. The parent inserts 6746 include parent data records that are to be deleted. The child inserts 6747 include child data records that are to be deleted.

5 **[00542]** In some embodiments, an outbound execute module 6704 receives the changes to the data for one or more applications and executes one or more of: an outbound insert module 6710 that inserts new data records, an outbound update module 6720 that updates existing data records, and an outbound delete module 6730 that deletes existing data records for one or more applications 6750. The one or more applications 6750 can also receive
10 changes to the data from an AOP synchronization process through the user_in: Hooks module 6772. The user_in:Hooks module 6772 can include an auto increment module 6774, which is described in more detail above with reference to Figures 27-29 above.

[00543] In some embodiments, the outbound insert process 6710 includes hooks 6711 and filters 6715. The hooks 6711 can include one or more of: hooks to a backdata update
15 module 6712 that performs an initial data synchronization (e.g., as described in Figures 18 and 19 above), hooks to a licensing module 6713 that verifies that licenses are valid (e.g., as described in Figures 44 and 57 above), and/or hooks to a user management module 6714 (e.g., as described in Figure 57 above). Note that the term “hooks” refers to a mechanism by which a given module can access another module. For example, a hook can include a
20 function, a procedure, and/or a method within a given module that can access functions of another module, an API function, procedure, and/or method, etc. In some embodiments, the backdata update module 6712 requeues new data records (e.g., data records to be added) for an application as updates that are updated in the outbound update process 6720. The filters 6715 can include an owner_id translation filter 6716 that translates owner_ids between
25 applications. Note that the term “filter” refers to a mechanism that can perform a specified operation on data. For example, this operation can include a translation operation that translates data values, a selection operation that selects data values based on specified criteria, etc.

[00544] In some embodiments, the outbound update process 6720 includes hooks 6721 and filters 6724. The hooks 6721 can include one or more of: hooks to a licensing module
30 6713 that verifies that licenses are valid and/or hooks to a user management module 6714.

The filters 6722 can include an owner_id translation filter 6716 that translates owner_ids between applications.

[00545] In some embodiments, the outbound delete process 6730 includes hooks 6731. The hooks 6731 can include one or more of: hooks to a licensing module 6713 that verifies
5 that licenses are valid and/or hooks to a user management module 6714.

[00546] Note that if the licensing modules 6713 determine that a license for an application is not valid, the outbound insert process 6710, the outbound update process 6720, and/or the outbound delete process 6730, respectively, are not performed. Similarly, if the user management modules 6714 determine that a user of an application does not have
10 sufficient privileges to change specified data for the one or more applications, the outbound insert process 6710, the outbound update process 6720, and/or the outbound delete process 6730, respectively, are not performed.

[00547] In some embodiments, as changes to the data for the one or more applications 6750 are detected by an inbound module 6752, the changes can be accumulated for one or
15 more of an AOP synchronization process and an application synchronization process. If the changes are being accumulated for an application synchronization process (6754, App Sync), the changes are accumulated into an account synchronization log 6762, which is then used by an outbound process 6764. If the changes are being accumulated for an AOP synchronization process (6754, AOP), the changes are accumulated into an outgoing synchronization log
20 6756. Changes from the outgoing_sync_log and the AOP user_in are then separated into “user out” tables 6758 as described above with reference to Figure 27 above. The user_out tables are then received by an AOP sync_in process 6760 on a receiving system (e.g., a server if the changes were made on a client, or a client if the changes were made on the server).

[00548] In some embodiments, an AOP sync_out process 6766 receives changes to
25 data for applications on a transmitting system made when the applications were operating in an AOP mode (e.g., as described above). These changes can be included in a user_in table 6768. The changes included in the user_in tables are then processed 6770. In some embodiments, the changes included in the user_in tables are received by the outbound process 6764, which incorporates the changes made to applications within a given computer
30 system (e.g., application synchronization) and changes made to applications on different computer systems (e.g., AOP synchronization). In some embodiments, the changes included in the user_in tables are processed by the auto increment module 6774 as described above.

After being processed by the auto increment module 6774, the changes included in the user_in tables are incorporated into the user_out tables 6758 and/or the one or more applications 6750. Note that the changes included in the user_in tables can be included in the user_out tables because an application synchronization operation may include changes both to the instance of the application on the server and on the client.

[00549] Figure 68 is a block diagram illustrating an exemplary process 6800 for handling parent-child relationships during a synchronization process, according to some embodiments. The exemplary process 6800 describes exemplary operations performed by the owner_id translation module 6716 in Figure 67. Parent-child relationships can be unenforced or can be enforced using foreign keys. Parent-child relationships can be used to associate records with each other. These associations can include a one-to-one (e.g., one parent to one child), a one-to-many (e.g., one parent to multiple children), and a many-to-many (e.g., many parents to many children) relationship. As illustrated in Figure 68, applications 6802 are being synchronized with each other. The applications 6802 can be any of the applications described above (e.g., web-based applications). The applications 6802 include users tables 6804, apply_users tables 6806, applications tables 6808, and databases tables 6810. A user with a given id is associated with an application with an associated id via the apply_users table 6806. For example, the apply_user table 6806-1 can include foreign keys referring to a data record in the users table 6804-1 associated with a given user (e.g., user_id) and a data record in the applications table associated with a given application (e.g., app_id). The apply_user table 6806-1 can also include a unique record id (e.g., "id") and an owner_id value that corresponds to the given user's instance within an application. The corresponding set of tables also exist for the application 6802-2.

[00550] In some embodiments, the parent-child relationships are defined in sync_vmap_parent tables 6812 and sync_vmap_child tables 6814. The sync_vmap_parent tables 6812 include an app_table field, an app_column field, a database_id field, a translate field, an auto_inc field, and a primary key field. The app_table and app_column fields specify an application table and column, respectively. The database_id field specifies the database_id in the databases table 6408 corresponding to the database including the app_table and app_column. The translate field specifies whether an owner_id translation is to be performed for a given parent-child relationship defined in the sync_vmap_parent tables 6812. The auto_inc field specifies that the parent value is an auto incrementing value (e.g. used in AOP sync). The primary key field indicates that the parent value is a primary key of a table

(e.g., used for AOP synchronization operations and application synchronization operations). For application synchronization operations, the primary key can indicate whether the EMS needs to generate a unique identifier for a given field. For AOP synchronization operations, the primary keys are required to be the same on the server and the client. Thus, the primary key can be used to reinforce the parent/child relationships. The sync_vmap_child tables include the app_table field, an app_column field, and a parent_id field.

[00551] In some embodiments, during a synchronization process, the changes to the data are stored one or more sync log entries. An exemplary sync log entry is illustrated in Figure 68. The exemplary sync log entry includes one or more of: sync_id (e.g., a unique identifier for the sync log entry), transaction (e.g., a unique identifier for a given database transaction), account_sync (e.g., the type of synchronization data – AOP sync, application sync, etc.), sync_column (e.g., the column to be synchronized), broadcast_table (e.g., a table in the source database), broadcast_column (e.g., a column in a table that in the source database includes the changed data), broadcast_database_id (e.g., a database_id in the databases table 6408 corresponding to the source database), broadcast_action (e.g., an action performed by the source database – insert, update, delete, etc.), broadcast_vmap_id (e.g., a unique identifier for a source vmap), datatypes (e.g., the data type of the changed data), new_bool (e.g., if a column is a Boolean value and an Insert or an Update has occurred, this stores the new Boolean value), old_bool (e.g., if a column is a Boolean value and an Update has occurred, this stores the old Boolean value), new_text (e.g., if a column is not a Boolean value and an Insert or an Update has occurred, this stores the new value), old_text (e.g., if a column is not a Boolean value and an Update has occurred, this stores the old value), group_ids (e.g., identifies ownership and record filtering for AOP and/or application synchronization operations when group ownership is declared), map_id (e.g., a unique identifier that is used to identify records across applications), owner_id (e.g., an identifier for the user that corresponds to the user's instance with an application), backdata (e.g., whether a backdata update is to be performed), from_aop (e.g., whether the sync log entry resulted from an AOP synchronization operation), subscribe_action (e.g., an action to be performed by the subscribing database – insert, update, delete, etc.), subscribe_database_id (e.g., a database_id in the databases table 6408 corresponding to the subscribing database), subscribe_table (e.g., a table in the subscribing database), subscribe_column (e.g., a column in a table that in the subscribing database includes the changed data), deploy_licensing_id (e.g., a unique identifier used to identify the licensing model for a given application), vmap_ids (e.g., a

unique identifier for a destination/subscribing vmap), and special (e.g., used to mark phrases – updating, complete, etc. – for auto-increment).

[00552] In some embodiments, every table in the application used by the synchronizer includes a column “eas_sync_map_id.” During a synchronization operation, the field

5 “map_id” includes the unique identifier that corresponds to the value in eas_sync_map_id column. This column can include a 32 character unique identifier which is used to identify records across applications. For example, when updating or deleting a value during synchronization, the value in the eas_sync_map_id column is used to match to the eas_sync_map_id in the destination app.

10 **[00553]** In some embodiments, the sync log entry 6816 includes data that allows the lookup of parent/children relationships. This lookup can be performed while the sync log entry is being processed. Note that the fields prefixed with “broadcast_” include data pertaining to the source database and the fields prefixed with “subscribe_” include data pertaining to the destination database.

15 **[00554]** In some embodiments, if a subscribing database includes a parent/child relationship defined on a subscribe_table and/or a subscribe_column, the sync log entry is treated as a special case and is handled in a different order. (e.g., see the outbound order 6740 in Figure 67). In some embodiments, a subscribing database is a database that has requested to receive changes to data made in another database. In some embodiments, if a given sync
20 log entry corresponds to a data record that has a parent-child relationship and the data record is marked as “translate” (e.g., via the sync_vmap_parent 6812 tables), the apply_user records for the source database are queried to determine a user associated with the owner_id for the changed data that the sync log entry is referencing. The apply_user records for the subscribing database are also queried to determine the owner_id for the subscribing database
25 associated with the determined user. The owner_id value of the subscribing database can then be used to replace the value of the owner_id in the sync log entry.

[00555] Figure 69 is a flow diagram of an exemplary process 6900 for translating owner IDs, according to some embodiments. The process 6900 may be governed by instructions that are stored in a computer readable storage medium and that are executed by
30 one or more processors of one or more servers. Each of the operations shown in Figure 69 may correspond to instructions stored in a computer memory or computer readable storage medium. The computer readable storage medium may include a magnetic or optical disk

storage device, solid state storage devices such as Flash memory, or other non-volatile memory device or devices. The computer readable instructions stored on the computer readable storage medium are in source code, assembly language code, object code, or other instruction format that is interpreted by one or more processors. For example, the process 6900 can be performed by the application synchronization module 6306 (e.g., in a client and/or a server) in Figure 63. The discussion of Figure 69 below describes the operations as being performed by the application synchronization module 6306 of a computer system (e.g., a client and/or a server) for the sake of clarity. However, it should be noted that other modules can perform all or a subset of these operations.

[00556] In Figure 69, the application synchronization module 6306 determines whether the app_column for a subscribing database is a vmap_parent or a vmap_child of a parent that has the “translate” field set to “true” (6902). If the “translate” field is set to “false” (6904, no), the application synchronization module 6306 continues with the synchronization (6910). If the “translate” field is set to “true” (6904, yes), the application synchronization module 6306 obtains the owner_id corresponding to the user of the subscribed database from the apply_user table (6906). Next, the application synchronization module 6306 replaces the app_column value with the owner_id from the apply_user table (6908). The application synchronization module 6306 then continues with the synchronization (6910).

[00557] Figure 70 is a block diagram illustrating an exemplary process 7000 for merging users between applications, according to some embodiments. Figure 71 is a flow diagram of an exemplary process 7100 for merging users between applications, according to some embodiments. It is desirable to merge users between applications because a single user may have different user identifiers (e.g., user_ids) in different applications. For example, the same user may have a user identifier of 2 in a first application and a user identifier of 6 for a second application. It is therefore desirable to merge these two users into a single user so that the synchronization of user data can be facilitated. The process 7100 corresponds to the process 7000 and are discussed together. Note that the circled numbers in Figure 70 correspond to the circled numbers in Figure 71. The process 7100 may be governed by instructions that are stored in a computer readable storage medium and that are executed by one or more processors of one or more servers. Each of the operations shown in Figure 71 may correspond to instructions stored in a computer memory or computer readable storage medium. The computer readable storage medium may include a magnetic or optical disk storage device, solid state storage devices such as Flash memory, or other non-volatile

memory device or devices. The computer readable instructions stored on the computer readable storage medium are in source code, assembly language code, object code, or other instruction format that is interpreted by one or more processors.

[00558] Figure 70 includes applications 7004-1 and 7004-2 and EMS 7008, which can be included in an application framework (e.g., the application framework 6300) on a computer system (not shown). A user is created in the applications 7004-1 and 7004-2 for a user 7002 (7102). For example, the user can have a user_id of 4 in the application 7004-1 and a user_id of 3 in the application 7004-2. A user is also created in the EMS 7008 (7104). For example, the user can have a user_id of 6 in the EMS 7008. In some embodiments, the EMS user_ids are applied to the applications 7004-1 and 7004-1 (7106) and the owner_id values are set (7108). Note that applying users to applications identifies the user's relationship and access to the application. The user's relationship is the "owner_id," which is the user's unique identifier for the application. Also note that a user cannot access an application using AOP unless the user is applied to that application and the user is allowed to use AOP (e.g., the AOP flag set for the user). Furthermore, applications may use this record to authenticate a user for an application. Thus, the application can query the EMS to determine whether the user has permission to use the application.

[00559] Returning to Figures 70 and 71, the user_ids are merged in the EMS (7110). Thus, user_id = 4 for the application 7004-1 and user_id = 3 for the application 7004-2 correspond to the same user_id = 6 in the EMS 7008.

[00560] At a later time, the user 7002 creates a task 7006 associated with the user_id = 3 in the application 7004-2 (7112). Note that although the task 7006 is described in this example, other objects can be created (e.g., calendar items, contacts, documents, etc.). Next, the task 7006 is synchronized with the EMS 7008 (7114). In doing so, the EMS 7008 determines the EMS user_id corresponding to the user_id = 3 for the application 7004-2 (e.g., user_id = 6). The EMS 7008 then translates the user_id in the task 7006 from user_id = 3 for the application 7004-2 to user_id = 4 for the application 7004-1 (7116). The EMS 7008 then sends the task 7006 with the translated user_id to the application 7004-1 (7118). The application 7004-1 receives the task 7006 with the translated user_id and creates a task in the database for the application 7004-1 corresponding to the task 7006 generated by the application 7004-2 (7120).

[00561] Figure 72 presents a block diagram of an exemplary server 7200, according to some embodiments. The server 7200 generally includes one or more processing units (CPU's) 7202, one or more network or other communications interfaces 7204, memory 7210, and one or more communication buses 7208 for interconnecting these components. The communication buses 7208 may include circuitry (sometimes called a chipset) that interconnects and controls communications between system components. The server 7200 may optionally include a display 7206 and one or more input devices 7205 (e.g., keyboard, mouse, trackpoint, etc.). The memory 7210 includes high-speed random access memory, such as DRAM, SRAM, DDR RAM or other random access solid state memory devices; and may include non-volatile memory, such as one or more magnetic disk storage devices, optical disk storage devices, flash memory devices, or other non-volatile solid state storage devices. The memory 7210 may optionally include one or more storage devices remotely located from the CPU(s) 7202. The memory 7210, or alternately the non-volatile memory device(s) within the memory 7210, comprises a computer readable storage medium. In some embodiments, the memory 7210 stores the following programs, modules and data structures, or a subset thereof: operating system 7211, network communication module 7212, an EMS module 7213, an account framework 7230, a software stack 7250, and/or auxiliary modules 7260.

[00562] The operating system 7211 includes procedures for handling various basic system services and for performing hardware dependent tasks. Network communication module 7212 can be used for connecting the server 7200 to other computers via the one or more communication network interfaces 7204 (wired or wireless) and one or more communication networks, such as the Internet, other wide area networks, local area networks, metropolitan area networks, and so on.

[00563] In some embodiments, the EMS module 7213 includes one or more of: an EMS database 7214 and/or an application synchronization module 7215. The application synchronization module can include one or more of: a data framework 7216, synchronization rules 7218, synchronization logs 7219, filters 7220, and hooks 7221. Note that these components and/or modules are described in more detail with reference to Figures 63-71, and 74-79. In some embodiments the data framework 7216 includes a synchronization data structure 7217 that can be used to synchronize data between applications. The synchronization data structure 7217 is described in more detail with reference to Figures 74-79.

[00564] In some embodiments, the account framework 7230 includes one or more of: an account directory 7231, an account database 7232, software packages 7233, and/or one or more applications 7240. The applications 7240 can include one or more of: an application database 7241, files 7243, and version control repository 7244. The files 7243 and the
5 version control repository 7244 can be associated with a vhost 7242. In some embodiments, the software packages 7233 can include utilities (e.g., phpmyadmin, phppgadmin, websvn, EDK, etc.), tools, and/or shared libraries (e.g., Zend, PHPunit, famfamfam, the Etelos Application Server Framework, etc.) used by applications within an account. These components and/or modules are described in more detail with reference to Figures 63-71, and
10 74-79.

[00565] In some embodiments, the software stack 7250 includes one or more of: an email module 7251, a web server module 7252, an authoring module 7253, and/or other services 7254. Email module 7251 can include any electronic mail program. For example, the email module 7251 can include email servers (e.g., Sendmail, Postfix, and qmail, etc.),
15 mail filtering programs (e.g., procmail, SpamAssassin, etc.), client email programs (e.g., elm, pine, Microsoft Outlook, etc.), etc. The web server module 7252 can include any application that can respond to client requests for pages, data, graphics, videos, documents, etc, hosted on a server coupled to clients and/or other servers via a network. For example, the web server module 7252 can include Apache HTTP Server, Microsoft Internet Information Server, etc.
20 The authoring module 7253 can include any application that allows a developer or a user to generate web-based applications. For example, the authoring module 7253 can include a text editor, a word processor, a web development environment (e.g., Microsoft Expression Web, Adobe Dreamweaver, etc.), etc. The other services 7254 can include other applications such as a Lightweight Data Access Protocol (LDAP) interface module and/or database, an Internet
25 Message Access Protocol (IMAP) module, a Post Office Protocol (POP) module, a Dovecot module (e.g., an IMAP and POP server), a web server (e.g., Apache HTTP server, Microsoft IIS Server, etc.), one or more database management systems (e.g., MySQL, PostgreSQL, etc.), logging applications (e.g., Syslog-ng, etc.), authentication services (e.g., saslauthd, etc.). Note that although Figure 72 illustrates a single module for each type of service, the software
30 stack 7250 can include any number of modules for a given type of service. For example, two different email modules can be included within software stack 7250.

[00566] In some embodiments, the auxiliary services module 7260 includes one or more of: famfamfam (icon images), phpMyAdmin (php web-based MySQL database

management interface), phpPGAdmin (php web-based PostgreSQL database management interface), WebSVN (php web-based Subversion interface), ImageMagick (image manipulation library), ZendFramework (php utility framework), IconCube Loaders (encrypted-php decryption library), libpng (PNG manipulation library), libjpeg (JPEG manipulation library), Neon (WebDAV client library), mcrypt (encryption library), and FreeType (font utilities library).

[00567] Each of the above identified elements may be stored in one or more of the previously mentioned memory devices, and corresponds to a set of instructions for performing a function described above. The above identified modules or programs (i.e., sets of instructions) need not be implemented as separate software programs, procedures or modules, and thus various subsets of these modules may be combined or otherwise re-arranged in various embodiments. In some embodiments, the memory 7210 may store a subset of the modules and data structures identified above. Furthermore, the memory 7210 may store additional modules and data structures not described above.

[00568] Although Figure 72 shows a “server,” Figure 72 is intended more as functional description of the various features that may be present in a set of servers than as a structural schematic of the embodiments described herein. In practice, and as recognized by those of ordinary skill in the art, items shown separately could be combined and some items could be separated. For example, some items shown separately in Figure 72 could be implemented on single servers and single items could be implemented by one or more servers. The actual number of servers used to implement an application server and how features are allocated among them will vary from one implementation to another, and may depend in part on the amount of data traffic that the system must handle during peak usage periods as well as during average usage periods.

[00569] Figure 73 presents a block diagram of an exemplary client computer system 7300, according to some embodiments. The client computer system 7300 generally includes one or more processing units (CPU's) 7302, one or more network or other communications interfaces 7304, memory 7310, and one or more communication buses 7308 for interconnecting these components. The communication buses 7308 may include circuitry (sometimes called a chipset) that interconnects and controls communications between system components. The client computer system 7300 includes a display 7306 and one or more input devices 7305 (e.g., keyboard, mouse, trackpoint, etc.). The memory 7310 includes

high-speed random access memory, such as DRAM, SRAM, DDR RAM or other random access solid state memory devices; and may include non-volatile memory, such as one or more magnetic disk storage devices, optical disk storage devices, flash memory devices, or other non-volatile solid state storage devices. The memory 7310 may optionally include one or more storage devices remotely located from the CPU(s) 7302. The memory 7310, or alternately the non-volatile memory device(s) within the memory 7310, comprises a computer readable storage medium. In some embodiments, the memory 7310 stores the following programs, modules and data structures, or a subset thereof: operating system 7311, network communication module 7312, an EMS module 7313, an account framework 7330, a software stack 7350, a browser module 7356, and/or auxiliary modules 7360.

[00570] The operating system 7311 includes procedures for handling various basic system services and for performing hardware dependent tasks. Network communication module 7312 can be used for connecting the server 7300 to other computers via the one or more communication network interfaces 7304 (wired or wireless) and one or more communication networks, such as the Internet, other wide area networks, local area networks, metropolitan area networks, and so on.

[00571] In some embodiments, the EMS module 7313 includes one or more of: an EMS database 7314 and/or an application synchronization module 7315. The application synchronization module can include one or more of: a data framework 7316, synchronization rules 7318, synchronization logs 7319, filters 7320, and hooks 7321. Note that these components and/or modules are described in more detail with reference to Figures 63-71 and 74-79. In some embodiments the data framework 7316 includes a synchronization data structure 7317 that can be used to synchronize data between applications. The synchronization data structure 7317 is described in more detail with reference to Figures 74-79.

[00572] In some embodiments, the account framework 7330 includes one or more of: an account directory 7331, an account database 7332, software packages 7333, and/or one or more applications 7340. The applications 7340 can include one or more of: an application database 7341, files 7343, and version control repository 7344. The files 7343 and the version control repository 7344 can be associated with a vhost 7342. In some embodiments, the software packages 7233 can include utilities (e.g., phpmyadmin, phppgadmin, websvn, EDK, etc.), tools, and/or shared libraries (e.g., Zend, PHPunit, famfamfam, the Etelos

Application Server Framework, etc.) used by applications within an account. These components and/or modules are described in more detail with reference to Figures 63-71, and 74-79.

[00573] In some embodiments, the software stack 7350 includes one or more of: an email module 7351, a web server module 7352, an authoring module 7353, and/or other services 7354. Email module 7351 can include any electronic mail program. For example, the email module 7351 can include email servers (e.g., Sendmail, Postfix, and qmail, etc.), mail filtering programs (e.g., procmail, SpamAssassin, etc.), client email programs (e.g., elm, pine, Microsoft Outlook, etc.), etc. The web server module 7352 can include any application that can respond to client requests for pages, data, graphics, videos, documents, etc, hosted on a server coupled to clients and/or other servers via a network. For example, the web server module 7352 can include Apache HTTP Server, Microsoft Internet Information Server, etc. The authoring module 7353 can include any application that allows a developer or a user to generate web-based applications. For example, the authoring module 7353 can include a text editor, a word processor, a web development environment (e.g., Microsoft Expression Web, Adobe Dreamweaver, etc.), etc. The other services 7354 can include other applications such as a Lightweight Data Access Protocol (LDAP) interface module and/or database, an Internet Message Access Protocol (IMAP) module, a Post Office Protocol (POP) module, a Dovecot module (e.g., an IMAP and POP server), a web server (e.g., Apache HTTP server, Microsoft IIS Server, etc.), one or more database management systems (e.g., MySQL, PostgreSQL, etc.), logging applications (e.g., Syslog-ng, etc.), authentication services (e.g., saslauthd, etc.). Note that although Figure 73 illustrates a single module for each type of service, the software stack 7350 can include any number of modules for a given type of service. For example, two different email modules can be included within software stack 7350.

[00574] In some embodiments, the auxiliary services module 7360 includes one or more of: famfamfam (icon images), phpMyAdmin (php web-based MySQL database management interface), phpPgAdmin (php web-based PostgreSQL database management interface), WebSVN (php web-based Subversion interface), ImageMagick (image manipulation library), ZendFramework (php utility framework), IconCube Loaders (encrypted-php decryption library), libpng (PNG manipulation library), libjpeg (JPEG manipulation library), Neon (WebDAV client library), mcrypt (encryption library), and FreeType (font utilities library).

[00575] In some embodiments, the browser module 7356 can include any application with a rendering engine that can access data and/or services on a local and/or a remote computer system and render the results so that a user can view the data and/or interact with the services. In some embodiments, browser module 7356 is a web browser.

5 [00576] Each of the above identified elements may be stored in one or more of the previously mentioned memory devices, and corresponds to a set of instructions for performing a function described above. The above identified modules or programs (i.e., sets of instructions) need not be implemented as separate software programs, procedures or modules, and thus various subsets of these modules may be combined or otherwise re-
10 arranged in various embodiments. In some embodiments, the memory 7310 may store a subset of the modules and data structures identified above. Furthermore, the memory 7310 may store additional modules and data structures not described above.

[00577] Figure 74 presents a block diagram illustrating an exemplary application synchronization module 7400, according to some embodiments. The application
15 synchronization module 7400 can be the application synchronization module 6304 in Figure 63. In some embodiments, the cross-application synchronization module 7400 includes one or more of: synchronization rules 7402, sync log 7410, data framework 7420, filters 7430, hooks 7440, an outbound insert module 7450, an outbound update module 7451, an outbound delete module 7452, an outbound process module 7453, an outbound execute
20 module 7454, an outbound order 7455, an inbound module 7456, and/or an AOP module 7457. These components and/or modules are described in more detail with reference to Figures 1-42 and 63-79.

[00578] In some embodiments, the synchronization rules 7402 include synchronization rules for one or more applications 7404. In some embodiments, the data framework 7420
25 includes a synchronization data structure 7422. The synchronization data structure 7422 is described in more detail with reference to Figure 75. In some embodiments, the filters 7430 include an owner_id translation module 7431 (e.g., as described in Figure 67). In some embodiments, the hooks 7440 include one or more of: a hook to a back data update module 7440, a hook to a licensing module 7441, a hook to a user management module 7442, and a
30 hook to an auto increment module 7443 (e.g., as described in Figure 67).

[00579] Figure 75 presents exemplary synchronization data structures 7500, according to some embodiments. The synchronization data structures 7500 include one or more of: an

eas_sync_app_log data structure 7502, an inbound data structure 7504, and an outbound data structure 7506. The eas_sync_app_log 7502 is used to store the changes made by the application. These changes can be tracked using database triggers on the tables and/or columns to be synchronized. The inbound data structure 7504 includes data that is to be processed by the application. This inbound data can be first placed in a temporary table prior to being processed. The outbound data structure 7506 includes data that is to be preprocessed and grouped together for outbound processing. This processed data can then be placed in an outgoing_sync_log and/or an account_sync_log. If the data is in an account_sync_log, the data is to be outbound processed and sent to the application databases. If the data is in the outgoing_sync_log, the data is to be sent to the user_out and/or the sync_out (for AOP) table to wait for an AOP sync. The data can be placed into the user_in or sync_in (for AOP) and processed as if it was in the account_sync_log.

[00580] Figure 76 is a flow diagram of an exemplary process 7600 for synchronizing applications, according to some embodiments. The process 7600 may be governed by instructions that are stored in a computer readable storage medium and that are executed by one or more processors of one or more servers. Each of the operations shown in Figure 76 may correspond to instructions stored in a computer memory or computer readable storage medium. The computer readable storage medium may include a magnetic or optical disk storage device, solid state storage devices such as Flash memory, or other non-volatile memory device or devices. The computer readable instructions stored on the computer readable storage medium are in source code, assembly language code, object code, or other instruction format that is interpreted by one or more processors.

[00581] As Figure 76 illustrates, changes made to a first data set in a plurality of data sets is detected (7602). At least a first subset of the changes are synchronized to a data framework (e.g., the data framework 7420) that facilitates data synchronization between the plurality of data sets (7204). In some embodiments, a data set and a data framework includes one or more of: one or more databases, one or more structured files, and one or more documents. The structured file can include one of: an extensible markup language (XML) file, a hypertext markup language (HTML) file, and a file including values separated by a delimiter. The delimiter can include one of: a comma, and a tab break. A document can include one of: a web document, a word processor document, a spreadsheet document, a presentation document, and an electronic mail message. In some embodiments, a change to a

data set includes one or more of: an update of data in the data set, an addition of data to the data set, and/or a deletion of data from the data set.

[00582] In some embodiments, synchronizing the at least the first subset of the changes to the data framework includes determining a mapping between data fields in a data structure for the first data set and data fields in a data structure for the data framework (7606) and synchronizing the data fields in the data structure for the first data set corresponding to the at least the first subset of the changes with the data fields in the data structure for the data framework based on the mapping (7608). In some embodiments, synchronizing the data fields includes translating values for the data fields in the data structure for the first data set to corresponding values for the data fields in the data structure for the data framework based on translation rules for the first data set (7610). In some embodiments, the translation rules and/or the mapping are included in the synchronization rules (e.g., the synchronization rules 6520, 6540 and 7218 in Figures 65 and 72, respectively).

[00583] In some embodiments, at least a second subset of the synchronized changes are synchronized from the data framework to a second data set in the plurality of data sets (7612). In some embodiments, synchronizing the at least the second subset of the synchronized changes from the data framework to a second data set in the plurality of data sets includes: determining a mapping between data fields in a data structure for the second data set and data fields in a data structure for the data framework (7614) and synchronizing the data fields in the data structure for the second data set with the data fields in the data structure for the data framework based on the mapping (7616). In some embodiments, synchronizing the data fields includes translating values for the data fields in the data framework to corresponding values for the data fields in the second data set based on translation rules for the second data (7618). In some embodiments, the translation rules and/or the mapping are included in the synchronization rules (e.g., the synchronization rules 6520, 6540 and 7218 in Figures 65 and 72, respectively).

[00584] In some embodiments, at least a third subset of the synchronized changes are synchronized from the data framework to a second data framework (7620).

[00585] Figure 77 is a flow diagram of an exemplary process 7700 for generating synchronization rules that are used to synchronize applications, according to some embodiments. The process 7700 may be governed by instructions that are stored in a computer readable storage medium and that are executed by one or more processors of one or

more servers. Each of the operations shown in Figure 77 may correspond to instructions stored in a computer memory or computer readable storage medium. The computer readable storage medium may include a magnetic or optical disk storage device, solid state storage devices such as Flash memory, or other non-volatile memory device or devices. The computer readable instructions stored on the computer readable storage medium are in source code, assembly language code, object code, or other instruction format that is interpreted by one or more processors.

[00586] As illustrated in Figure 77, a first data set in a plurality of data sets that is to be synchronized with a data framework is identified (7702). A mapping between one or more data fields in a data structure for the first data set and one or more data fields in a data structure for the data framework is determined (7704). Synchronization rules for the first data set based on the determined mapping are generated (7706).

[00587] In some embodiments, at least a subset of the one or more data fields for the first data set that include data values that are to be translated to corresponding data values for the data fields for the data framework are identified (7708). Translation rules for translating data values for the at least the subset of the data fields for the first data set to corresponding data values for the data fields for the data framework are generated (7710). In some embodiments, the translation rules are included in the synchronization rules (e.g., the synchronization rules 6520, 6540 and 7218 in Figures 65 and 72, respectively).

[00588] In some embodiments, the first data set and the data framework include one or more of: one or more databases; one or more structured files; and one or more documents. In some embodiments, the first data set and the data framework include one or more databases. The mapping can include a mapping between columns of one or more database tables for the first data set and columns of one or more database tables for the data framework.

[00589] Figure 78 is a flow diagram of an exemplary process 7800 for synchronizing applications, according to some embodiments. The process 7800 may be governed by instructions that are stored in a computer readable storage medium and that are executed by one or more processors of one or more servers. Each of the operations shown in Figure 78 may correspond to instructions stored in a computer memory or computer readable storage medium. The computer readable storage medium may include a magnetic or optical disk storage device, solid state storage devices such as Flash memory, or other non-volatile memory device or devices. The computer readable instructions stored on the computer

readable storage medium are in source code, assembly language code, object code, or other instruction format that is interpreted by one or more processors.

[00590] As illustrated in Figure 78, changes made to a first data set for a first web-based application in an account are detected (7802). At least a second data set for a second web-based application in the account is identified (7804), wherein the second data set includes at least a subset of the data included in the first data set. Synchronization rules to synchronize the second data set with the first data set are applied (7806).

[00591] In some embodiments, the synchronization rules include: one or more rules to synchronize at least a subset of data fields in a data structure for the first data set with data fields in a data structure for a data framework, and/or one or more rules to synchronize at least a subset of the data fields in the data structure for the data framework with data fields in a data structure for the second data set.

[00592] In some embodiments, the account includes a plurality of data sets and corresponding web-based applications.

[00593] In some embodiments, the synchronization rules include one or more of: a mapping between fields in the first data set and fields in the second data set and/or rules for resolving data conflicts between the first data set and the second data set.

[00594] Figure 79 presents exemplary synchronization rules data structures 7900, according to some embodiments. The synchronization rules data structures include a sync_vmap data structure 7902, a sync_vmap_parent data structure 7904, and a sync_vmap_child data structure 7906.

[00595] Screenshots

[00596] In the following figures (descriptions of screenshots), all references to modules (e.g., marketplace module 6122, user account access module 6172, deployer module 6152, packager module 6136, and marketplace database management module 6180) relate to modules shown in the system of Figure 61. These screen shots are exemplary screenshots of user interfaces that display information for, and enable user interaction with, the various applications and components of the application marketplace of the present invention.

[00597] FIG. 101 is an exemplary screenshot 10100 of an Account User Management Interface, generated by User Account Access Module 6172. The information reflected here is stored in a user database accessed by User Account Database Management Module.

[00598] An Account User Management Interface allows a user to activate services and application for different users. Merging users together between applications enables better synchronization of data between and across applications. Field 10102 displays a user's first name; field 10104 displays a user's last name; field 10106 displays a user's phone number; field 10108 displays a user's email address; field 10110 displays a user's office; field 10112 displays a user's department; field 10114 is a matrix of user access information. These are exemplary fields; in other embodiments different fields may be created.

[00599] FIG. 102 is an exemplary screenshot 10200 of an application packager, generated by Packager Module 6136. The information reflected here is stored in a user database accessed by User Account Database Management Module 6188.

[00600] The Application Packager allows a user to package up the user's application for distribution in the Marketplace or move the application to another hosted solution (e.g. an Etelos hosted solution). Field 10202, and 10204 allow a user to input the user's Marketplace Username, and user's Marketplace password respectively.

[00601] FIG. 103 is an exemplary screenshot 10300 of an application packager, generated by Packager Module 6136. The information entered here is stored in a database accessed by Deployment Database Management Module 6184. Package a new full app (with the associated Begin button) 10302 allow a user to begin packaging a new application. Package an update to an app 10304, allows a user to package an update to an application.

[00602] FIG. 104 is an exemplary screenshot 10400 of an application packager, generated by Packager Module. The information entered here is stored in a database accessed by Deployment Database Management Module 6184. Field 10402 allows a user to enter an application name which the user wishes to package. Field 10404 allows a user to specify a script associated with the package.

[00603] FIG. 105 is an exemplary screenshot 10500 of an application packager, generated by Packager Module 6136. The information entered here is stored in a database accessed by Deployment Database Management Module 6184. Package a new full app (with the associated Begin button) 10502 allow a user to begin packaging a new application. Package an update to an app, 10504, allows a user to package an update to an application.

[00604] FIG. 106 is an exemplary screenshot 10600 of an application packager, generated by Packager Module 6136. The information entered here is stored in a database accessed by Deployment Database Management Module 6184. Field 10602 allows a user to

enter an application name which the user wishes to package. Field 10604 allows a user to specify other data.

[00605] FIG. 107 is an exemplary screenshot 10700 of an application packager, generated by Packager Module 6136. The information entered here is used by Permissions
5 Module 6128 to verify the user's credentials for a selected destination store (e.g. an Etelos store).

[00606] FIG. 108 is an exemplary screenshot 10800 of an application packager, generated by Packager Module 6136. Field 10802 displays a set of icons associated with packaging of a software application for distribution.

10 [00607] FIG. 109 is an exemplary screenshot 10900 of an application packager, generated by Packager Module 6136. Field 10902 shows a file structure for a packaged application.

[00608] FIG. 110 is an exemplary screenshot 11000 of an application packager, generated by Packager Module 6136. Field 11002 allows a user to view Binders and Tables
15 associated with a packaged application. Field 11004 allows a user to select a specific binder and/or table.

[00609] FIG. 111 is an exemplary screenshot 11100 of an application packager, generated by Packager Module 6136 for specifying a script associated with a packaged application.

20 [00610] FIG. 112 is an exemplary screenshot 11200 of an application packager, generated by Packager Module 6136. Scheduler, 11202, allows a user to schedule software applications for packaging updates. Database Broadcast/Subscribe rules, 11204, allows an administrator to specify broadcasting and subscription rules for updates based on an application's license as stored in licensing database management module 6182.

25 [00611] FIG. 113 is an exemplary screenshot 11300 of an application packager, generated by Packager Module 6136.

[00612] FIG. 114 is an exemplary screenshot 11400 of an Account Information Details, generated by User Accounts Module 6156.

[00613] Account Information Details shows the user the detailed information
30 associated with each application installed on the user's account. Field 11402 includes examples of information that may be included in one embodiment.

[00614] FIG. 115 is an exemplary screenshot 11500 of a Development Environment that can be used to create or modify applications. Files panel 11502 displays the file structure associated with an application. Binders panel 11504 displays Binders associated with the application. Panel 11506 can display the contents of selected files, folders, and/or objects.

5 [00615] FIG. 116 is an exemplary screenshot 11600 of an application packager, generated by Packager Module 6136 similar to FIG. 103.

[00616] FIG. 117 is an exemplary screenshot 11700 of an AOP sync rules manager, generated by an application synchronization module (e.g., the application synchronization module 6306 in Figure 63). The AOP sync rules manager 11702 displays current
10 synchronization rules as defined by a user. Column 11704 displays the Code associated with an application. The code column includes the code to execute upon rule execution. Column 11706 displays the status of a rule associated with an application. Column 11708 displays names of applications and/or database for which there is a sync rule defined. Column 11710 and 11712 display a table and a column, respectively, in the application database for which
15 there is a defined sync rule. Button 11714 opens an interface to specify whether a given column in the application database is an autoincrementing column. Button 11716 allows a user to create a new sync rule.

[00617] FIGs. 118-121 are exemplary screenshots of an AOP sync rules manager. FIG. 118 is an exemplary screenshot 11800 of an AOP sync rules manager, allowing a user to
20 define a new sync rule. Field 11802 allows a user to specify the application and/or database for which the user wishes to define a new sync rule. Field 11804 allows the user to choose a table associated with the application and/or database for which the user wishes to define a sync rule.

[00618] FIG. 119 is an exemplary screenshot 11900 of an AOP sync rules manager, allowing a user to define a new sync rule. Field 11902 allows a user to specify an application
25 and/or a database for which the user wishes to define a new sync rule. Field 11904 allows the user to choose a table associated with the application and/or database for which the user wishes to define a new sync rule. Field 11906 allows the user to specify a Column associated with the application and/or database for which the user is defining a new sync rule. Fields
30 11908, 11910, and 11912 allow a user to define a table in the application framework (e.g., Framework Sync Table) to which the table defined in field 11904 is synchronized, an associated column (e.g., Framework Sync Column) in the table defined in field 11904 to

which the column defined in field 11906 is synchronized, and a scope of the synchronization rule, respectively.

[00619] FIG. 120 is an exemplary screenshot 12000 of an AOP sync rules manager, allowing a user to define a new sync rule. Field 12002 allows a user to specify an application and/or a database for which the user wishes to define a new sync rule. Field 12004 allows the user to choose a table associated with the application and/or database for which the user wishes to define a new sync rule. Field 12006 allows the user to specify a Column associated with the application and/or database for which the user is defining a new sync rule. Field 12008 allows a user to select when and what types of changes are synchronized. For example, changes can be synchronized when there are one or more of the following: inserts (e.g., new data added), and updates (e.g., existing data is updated), deletes (e.g., existing data is deleted), etc. Fields 12010 and 12012 allow a user to define a table in the application framework (e.g., Framework Sync Table) to which the table defined in field 12004 is synchronized, and a scope of the synchronization rule, respectively.

[00620] FIG. 121 is an exemplary screenshot 12100 of an AOP sync rules manager, allowing a user to define a new sync rule. Field 12102 allows a user to specify an application and/or a database for which the user wishes to define a new sync rule. Field 12104 allows the user to choose a table associated with the application and/or database for which the user wishes to define a new sync rule. Field 12106 allows the user to specify a Column associated with the application and/or database for which the user is defining a new sync rule. Field 12108 allows a user to select when and what types of changes are synchronized. Fields 12110, 12112, and 12114 allow a user to define a table in the application framework (e.g., Framework Sync Table) to which the table defined in field 12104 is synchronized, an associated column (e.g., Framework Sync Column) in the table defined in field 12104 to which the column defined in field 12106 is synchronized, and a scope of the synchronization rule, respectively.

[00621] FIG. 122 is an exemplary screenshot 12200 of a Group Manager, generated by an access module (e.g., the access module 6170 in Figure 61). In some embodiments, a user may define an application and/or database 12202, group 12204, membership 12206, membership id 12208, data type 12210, group primary key 12212, membership group foreign key 12214, and Data id 12216 associated with an application. By clicking the Create button

12218 a user may create a group definition associated with an application. By clicking the Cancel button 12220, a user may cancel the group definition associated with an application.

[00622] FIG. 123 is an exemplary screenshot 12300 of a sync rules manager wherein a user generates an ID Translation Definition for parent-child relationships. The user may
5 define a Vmap Parent 12302 by defining a table 12304 and a column 12306 for an application and/or a database that is a parent for another column in a table. The user may save changes by clicking the Save button 12308, or cancel changes by clicking the Cancel button 12310.

[00623] FIG. 124 is an exemplary screenshot 12400 of a sync rules manager wherein a user generates an ID Translation Definition. Field 12402 includes applications for which
10 there is an ID Translation Definition. Field 12404 can display the Vmap parent relationships described in FIG. 123. The button Edit 12406 allows a user to edit the relationships information associated with an application.

[00624] FIG. 125 is an exemplary screenshot 12500 of a sync rules manager wherein a user may use Task Editor 12502 to add scripts to a sync rule. The user may specify whether
15 the task is to be performed before or after syncing using field 12504. The user may also specify the type of scripts using the field 12506 (e.g., SQL). The user may further specify in field 12508 whether the script is to evaluate or to execute.

[00625] FIG. 126 is an exemplary screenshot 12600 of a sync rules manager wherein a user may use Task Editor 12602 to add script details to a sync rule similar to FIG. 125.

[00626] FIG. 127 is an exemplary screenshot 12700 of a sync rules manager wherein a user can manage synchronization rules associated with an application. The user may access
20 help information by clicking the question mark button 12702. The user may view a scope of a rule in field 12704, a code in field 12706, a status of a rule in field 12708, when a rule is applied in field 12710, an application and/or a database in field 12712, an application table in
25 field 12714, an application column in field 12716, an account table in field 12718, application account in field 12720. The user may specify an ID Translation by clicking on the plus button 12722, or create a new sync rule by clicking on the plus button 12724.

[00627] FIG. 128 is an exemplary screenshot 12800 of a sync rules manager wherein a user can manage synchronization rules associated with an application similar to FIG. 127.

[00628] FIG. 129 is an exemplary screenshot 12900 of a support page (e.g. an Etelos Support page) wherein a user may use a Storefront 12902 (e.g. Etelos "My Storefront") to

add product or services to a marketplace (e.g. an Etelos marketplace) generated by marketplace module 6122. Field 12904 displays a store listing; field 12906 displays a status of an application; field 12908 displays a type of an application.

[00629] FIG. 130 is an exemplary screenshot 13000 of a Product/Service Basic Information Page generated by listing module 6124. A user may specify a product name in field 13002, short description in field 13004, product type in field 13006, and item status in field 13008. Store Item Details appears in the field 13010. The user may create a Product/Service Basic Information page by clicking on the button Create A Copy 13012.

[00630] FIG. 131 is an exemplary screenshot 13100 of a staging application page. In application file field 13102, a user can specify an application file for e.g. the VBS registration website manager (in one embodiment, VBS is an example software application listed in the marketplace). In application size field 13104, a user can specify the application size. In install time field 13106, a user can specify the estimated install time for the application.

[00631] FIG. 132 is an exemplary screenshot 13200 of an edit licensing page wherein a user may edit licensing for “a registration website manager” 13202. The edit licensing for “registration website” is generated by the licensing module 6126. The information entered on this page is stored by licensing database management module 6182.

[00632] Field 13204 displays the name, description, or billing, retail, wholesale, and type (per account or per user) information associated with a packaged application for which the user may wish to edit licensing information. In field 13206, the user may select an install type for an application. In field 13208, the user may type a description of the application. In field 13210, the user may specify a license version. In field 13212, the user may specify a type of license associated with the application. In field 13214, the user may specify a quantity of licenses associated with the application (allowance of application features like user groups, max records on quantifiable items like tasks, documents, projects, sales reps, etc...). In field 13216, the user may specify whether the license is a trial version. In field 13218, the user may specify a billing type associated with a license associated with the application. In field 13220, the user may specify whether the application is hosted using bundled hosting. In field 13222, the user may specify a hosting provider associated with the application. In field 13224, the user may specify a retail pricing associated with the application. In field 13226, the user may specify a resale term to be associated with a license associated with the application. In field 13228, the user may specify a wholesale pricing associated with the

application. In field 13230, the user may specify a license type associated with a source code associated with the application (e.g. whether source code is licensed as an open source license, etc.). In field 13232, the user may specify whether the source code associated with the application is downloadable. In field 13234, the user may specify whether application synchronization (syncing) is enabled. In field 13236, the user may specify whether the application is Application on Plane (AOP) enabled. In field 13238, the user may specify whether multiple instances of the application are allowed. In field 13240, the user may specify an external product ID associated with the application. In field 13242, the user may specify whether there is an ad server associated with the application. In field 13244, the user may specify the status of a license associated with the application (e.g. whether the license is on or off). These fields are exemplary, other embodiments may incorporate different fields and editing options concerning licensing associated with an application.

[00633] FIG. 133 is an exemplary screenshot 13300 of a setup marketing information page generated by the listing module 6124. A product review and store categories page 13302 includes information used to display an application in a catalog and search result. In field 13304, a user may upload a product image to be displayed with an application. In field 13306, a user may specify a desired turnkey rating associated with an application. In field 13308, a user may specify an internet address associated with a provider of an application. In field 13310, a user may specify different categories associated with an application. In field 13312, a user may specify store items related to an application. In field 13314, a user may specify keywords for which an application may appear in a search result. In field 13316, a user may specify a type of hosting associated with an application.

[00634] FIG. 134 is an exemplary screenshot 13400 of a setup marketing full page generated by the listing module 6124 whereby a user may describe his applications. In field 13402, a user may specify a web address to redirect a purchaser to the user's marketing material. In field 13404, a user may write a description of an application which the user is selling through marketplace module 6122.

[00635] FIG. 135 is an exemplary screenshot of a setup product features full page 13500 generated by the listing module 6124. In field 13502, a user may turn on WYSIWYG. In field 13504, a user may write a description of features in the user's application. The user's application is available for sale through marketplace module 6122.

[00636] FIG. 136 is an exemplary screenshot of a product demo page 13600 generated by the listing module 6124. The information entered here is stored by marketplace database management module 6180. A user may access this product demos page 13602 to upload screenshots, and videos associated with the user's applications available through Marketplace module 6122. In field 13604, a user may add a descriptive text describing the features of the user's application. In field 13606, a user may upload videos to be used as a demo associated with an application. In field 13608, a user may upload screenshots to be used as a demo associated with an application. In field 13610, a user may view previously uploaded screenshots associated with an application.

[00637] FIG. 137 is an exemplary screenshot 13700 of an "Add a Blog" feed page, executed by marketplace module 6122. A user can set up a blog feed for a store item using add a Blog feed page 13702 (e.g. "add a Blog feed for Etelos CRM"). In field 13704, a user may provide a feed link. In field 13708, a user may specify a feed status (e.g. on or off). In field 13708, a user may specify whether blogs should be displayed on the user's store. Using buttons 13710, 13712, and 13714, a user may save changes, update feed, or review feed respectively.

[00638] FIG. 138 is an exemplary screenshot 13800 of a setup frequently asked questions (FAQs) page generated by the listing module 6124. Information entered here is stored in the Marketplace database management module 6180. In field 13802, a user may enter FAQs about the user's application. Using the checkbox 13804, a user can display the FAQ tab on the user's store in Marketplace module 6122.

[00639] FIG. 139 is an exemplary screenshot 13900 of a setup getting started information page generated by the listing module 6124. The information entered here is stored by marketplace database management module 6180. In field 13902, an administrator may write a getting started page associated with an application which a user will see when starting the application. In field 13904, an administrator may write a getting started email associated with an application which a user will receive when the application is installed. In field 13906, an administrator may write a text associated with an application. The text would only appear if a user has purchased a trial version of the application.

[00640] FIG. 140 is an exemplary screenshot 14000 of a support information page generated by marketplace database management module 6180. In field 14002, an

administrator may write a support page to be displayed to the licensees of the administrator's application in Marketplace module 6122.

[00641] FIG. 141 is an exemplary screenshot 14100 of an about us page generated by the listing module 6124. The information entered here is stored by marketplace database management module 6180. Icon 14102, Cart, enables a user to access items in the user's shopping cart. Icon 14104, Account, enables a user to access the users account information. In field 14106, an administrator may specify the administrator's company name. In field 14108, an administrator may specify the administrator's company website link. In field 14110, an administrator can specify a contact email associated with the administrator's application. In field 14112, an administrator may write an "about us" section to be included with the administrator's application by Marketplace module 6122.

[00642] FIG. 142 is an exemplary screenshot 14200 of a product upgrade page generated in some embodiments by the listing module 6124, or in some embodiments by the licensing module. The information entered here is stored by marketplace database management module 6180. Icon 14202, Cart, enables an administrator to access items in the user's shopping cart. Cart icon 14204, and account icon 15204 operate as previously described. Using checkbox 14206, an administrator may specify whether an upgrade tab should be displayed in the administrator's store by marketplace module 6122. In field 14208, an administrator can enter a name for an upgrade. In field 14210, an administrator can write a description of an upgrade. In field 14212, an administrator may specify upgrade licensing information, e.g., the version or features of a license to upgrade to. In field 14214, an administrator may specify the status of an upgrade (e.g. if the upgrade is active). Using button 14216, an administrator may create the upgrade. Field 14218, displays upgrade information for an application an administrator is currently viewing.

[00643] FIG. 143 is an exemplary screenshot 14300 of a "My Store Style" page generated by the listing module 6124. The information entered here is stored by marketplace database management module 6180. An administrator can use the "My store style" page to setup the look and feel of his regular and/or syndicated store pages. For example, in one embodiment, in field 14302, an administrator can specify page background color, page headings font color, body font color, and general font. In field 14304, an administrator may write or define a page header to be displayed in the administrator's store. In field 14306, an

administrator may write or define a page footer to be displayed in the administrator's store in Marketplace module 6122.

[00644] FIG. 144 is an exemplary screenshot 14400 of a pricing grid and purchase/license link setup page generated by the listing module 6124. The information entered here is stored by marketplace database management module 6180. Pricing grid and purchase/license link setup page 14402 enables an administrator to set up items that will appear in the pricing grid on the Marketing page for a product in Marketplace module 6122. In field 14404, an administrator may enter the name of an application for which the administrator is setting up a pricing grid. In field 14406, an administrator may enter a price associated with an application. In field 14408, an administrator may provide a web address to redirect a prospective purchaser once the purchaser clicks a buy now link. In field 14410, an administrator may choose the link text. In field 14412, the administrator may enter sorting order information. In field 14414, the administrator may specify a status of a pricing grid (e.g. active). Using the create button 14416, an administrator can create the pricing grid the administrator has specified.

[00645] FIG. 145 is an exemplary screenshot 14500 of a web services "Post Data" setup page generated by the listing module 6124. The information entered here is stored by Marketplace database management module 6180. In field 14502, an administrator may specify a web address for a purchaser's purchase/license information to be posted to. Using radio buttons 14504, an administrator may specify in what format he prefers to receive a user's purchase/license information. Using the save changes button 14506, an administrator may save the changes the administrator has created. In field 14508, an administrator may provide a sample of the format in which he prefers to receive a purchase/license information.

[00646] FIG. 146 is an exemplary screenshot 14600 of a support page (e.g. an Etelos support page) wherein a user may use a Storefront (e.g. Etelos "My Storefront") to add product or services to a marketplace (e.g. an Etelos marketplace) generated by marketplace module 6122. Field 14602 displays a store listing and link; field 14604 displays a status of an application; field 14606 displays a type of an application; field 14608 displays tools available to a user. For example a user administrator can preview a store listing, number of people who have purchased from the store, and other information related to an application.

[00647] FIG. 147 is an exemplary screenshot 14700 of a support page (e.g. an Etelos support page) generated by marketplace module 6122. Using drop down menu 14702, a user

may select which application the user would like to deploy to the user's customers. Using drop down menu 14704, a user may select which licenses the user would like to upgrade. Using drop down menu 14706, a user may specify a server filter. Using the begin upgrade button 14708, a user may start an upgrade process.

5 **[00648]** FIG. 148 is an exemplary screenshot 14800 of a support page (e.g. an Etelos support page) generated by marketplace module 6122. In field 14802, a user may add new domains to the user's account. A user may view an account name in field 14804, its associated default domain name in field 14806, and its status in field 14808. A user may add a domain to an account by clicking on the button Add Domain 14810.

10 **[00649]** FIG. 149 is an exemplary screenshot 14900 of a support page (e.g. an Etelos support page) generated by marketplace module 6122. A user may use the page "create a sub domain for" 14902 to create a subdomain associated with an account. In field 14904, a user may type a domain the user would like to associate with the user's account.

15 **[00650]** FIG. 150 is an exemplary screenshot 15000 of a store product list generated by the marketplace module 6122. The information reflected here is stored in marketplace database management module 6180, including product name column 15002, store owner column 15004, company column 15006, status column 15008, and export option 15010.

20 **[00651]** FIG. 151 is an exemplary screenshot 15100 of a store listing report (e.g. a store listing 15110, "Store listing report for Etelos CRM") generated by transactions report module 6132. In one embodiment, the store listing report 15110 utilizes a table including the following rows and columns. Column 15102 displays a running total for every row; column 15104 displays a total year to date for every row; columns in the field 15106 displays a total for a month; column 15108 displays a total for financial quarter 1. Rows 15112 display various totals for store listings. Row 15114 displays accounts that are later cancelled or
25 deactivated. Rows in 15116 provide a break down between unused, dormant, dry accounts, and active accounts. Rows in 15118 provide a number of registered users and accounts registered without a registered user. Rows in 15120 provide a number of billable users, free users, and a total number of users. These columns and rows are exemplary, and in other embodiments different information might be provided in a store listing table.

30 **[00652]** FIG. 152 is an exemplary screenshot 15200 of a store listing report (e.g. a store listing 15110, "Store listing report for Etelos CRM") generated by marketplace module 6122 similar to FIG. 151.

[00653] FIG. 153 is an exemplary screenshot 15300 of a licensing transaction report (also known as a bottleneck report) generated by transactions report module 6132. This report includes all non-hosting services, non-cancelled, non-automated billing items. In one embodiment, the report includes the following information: field 15302, displays when an item was created, field 15304 displays the name-email, and company associated with an item; field 15306 displays a check mark if an item is store registered; field 15308 displays a check mark if an item is associated with a credit card; field 15310 displays an item's billing detail and associated domain; field 15312 displays a check mark if an item is checked out; field 15314 displays a check mark if an item is processed; field 15316 displays a check mark and a time stamp to indicate whether licensing terms associated with an item are accepted; field 15318 displays a check mark to indicate whether an item is downloaded; field 15320 displays a check mark if an installation for an item has started; field 15322 displays a check mark to indicate whether an item is licensed; field 15324 displays a check mark to indicate whether an installation of an item is complete; and field 15326 displays a check mark to indicate whether a user associated with an item is registered.

[00654] FIG. 154 is an exemplary screenshot 15400 of a support page (e.g. an Etelos support page) generated by Marketplace module 6122 wherein a vendor or administrator may edit an application information using the field 15402. Using the field 15404, the vendor may view or select the vendor's packaged applications available for editing. Field 15406 displays the names of a vendor's applications available for editing. Field 15408 indicates whether an application has been deployed in Marketplace module 6122. Field 15410 indicates an application type; field 15412 indicates a date where an application was created; field 15414 indicates the grouping information associated with an application; field 15416 indicates an application's version; field 15418 enables a user to archive an application using a check box.

[00655] FIG. 155 is an exemplary screenshot of a Marketplace 15500 (e.g. an Etelos Marketplace) generated by Marketplace module 6122. Tab 15502 displays the available storefronts stored by Marketplace database management module 6180. Tab 15504 displays the categories of available applications. Tab 15506 displays the most recent available applications. Tab 15508 displays a list of featured applications. Box 15510 indicates the steps a user takes to install and use the user's applications in a Marketplace.

[00656] FIG. 156 is an exemplary screenshot 15600 of a hosting and development environment (e.g. an Etelos Hosting and Dev Environment for BT Web21C SDK 15602)

wherein a user may purchase/license an application available through Marketplace module 6122. Tab 15604 displays a purchasing information 15620 including purchasing options; tab 15606 displays available features of an application; tab 15608 displays available demos associated with an application; tab 15610 displays FAQs associated with an application; tab 15612 displays available forums associated with an application; tab 15614 displays a support page associated with an application. A user may purchase/license an application by clicking the Buy Now button 15616. A user may obtain a free trial of an application by clicking on a Free Trial button 15618.

[00657] FIG. 157 is an exemplary screenshot of a Marketplace 15700 (e.g. an Etelos Marketplace) generated by Marketplace module 6122. 15702 and 15704 are examples of storefronts available through Marketplace database management module 6180.

[00658] FIG. 158 is an exemplary screenshot 15800 of a licensing page. Field 15802 includes licensing terms which a user may accept by clicking the Accept button 15804, or decline by clicking the decline button 15806.

[00659] FIG. 159 is an exemplary screenshot 15900 of a support page (e.g. an Etelos support page) generated by Marketplace module 6122 including a user's billing 15902 and transaction 15904 history, including date 15906, order ID 15908, name 15910, description 15912, and price 15914.

[00660] FIG. 160 is an exemplary screenshot 16000 of a shopping cart page. Field 16002 displays a date when a purchase/license is occurring; field 16003 provides a description; field 16004 displays a quantity of applications a user is purchasing; field 16006 displays a total cost of items in a user's shopping cart.

[00661] FIG. 161 is an exemplary screenshot 16100 of a Getting Started page for a CRM application listing (e.g. an Etelos CRM test listing page) where a user can initialize an application.

[00662] FIG. 162 is an exemplary screenshot 16200 of a Getting Started page for a developer toolkit app (e.g. an Etelos V6 developer toolkit page) where a user can initialize a development environment generated by listing module 6124 to define an application.

[00663] FIG. 163 is an exemplary screenshot 16300 of a support page (e.g. an Etelos support page) generated by marketplace module 6122. Field 16302 describes an exemplary

user idea called “Syndicated project management app”. This description is stored by the marketplace database management module 6180.

[00664] FIG 164 is an exemplary screen shot 16400 from the bottom portion of the screenshot 16300 in FIG 163. In field 16402, Add a Comment, a user may add a comment to a user idea reflected in FIG. 163. To add a comment a user enters the user’s email address in the field 16404, and the user’s name in the field 16406. The user may include a web address in the field 16408. The user can type a comment in the field 16410. The user can add a comment by clicking on the button Add A Comment 16412. These comments are stored by the marketplace database management module 6180.

[00665] FIG. 165 is an exemplary screenshot 16500 of a support page (e.g. an Etelos support page) generated by marketplace module 6122. Using button 16502, a user may start a new idea. In field 16504, a user may view most popular ideas as voted by other users.

[00666] FIG. 166 is an exemplary screenshot 16600 of an installation page generated by application deployer module 1650. Installation in progress page 16602 displays which applications are being installed, estimated install times, status, and progress for each application installation. Field 16604 displays a name of an application being installed; field 16606 displays an estimated time for an application installation; field 16608 displays an installation status; field 16610 displays a progress bar associated with an application installation.

[00667] FIG. 167 is an exemplary screenshot 16700 of an installation processing page generated by application deployment module 6150. A progress bar 16702 displays a progress of a user’s request for installation. This page is displayed to a user during installation of a software application.

[00668] FIG. 168 is an exemplary screenshot 16800 of an installation processing page similar to FIG. 166. This page shows that the installation has completed (16806) and gives a link to get started (16808) with the deployed application.

[00669] FIG. 169 is an exemplary screenshot 16900 of a support page (e.g. an Etelos support page) generated by marketplace module 6122. This page is presented to a user when configuring accounts and applications. My developer accounts 16902 enables a user to select an account name to edit its properties, or add an account manually. Drop down menu 16904 enables a user to choose the type of account to generate. Button “Build an App” 16906 enables a user to build an application for a particular account in the list of developer accounts

16902. Button “Add Domain” 16908 enables a user to add a domain associated with an application.

[00670] FIG. 170 is an exemplary screenshot 17000 of a support page (e.g. an Etelos support page) generated by marketplace module 6122. This page is displayed to a user when
5 configuring their account to run software apps. Payment information 17002 displays an account payment information. In one embodiment payment information field 17004 includes the last four digits of a credit card number associated with an account.

[00671] FIG. 171 is an exemplary screenshot 17100 of a marketplace page, generated by marketplace module 6122. This page is presented to a user searching for software
10 applications in the marketplace. Marketplace search result 17102 displays the results for an exemplary search. Support forum search results 17104 provides a link to search results in Marketplace forum stored in Marketplace database management module 6180. Knowledge base search result 17106 provides a link to search results in Marketplace knowledge base stored in Marketplace database management module 6180.

[00672] FIG. 172 is an exemplary screenshot 17200 of a marketplace shopping cart page, generated by marketplace module 6122, similar to FIG. 141. This screen shows a shopping cart with a software application for licensing, displayed to a prospective licensee.

[00673] FIG. 173 is an exemplary screenshot 17300 of a support page (e.g. an Etelos Support page) generated by Marketplace module 6122 displayed to a user, so the user can
20 edit an application or deploy an application. Field 17302 displays names of available applications; field 17304 displays the application type associated with an application; field 17306 displays the grouping information associated with an application; field 17308 displays an application version managed by the packager module 6136. A user may archive an application by checking the checkboxes in field 17310.

[00674] FIG. 174 is an exemplary screenshot 17400 of a support page (e.g. an Etelos Support page) generated by Marketplace module 6122, wherein a user can learn about an application by reviewing the information in a knowledge base (e.g. an Etelos knowledge base). This knowledge base information is managed by marketplace database management module 6180.

[00675] FIG. 175 is an exemplary screenshot 17500 of a support page (e.g. an Etelos Support page) generated by Marketplace module 6122 wherein installation of software application information associated with a user’s account is displayed. Field 17502 displays

install dates; field 17504 displays product information; field 17506 displays licensing information; field 17508 provides links to more installation information; field 17510 includes buttons for a user to cancel an installation or reinstall an installation.

[00676] FIG. 176 is an exemplary screenshot 17600 of a support page (e.g. an Etelos Support page) similar to FIG. 129 generated by marketplace module 6122.

[00677] FIG. 177 is an exemplary screenshot 17700 of a support page (e.g. an Etelos Support page) generated by marketplace module 6122. Discussion topics 17702 includes topics available in a forum page stored by marketplace database management module 6180. Column 17704 includes discussion topics threads; column 17706 displays a date on which a forum thread was last posted; column 17708 displays number of replies to a forum thread; column 17710 displays number of times a forum thread has been viewed; column 17712 enables a user to subscribe to a forum thread by checking a checkbox.

[00678] FIG. 178 is an exemplary screenshot 17800 of a support page (e.g. an Etelos Support page) generated by Marketplace module 6122 that includes information on getting started for business users. This page is displayed to a user seeking support information regarding an application.

[00679] FIG. 179 is an exemplary screenshot 17900 of a support page (e.g. an Etelos Support page) generated by Marketplace module 6122. Your installed applications 17902 enables a user to view the user's installed applications. Field 17904 includes links to other support areas e.g. getting started, account, forums, or knowledge base.

[00680] FIG. 180 is an exemplary screenshot 18000 of a support page (e.g. an Etelos Support page) generated by Marketplace module 6122. Field 18002 includes information on an exemplary web developing application called EASE made available through Marketplace module 6122. This page is accessed by an application developer/vendor using the marketplace (e.g. Etelos system).

[00681] FIG. 181 is an exemplary screenshot 18100 of a support page (e.g. an Etelos Support page) generated by Marketplace module 6122 wherein a user can modify the user's profile information. The information entered here is stored by user account access module 6172.

[00682] FIG. 182 is an exemplary screenshot 18200 of a Marketplace homepage, generated by marketplace module 6122..

[00683] FIG. 183 is an exemplary screenshot 18300 of a Marketplace page, generated by marketplace module 6122, where an application is displayed for sale (e.g. Etelos Hosting and Dev Environment for BT Web21C SDK). This page shows reviews by users and a description of software applications.

5 [00684] FIG. 184 is an exemplary screenshot 18400 of a Marketplace page, generated by marketplace module 6122, where an application is displayed for sale (e.g. Etelos Hosting and Dev Environment for BT Web21C SDK). In field Reviews 18402, a user may post a review associated with an application. In field 18404 the user may type a title for the user's review; in field 18406 the user may type the text of the user's review. Radio buttons 18408-
10 18416 enable the user to rate the user's satisfaction with different aspects of Marketplace services.

[00685] FIG. 185 is an exemplary screenshot 18500 of a Marketplace page, generated by marketplace module 6122, where an application is displayed for sale (e.g. Etelos Hosting and Dev Environment for BT Web21C SDK). A user may purchase/license an application by
15 clicking the button 18502 labeled Buy Now. A user may purchase/license a free trial of an application by clicking the button 18504 labeled Free Trial. The features tab 18506 is activated wherein information about features of the application are displayed.

[00686] FIG. 186 is an exemplary screenshot 18600 of a Marketplace page, generated by marketplace module 6122, where an application is displayed for sale (e.g. Etelos Hosting
20 and Dev Environment for BT Web21C SDK). A demos tab 18602 shows videos or screenshots of the application. This screen is displayed to a user looking for information about a software application.

[00687] FIG. 187 is an exemplary screenshot 18700 of a Marketplace page, generated by marketplace module 6122, where an application is displayed for sale (e.g. Etelos Hosting
25 and Dev Environment for BT Web21C SDK) and the FAQs tab 18702 is activated wherein user's frequently asked questions associated with the application are displayed.

[00688] FIG. 188 is an exemplary screenshot 18800 of a Marketplace page, generated by marketplace module 6122, where an application is displayed for sale (e.g. Etelos Hosting
30 and Dev Environment for BT Web21C SDK) and the forum tab 18802 is activated wherein forum discussions associated with the application are displayed.

[00689] FIG. 189 is an exemplary screenshot 18900 of a Marketplace page, generated by marketplace module 6122, where an application is displayed for sale (e.g. Etelos Hosting

and Dev Environment for BT Web21C SDK) and the support tab 18902 is activated wherein the support information associated with the application is displayed.

[00690] The foregoing description, for purpose of explanation, has been described with reference to specific embodiments. However, the illustrative discussions above are not intended to be exhaustive or to limit the invention to the precise forms disclosed. Many modifications and variations are possible in view of the above teachings. The embodiments were chosen and described in order to best explain the principles of the invention and its practical applications, to thereby enable others skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated.

[00691] **Start of: Software Marketplace and Distribution System support for Foreign Prosecution.**

[00692] 1. A computer implemented method, comprising:

[00693] at one or more servers, hosting a marketplace application:

[00694] receiving from a vendor a software application for distribution;

[00695] associating license terms with the software application;

[00696] making the software application available for distribution through the marketplace application; and

[00697] deploying the software application to one or more user accounts on one or more hosting servers, in accordance with the license terms.

[00698] 2. The computer-implemented method of claim 1, further comprising presenting a description of the license terms to a user.

[00699] 3. The computer-implemented method of claim 1, wherein the on one or more hosting servers are physically separate from the one or more servers hosting the marketplace application.

[00700] 4. The computer-implemented method of claim 1, wherein the deploying is performed after a user request to deploy the software application.

[00701] 5. The computer-implemented method of claim 1, wherein the deploying includes downloading the software to the one or more user accounts.

[00702] 6. The computer-implemented method of claim 1, wherein the deploying includes activating a flag associated with the software application in the one or more user accounts.

[00703] 7. The computer-implemented method of claim 6, where the flag enables
5 for user the software application.

[00704] 8. The computer-implemented method of claim 1, wherein the deploying includes activating a license for the software application in the one or more user accounts.

[00705] 9. The computer-implemented method of claim 1, wherein the deploying includes providing the software application for hosting by a user on hosting servers
10 associated with the user.

[00706] 10. The computer-implemented method of claim 1, wherein distribution to a user through the marketplace application includes through a website associated with the marketplace application.

[00707] 11. The computer-implemented method of claim 1, wherein distribution to
15 a user through the marketplace application includes through a client associated with the marketplace application.

[00708] 12. The computer-implemented method of claim 1, comprising packaging the software application for distribution via the marketplace application hosted by the one or more servers.

[00709] 13. The computer-implemented method of claim 12, comprising storing a
20 packaged software application in an application repository.

[00710] 14. The computer implemented method of claim 12 wherein packaging includes preparing an update to a previously deployed software application, where the update requires the previously deployed software application to function.

[00711] 15. The computer implemented method of claim 12 wherein packaging
25 includes preparing a standalone distribution for a software application.

[00712] 16. The computer implemented method of claim 15 wherein the standalone distribution includes a software application and one or more updates to the application.

[00713] 17. The computer implemented method of claims 14, 15, 16 wherein the
30 update is deployed to the one or more user accounts selected from the group consisting of a

push method, a subscription (pull) method, and a hybrid method, in accordance with the license terms.

[00714] 18. The computer-implemented method of claim 1, wherein the making available is performed by a listing manager.

5 [00715] 19. The computer implemented method of claim 18 wherein the listing manager includes a store listing for licensing the software application.

[00716] 20. The computer-implemented method of claim 1, wherein the deploying is performed by an application deployer.

10 [00717] 21. The computer-implemented method of claim 1, comprising hosting the deployed software application for the one or more user accounts.

[00718] 22. The computer implemented method of claims 1 or 19 wherein the marketplace application includes technical support for the software application.

15 [00719] 23. The computer implemented method of claims 1, 19 or 22, wherein making the software application available for distribution includes at least one selected from the group consisting of: determining a user account type, and based on the user account type, preparing to deploy a software application to the user account, or generating a new user account compatible with the software application and preparing to deploying the software application to the new user account.

[00720] 24. A server system, comprising:

20 [00721] one or more processors;

[00722] memory; and

[00723] one or more programs stored in the memory, the one or more programs comprising instructions for at one or more servers, host a marketplace application:

[00724] receiving from a vendor a software application for distribution;

25 [00725] associate license terms with the software application;

[00726] make the software application available for distribution through the marketplace application; and

[00727] deploy the software application to one or more user accounts on one or more hosting servers, in accordance with the license terms.

[00728] 25. The server system of claim 24, further comprising instructions to present a description of the license terms to a user.

[00729] 26. The server system of claim 24, wherein the on one or more hosting servers are physically separate from the one or more servers hosting the marketplace application.

[00730] 27. The server system of claim 24, wherein the instructions to deploy include instructions to deploy the software application after a user request.

[00731] 28. The server system of claim 24, wherein the instructions to deploy include instructions to download the software to the one or more user accounts.

[00732] 29. The server system of claim 24, wherein the instructions to deploy include instructions to activate a flag associated with the software application in the one or more user accounts.

[00733] 30. The server system of claim 29, where the flag enables for user the software application.

[00734] 31. The server system of claim 24, wherein the instructions to deploy include instructions to activate a license for the software application in the one or more user accounts.

[00735] 32. The server system of claim 24, wherein the instructions to deploy include instructions to provide the software application for hosting by a user on hosting servers associated with the user.

[00736] 33. The server system of claim 24, wherein the instructions to distribute to a user through the marketplace application includes instructions to distribute through a website associated with the marketplace application.

[00737] 34. The server system of claim 24, wherein the instructions to distribute to a user through the marketplace application includes instructions to distribute through a client associated with the marketplace application.

[00738] 35. The server system of claim 24, comprising instructions to package the software application for distribution via the marketplace application hosted by the one or more servers.

[00739] 36. The server system of claim 35, comprising instructions to store a packaged software application in an application repository.

[00740] 37. The server system of claim 35 wherein instructions to package include instructions to prepare an update to a previously deployed software application, where the update requires the previously deployed software application to function.

[00741] 38. The server system of claim 35 wherein instructions to package include instructions to preparing a standalone distribution for a software application.

[00742] 39. The server system of claim 38 wherein the standalone distribution includes a software application and one or more updates to the application.

[00743] 40. The server system of claims 37, 38 or 39 where the update is deployed to the one or more user accounts selected from the group consisting of a push method, a subscription (pull) method, and a hybrid method, in accordance with the license terms.

[00744] 41. The server system of claim 24, wherein the instructions to make available includes instructions for a listing manager.

[00745] 42. The server system of claim 41 wherein the listing manager includes a store listing for licensing the software application.

[00746] 43. The server system of claim 24, wherein the instructions to deploy include instructions for an application deployer.

[00747] 44. The server system of claim 24, comprising instructions to host the deployed software application for the one or more user accounts.

[00748] 45. The server system of claim 24 or 42 wherein the marketplace application includes technical support for the software application.

[00749] 46. The server system of claim 24, 42, or 45, wherein instructions to make the software application available for distribution includes instructions for at least one selected from the group consisting of: determining a user account type, and based on the user account type, preparing to deploy a software application to the user account, or generating a new user account compatible with the software application and preparing to deploying the software application to the new user account.

[00750] 47. A computer readable storage medium storing one or more programs configured for execution by a computer, the one or more programs comprising instructions to at one or more servers, hosting a marketplace application:

[00751] receive from a vendor a software application for distribution;

5 [00752] associate license terms with the software application;

[00753] make the software application available for distribution through the marketplace application; and

[00754] deploy the software application to one or more user accounts on one or more hosting servers, in accordance with the license terms.

10 [00755] 48. A computer implemented method, comprising:

[00756] at one or more marketplace servers hosting a marketplace application, in response to a request from a syndicated server to distribute a software application from the marketplace:

[00757] identifying one or more user accounts associated with the request;

15 [00758] verifying that the one or more user accounts has permission to use the software application; and

[00759] deploying the software application to the one or more user accounts, in accordance with license terms associated with the software application.

20 [00760] 49. The computer-implemented method of claim 48, wherein deploying includes presenting for deployment to a user.

[00761] 50. The computer-implemented method of claim 48, comprising making the software application available for distribution through the syndicated server.

25 [00762] 51. The computer-implemented method of claim 48 or 49, wherein deploying includes providing through an application deployer, across a network to the one or more user accounts, a software application stored at the application repository.

[00763] 52. The computer-implemented method of claim 51, wherein deploying includes selecting one of a plurality of software applications, compatible with the one or more user accounts, from the application repository.

[00764] 53. The computer-implemented method of claim 51 or 52, wherein the application repository stores software applications in a plurality of states, including at least one selected from the group consisting of a ready to deploy state, an undergoing quality assurance state, a ready to submit for quality assurance state, and an unfinished state.

5 [00765] 54. A server system, comprising:

[00766] one or more processors;

[00767] memory; and

[00768] one or more programs stored in the memory, the one or more programs comprising instructions to at one or more marketplace servers hosting a marketplace application, in response to a request from a syndicated server to distribute a software application from the marketplace:

10

[00769] identify one or more user accounts associated with the request;

[00770] verify that the one or more user accounts has permission to use the software application; and

15 [00771] deploy the software application to the one or more user accounts, in accordance with license terms associated with the software application.

[00772] 55. The server system of claim 54, wherein instructions to deploy include instructions to present for deployment to a user.

[00773] 56. The server system of claim 54, further comprising instructions to make the software application available for distribution through the syndicated server.

20

[00774] 57. The server system of claim 54 or 55, wherein instructions to deploy include instructions to provide through an application deployer, across a network to the one or more user accounts, a software application stored at the application repository.

[00775] 58. The server system of claim 57, wherein instructions to deploy include instructions to select one of a plurality of software applications, compatible with the one or more user accounts, from the application repository.

25

[00776] 59. The server system of claim 57 or 58, wherein the application repository stores software applications in a plurality of states, including at least one selected from the group consisting of a ready to deploy state, an undergoing quality assurance state, a ready to submit for quality assurance state, and an unfinished state.

30

[00777] 60. A computer readable storage medium storing one or more programs configured for execution by a computer, the one or more programs comprising instructions for at one or more marketplace servers hosting a marketplace application, in response to a request from a syndicated server to distribute a software application from the marketplace:

5 [00778] identifying one or more user accounts associated with the request;

[00779] verifying that the one or more user accounts has permission to use the software application; and

[00780] deploying the software application to the one or more user accounts, in accordance with license terms associated with the software application.

10 [00781] Data Structure for purchasing, deploying, hosting programs:

[00782] 61. A computer system, comprising:

[00783] one or more processors;

[00784] memory; and

15 [00785] one or more programs stored in the memory, the one or more programs comprising instructions for implementing:

[00786] a program module configured to provide a software application for distribution in response to an access request from a user;

[00787] a program module configured to receive and deploy the software application from the marketplace module to an account on one or more servers; and

20 [00788] a program module configured to provide at least one or more user accounts, from which the user accesses the software application.

[00789] 62. The system of claim 61, wherein the one or more user accounts receives a link to the deployed software application, wherein the link permits the one or more user accounts to access the deployed software application.

25 [00790] 63. The system of claim 61, wherein the program module configured to provide a software application for distribution is a marketplace module.

[00791] 64. The system of claim 61, wherein the program module configured to provide at least one or more user accounts is a one or more user accounts module

[00792] 65. The system of claim 61, wherein the program module configured to receive and deploy a software application to the one or more user accounts module is a deployment module.

[00793] 66. The system of claim 61, comprising an application repository storing software applications ready for distribution.

[00794] 67. The system of claim 61, comprising a licensing module configured to provide a license associated with the software application, and to ensure that the deploying is performed in accordance with the license.

[00795] 68. The system of claim 64, wherein the licensing module is configured to verify user permission to access a deployed software application prior to executing the application.

[00796] 69. The system of claim 61, comprising a billing module configured to receive payment associated with the one or more user accounts for deployment of the selected software application.

[00797] 70. The system of claim 69, wherein the billing module is configured to receive subscription payments associated with the one or more user accounts.

[00798] 71. The system of claim 69, wherein an amount of the payment varies in accordance with license terms associated with the software application.

[00799] 72. The system of claim 69, wherein the billing module is configured to receive from the user a promotional code prior to processing a payment, and the billing module is configured to process the payment based on the promotional code.

[00800] 73. The system of claim 61, comprising a listing manager module for managing marketplace content related to the software application.

[00801] 74. The system of claim 61, comprising a deployer module for accessing the software application from the application repository and deploying the software application in the hosting infrastructure module.

[00802] 75. The system of claim 61, wherein the access request comprises a selection instruction for a software application compatible with a framework associated with the one or more user accounts.

[00803] 76. The system of claim 61, wherein the user accesses the software application through a one or more user accounts registered with the hosting infrastructure module.

[00804] 77. The system of claim 61, wherein following installation, the hosting
5 infrastructure module sends an installation confirmation to the marketplace module.

[00805] 78. The computer implemented method of any one of claims 61 to 77, wherein the software application is a web-based software application, executed at one or more servers.

[00806] **End of: Software Marketplace and Distribution System support for
10 Foreign Prosecution.**

[00807]

[00808] **Start of: Software Licensing and Enforcement System support for
Foreign Prosecution.**

[00809] 1. A computer implemented method, comprising:

15 [00810] at one or more servers, hosting a marketplace application:

[00811] receiving from a vendor a software application for distribution;

[00812] generating license terms in response to a selection by the vendor from options provided by the marketplace application;

[00813] associating the license terms with the software application, and

20 [00814] making the software application available for distribution through the marketplace application, in accordance with the license terms.

[00815] 2. The computer implemented method of claim 1, further comprising deploying the software application to one or more user accounts on one or more hosting servers, in accordance with the license terms.

25 [00816] 3. The computer implemented method of claim 2, where the deploying is in response to a payment associated with the one or more user accounts.

[00817] 4. The computer implemented method of claim 1, wherein the license terms include at least one of an open source license, a closed license, a source code license, an executable license, and a repackaging license.

[00818] 5. The computer implemented method of claim 4, wherein the repacking license determines whether a user of the software application is permitted to repackage and redistribute the software application.

[00819] 6. The computer implemented method of claim 5, wherein the repacking license has an associated royalty.

[00820] 7. The computer implemented method of claim 6, wherein the associated royalty is one selected from the group consisting of a wholesale royalty, a retail royalty, and a flat fee.

[00821] 8. The computer implemented method of claims 1,, 4, or 5, wherein a license manager determines user permissions for installation, activation, and access to features of applications.

[00822] 9. The computer implemented method of claim 1 wherein the options provided by the marketplace application includes an option to use license terms supplied by the vendor.

[00823] 10. The computer implemented method of claims 1, 4, 5, 8, or 9 including displaying licensing events for a respective software application made available for distribution through the marketplace application.

[00824] 11. The computer-implemented method of claim 10, wherein displaying includes displaying the licensing events to a respective vendor associated with the respective software application.

[00825] 12. The computer-implemented method of claim 1, comprising storing the license terms in a licensing manager.

[00826] 13. The computer implemented method of claim 1, wherein a price associated with the software application is stored at a licensing manager separately from the software application.

[00827] 14. The computer implemented method of claim 13, wherein the price is dynamically adjusted by the licensing manager in response to a selection by the software vendor.

[00828] 15. The computer implemented method of claim 1, wherein at least one selected from the group consisting of access duration, features, and price, is dynamically adjusted by a licensing manager in response to a selection by the software vendor.

[00829] 16. The computer implemented method of claim 1, 2, wherein the one or
5 more user accounts are stored separately from the marketplace application.

[00830] 17. The computer implemented method of claims 1 or 16 further comprising processing a payment associated with the software application.

[00831] 18. The computer implemented method of claim 17, further comprising,
prior to processing a payment, receiving from the user a promotional code, and processing the
10 payment based on the promotional code.

[00832] 19. The computer implemented method of claim 17, comprising storing a record of the processed payment in a billing record.

[00833] 20. The computer implemented method of claim 19, comprising prior to
executing the deployed application, comparing a user identifier associated with the one or
15 more user accounts and an application id associated with the deployed application against a billing manager to verify that a valid payment has been recorded.

[00834] 21. The computer implemented methods of claims 1, 16, 17, or 20, comprising verifying that the one or more user accounts has permission to execute the deployed application, and in the event of a verification failure, warning the user.

[00835] 22. The computer implemented method of claim 21, wherein the verifying
is performed periodically, and following a plurality of verification failures, preventing the
20 one or more user accounts from executing the deployed application.

[00836] 23. The computer implemented method of claims 21 or 22, wherein the
verifying includes checking for multiple instances of the software application being
25 simultaneously executed by the one or more user accounts.

[00837] 24. The computer implemented method of claim 16, comprising
preventing access by the user to data stored at the one or more servers, upon determining that
the one or more user accounts has been disabled.

[00838] 25. The computer implemented method of claim 16, further comprising deploys a license key associated with the software application to a user account associated with a licensee of the software application.

[00839] 26. The computer implemented method of claim 25, further comprising communicating to the software vendor that the software application has been licensed by the user.

[00840] 27. The computer implemented method of claim 26, wherein the communicating is performed via an application programming interface call.

[00841] 28. A server system, comprising:

[00842] one or more processors;

[00843] memory; and

[00844] one or more programs stored in the memory, the one or more programs comprising instructions for, at one or more servers hosting a marketplace application:

[00845] receiving from a vendor a software application for distribution;

[00846] generating license terms in response to a selection by the vendor from options provided by the marketplace application;

[00847] associating the license terms with the software application, and

[00848] making the software application available for distribution through the marketplace application, in accordance with the license terms.

[00849] 29. The computer implemented method of claim 28, further comprising deploying the software application to one or more user accounts on one or more hosting servers, in accordance with the license terms.

[00850] 30. The computer implemented method of claim 29, where the deploying is in response to a payment associated with the one or more user accounts.

[00851] 31. The computer implemented method of claim 28, wherein the license terms include at least one of an open source license, a closed license, a source code license, an executable license, and a repackaging license.

[00852] 32. The computer implemented method of claim 28, wherein the repacking license determines whether a user of the software application is permitted to repackage and redistribute the software application.

[00853] 33. The computer implemented method of claim 32, wherein the repacking license has an associated royalty.

[00854] 34. The computer implemented method of claim 33, wherein the associated royalty is one selected from the group consisting of a wholesale royalty, a retail royalty, and a flat fee.

[00855] 35. The computer implemented method of claims 28, 32, or 33, wherein a license manager determines user permissions for installation, activation, and access to features of applications.

[00856] 36. The computer implemented method of claim 28 wherein the options provided by the marketplace application includes an option to use license terms supplied by the vendor.

[00857] 37. The computer implemented method of claims 28, 32, 33, 35, or 36 including displaying licensing events for a respective software application made available for distribution through the marketplace application.

[00858] 38. The computer-implemented method of claim 37, wherein displaying includes displaying the licensing events to a respective vendor associated with the respective software application.

[00859] 39. The computer-implemented method of claim 28, comprising storing the license terms in a licensing manager.

[00860] 40. The computer implemented method of claim 28, wherein a price associated with the software application is stored at a licensing manager separately from the software application.

[00861] 41. The computer implemented method of claim 40, wherein the price is dynamically adjusted by the licensing manager in response to a selection by the software vendor.

[00862] 42. The computer implemented method of claim 28, wherein at least one selected from the group consisting of access duration, features, and price, is dynamically adjusted by a licensing manager in response to a selection by the software vendor.

[00863] 43. The computer implemented method of claim 29, wherein the one or more user accounts are stored separately from the marketplace application.

[00864] 44. The computer implemented method of claims 28 or 43 further comprising processing a payment associated with the software application.

[00865] 45. The computer implemented method of claim 44, further comprising, prior to processing a payment, receiving from the user a promotional code, and processing the payment based on the promotional code.

[00866] 46. The computer implemented method of claim 44, comprising storing a record of the processed payment in a billing record.

[00867] 47. The computer implemented method of claim 44, comprising prior to executing the deployed application, comparing a user identifier associated with the one or more user accounts and an application id associated with the deployed application against a billing manager to verify that a valid payment has been recorded.

[00868] 48. The computer implemented methods of claims 28, 43, 44, and 47 comprising verifying that the one or more user accounts has permission to execute the deployed application, and in the event of a verification failure, warning the user.

[00869] 49. The computer implemented method of claim 48, wherein the verifying is performed periodically, and following a plurality of verification failures, preventing the one or more user accounts from executing the deployed application.

[00870] 50. The computer implemented method of claims 48 or 49, wherein the verifying includes checking for multiple instances of the software application being simultaneously executed by the one or more user accounts.

[00871] 51. The computer implemented method of claim 43, comprising preventing access by the user to data stored at the one or more servers, upon determining that the one or more user accounts has been disabled.

[00872] 52. The computer implemented method of claim 43, further comprising
deploys a license key associated with the software application to a user account associated
with a licensee of the software application.

[00873] 53. The computer implemented method of claim 52, further comprising
5 communicating to the software vendor that the software application has been licensed by the
user.

[00874] 54. The computer implemented method of claim 53, wherein the
communicating is performed via an application programming interface call.

[00875] 55. A computer readable storage medium storing one or more programs
10 configured for execution by a computer, the one or more programs comprising instructions
for

[00876] at one or more servers, hosting a marketplace application:

[00877] receiving from a vendor a software application for distribution;

[00878] generating license terms in response to a selection by the vendor from options
15 provided by the marketplace application;

[00879] associating the license terms with the software application, and

[00880] making the software application available for distribution through the
marketplace application, in accordance with the license terms.

[00881] 56. A computer implemented method, comprising:

[00882] 20 at one or more servers hosting a marketplace application:

[00883] making a software application available for distribution through the
marketplace application;

[00884] receiving a user request to license the software application; and

[00885] providing the software application for deployment to one or more user
25 accounts, hosted on one or more servers.

[00886] 57. The computer implemented method of claim 56, where providing the
software application for deployment includes providing the software application across a
network for deployment at one or more servers associated with the user, hosting the one or
more user accounts.

[00887] 58. The computer implemented method of claim 56, further comprising processing a selection by the user that includes adding the software application to a cart associated with the user, and checking out the cart.

[00888] 59. The computer implemented method of claim 56, comprising prior to
5 deploying, presenting a description of license terms associated with the software application to the user.

[00889] 60. The computer implemented method of claim 57, wherein the license terms are specified by a vendor associated with the software application.

[00890] 61. The computer implemented method of claim 60, wherein specifying
10 the license terms includes selecting the license terms from a plurality of options provided by a licensing engine associated with the one or more servers hosting a marketplace application.

[00891] 62. The computer implemented method of claim 57, 60, or 61 comprising validating that the request to license the software application complies with the license terms.

[00892] 63. The computer implemented method of claim 56, 57, 60, 61 or 62
15 wherein the request to license the software application includes a payment selected from the group consisting of a cash payment, a credit payment, and a prospective future payment.

[00893] 64. The computer implemented method of claim 56, 57, 60, 61, 62 or 63 wherein the software application is a web-based software application, executed at one or more servers.

[00894] 65. A server system, comprising:
20

[00895] one or more processors;

[00896] memory; and

[00897] one or more programs stored in the memory, the one or more programs comprising instructions to at one or more servers hosting a marketplace application:

[00898] make a software application available for distribution through the marketplace application;
25

[00899] receive a user request to license the software application; and

[00900] provide the software application for deployment to one or more user accounts, hosted on one or more servers.

[00901] 66. The server system of claim 65, wherein the instructions to provide the software application for deployment includes instructions to provide the software application across a network for deployment at one or more servers associated with the user, hosting the one or more user accounts.

5 [00902] 67. The server system of claim 65, further comprising instructions to process a selection by the user that includes adding the software application to a cart associated with the user, and checking out the cart.

[00903] 68. The server system of claim 65, further comprising instructions for prior to deploying, presenting a description of license terms associated with the software
10 application to the user.

[00904] 69. The server system of claim 68, further comprising instructions to receive license terms specified by a vendor associated with the software application.

[00905] 70. The server system of claim 69, wherein receiving license terms specified by a vendor include receiving a selection of license terms from a plurality of
15 options provided by a licensing engine associated with the one or more servers hosting a marketplace application.

[00906] 71. The server system of claim 66, 69, or 70 further comprising instructions to validate that the request to license the software application complies with the license terms.

20 [00907] 72. The server system of claims 65, 66, 69, 70, or 71 wherein the request to license the software application includes a payment selected from the group consisting of a cash payment, a credit payment, and a prospective future payment.

[00908] 73. The server system of claims 65, 66, 69, 70, 71, or 72 wherein the software application is a web-based software application, executed at one or more servers.

25 [00909] 74. A computer readable storage medium storing one or more programs configured for execution by a computer, the one or more programs comprising instructions for at one or more servers hosting a marketplace application:

[00910] making a software application available for distribution through the marketplace application;

30 [00911] receiving a user request to license the software application; and

[00912] providing the software application for deployment to one or more user accounts, hosted on one or more servers.

[00913] **End of: Software Licensing and Enforcement System support for Foreign Prosecution.**

What is claimed is:

1. A computer implemented method, comprising:
at one or more servers, hosting a marketplace application:
receiving from a vendor a software application for distribution;
5 associating license terms with the software application;
making the software application available for distribution through the marketplace
application; and
deploying the software application to one or more user accounts on one or more
hosting servers, in accordance with the license terms.
- 10 2. The computer-implemented method of claim 1, further comprising presenting a
description of the license terms to a user.
3. The computer-implemented method of claim 1, wherein the on one or more hosting
servers are physically separate from the one or more servers hosting the marketplace
application.
- 15 4. The computer-implemented method of claim 1, wherein the deploying is performed
after a user request to deploy the software application.
5. The computer-implemented method of claim 1, wherein the deploying includes
downloading the software to the one or more user accounts.
6. The computer-implemented method of claim 1, wherein the deploying includes
20 activating a flag associated with the software application in the one or more user accounts.
7. The computer-implemented method of claim 6, where the flag enables for user the
software application.
8. The computer-implemented method of claim 1, wherein the deploying includes
activating a license for the software application in the one or more user accounts.
- 25 9. The computer-implemented method of claim 1, wherein the deploying includes
providing the software application for hosting by a user on hosting servers associated with the
user.

10. The computer-implemented method of claim 1, wherein distribution to a user through the marketplace application includes through a website associated with the marketplace application.

11. The computer-implemented method of claim 1, wherein distribution to a user through the marketplace application includes through a client associated with the marketplace application.

12. The computer-implemented method of claim 1, comprising packaging the software application for distribution via the marketplace application hosted by the one or more servers.

13. The computer-implemented method of claim 12, comprising storing a packaged software application in an application repository.

14. The computer implemented method of claim 12 wherein packaging includes preparing an update to a previously deployed software application, where the update requires the previously deployed software application to function.

15. The computer implemented method of claim 12 wherein packaging includes preparing a standalone distribution for a software application.

16. The computer implemented method of claim 15 wherein the standalone distribution includes a software application and one or more updates to the application.

17. The computer implemented method of claim 14, wherein the update is deployed to the one or more user accounts selected from the group consisting of a push method, a subscription (pull) method, and a hybrid method, in accordance with the license terms.

18. The computer-implemented method of claim 1, wherein the making available is performed by a listing manager.

19. The computer implemented method of claim 18 wherein the listing manager includes a store listing for licensing the software application.

20. The computer-implemented method of claim 1, wherein the deploying is performed by an application deployer.

21. The computer-implemented method of claim 1, comprising hosting the deployed software application for the one or more user accounts.

22. The computer implemented method of claim 1 wherein the marketplace application includes technical support for the software application.

23. The computer implemented method of claim 1, wherein making the software application available for distribution includes at least one selected from the group consisting of: determining a user account type, and based on the user account type, preparing to deploy a software application to the user account, or generating a new user account compatible with the software application and preparing to deploying the software application to the new user account.

24. A server system, comprising:
one or more processors;
memory; and
one or more programs stored in the memory, the one or more programs comprising instructions for at one or more servers, host a marketplace application:
receiving from a vendor a software application for distribution;
associate license terms with the software application;
make the software application available for distribution through the marketplace application; and
deploy the software application to one or more user accounts on one or more hosting servers, in accordance with the license terms.

25. The server system of claim 24, further comprising instructions to present a description of the license terms to a user.

26. The server system of claim 24, wherein the on one or more hosting servers are physically separate from the one or more servers hosting the marketplace application.

27. The server system of claim 24, wherein the instructions to deploy include instructions to deploy the software application after a user request.

28. The server system of claim 24, wherein the instructions to deploy include instructions to download the software to the one or more user accounts.

29. The server system of claim 24, wherein the instructions to deploy include instructions to activate a flag associated with the software application in the one or more user accounts.

30. The server system of claim 29, where the flag enables for user the software application.

5 31. The server system of claim 24, wherein the instructions to deploy include instructions to activate a license for the software application in the one or more user accounts.

32. The server system of claim 24, wherein the instructions to deploy include instructions to provide the software application for hosting by a user on hosting servers associated with the user.

10 33. The server system of claim 24, wherein the instructions to distribute to a user through the marketplace application includes instructions to distribute through a website associated with the marketplace application.

34. The server system of claim 24, wherein the instructions to distribute to a user through the marketplace application includes instructions to distribute through a client associated with
15 the marketplace application.

35. The server system of claim 24, comprising instructions to package the software application for distribution via the marketplace application hosted by the one or more servers.

36. The server system of claim 35, comprising instructions to store a packaged software application in an application repository.

20 37. The server system of claim 35 wherein instructions to package include instructions to prepare an update to a previously deployed software application, where the update requires the previously deployed software application to function.

38. The server system of claim 35 wherein instructions to package include instructions to preparing a standalone distribution for a software application.

25 39. The server system of claim 38 wherein the standalone distribution includes a software application and one or more updates to the application.

40. The server system of claim 37, where the update is deployed to the one or more user accounts selected from the group consisting of a push method, a subscription (pull) method, and a hybrid method, in accordance with the license terms.

41. The server system of claim 24, wherein the instructions to make available includes
5 instructions for a listing manager.

42. The server system of claim 41 wherein the listing manager includes a store listing for licensing the software application.

43. The server system of claim 24, wherein the instructions to deploy include instructions for an application deployer.

10 44. The server system of claim 24, comprising instructions to host the deployed software application for the one or more user accounts.

45. The server system of claim 24 wherein the marketplace application includes technical support for the software application.

46. The server system of claim 24, wherein instructions to make the software application
15 available for distribution includes instructions for at least one selected from the group consisting of: determining a user account type, and based on the user account type, preparing to deploy a software application to the user account, or generating a new user account compatible with the software application and preparing to deploying the software application to the new user account.

20 47. A computer readable storage medium storing one or more programs configured for execution by a computer, the one or more programs comprising instructions to at one or more servers, hosting a marketplace application:

receive from a vendor a software application for distribution;

associate license terms with the software application;

25 make the software application available for distribution through the marketplace application; and

deploy the software application to one or more user accounts on one or more hosting servers, in accordance with the license terms.

48. A computer implemented method, comprising:
at one or more marketplace servers hosting a marketplace application, in response to a
request from a syndicated server to distribute a software application from the
marketplace:

5 identifying one or more user accounts associated with the request;
verifying that the one or more user accounts has permission to use the software
application; and
deploying the software application to the one or more user accounts, in accordance
with license terms associated with the software application.

10 49. The computer-implemented method of claim 48, wherein deploying includes
presenting for deployment to a user.

50. The computer-implemented method of claim 48, comprising making the software
application available for distribution through the syndicated server.

15 51. The computer-implemented method of claim 48 wherein deploying includes
providing through an application deployer, across a network to the one or more user accounts,
a software application stored at the application repository.

52. The computer-implemented method of claim 51, wherein deploying includes selecting
one of a plurality of software applications, compatible with the one or more user accounts,
from the application repository.

20 53. The computer-implemented method of claim 51, wherein the application repository
stores software applications in a plurality of states, including at least one selected from the
group consisting of a ready to deploy state, an undergoing quality assurance state, a ready to
submit for quality assurance state, and an unfinished state.

25 54. A server system, comprising:
one or more processors;
memory; and
one or more programs stored in the memory, the one or more programs comprising
instructions to at one or more marketplace servers hosting a marketplace
application, in response to a request from a syndicated server to distribute a
30 software application from the marketplace:

identify one or more user accounts associated with the request;
verify that the one or more user accounts has permission to use the software
application; and
deploy the software application to the one or more user accounts, in accordance with
5 license terms associated with the software application.

55. The server system of claim 54, wherein instructions to deploy include instructions to present for deployment to a user.

56. The server system of claim 54, further comprising instructions to make the software application available for distribution through the syndicated server.

10 57. The server system of claim 54, wherein instructions to deploy include instructions to provide through an application deployer, across a network to the one or more user accounts, a software application stored at the application repository.

58. The server system of claim 57, wherein instructions to deploy include instructions to select one of a plurality of software applications, compatible with the one or more user
15 accounts, from the application repository.

59. The server system of claim 57, wherein the application repository stores software applications in a plurality of states, including at least one selected from the group consisting of a ready to deploy state, an undergoing quality assurance state, a ready to submit for quality assurance state, and an unfinished state.

20 60. A computer readable storage medium storing one or more programs configured for execution by a computer, the one or more programs comprising instructions for at one or more marketplace servers hosting a marketplace application, in response to a request from a syndicated server to distribute a software application from the marketplace:

25 identifying one or more user accounts associated with the request;
verifying that the one or more user accounts has permission to use the software application; and
deploying the software application to the one or more user accounts, in accordance with license terms associated with the software application.

61. A computer system, comprising:

one or more processors;

memory; and

one or more programs stored in the memory, the one or more programs comprising instructions for implementing:

5 a program module configured to provide a software application for distribution in response to an access request from a user;

 a program module configured to receive and deploy the software application from the marketplace module to an account on one or more servers; and

10 a program module configured to provide at least one or more user accounts, from which the user accesses the software application.

62. The system of claim 61, wherein the one or more user accounts receives a link to the deployed software application, wherein the link permits the one or more user accounts to access the deployed software application.

15 63. The system of claim 61, wherein the program module configured to provide a software application for distribution is a marketplace module.

64. The system of claim 61, wherein the program module configured to provide at least one or more user accounts is a one or more user accounts module

20 65. The system of claim 61, wherein the program module configured to receive and deploy a software application to the one or more user accounts module is a deployment module.

66. The system of claim 61, further comprising an application repository storing software applications ready for distribution.

25 67. The system of claim 61, further comprising a licensing module configured to provide a license associated with the software application, and to ensure that the deploying is performed in accordance with the license.

68. The system of claim 64, wherein the licensing module is configured to verify user permission to access a deployed software application prior to executing the application.

69. The system of claim 61, further comprising a billing module configured to receive payment associated with the one or more user accounts for deployment of the selected software application.

70. The system of claim 69, wherein the billing module is configured to receive
5 subscription payments associated with the one or more user accounts.

71. The system of claim 69, wherein an amount of the payment varies in accordance with license terms associated with the software application.

72. The system of claim 69, wherein the billing module is configured to receive from the user a promotional code prior to processing a payment, and the billing module is configured
10 to process the payment based on the promotional code.

73. The system of claim 61, further comprising a listing manager module for managing marketplace content related to the software application.

74. The system of claim 61, further comprising a deployer module for accessing the software application from the application repository and deploying the software application
15 in the hosting infrastructure module.

75. The system of claim 61, wherein the access request comprises a selection instruction for a software application compatible with a framework associated with the one or more user accounts.

76. The system of claim 61, wherein the user accesses the software application through a
20 one or more user accounts registered with the hosting infrastructure module.

77. The system of claim 61, wherein following installation, the hosting infrastructure module sends an installation confirmation to the marketplace module.

78. The system of claim 61, wherein the software application is a web-based software application, executed at one or more servers.

25 79. A computer implemented method, comprising:
at one or more servers, hosting a marketplace application:
receiving from a vendor a software application for distribution;

generating license terms in response to a selection by the vendor from options
provided by the marketplace application;
associating the license terms with the software application, and
making the software application available for distribution through the marketplace
application, in accordance with the license terms.

80. The computer implemented method of claim 79, further comprising deploying the software application to one or more user accounts on one or more hosting servers, in accordance with the license terms.

81. The computer implemented method of claim 80, where the deploying is in response to a payment associated with the one or more user accounts.

82. The computer implemented method of claim 79, wherein the license terms include at least one of an open source license, a closed license, a source code license, an executable license, and a repacking license.

83. The computer implemented method of claim 82, wherein the repacking license determines whether a user of the software application is permitted to repackage and redistribute the software application.

84. The computer implemented method of claim 83, wherein the repacking license has an associated royalty.

85. The computer implemented method of claim 84, wherein the associated royalty is one selected from the group consisting of a wholesale royalty, a retail royalty, and a flat fee.

86. The computer implemented method of claim 79, wherein a license manager determines user permissions for installation, activation, and access to features of applications.

87. The computer implemented method of claim 79 wherein the options provided by the marketplace application includes an option to use license terms supplied by the vendor.

88. The computer implemented method of claim 79, including displaying licensing events for a respective software application made available for distribution through the marketplace application.

89. The computer-implemented method of claim 88, wherein displaying includes sending for display the licensing events to a respective vendor associated with the respective software application.

5 90. The computer-implemented method of claim 79, further comprising storing the license terms in a licensing manager.

91. The computer implemented method of claim 79, wherein a price associated with the software application is stored at a licensing manager separately from the software application.

92. The computer implemented method of claim 91, wherein the price is dynamically adjusted by the licensing manager in response to a selection by the software vendor.

10 93. The computer implemented method of claim 79, wherein at least one selected from the group consisting of access duration, features, and price, is dynamically adjusted by a licensing manager in response to a selection by the software vendor.

94. The computer implemented method of claim 79, wherein the one or more user accounts are stored separately from the marketplace application.

15 95. The computer implemented method of claim 79 further comprising processing a payment associated with the software application.

96. The computer implemented method of claim 95, further comprising, prior to processing a payment, receiving from the user a promotional code, and processing the payment based on the promotional code.

20 97. The computer implemented method of claim 95, further comprising storing a record of the processed payment in a billing record.

98. The computer implemented method of claim 97, further comprising prior to executing the deployed application, comparing a user identifier associated with the one or more user accounts and an application id associated with the deployed application against a billing
25 manager to verify that a valid payment has been recorded.

99. The computer implemented methods of claim 79, further comprising verifying that the one or more user accounts has permission to execute the deployed application, and in the event of a verification failure, warning the user.

100. The computer implemented method of claim 99, wherein the verifying is performed periodically, and following a plurality of verification failures, preventing the one or more user accounts from executing the deployed application.

101. The computer implemented method of claims 99, wherein the verifying includes checking for multiple instances of the software application being simultaneously executed by the one or more user accounts.

102. The computer implemented method of claim 94, further comprising preventing access by the user to data stored at the one or more servers, upon determining that the one or more user accounts has been disabled.

103. The computer implemented method of claim 94, further comprising deploying a license key associated with the software application to a user account associated with a licensee of the software application.

104. The computer implemented method of claim 103, further comprising communicating to the software vendor that the software application has been licensed by the user.

105. The computer implemented method of claim 104, wherein the communicating is performed via an application programming interface call.

106. A server system, comprising:
one or more processors;
memory; and
one or more programs stored in the memory, the one or more programs comprising instructions for, at one or more servers hosting a marketplace application:
receiving from a vendor a software application for distribution;
generating license terms in response to a selection by the vendor from options provided by the marketplace application;
associating the license terms with the software application, and

making the software application available for distribution through the marketplace application, in accordance with the license terms.

107. The server system of claim 106, further comprising instructions for deploying the software application to one or more user accounts on one or more hosting servers, in accordance with the license terms.

108. The server system of claim 107, where the deploying is in response to a payment associated with the one or more user accounts.

109. The server system of claim 106, wherein the license terms include at least one of an open source license, a closed license, a source code license, an executable license, and a repacking license.

110. The server system of claim 106, wherein the repacking license determines whether a user of the software application is permitted to repackage and redistribute the software application.

111. The server system of claim 110, wherein the repacking license has an associated royalty.

112. The server system of claim 111, wherein the associated royalty is one selected from the group consisting of a wholesale royalty, a retail royalty, and a flat fee.

113. The server system of claim 106, further comprising instructions for license manager to determines user permissions for installation, activation, and access to features of applications.

114. The server system of claim 106 wherein the options provided by the marketplace application includes an option to use license terms supplied by the vendor.

115. The server system of claim 106, including instructions to display licensing events for a respective software application made available for distribution through the marketplace application.

116. The server system of claim 115, wherein displaying includes sending for display the licensing events to a respective vendor associated with the respective software application.

117. The server system of claim 106, further comprising instructions to store the license terms in a licensing manager.

118. The server system of claim 106, wherein a price associated with the software application is stored at a licensing manager separately from the software application.

5 119. The server system of claim 118, wherein the price is dynamically adjusted by the licensing manager in response to a selection by the software vendor.

120. The server system of claim 106, wherein at least one selected from the group consisting of access duration, features, and price, is dynamically adjusted by a licensing manager in response to a selection by the software vendor.

10 121. The server system of claim 107, wherein the one or more user accounts are stored separately from the marketplace application.

122. The server system of claim 106 further comprising instructions to process a payment associated with the software application.

15 123. The server system of claim 122, further comprising instructions for, prior to processing a payment, receiving from the user a promotional code, and processing the payment based on the promotional code.

124. The server system of claim 122, further comprising instructions for storing a record of the processed payment in a billing record.

20 125. The server system of claim 122, further comprising instructions for prior to executing the deployed application, comparing a user identifier associated with the one or more user accounts and an application id associated with the deployed application against a billing manager to verify that a valid payment has been recorded.

25 126. The server system of claim 106, further comprising instructions for verifying that the one or more user accounts has permission to execute the deployed application, and in the event of a verification failure, warning the user.

127. The server system of claim 126, wherein the verifying is performed periodically, and following a plurality of verification failures, preventing the one or more user accounts from executing the deployed application.

128. The server system of claim 126, wherein the verifying includes checking for multiple instances of the software application being simultaneously executed by the one or more user accounts.

129. The server system of claim 121, further comprising instructions for preventing access by the user to data stored at the one or more servers, upon determining that the one or more user accounts has been disabled.

130. The server system of claim 121, further comprising instructions for deploying a license key associated with the software application to a user account associated with a licensee of the software application.

131. The server system of claim 130, further comprising instructions for communicating to the software vendor that the software application has been licensed by the user.

132. The server system of claim 131, wherein the communicating is performed via an application programming interface call.

133. A computer readable storage medium storing one or more programs configured for execution by a computer, the one or more programs comprising instructions for

at one or more servers, hosting a marketplace application:

receiving from a vendor a software application for distribution;

generating license terms in response to a selection by the vendor from options

provided by the marketplace application;

associating the license terms with the software application, and

making the software application available for distribution through the marketplace

application, in accordance with the license terms.

134. A computer implemented method, comprising:

at one or more servers hosting a marketplace application:

making a software application available for distribution through the marketplace application;

receiving a user request to license the software application; and
providing the software application for deployment to one or more user accounts,
hosted on one or more servers.

135. The computer implemented method of claim 134, where providing the software
application for deployment includes providing the software application across a network for
deployment at one or more servers associated with the user, hosting the one or more user
accounts.

136. The computer implemented method of claim 134, further comprising processing a
selection by the user that includes adding the software application to a cart associated with
the user, and checking out the cart.

137. The computer implemented method of claim 134, further comprising prior to
deploying, presenting a description of license terms associated with the software application
to the user.

138. The computer implemented method of claim 135, wherein the license terms are
specified by a vendor associated with the software application.

139. The computer implemented method of claim 138, wherein specifying the license
terms includes selecting the license terms from a plurality of options provided by a licensing
engine associated with the one or more servers hosting a marketplace application.

140. The computer implemented method of claim 135, further comprising validating that
the request to license the software application complies with the license terms.

141. The computer implemented method of claim 134, wherein the request to license the
software application includes a payment selected from the group consisting of a cash
payment, a credit payment, and a prospective future payment.

142. The computer implemented method of claim 134, wherein the software application is
a web-based software application, executed at one or more servers.

143. A server system, comprising:
one or more processors;
memory; and

one or more programs stored in the memory, the one or more programs comprising instructions to at one or more servers hosting a marketplace application: make a software application available for distribution through the marketplace application;

5 receive a user request to license the software application; and
provide the software application for deployment to one or more user accounts, hosted on one or more servers.

144. The server system of claim 143, wherein the instructions to provide the software application for deployment includes instructions to provide the software application across a
10 network for deployment at one or more servers associated with the user, hosting the one or more user accounts.

145. The server system of claim 143, further comprising instructions to process a selection by the user that includes adding the software application to a cart associated with the user, and checking out the cart.

15 146. The server system of claim 143, further comprising instructions for prior to deploying, presenting a description of license terms associated with the software application to the user.

147. The server system of claim 146, further comprising instructions to receive license terms specified by a vendor associated with the software application.

148. The server system of claim 147, wherein instructions to receive license terms
20 specified by a vendor include instructions to receive a selection of license terms from a plurality of options provided by a licensing engine associated with the one or more servers hosting a marketplace application.

149. The server system of claim 144, further comprising instructions to validate that the request to license the software application complies with the license terms.

25 150. The server system of claim 143, wherein the request to license the software application includes a payment selected from the group consisting of a cash payment, a credit payment, and a prospective future payment.

151. The server system of claim 143, wherein the software application is a web-based software application, executed at one or more servers.

152. A computer readable storage medium storing one or more programs configured for execution by a computer, the one or more programs comprising instructions for at one or more servers hosting a marketplace application:

making a software application available for distribution through the marketplace

5 application;

receiving a user request to license the software application; and

providing the software application for deployment to one or more user accounts,
hosted on one or more servers.

153. A method, comprising:

10 detecting changes made to a first data set in a plurality of data sets; and

synchronizing at least a first subset of the changes to a data framework that facilitates
data synchronization between the plurality of data sets.

154. The method of claim 153, wherein synchronizing the at least the first subset of the changes to the data framework includes:

15 determining a mapping between data fields in a data structure for the first data set and
data fields in a data structure for the data framework; and

synchronizing the data fields in the data structure for the first data set corresponding
to the at least the first subset of the changes with the data fields in the data
structure for the data framework based on the mapping.

20 155. The method of claim 154, wherein synchronizing the data fields includes translating
values for the data fields in the data structure for the first data set to corresponding values for
the data fields in the data structure for the data framework based on translation rules for the
first data set.

156. The method of claim 153, including synchronizing at least a second subset of the
25 synchronized changes from the data framework to a second data set in the plurality of data
sets.

157. The method of claim 156, wherein synchronizing the at least the second subset of the
synchronized changes from the data framework to a second data set in the plurality of data
sets includes:

determining a mapping between data fields in a data structure for the second data set and data fields in a data structure for the data framework;
synchronizing the data fields in the data structure for the second data set with the data fields in the data structure for the data framework based on the mapping.

5 158. The method of claim 157, wherein synchronizing the data fields includes translating values for the data fields in the data framework to corresponding values for the data fields in the second data set based on translation rules for the second data set.

159. The method of claim 153, including synchronizing at least a third subset of the synchronized changes from the data framework to a second data framework.

10 160. The method of claim 153, wherein a data set and a data framework includes one or more of:

one or more databases;
one or more structured files; and
one or more documents.

15 161. The method of claim 160, wherein a structured file includes one of:
an extensible markup language (XML) file;
a hypertext markup language (HTML) file; and
a file including values separated by a delimiter.

20 162. The method of claim 161, wherein the delimiter can include one of:
a comma; and
a tab break.

25 163. The method of claim 160, wherein a document includes one of:
a web document;
a word processor document;
a spreadsheet document;
a presentation document; and
an electronic mail message.

164. The method of claim 153, wherein a change to a data set includes one or more of:
an update of data in the data set

an addition of data to the data set; and
a deletion of data from the data set.

165. A computer readable storage medium storing one or more programs configured for execution by a computer, the one or more programs comprising instructions for:

5 detecting changes made to a first data set in a plurality of data sets; and
synchronizing at least a first subset of the changes to a data framework that facilitates data synchronization between the plurality of data sets.

166. The computer readable storage medium of claim 165, wherein synchronizing the at least the first subset of the changes to the data framework includes:

10 determining a mapping between data fields in a data structure for the first data set and data fields in a data structure for the data framework; and
synchronizing the data fields in the data structure for the first data set corresponding to the at least the first subset of the changes with the data fields in the data structure for the data framework based on the mapping.

15 167. The computer readable storage medium of claim 166, wherein synchronizing the data fields includes translating values for the data fields in the data structure for the first data set to corresponding values for the data fields in the data structure for the data framework based on translation rules for the first data set.

20 168. The computer readable storage medium of claim 165, including synchronizing at least a second subset of the synchronized changes from the data framework to a second data set in the plurality of data sets.

169. The computer readable storage medium of claim of claim 168, wherein synchronizing the at least the second subset of the synchronized changes from the data framework to a second data set in the plurality of data sets includes:

25 determining a mapping between data fields in a data structure for the second data set and data fields in a data structure for the data framework;
synchronizing the data fields in the data structure for the second data set with the data fields in the data structure for the data framework based on the mapping.

30 170. The computer readable storage medium of claim of claim 169, wherein synchronizing the data fields includes translating values for the data fields in the data framework to

corresponding values for the data fields in the second data set based on translation rules for the second data set.

171. The computer readable storage medium of claim of claim 165, including
synchronizing at least a third subset of the synchronized changes from the data framework to
5 a second data framework.

172. A system, comprising:
one or more processors;
memory; and
one or more programs stored in the memory, the one or more programs comprising
10 instructions to:
detect changes made to a first data set in a plurality of data sets; and
synchronize at least a first subset of the changes to a data framework that facilitates
data synchronization between the plurality of data sets.

173. The system of claim 172, wherein the instructions to synchronize the at least the first
15 subset of the changes to the data framework includes instructions to:
determine a mapping between data fields in a data structure for the first data set and
data fields in a data structure for the data framework; and
synchronize the data fields in the data structure for the first data set corresponding to
the at least the first subset of the changes with the data fields in the data
20 structure for the data framework based on the mapping.

174. The system of claim 173, wherein the instructions to synchronize the data fields
includes instructions to translate values for the data fields in the data structure for the first
data set to corresponding values for the data fields in the data structure for the data
framework based on translation rules for the first data set.

25 175. The system of claim 172, including instructions to synchronize at least a second
subset of the synchronized changes from the data framework to a second data set in the
plurality of data sets.

176. The system of claim 175, wherein the instructions to synchronize the at least the
second subset of the synchronized changes from the data framework to a second data set in
30 the plurality of data sets includes instructions to:

determine a mapping between data fields in a data structure for the second data set
and data fields in a data structure for the data framework;
synchronize the data fields in the data structure for the second data set with the data
fields in the data structure for the data framework based on the mapping.

5 177. The system of claim 176, wherein the instructions to synchronize the data fields
includes instructions to translate values for the data fields in the data framework to
corresponding values for the data fields in the second data set based on translation rules for
the second data set.

10 178. The system of claim 172, including instructions to synchronize at least a third subset
of the synchronized changes from the data framework to a second data framework.

179. A method, comprising:
identifying a first data set in a plurality of data sets that is to be synchronized with a
data framework;
determining a mapping between one or more data fields in a data structure for the first
15 data set and one or more data fields in a data structure for the data framework;
and
generating synchronization rules for the first data set based on the determined
mapping.

20 180. The method of claim 179, including:
identifying at least a subset of the one or more data fields for the first data set that
include data values that are to be translated to corresponding data values for
the data fields for the data framework; and
generating translation rules for translating data values for the at least the subset of the
data fields for the first data set to corresponding data values for the data fields
25 for the data framework.

181. The method of claim 179, wherein the first data set and the data framework includes
one or more of:

one or more databases;
one or more structured files; and
30 one or more documents.

182. The method of claim 179,
wherein the first data set and the data framework include one or more databases; and
wherein the mapping includes a mapping between columns of one or more database
tables for the first data set and columns of one or more database tables for the
data framework.

183. A computer readable storage medium storing one or more programs configured for
execution by a computer, the one or more programs comprising instructions for:
identifying a first data set in a plurality of data sets that is to be synchronized with a
data framework;
determining a mapping between one or more data fields in a data structure for the first
data set and one or more data fields in a data structure for the data framework;
and
generating synchronization rules for the first data set based on the determined
mapping.

184. The computer readable storage medium of claim 183, including:
identifying at least a subset of the one or more data fields for the first data set that
include data values that are to be translated to corresponding data values for
the data fields for the data framework; and
generating translation rules for translating data values for the at least the subset of the
data fields for the first data set to corresponding data values for the data fields
for the data framework.

185. A system, comprising:
one or more processors;
memory; and
one or more programs stored in the memory, the one or more programs comprising
instructions to:
identify a first data set in a plurality of data sets that is to be synchronized with a data
framework;
determine a mapping between one or more data fields in a data structure for the first
data set and one or more data fields in a data structure for the data framework;
and
generate synchronization rules for the first data set based on the determined mapping.

186. The system of claim 185, including instructions to:
identify at least a subset of the one or more data fields for the first data set that include
data values that are to be translated to corresponding data values for the data
fields for the data framework; and
5 generate translation rules for translating data values for the at least the subset of the
data fields for the first data set to corresponding data values for the data fields
for the data framework.
187. A method, comprising:
detecting changes made to a first data set for a first web-based application in an
10 account;
identifying at least a second data set for a second web-based application in the
account, wherein the second data set includes at least a subset of the data
included in the first data set; and
applying synchronization rules to synchronize the second data set with the first data
15 set.
188. The method of claim 187, wherein the synchronization rules include:
one or more rules to synchronize at least a subset of data fields in a data structure for
the first data set with data fields in a data structure for a data framework; and
one or more rules to synchronize at least a subset of the data fields in the data
20 structure for the data framework with data fields in a data structure for the
second data set.
189. The method of claim 187, wherein the account includes a plurality of data sets and
corresponding web-based applications.
190. The method of claim 187, wherein the synchronization rules include one or more of:
25 a mapping between fields in the first data set and fields in the second data set; and
rules for resolving data conflicts between the first data set and the second data set.
191. The method of claim 187, wherein a data set includes one or more of:
one or more databases;
one or more structured files; and
30 one or more documents.

192. The method of claim 191, wherein a structured file includes one of:
an extensible markup language (XML) file;
a hypertext markup language (HTML) file; and
a file including values separated by a delimiter.

5 193. The method of claim 192, wherein the delimiter can include one of:
a comma; and
a tab break.

194. The method of claim 191, wherein a document includes one of:
a web document;
10 a word processor document;
a spreadsheet document;
a presentation document; and
an electronic mail message.

195. The method of claim 187, wherein a change to a data set includes one or more of:
15 an update of data in the data set
an addition of data to the data set; and
a deletion of data from the data set.

196. A computer readable storage medium storing one or more programs configured for
execution by a computer, the one or more programs comprising instructions for:
20 detecting changes made to a first data set for a first web-based application in an
account;
identifying at least a second data set for a second web-based application in the
account, wherein the second data set includes at least a subset of the data
included in the first data set; and
25 applying synchronization rules to synchronize the second data set with the first data
set.

197. The computer readable storage medium of claim 196, wherein the synchronization
rules include:
one or more rules to synchronize at least a subset of data fields in a data structure for
30 the first data set with data fields in a data structure for a data framework; and

one or more rules to synchronize at least a subset of the data fields in the data structure for the data framework with data fields in a data structure for the second data set.

198. The computer readable storage medium of claim 196, wherein the account includes a plurality of data sets and corresponding web-based applications.

199. The computer readable storage medium of claim 196, wherein the synchronization rules include one or more of:

a mapping between fields in the first data set and fields in the second data set; and rules for resolving data conflicts between the first data set and the second data set.

200. The computer readable storage medium of claim 196, wherein a change to a data set includes one or more of:

an update of data in the data set
an addition of data to the data set; and
a deletion of data from the data set.

201. A system, comprising:

one or more processors;
memory; and

one or more programs stored in the memory, the one or more programs comprising instructions to:

detect changes made to a first data set for a first web-based application in an account;
identify at least a second data set for a second web-based application in the account, wherein the second data set includes at least a subset of the data included in the first data set; and
apply synchronization rules to synchronize the second data set with the first data set.

202. The system of claim 201, wherein the synchronization rules include:

one or more rules to synchronize at least a subset of data fields in a data structure for the first data set with data fields in a data structure for a data framework; and
one or more rules to synchronize at least a subset of the data fields in the data structure for the data framework with data fields in a data structure for the second data set.

203. The system of claim 201, wherein the account includes a plurality of data sets and corresponding web-based applications.

204. The system of claim 201, wherein the synchronization rules include one or more of:
a mapping between fields in the first data set and fields in the second data set; and
rules for resolving data conflicts between the first data set and the second data set.

205. The system of claim 201, wherein a change to a data set includes one or more of:
an update of data in the data set
an addition of data to the data set; and
a deletion of data from the data set.

206. A computer system, comprising:
one or more processors;
memory; and
one or more programs and data structures stored in the memory, the one or more
programs and data structures including:
an application data structure configured to store data and program files for a
web-based application;
an account data structure configured to store one or more instances of the
application data structure for an account, wherein an instance of the
application data structure corresponds to an instance of a web-based
application; and
a synchronization module configured to synchronize data between web-based
applications within an account based on synchronization rules.

207. The computer system of claim 206, including a framework configured to store one or more instances of the account data structure, wherein an instance of the account data structure corresponds to an account.

208. The computer system of claim 206, including a synchronization database configured to store the synchronization rules for synchronizing data between web-based applications.

209. The computer system of claim 206, including a framework database configured to store information about accounts and web-based applications for the accounts in the framework.

210. The computer system of claim 206, wherein an instance of the application data structure includes one or more of:

- an application database configured to store data for a web-based application;
- files for the web-based application; and
- 5 a version control repository for the web-based application.

211. The computer system of claim 206, wherein an instance of the account data structure includes one or more of:

- one or more support software packages; and
- an account database configured to store metadata for accounts.

10 212. The computer system of claim 211, wherein the one or more support software packages include one or more of:

- utilities;
- tools; and
- shared libraries.

15 213. The computer system of claim 206, including:

- a web server;
- one or more scripting languages; and
- an operating system.

20 214. A computer readable storage medium storing one or more programs and data structures configured for execution by a computer,

the one or more data structures comprising:

- an application data structure configured to store data and program files for a web-based application;
- an account data structure configured to store one or more instances of the application data structure for an account, wherein an instance of the application data structure corresponds to an instance of a web-based application; and
- 25

the one or more programs comprising instructions for synchronizing data between web-based applications within an account based on synchronization rules.

215. The computer readable storage medium of claim 214, including a framework configured to store one or more instances of the account data structure, wherein an instance of the account data structure corresponds to an account.

216. The computer readable storage medium of claim 214, including a synchronization database configured to store the synchronization rules for synchronizing data between web-based applications.

217. The computer readable storage medium of claim 214, including a framework database configured to store information about accounts and web-based applications for the accounts in the framework.

218. The computer readable storage medium of claim 214, wherein an instance of the application data structure includes one or more of:

- an application database configured to store data for a web-based application;
- files for the web-based application; and
- a version control repository for the web-based application.

219. The computer readable storage medium of claim 214, wherein an instance of the account data structure includes one or more of:

- one or more support software packages; and
- an account database configured to store metadata for accounts.

220. The computer readable storage medium of claim 219, wherein the one or more support software packages include one or more of:

- utilities;
- tools; and
- shared libraries.

221. The computer readable storage medium of claim 214, including:

- a web server;
- one or more scripting languages; and
- an operating system.

222. A method, comprising:

providing on a computer system a local software stack configured to provide local web services for dynamic, web-based applications that are executed on the computer system when it is offline; and

when the computer system is offline, executing on the computer system a first dynamic, web-based application using the web services provided by the local software stack, such that functionality of the first dynamic, web-based application when the computer system is offline is substantially similar to functionality of the first dynamic, web-based application when the computer system is online.

10 223. The method of claim 222, wherein the first web-based application is a database-driven web-based application.

224. The method of claim 222, wherein architecture of the first dynamic, web-based application is not modified to provide the functionality when the computer system is offline.

225. The method of claim 222, further comprising:

15 in response to detecting a network connection with an application server,
synchronizing with the application server changes in information associated with the first dynamic, web-based application due to its offline execution.

226. The method of claim 222, wherein a user of the computer system initiates execution of the first, dynamic web-based application when the computer system is offline by directing
20 a web browser on the first computer system to a specified universal resource locator (URL) that is associated with the computer system instead of a remote application server.

227. The method of claim 226, wherein the specified URL is associated with the computer system through a modified hosts file on the first computer system.

228. The method of claim 226, wherein the specified URL is associated with the computer
25 system through a modified hosts file on the first computer system.

229. The method of claim 226, wherein the specified URL is associated with the first computer system through a dynamic naming server (DNS) record on the first computer system.

230. The method of claim 222, wherein the local software stack comprises: a web server responsive to browser-issued commands, a database management system, and a dynamic scripting language.

231. The method of claim 230, wherein the web server is Apache web server, the database management system is MySQL, and the dynamic scripting language is at least one of PHP, Python, Perl, or Ruby.

232. The method of claim 230, wherein the first dynamic, web-based application communicates with the local software stack using TCP/IP.

233. A system, comprising:

one or more processors;

memory; and

one or more programs stored in the memory, the one or more programs comprising instructions to:

provide on a computer system a local software stack configured to provide local web services for dynamic, web-based applications that are executed on the computer system when it is offline; and

when the computer system is offline, execute on the computer system a first dynamic, web-based application using the web services provided by the local software stack, such that functionality of the first dynamic, web-based application when the computer system is offline is substantially similar to functionality of the first dynamic, web-based application when the computer system is online.

234. A computer readable storage medium storing one or more programs for execution by a computer, the one or more programs comprising instructions to:

provide on a computer system a local software stack configured to provide local web services for dynamic, web-based applications that are executed on the computer system when it is offline; and

when the computer system is offline, execute on the computer system a first dynamic, web-based application using the web services provided by the local software stack, such that functionality of the first dynamic, web-based application when the computer system is offline is substantially similar to functionality of the first dynamic, web-based application when the computer system is online.

235. A method, comprising:

using a dynamic, web-based application configured to interact over a network
connection with an application server to provide desired functionality;
disconnecting the network connection from the application server; and
5 continuing to use the web-based application while still providing the desired
functionality.

236. The method of claim 235, wherein the dynamic, web-based application is a database-driven web-based application.

237. The method of claim 235, wherein architecture of the dynamic, web-based
10 application is not modified to provide the functionality when the computer system is offline.

238. The method of claim 235, further comprising:

in response to detecting a network connection with an application server,
synchronizing with the application server changes in information associated
with the dynamic, web-based application due to its offline execution.

15 239. The method of claim 235, wherein a user of the computer system initiates execution
of the dynamic, web-based application when the computer system is offline by directing a
web browser on the first computer system to a specified universal resource locator (URL) that
is associated with the computer system instead of a remote application server.

240. A system, comprising:

20 one or more processors;
memory; and
one or more programs stored in the memory, the one or more programs comprising
instructions to:
use a dynamic, web-based application configured to interact over a network
25 connection with an application server to provide desired functionality;
disconnect the network connection from the application server; and
continue to use the web-based application while still providing the desired
functionality.

241. A computer readable storage medium storing one or more programs for execution by
30 a computer, the one or more programs comprising instructions to:

use a dynamic, web-based application configured to interact over a network
connection with an application server to provide desired functionality;
disconnect the network connection from the application server; and
continue to use the web-based application while still providing the desired
functionality.

242. A method, comprising:

for a web-based application that is not designed to be used while a first computer
system on which the web-based application executes does not have a network
connection to an application server,
making changes to the web-based application while the first computer system
does not have a network connection to the application server; and
tracking the changes made to the web-based application while the first
computer system is disconnected from the application server;
when the network connection between the first computer system and the application
server is reestablished,
synchronizing changes for the web-based application made on the first
computer system with the web-based application on the application
server; and
synchronizing changes for the web-based application made on the application
server with the web-based application on the first computer system.

243. The method of claim 242, further comprising tracking the date and time when the
changes to the web-based application were made.

244. The method of claim 242, wherein the changes are synchronized when a network
connection between the first computer system and the application server is reestablished.

245. The method of claim 242 wherein the changes are synchronized between different
types of database management systems (DBMS).

246. The method of claim 242 wherein the changes are synchronized between the same
types of database management system (DBMS).

247. The method of claim 242, wherein the changes are synchronized directly with a
database management system (DBMS) associated with the web-based application.

248. The method of claim 242, wherein the changes are synchronized directly with file structures associated with the web-based application.

249. The method of claim 242, wherein for a newly-installed instance of the web-based application on the first computer system, the method further comprises performing a complete synchronization of all records in the web-based application on the application server with the web-based application on the first computer system.

250. The method of claim 242, wherein the changes include one or more of:
changes to data in the web-based application;
changes to data structures in the web-based application; and
changes to files for the web-based application.

251. A system, comprising:
one or more processors;
memory; and
one or more programs stored in the memory, the one or more programs comprising instructions to:
for a web-based application that is not designed to be used while a first computer system on which the web-based application executes does not have a network connection to an application server,
make changes to the web-based application while the first computer system does not have a network connection to the application server; and
track the changes made to the web-based application while the first computer system is disconnected from the application server;
when the network connection between the first computer system and the application server is reestablished,
synchronize changes for the web-based application made on the first computer system with the web-based application on the application server; and
synchronize changes for the web-based application made on the application server with the web-based application on the first computer system.

252. A computer readable storage medium storing one or more programs for execution by a computer, the one or more programs comprising instructions to:

for a web-based application that is not designed to be used while a first computer system on which the web-based application executes does not have a network connection to an application server,
make changes to the web-based application while the first computer system
5 does not have a network connection to the application server; and
track the changes made to the web-based application while the first computer system is disconnected from the application server;
when the network connection between the first computer system and the application server is reestablished,
10 synchronize changes for the web-based application made on the first computer system with the web-based application on the application server; and
synchronize changes for the web-based application made on the application server with the web-based application on the first computer system.

253. A method, comprising:

15 for a web-based application that is being used on a first computer system while the first computer system is disconnected from an application server,
in response to detecting a network connection to the application server:
synchronizing changes for the web-based application made on the first computer system with the web-based application on the application
20 server; and
synchronizing changes for the web-based application made on the application server with the web-based application on the first computer system.

254. The method of claim 253, wherein for a newly-installed instance of the web-based application on the first computer system, the method further comprises performing a
25 synchronization of records in the web-based application on the application server with the web-based application on the first computer system.

255. The method of claim 253, wherein the changes include one or more of:
changes to data in the web-based application;
changes to data structures in the web-based application; and
30 changes to files for the web-based application.

256. A system, comprising:

one or more processors;

memory; and

one or more programs stored in the memory, the one or more programs comprising instructions to:

- 5 for a web-based application that is being used on a first computer system while the first computer system is disconnected from an application server, in response to detecting a network connection to the application server: synchronize changes for the web-based application made on the first computer system with the web-based application on the application server; and
- 10 synchronize changes for the web-based application made on the application server with the web-based application on the first computer system.

257. A computer readable storage medium storing one or more programs for execution by a computer, the one or more programs comprising instructions to:

- 15 for a web-based application that is being used on a first computer system while the first computer system is disconnected from an application server, in response to detecting a network connection to the application server: synchronize changes for the web-based application made on the first computer system with the web-based application on the application server; and
- 20 synchronize changes for the web-based application made on the application server with the web-based application on the first computer system.

258. A method, comprising:

- on a first computer system that is disconnected from an application server, using a web-based application that is configured to interact over a network connection with the application server to provide specified
- 25 functionality;
- when the network connection is reestablished, synchronizing changes made to the web-based application while the first computer system was disconnected from the application server; and
- 30 if a conflict between the first computer system and the application server exists, resolving the conflict so that both the first computer system and the application server are synchronized with each other.

259. The method of claim 258, wherein a conflict exists if the first computer system includes a first set of records that are not included on the application server, and wherein resolving the conflict so that both the first computer system and the application server are synchronized with each other involves:

- 5 sending the first set of records to the application server;
- receiving new identifiers for the first set of records, wherein the new identifiers are assigned by the application server when the first set of records are added to the application server; and
- updating all references to the old identifiers for the first set of records with the new
- 10 identifiers for the first set of records.

260. The method of claim 258, wherein a conflict exists if the application server includes a second set of records that are not included on the first computer system, and wherein resolving the conflict so that both the first computer system and the application server are synchronized with each other involves:

- 15 receiving the second set of records from the application server;
- determining whether identifiers for the second set of records are already being used by the first computer system; and
- if the identifiers for the second set of records are not being used by the first computer system, inserting the second set of records.

- 20 261. The method of claim 260, wherein if the identifiers for the second set of records are already being used by the first computer system, the method further comprises:

- determining a third set of records which include identifiers which conflict with the identifiers for the second set of records;
- sending the third set of records to the application server;
- 25 receiving new identifiers for the third set of records, wherein the new identifiers are assigned by the application server when the third set of records are added to the application server;
- updating all references to the old identifiers for the third set of records with the new identifiers for the third set of records; and
- 30 inserting the second set of records.

262. The method of claim 258, wherein if a conflict between the first computer system and the application server does not exist, the method further comprises updating records on the

first computer system so that the first computer system is synchronized with the application server.

263. A system, comprising:

one or more processors;

memory; and

one or more programs stored in the memory, the one or more programs comprising instructions to:

on a first computer system that is disconnected from an application server,

use a web-based application that is configured to interact over a network

connection with the application server to provide specified functionality;

when the network connection is reestablished, synchronize changes made to the web-based application while the first computer system was disconnected from the application server; and

if a conflict between the first computer system and the application server exists, resolve the conflict so that both the first computer system and the application server are synchronized with each other.

264. A computer readable storage medium storing one or more programs for execution by a computer, the one or more programs comprising instructions to:

on a first computer system that is disconnected from an application server,

use a web-based application that is configured to interact over a network connection with the application server to provide specified functionality;

when the network connection is reestablished, synchronize changes made to the web-based application while the first computer system was disconnected from the application server; and

if a conflict between the first computer system and the application server exists, resolve the conflict so that both the first computer system and the application server are synchronized with each other.

265. A method, comprising:

creating a first set of records with a corresponding first set of identifiers in a first database;

synchronizing the first database with a second database;
if the corresponding first set of identifiers already exists in the second database,
receiving a new set of identifiers for the first set of records from the second
database, wherein the new set of identifiers is assigned to the first set
of records when the first set of records is added to the second database;
and
updating all references to the corresponding first set of identifiers for the first
set of records with the new set of identifiers for the first set of records.

266. The method of claim 265, wherein if the corresponding first set of identifiers does not
exist in the second database, the first set of records is inserted into the second database using
the corresponding first set of identifiers.

267. The method of claim 265, wherein if the second database includes a second set of
records that does not exist in the first database, the method further comprises:

receiving the second set of records from the second database;
sending the first set of records to the second database;
receiving new identifiers for the first set of records, wherein the new identifiers are
assigned by the second database when the first set of records are added to the
second database;
updating all references to the old identifiers for the first set of records with the new
identifiers for the first set of records; and
inserting the second set of records.

268. The method of claim 265, wherein the first database and the second database are used
for a web-based application that requires a network connection with an application server to
provide specified functionality.

269. The method of claim 265, wherein the web-based application was not designed to be
used without a network connection to the application server.

270. The method of claim 265, wherein the first database is on a client computer system.

271. The method of claim 265, wherein the second database is on an application server.

272. A system, comprising:

one or more processors;
memory; and
one or more programs stored in the memory, the one or more programs comprising instructions to:

- 5 create a first set of records with a corresponding first set of identifiers in a first database;
synchronize the first database with a second database;
if the corresponding first set of identifiers already exists in the second database,
10 receive a new set of identifiers for the first set of records from the second database, wherein the new set of identifiers is assigned to the first set of records when the first set of records is added to the second database;
and
update all references to the corresponding first set of identifiers for the first set of records with the new set of identifiers for the first set of records.

- 15 273. A computer readable storage medium storing one or more programs for execution by a computer, the one or more programs comprising instructions to:
create a first set of records with a corresponding first set of identifiers in a first database;
synchronize the first database with a second database;
20 if the corresponding first set of identifiers already exists in the second database,
receive a new set of identifiers for the first set of records from the second database, wherein the new set of identifiers is assigned to the first set of records when the first set of records is added to the second database;
and
25 update all references to the corresponding first set of identifiers for the first set of records with the new set of identifiers for the first set of records.

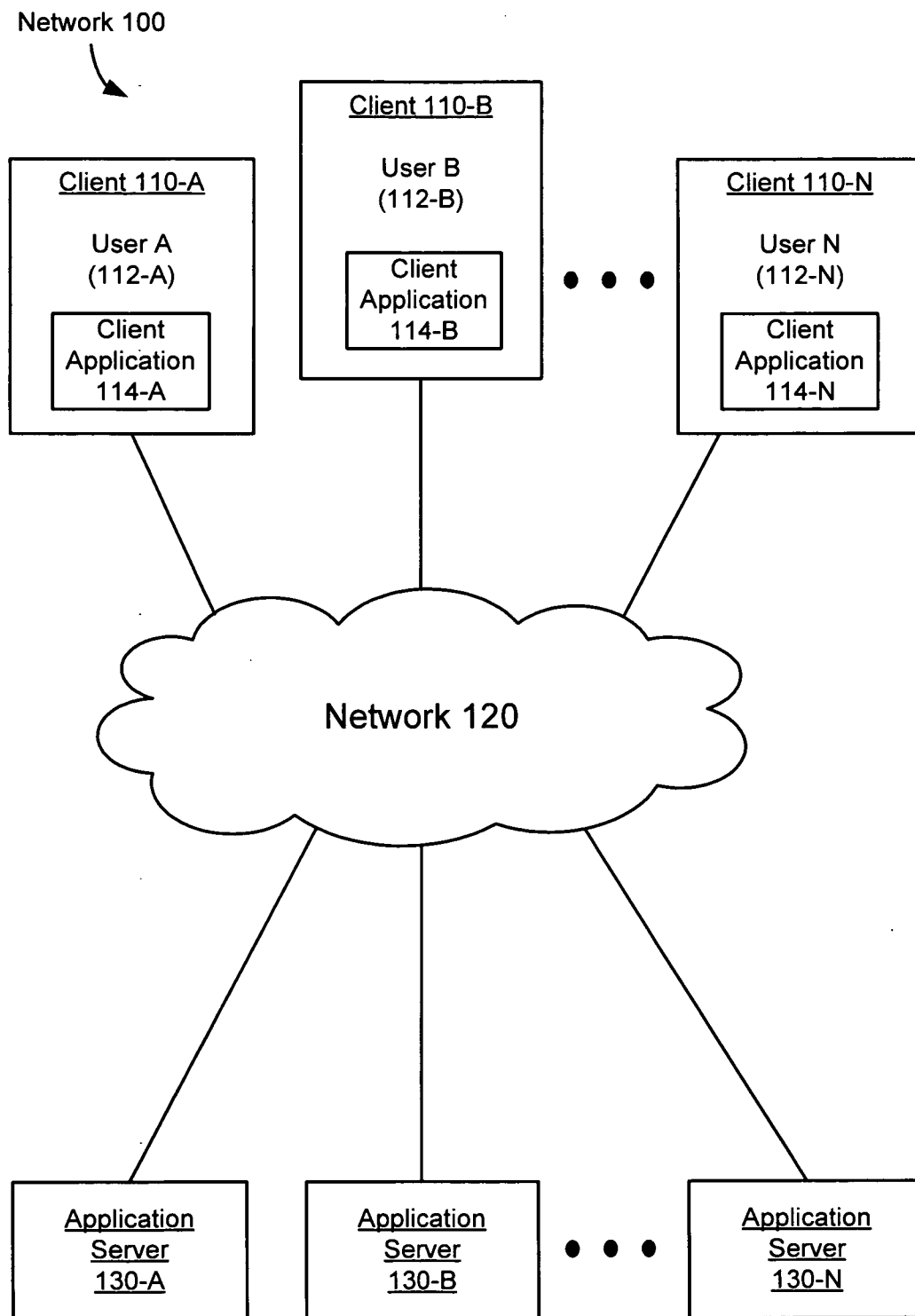


Figure 1

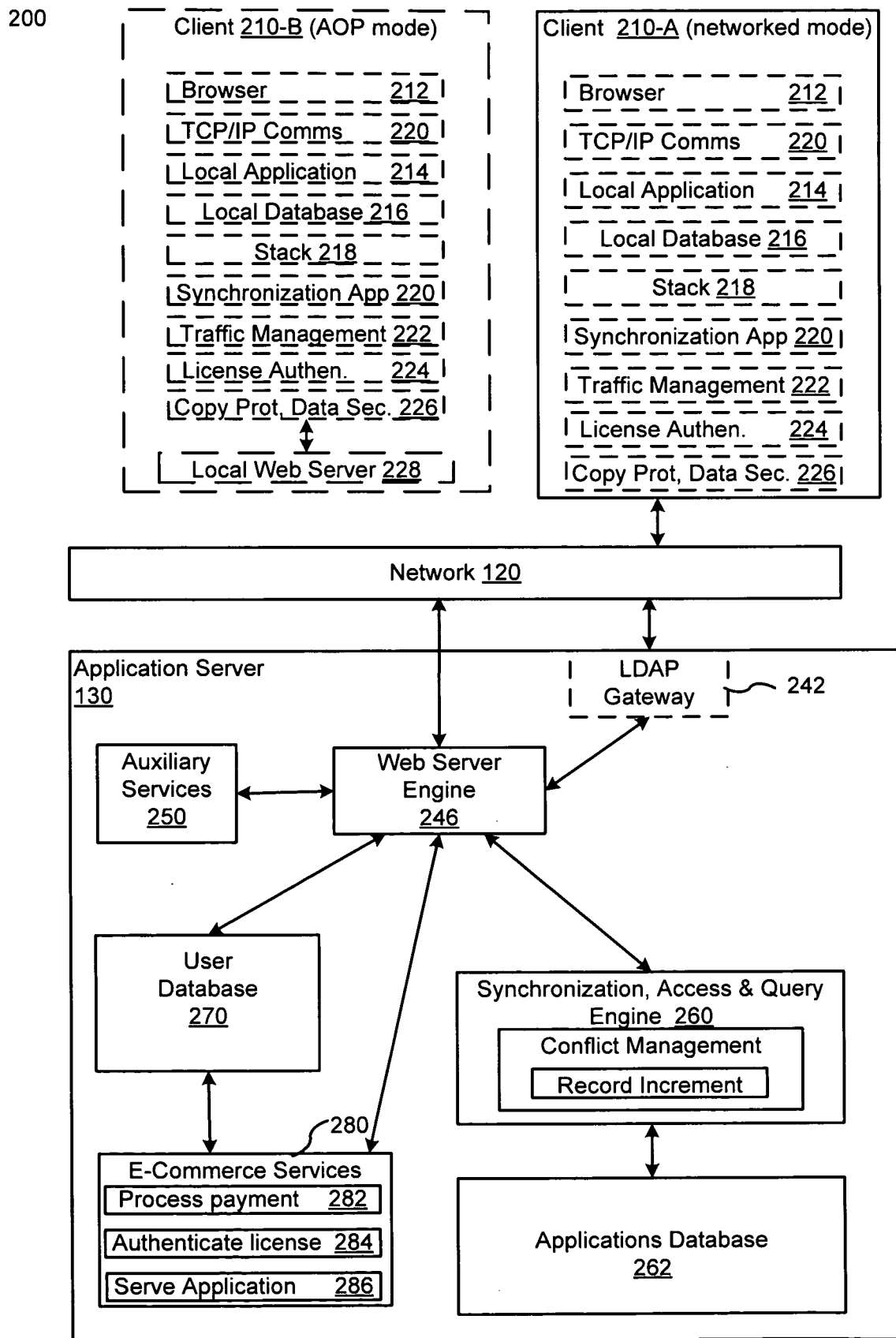


Figure 2

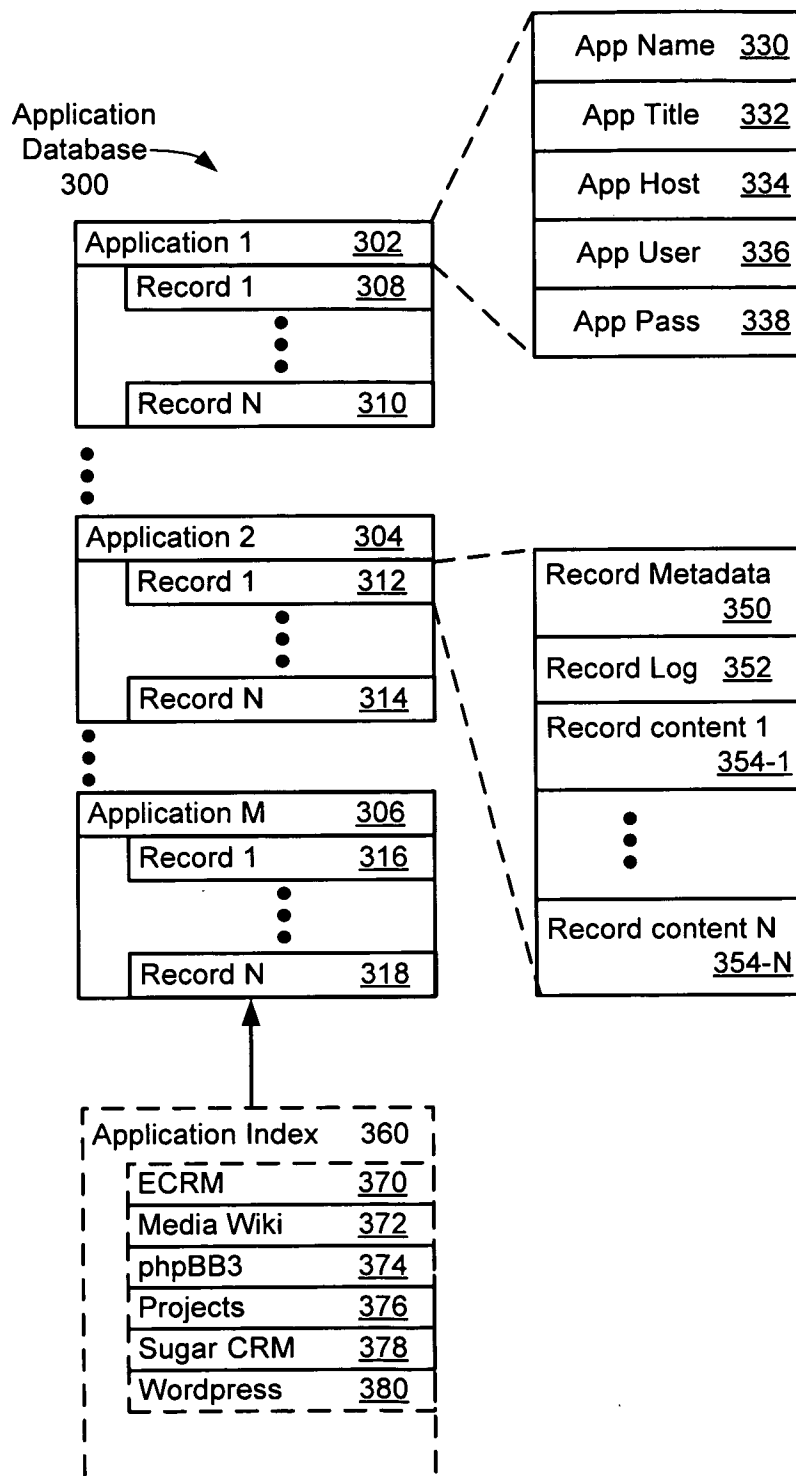


Figure 3

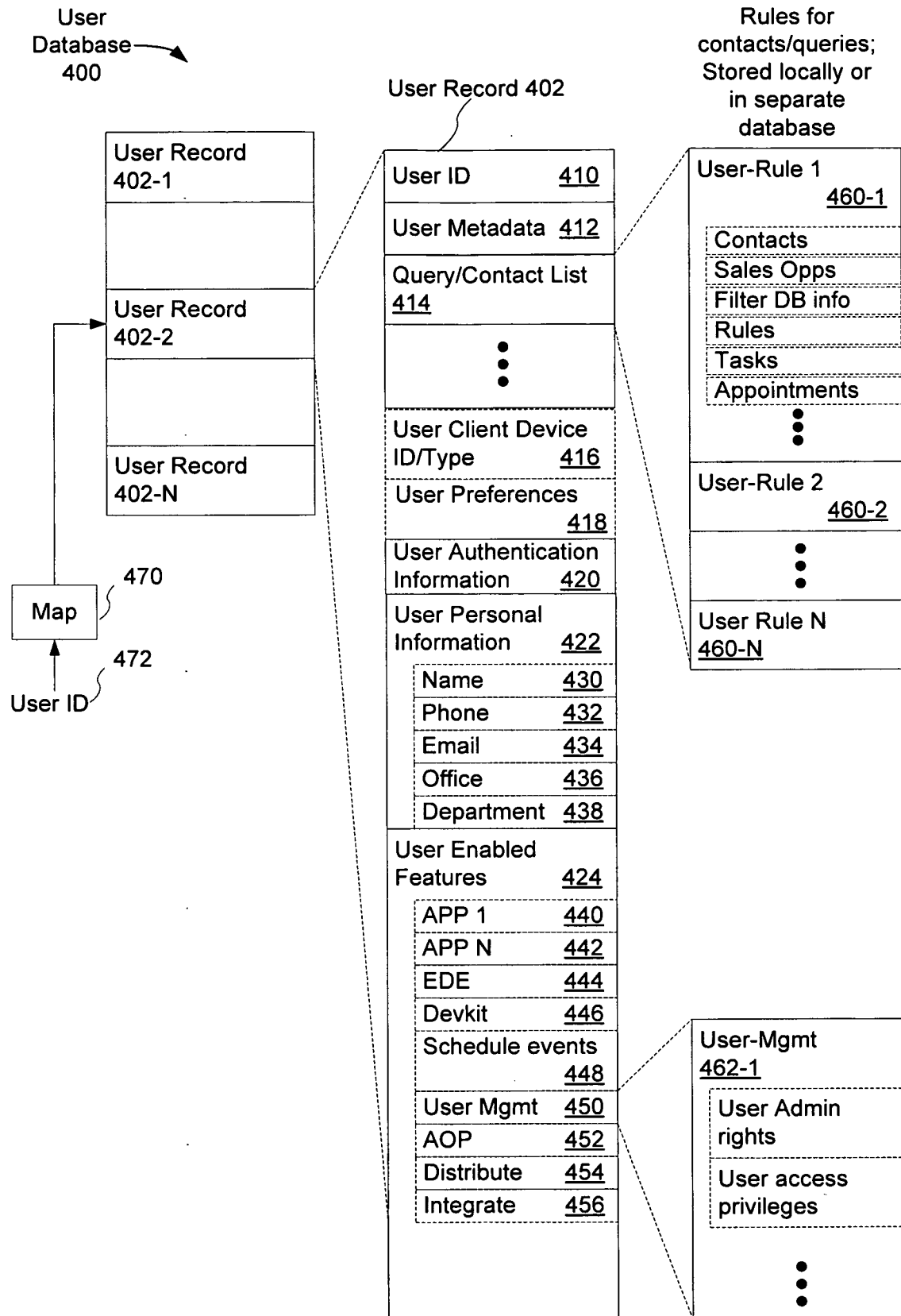


Figure 4

Sync Engine
500

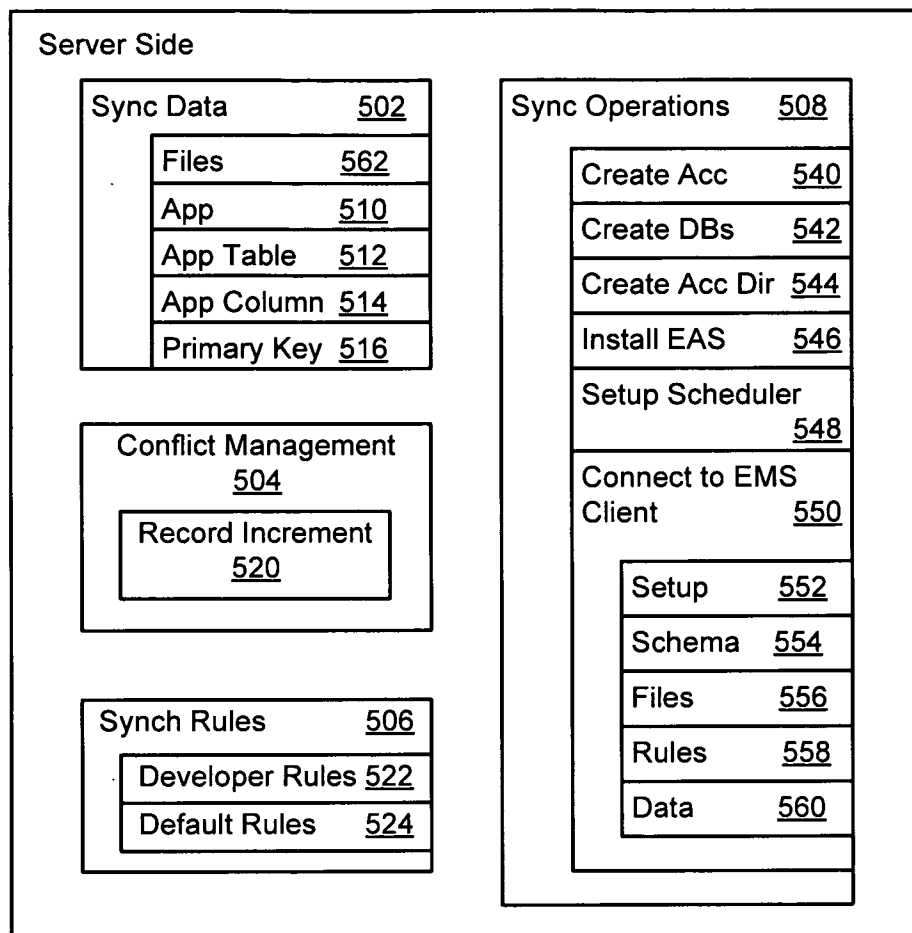


Figure 5

Sync Engine
600

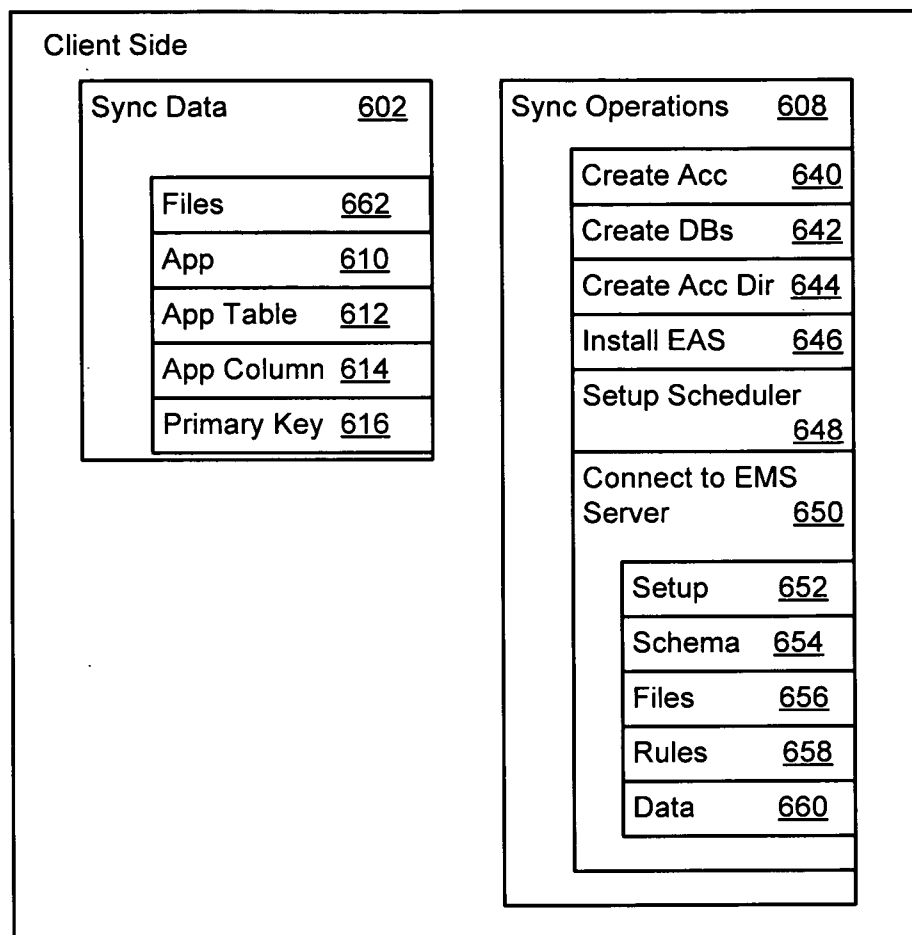


Figure 6

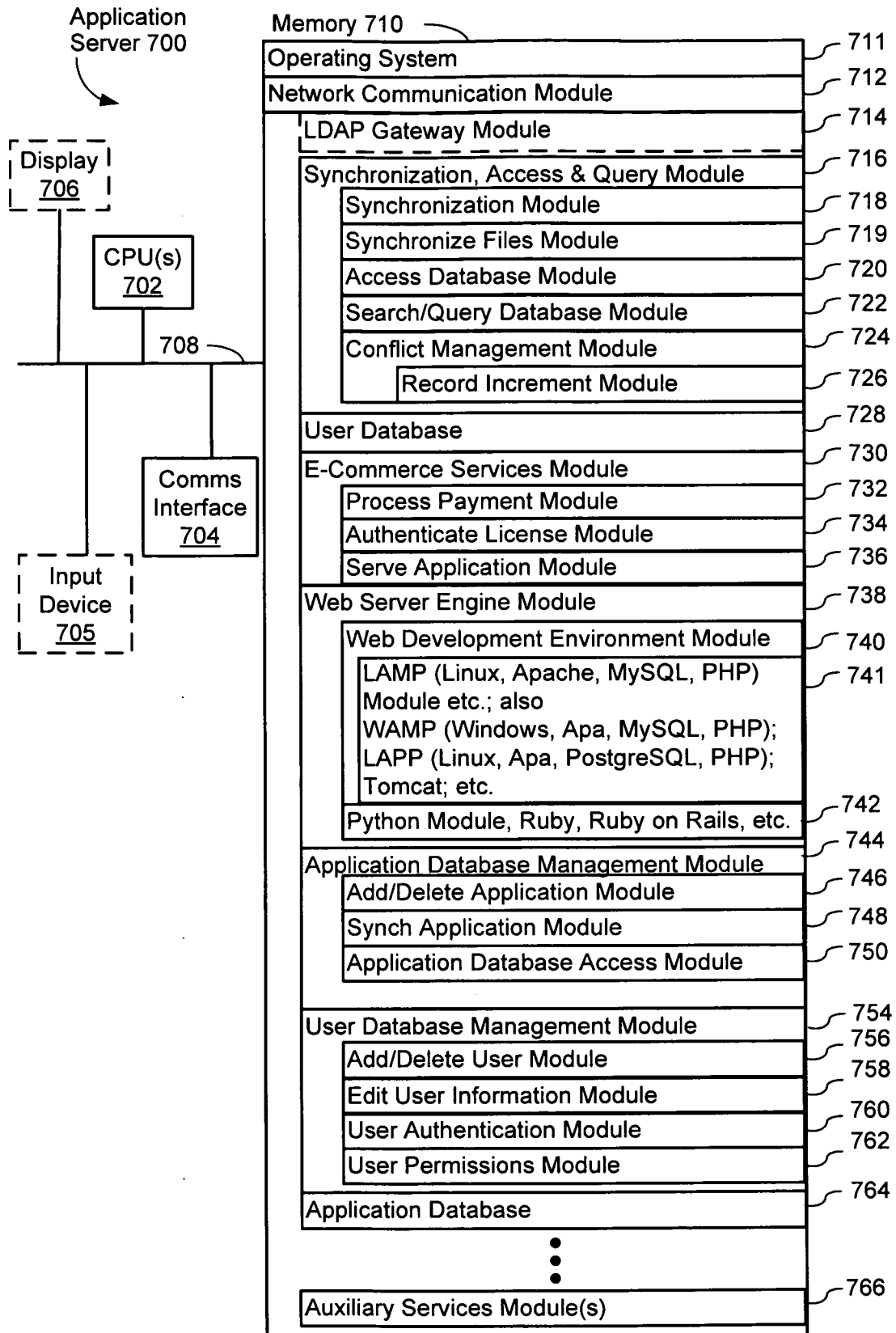


Figure 7

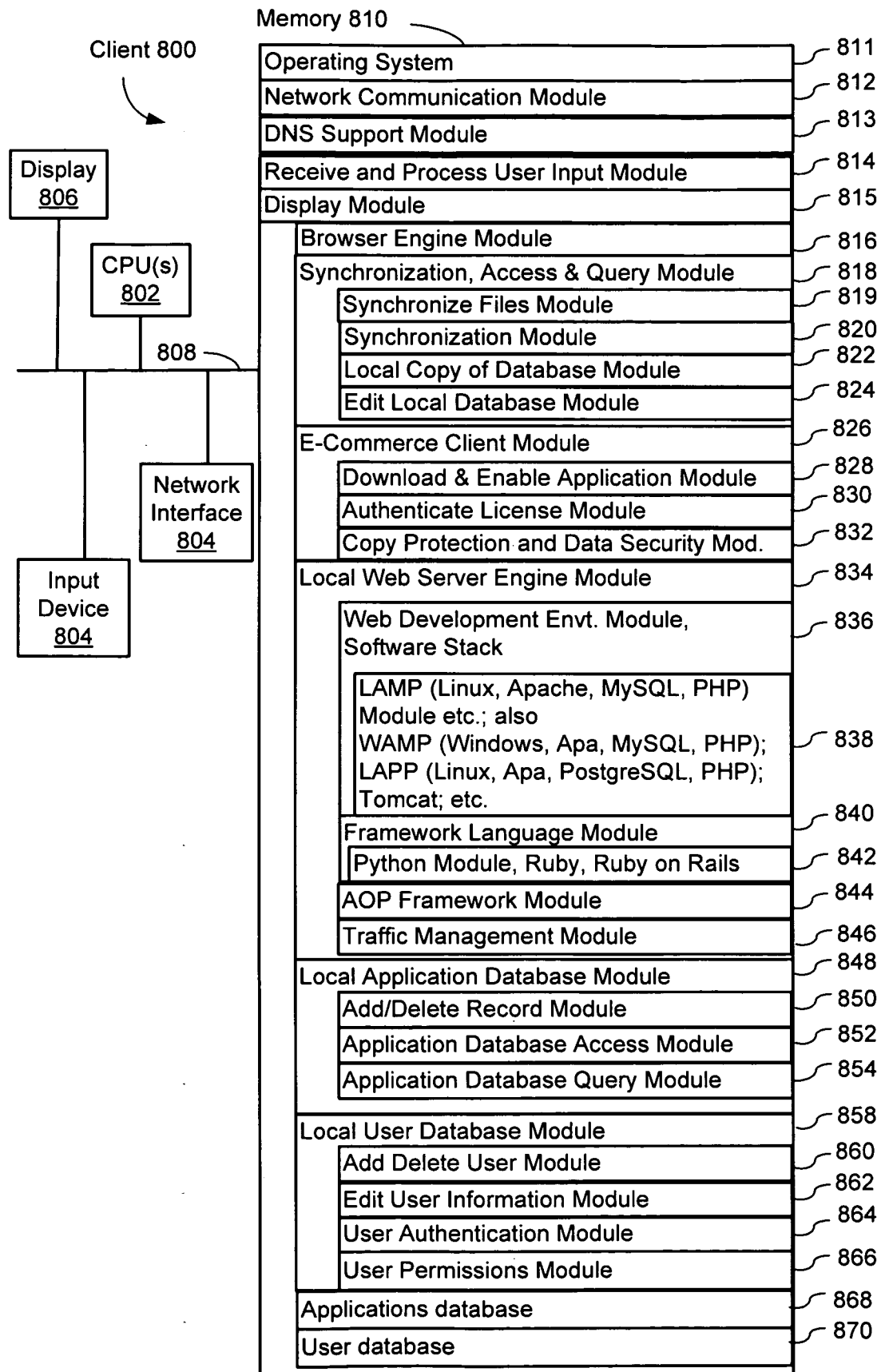


Figure 8

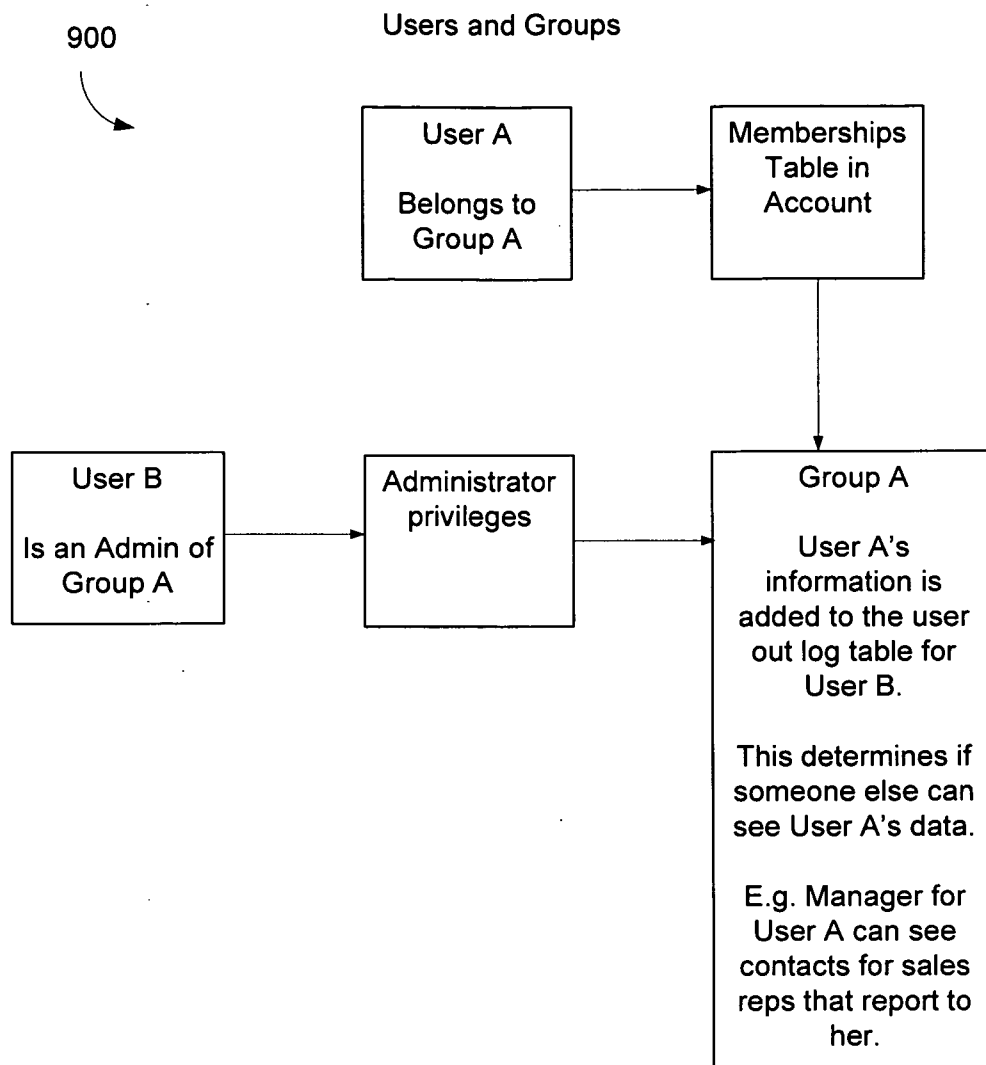


Figure 9

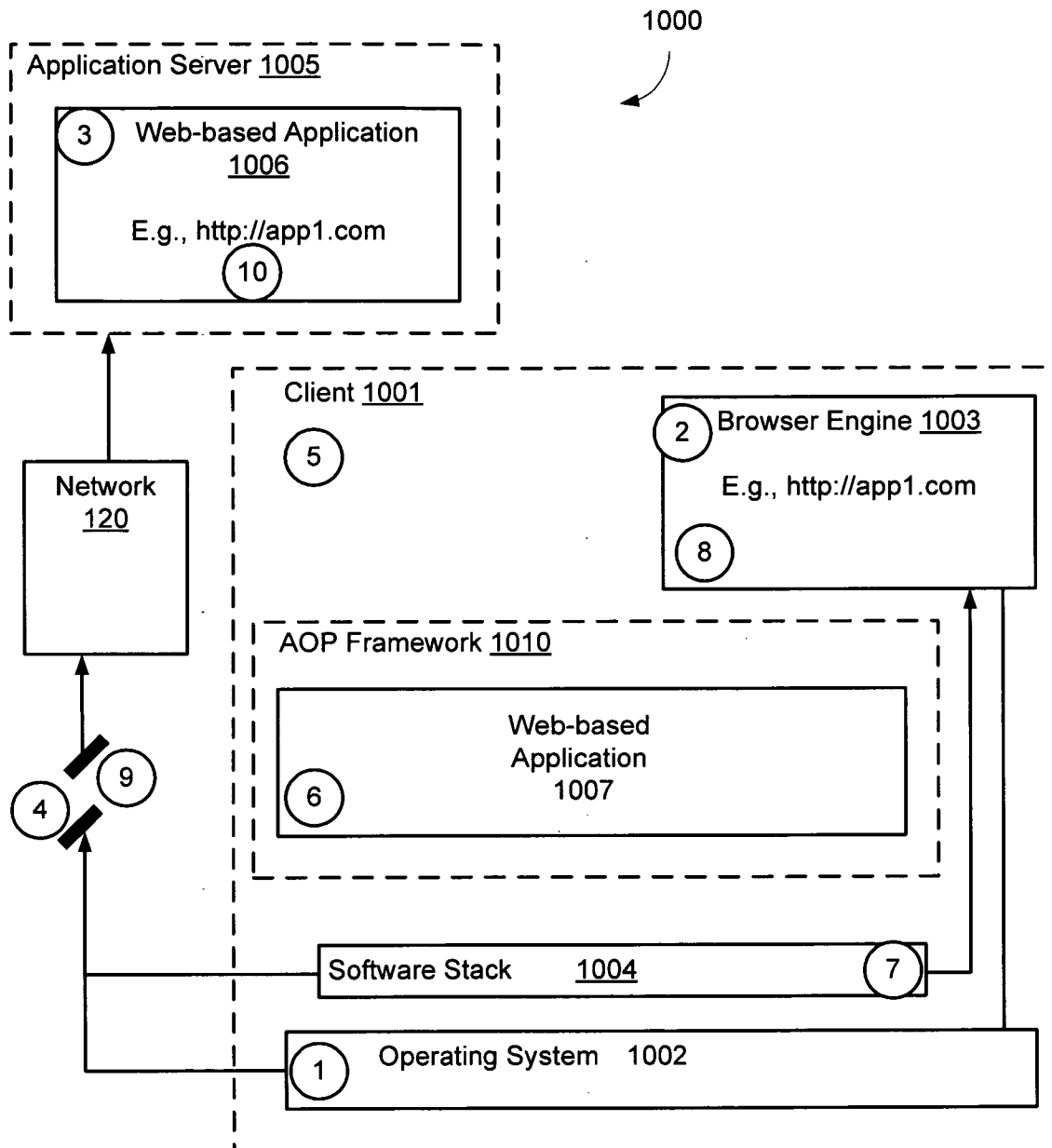
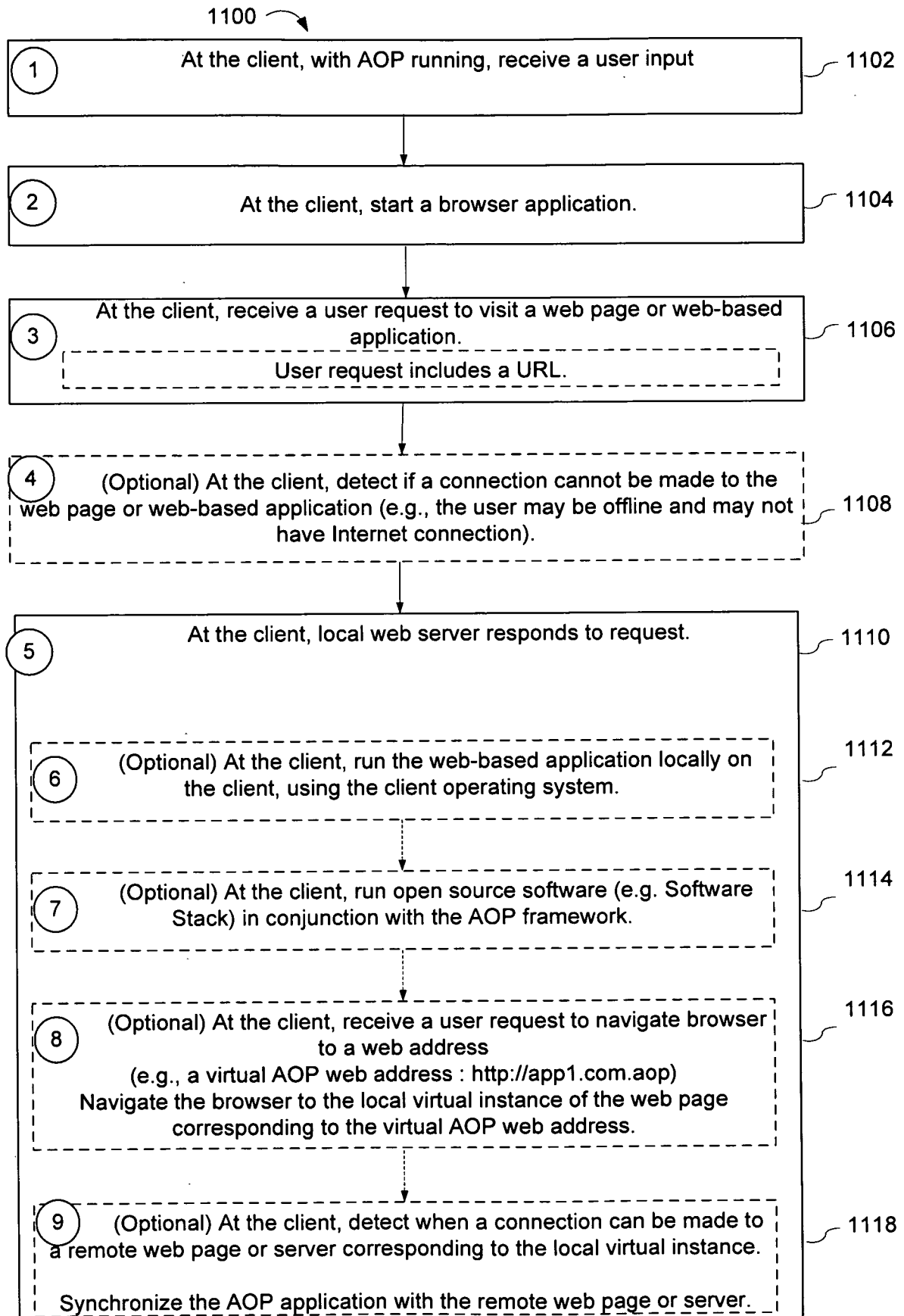


Figure 10



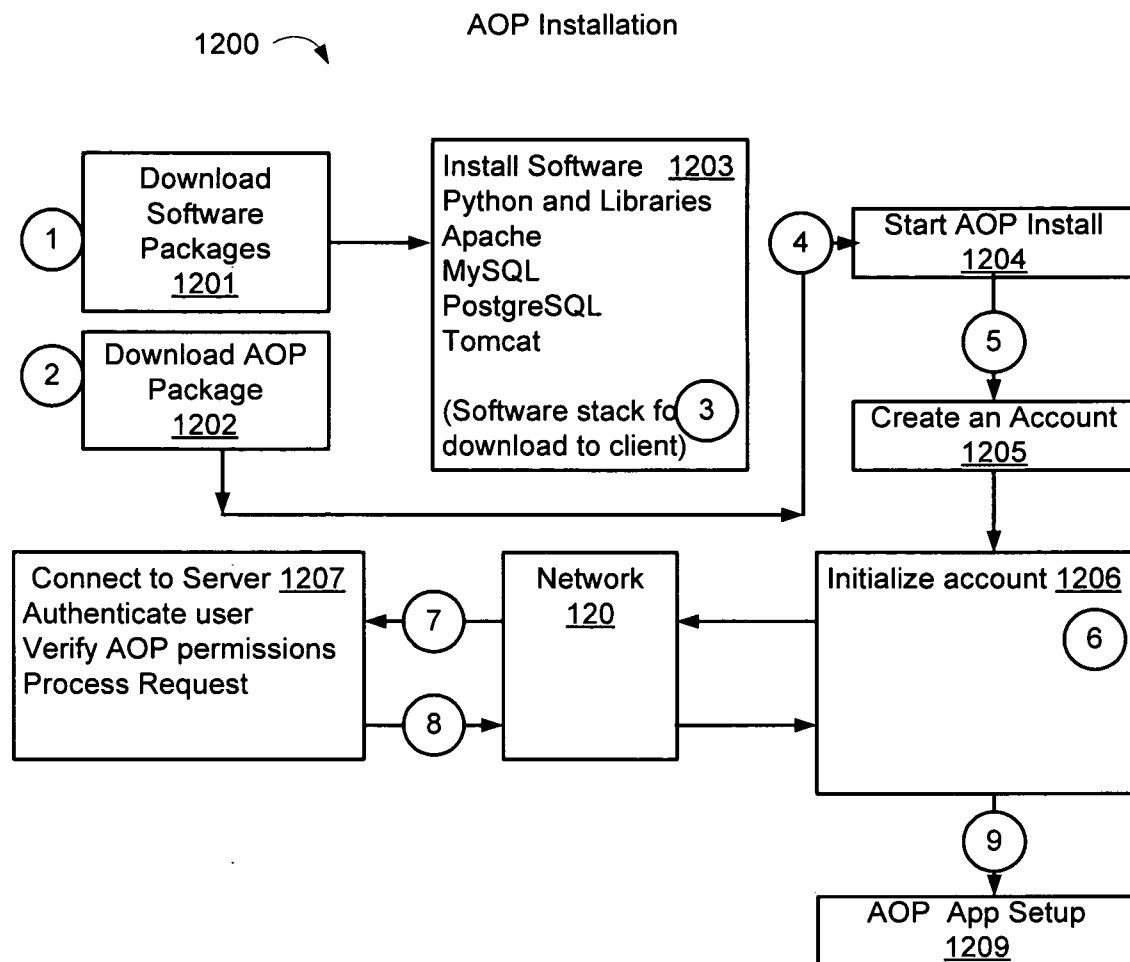


Figure 12

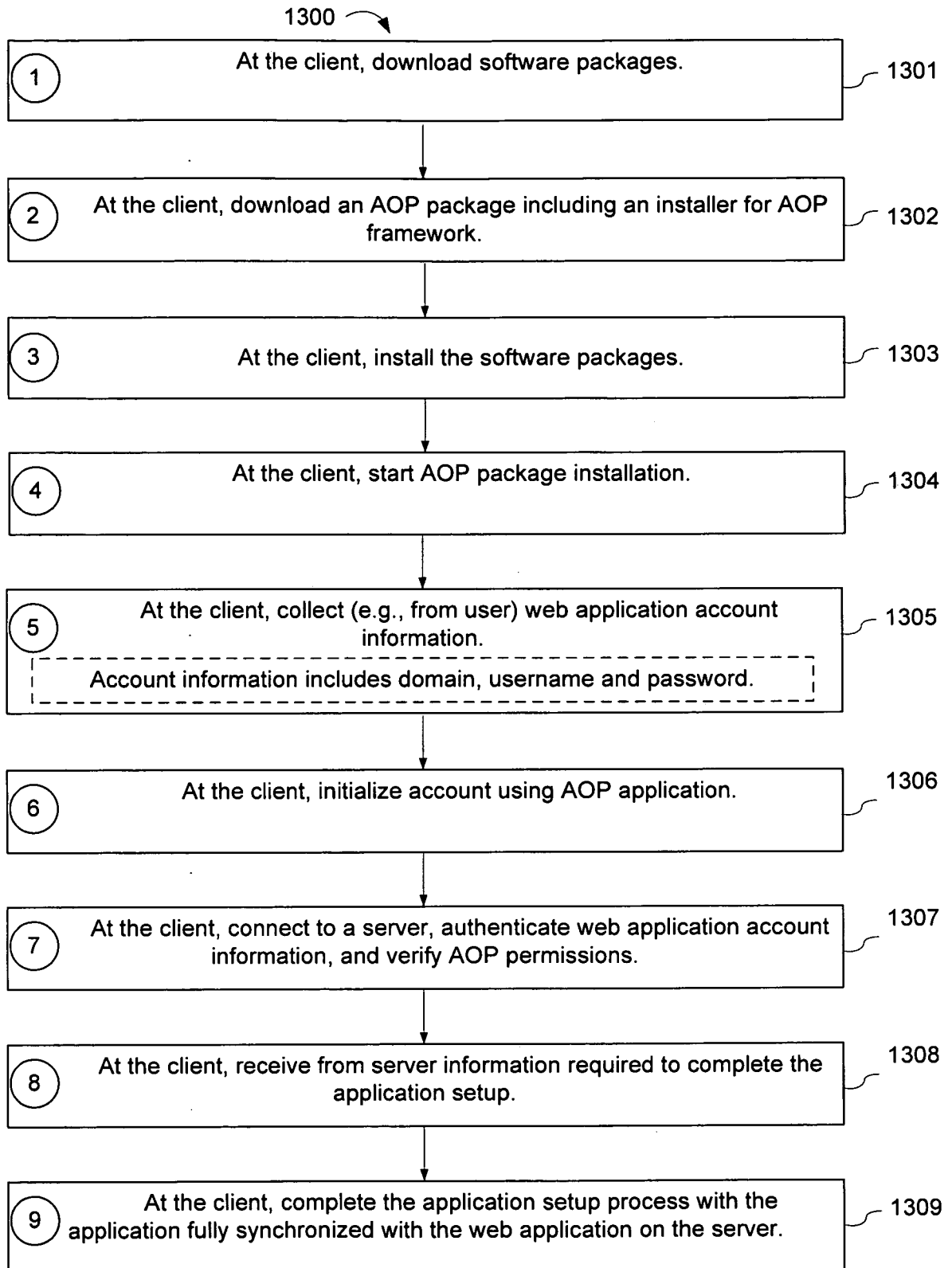


Figure 13

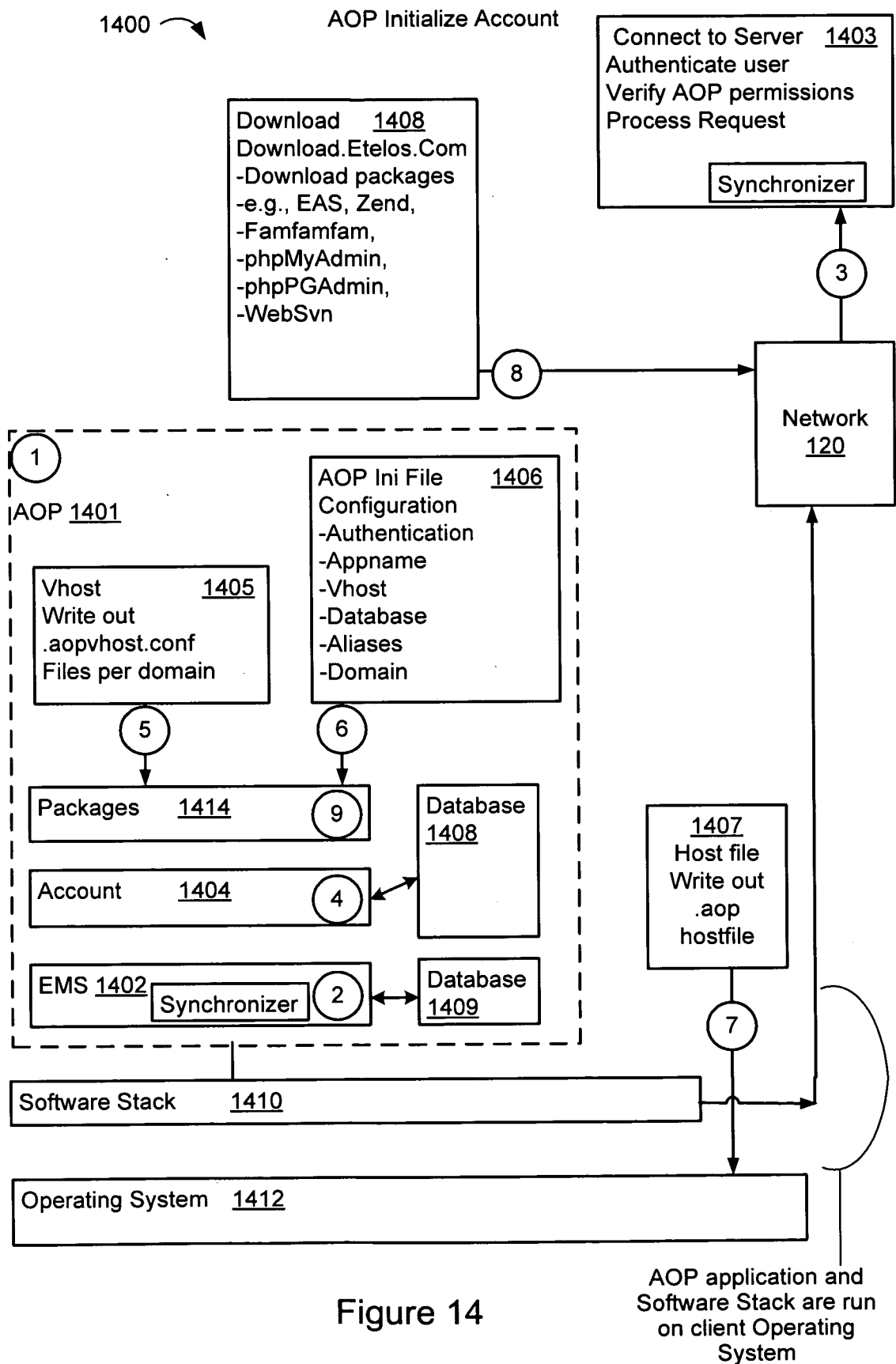


Figure 14

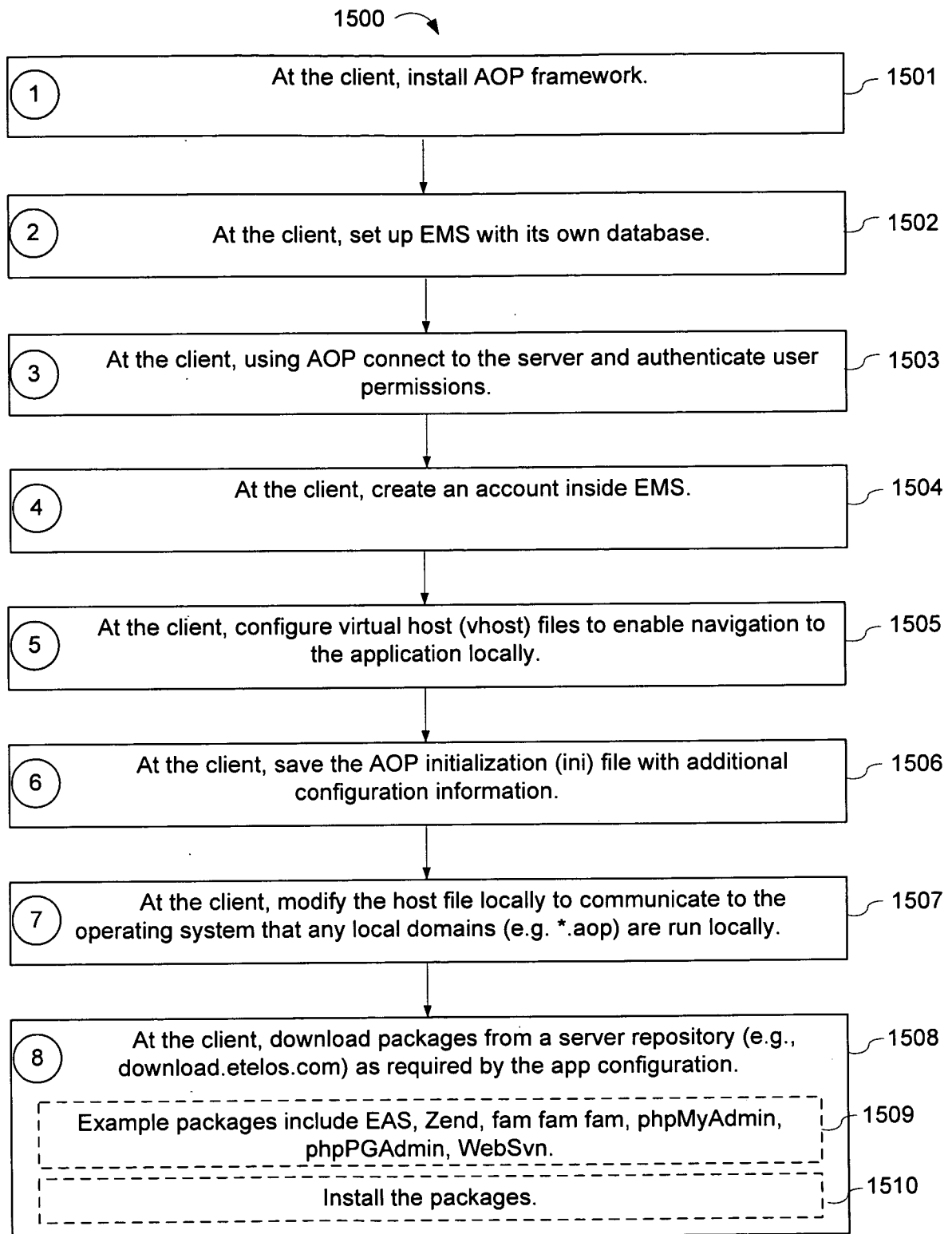


Figure 15

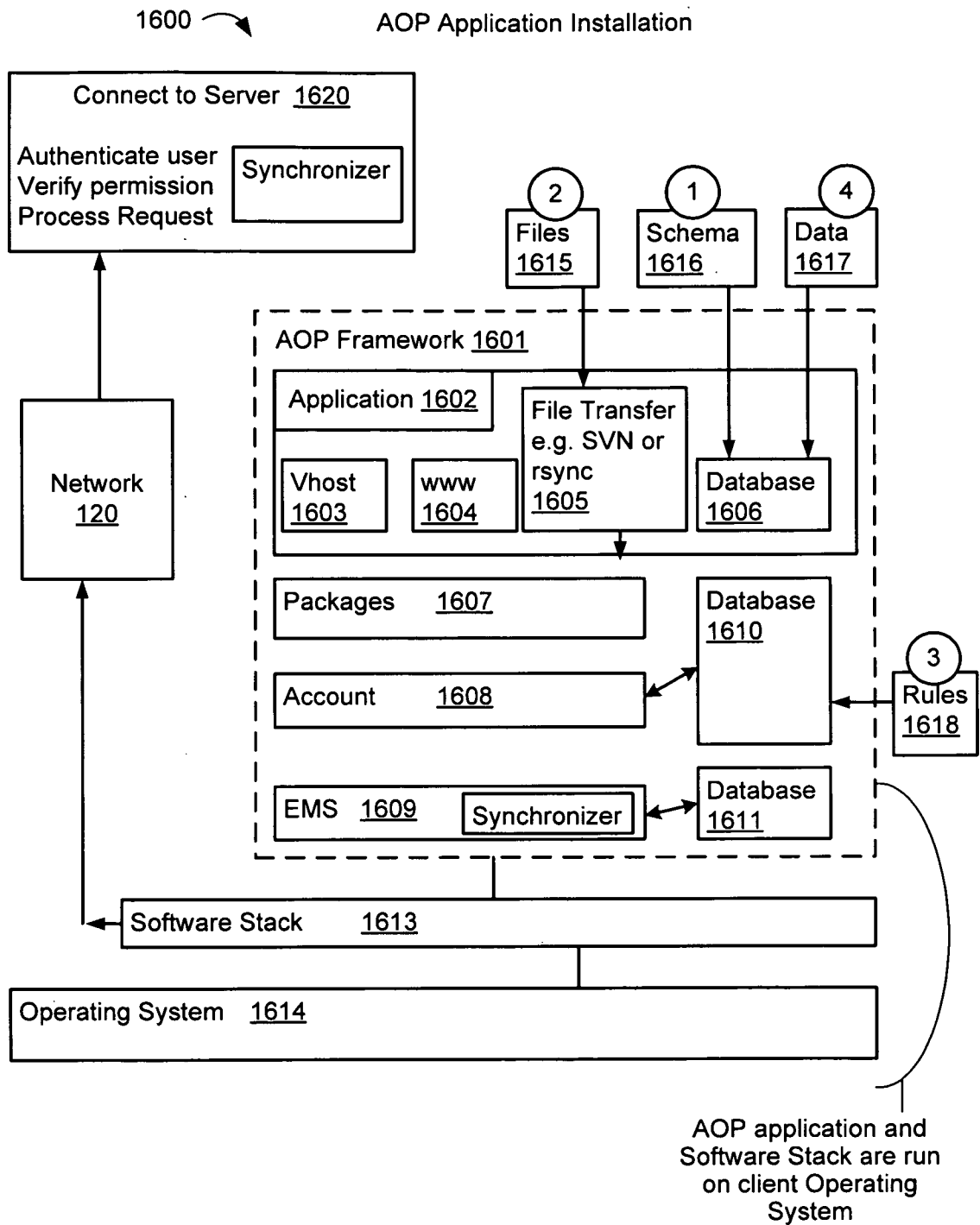


Figure 16

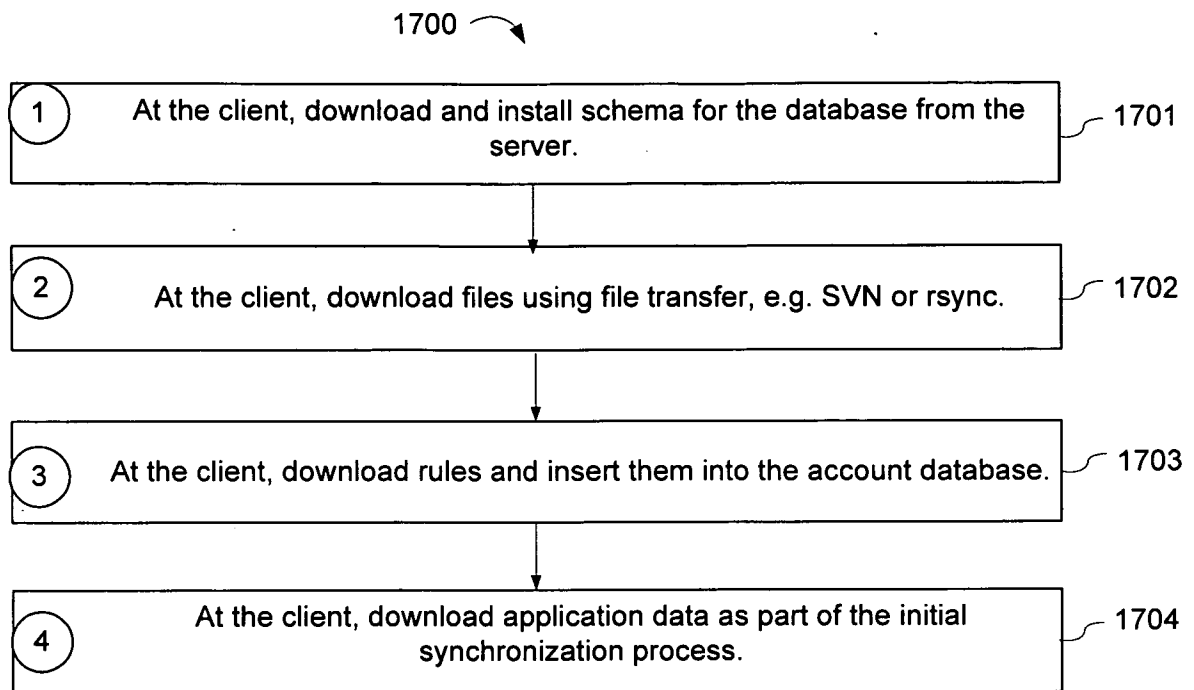


Figure 17

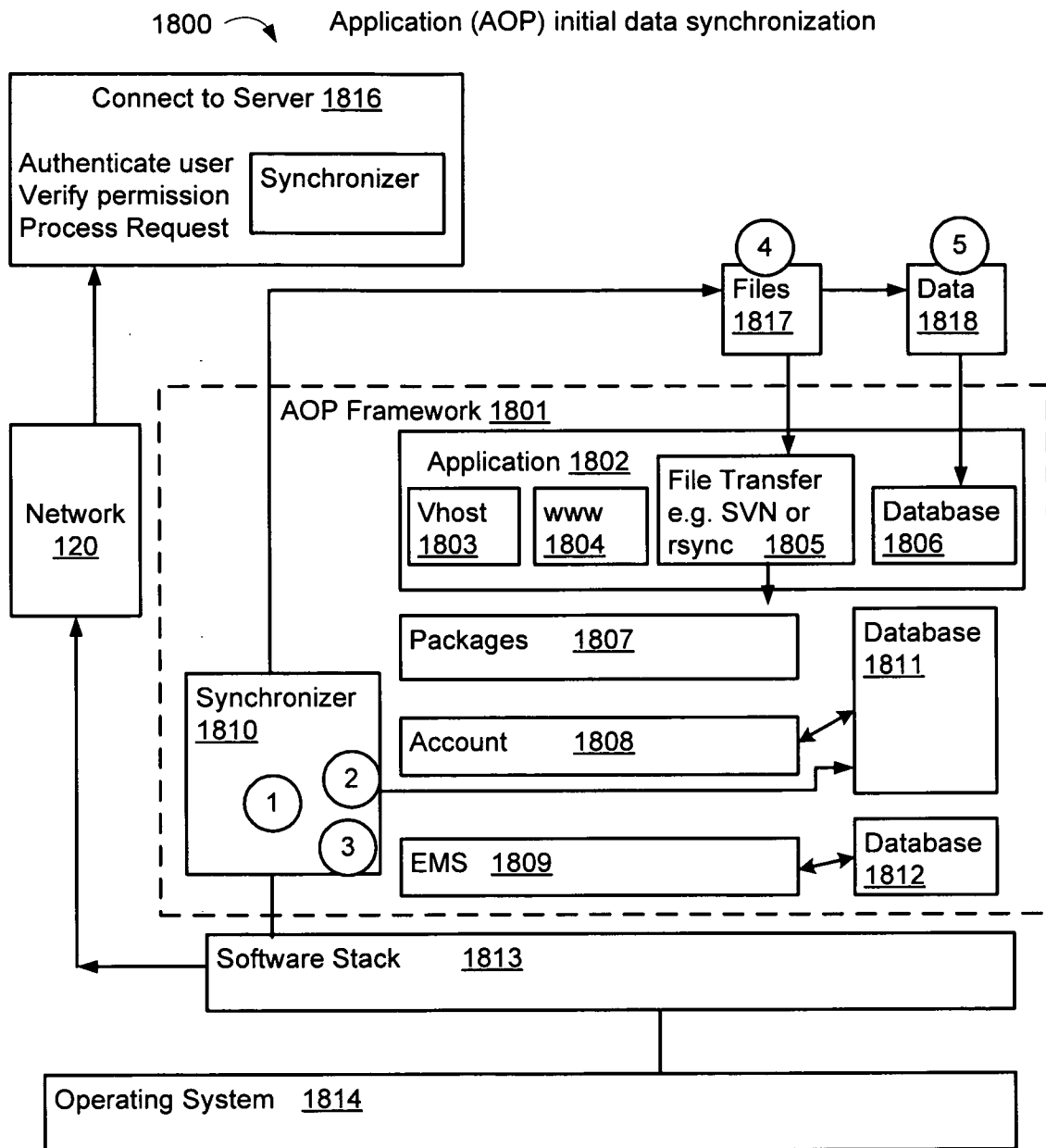


Figure 18

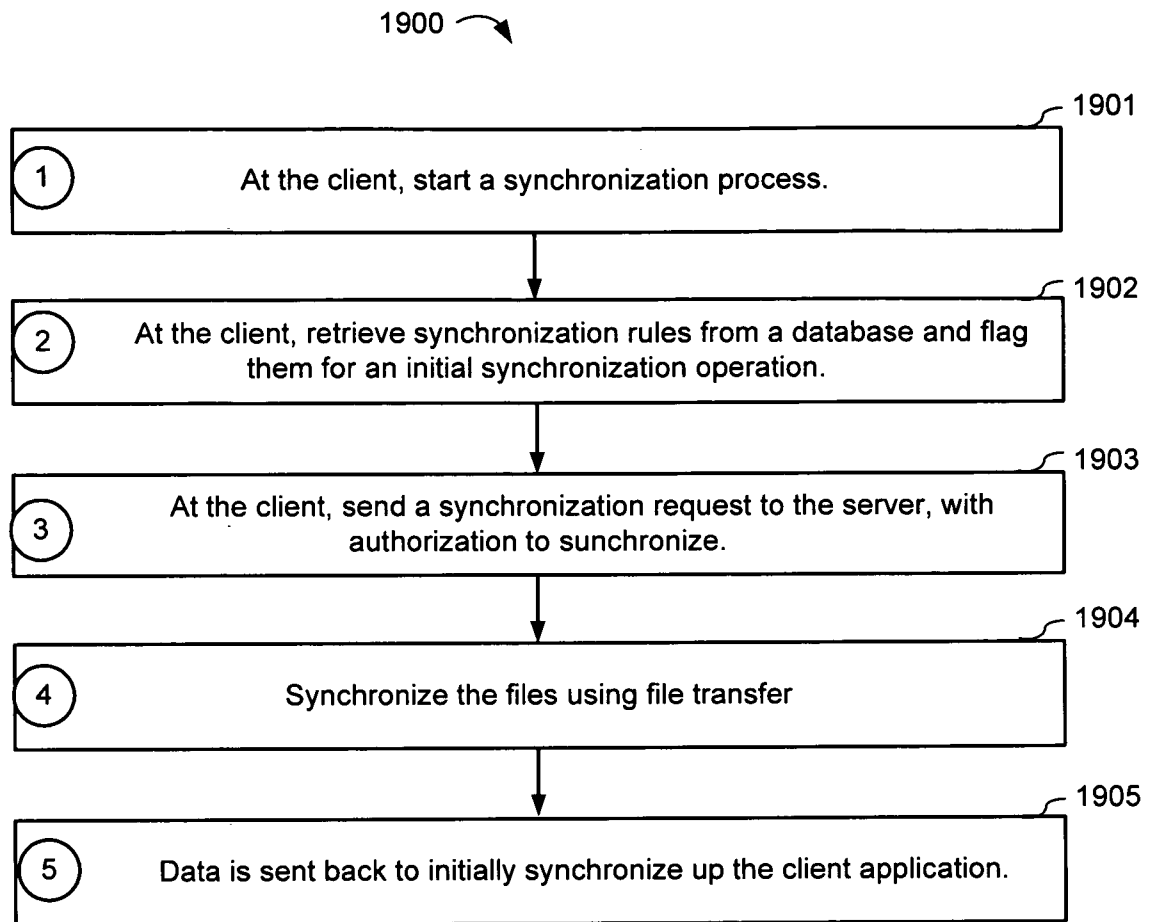


Figure 19

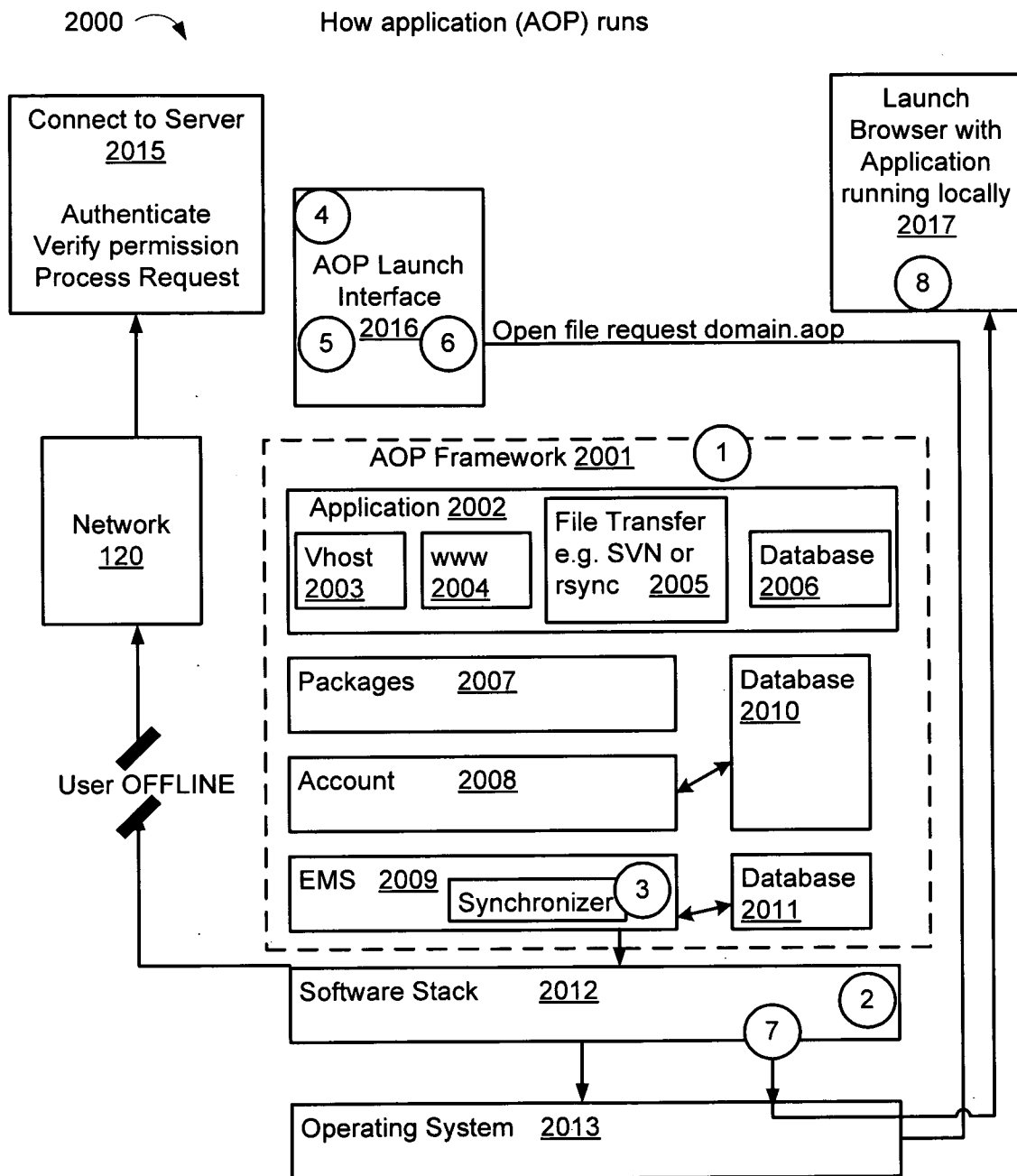


Figure 20

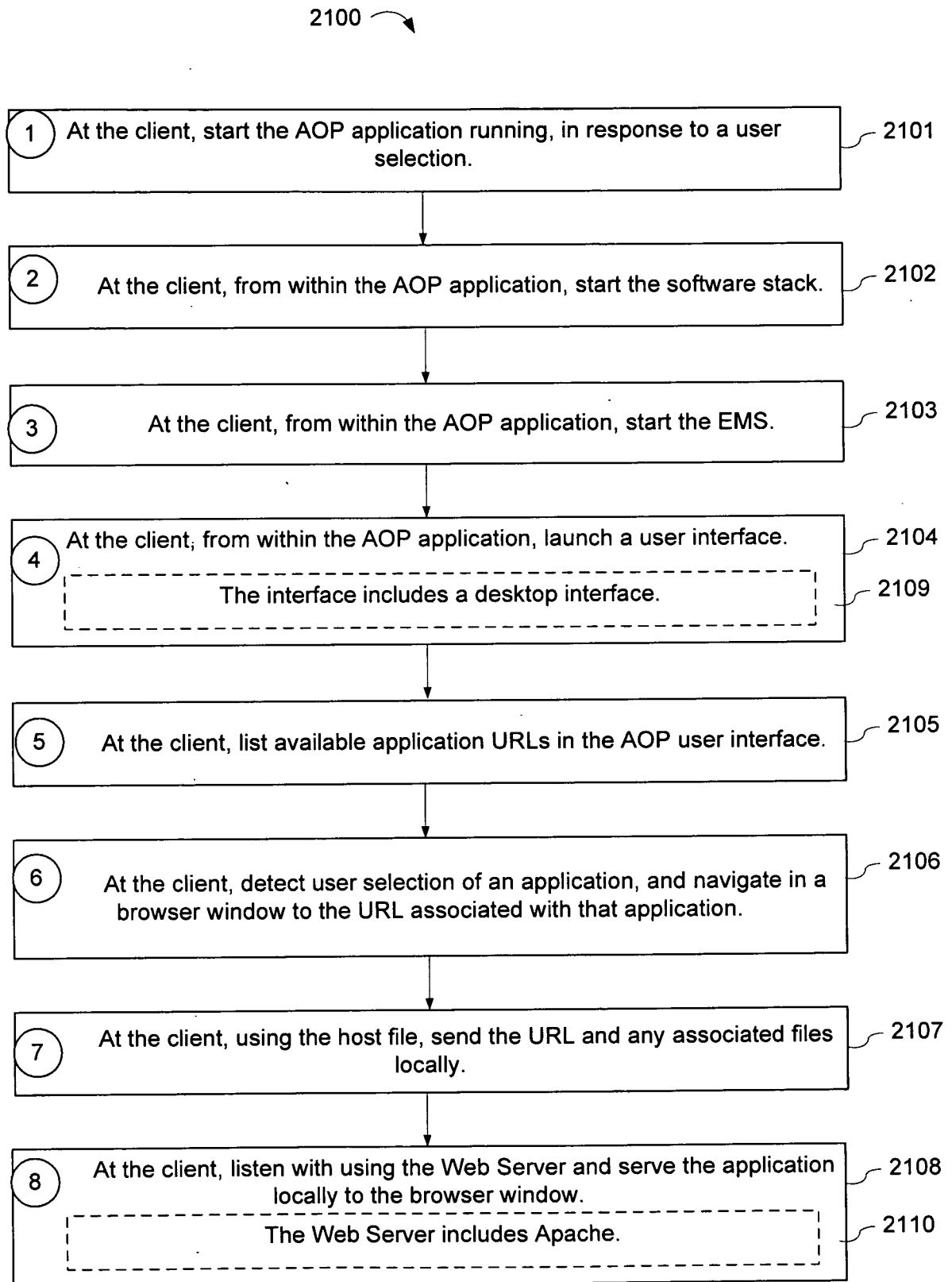


Figure 21

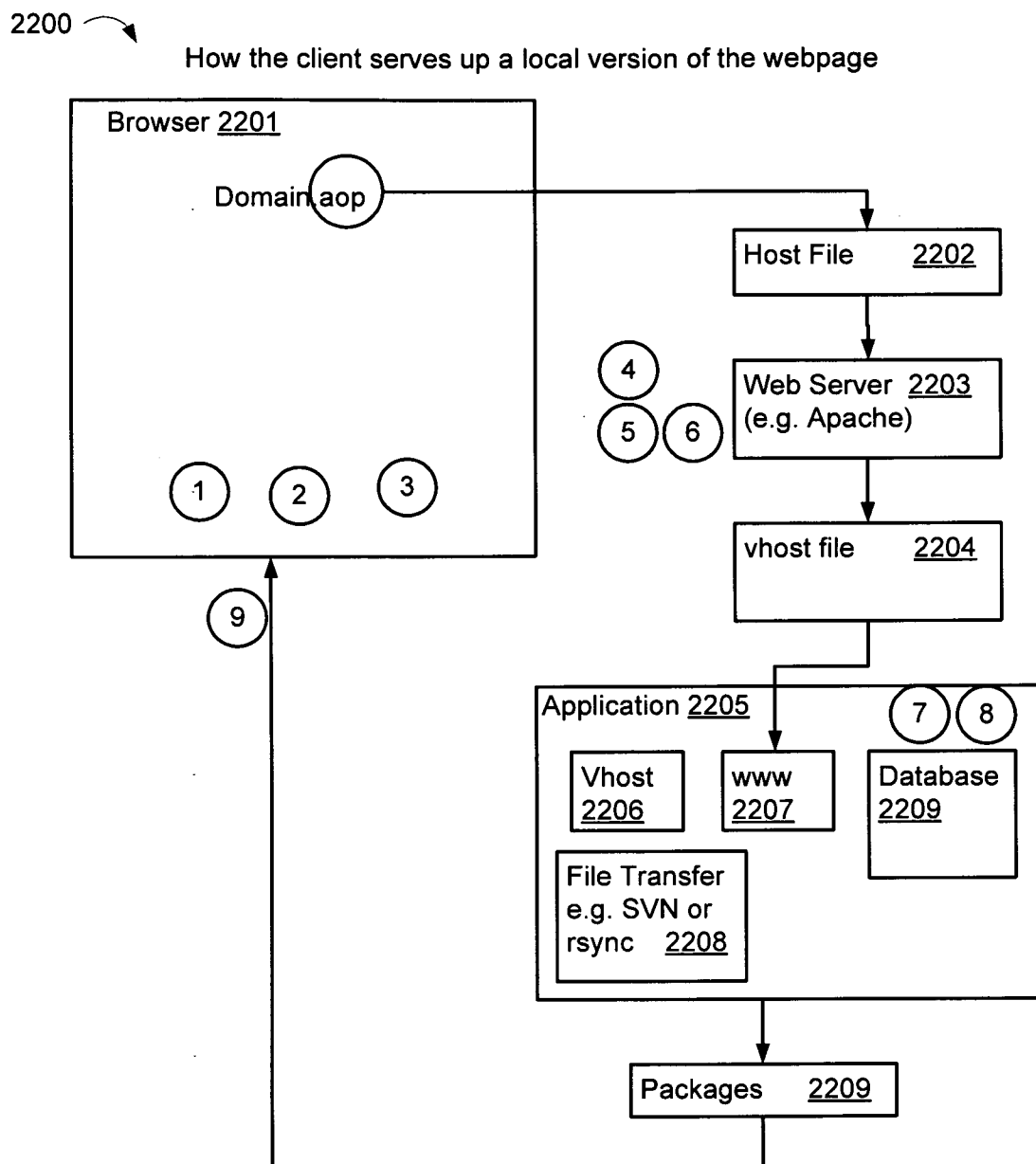


Figure 22

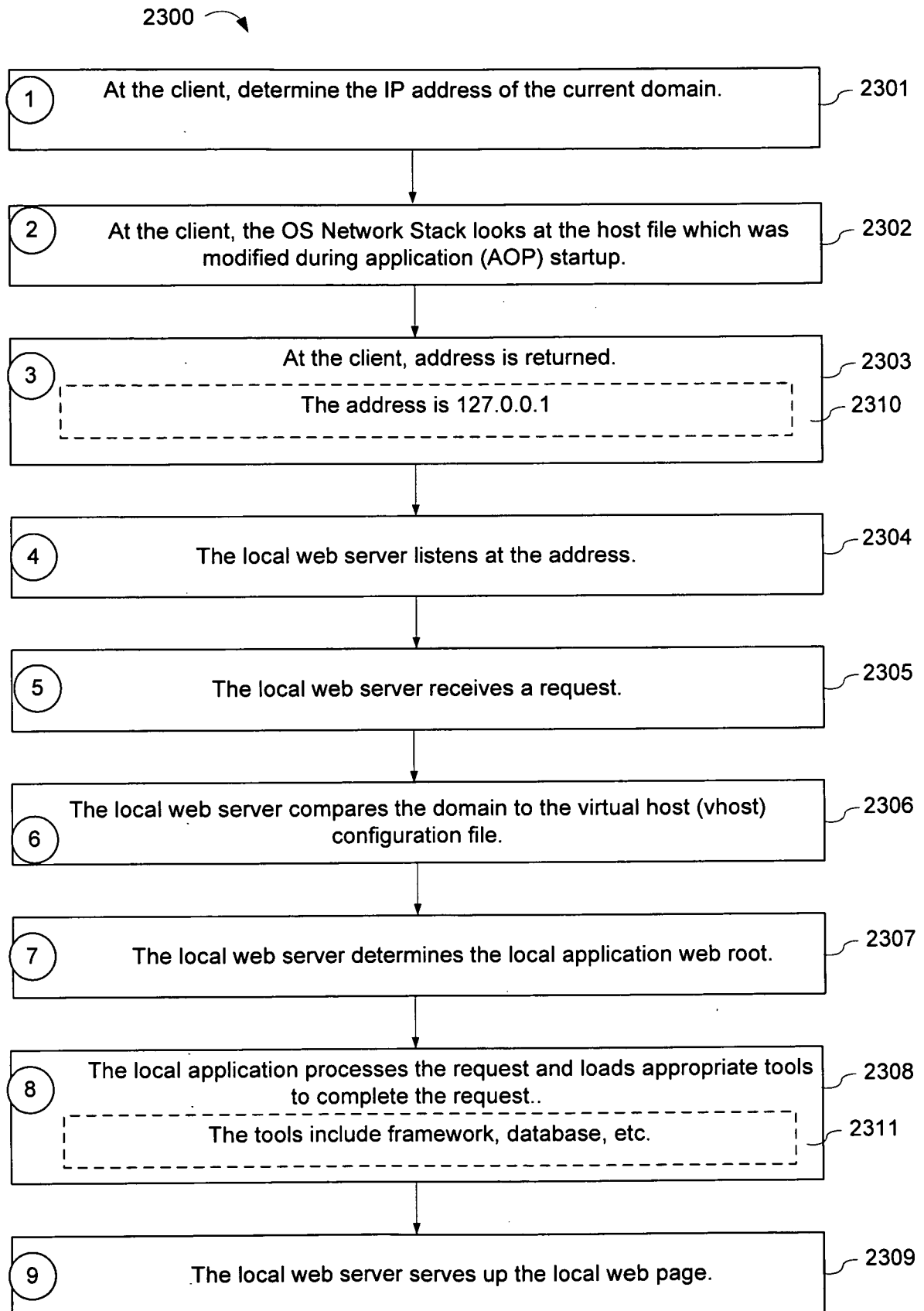


Figure 23

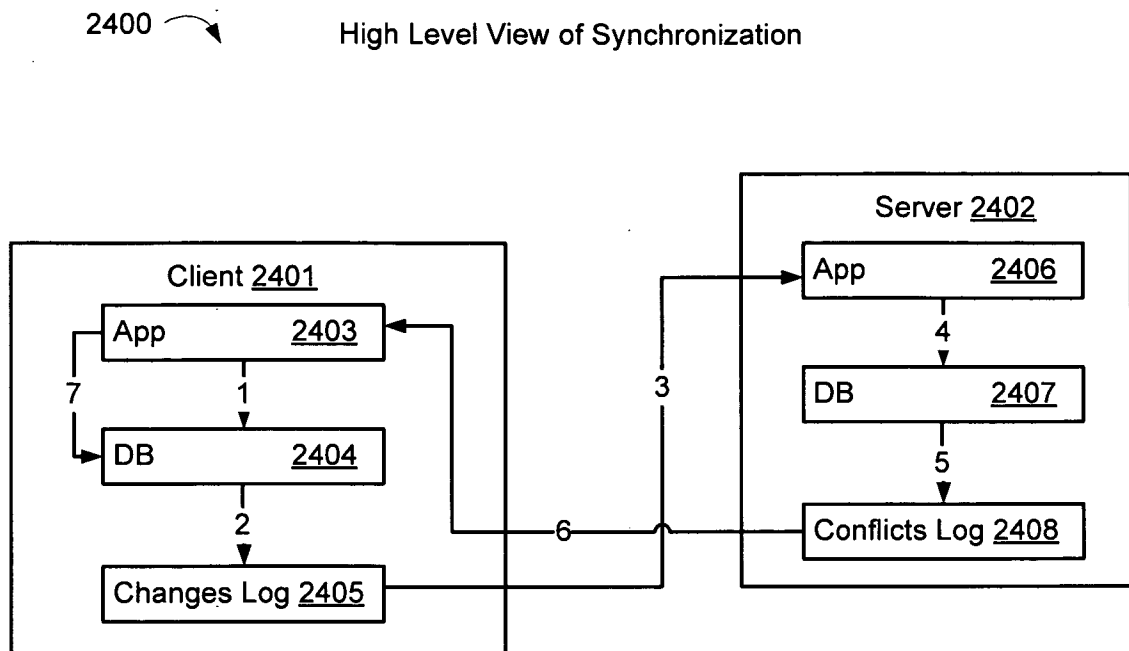


Figure 24

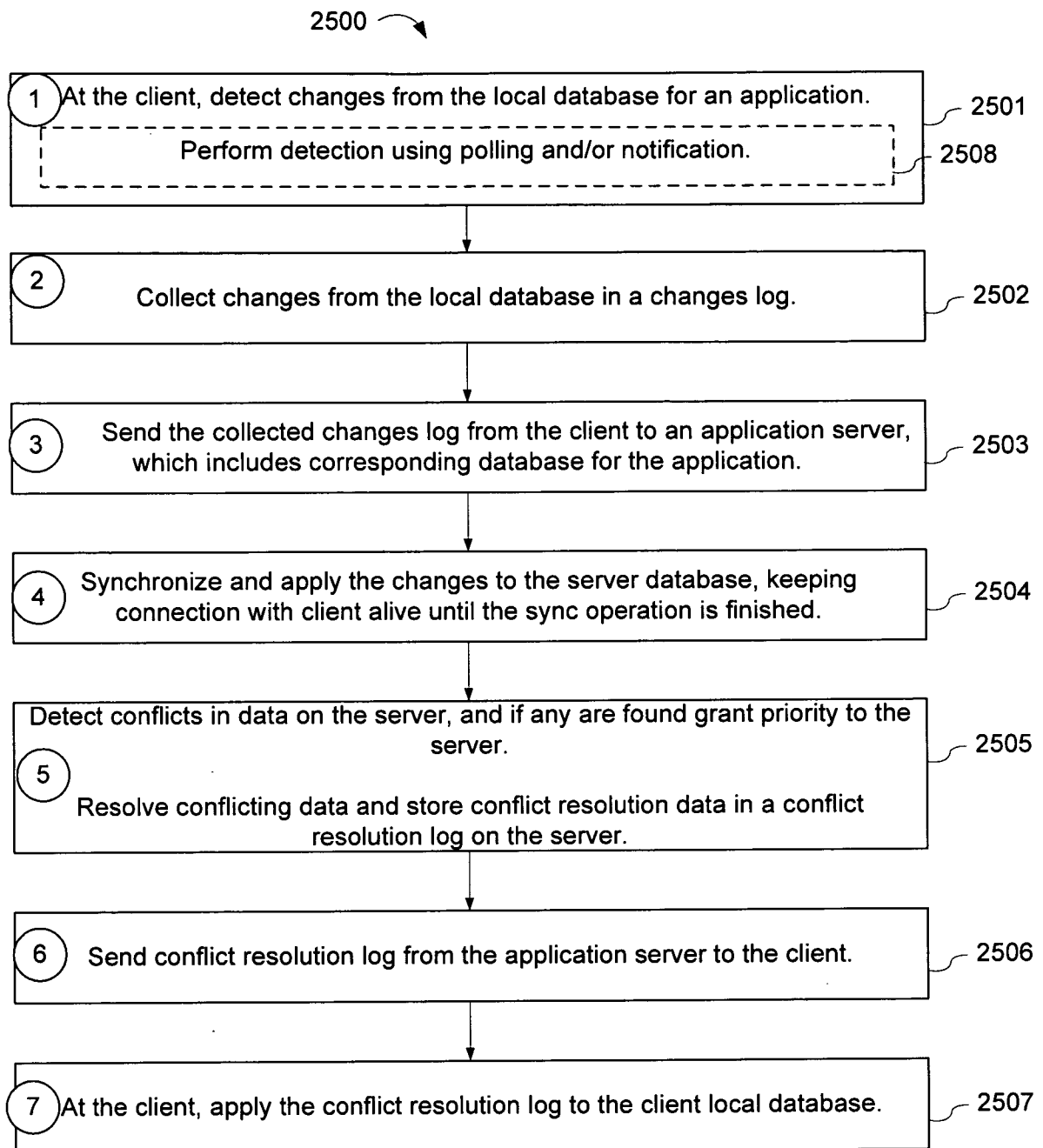


Figure 25

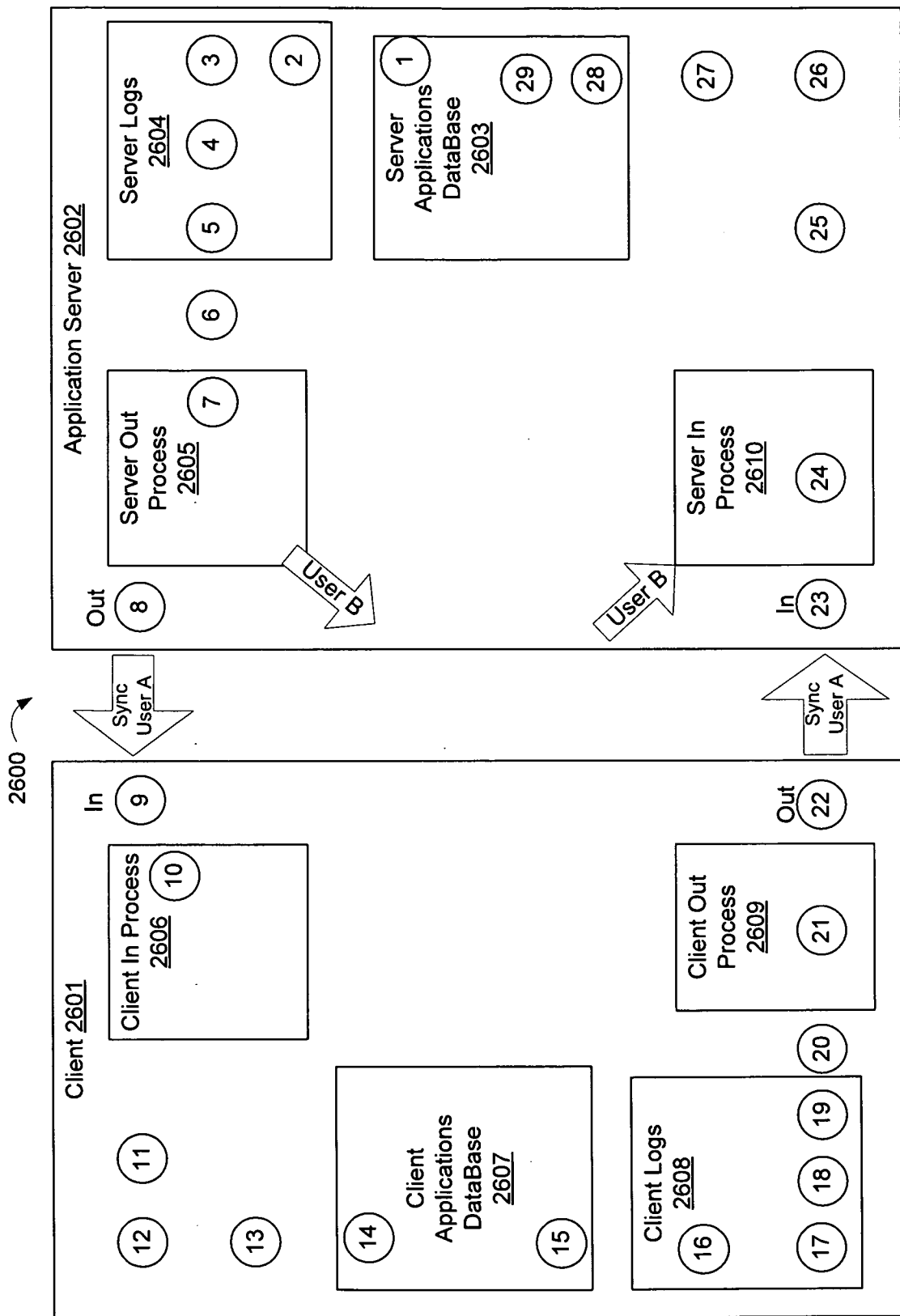


Figure 26

Figure 26

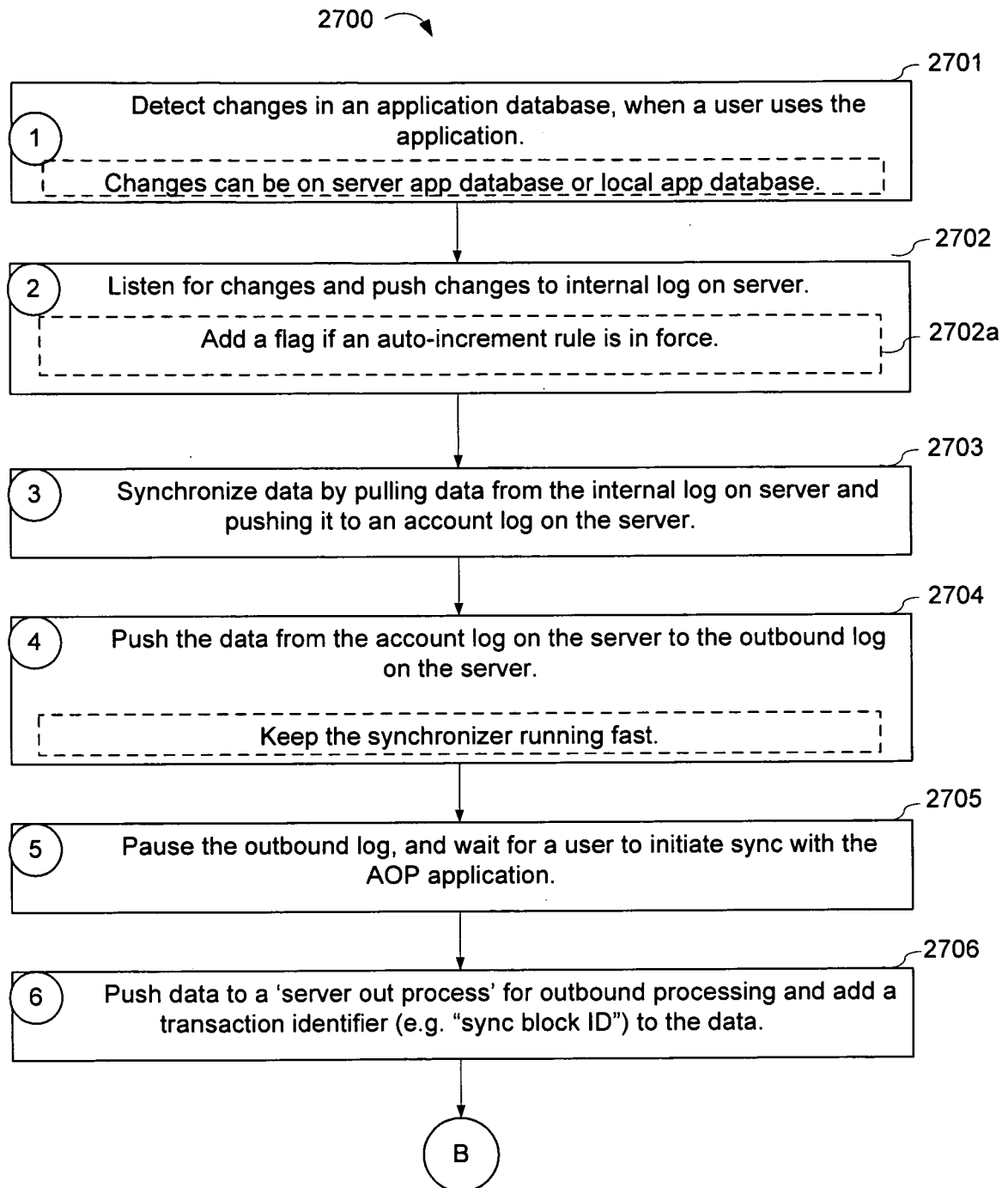


Figure 27A

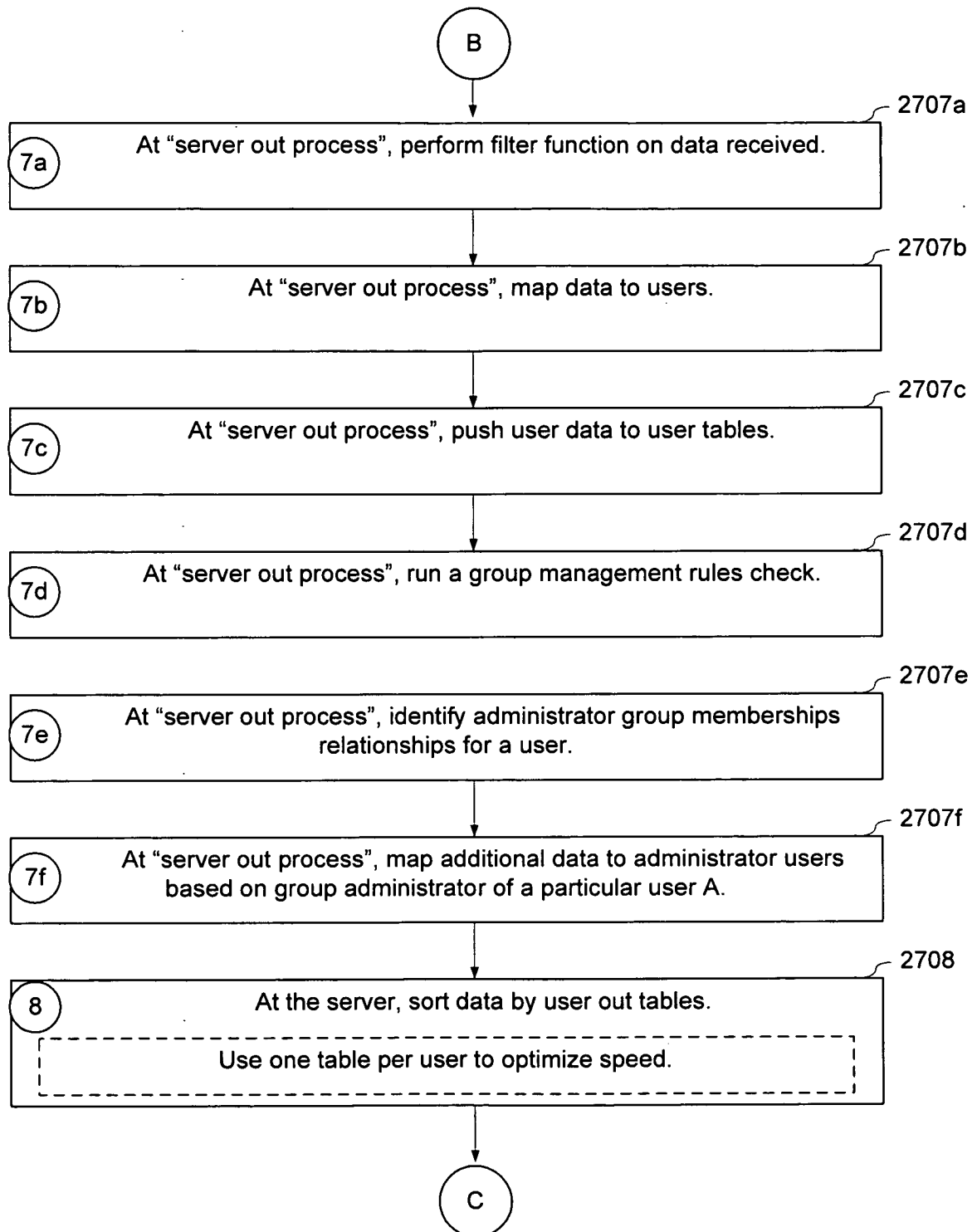


Figure 27B

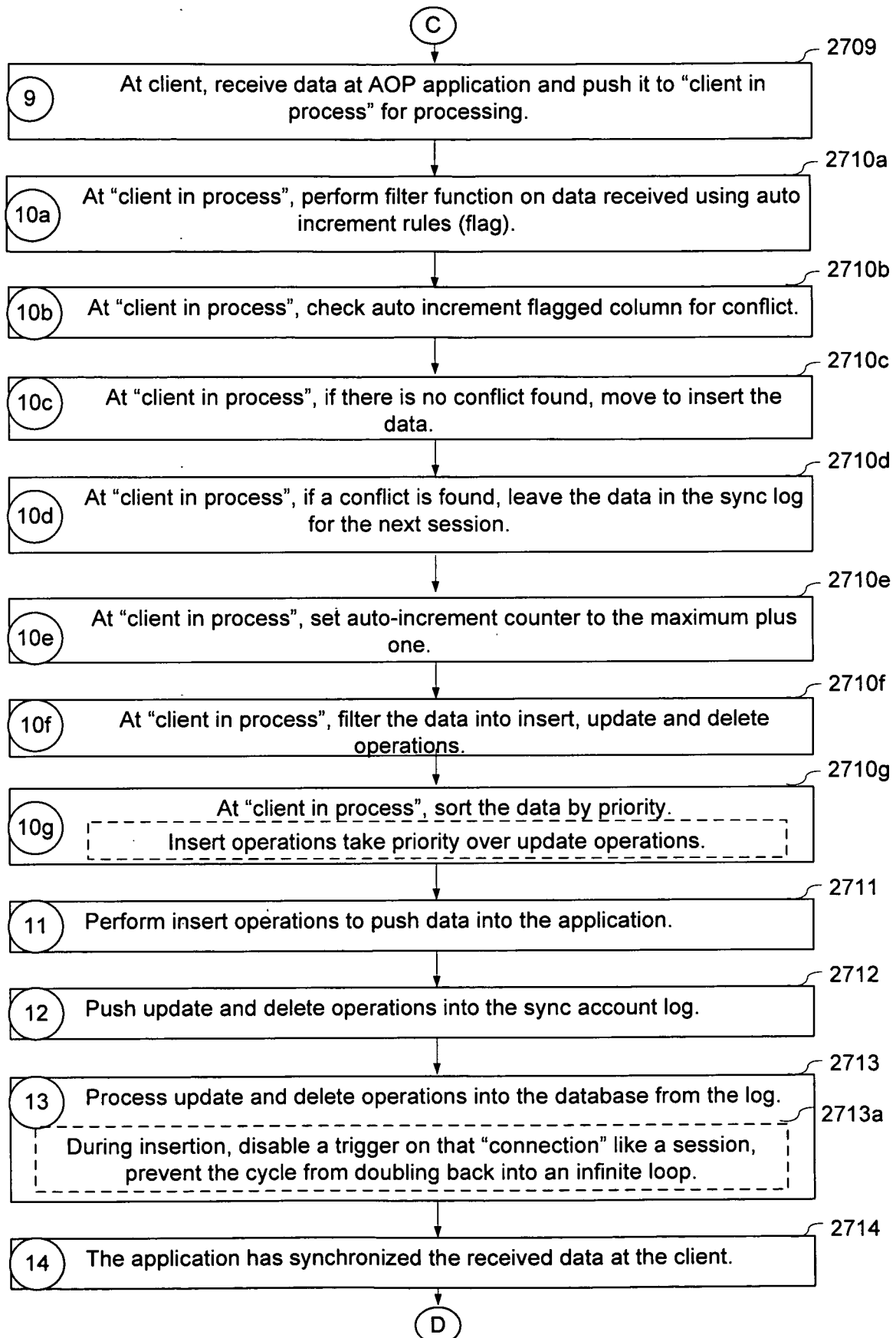


Figure 27C

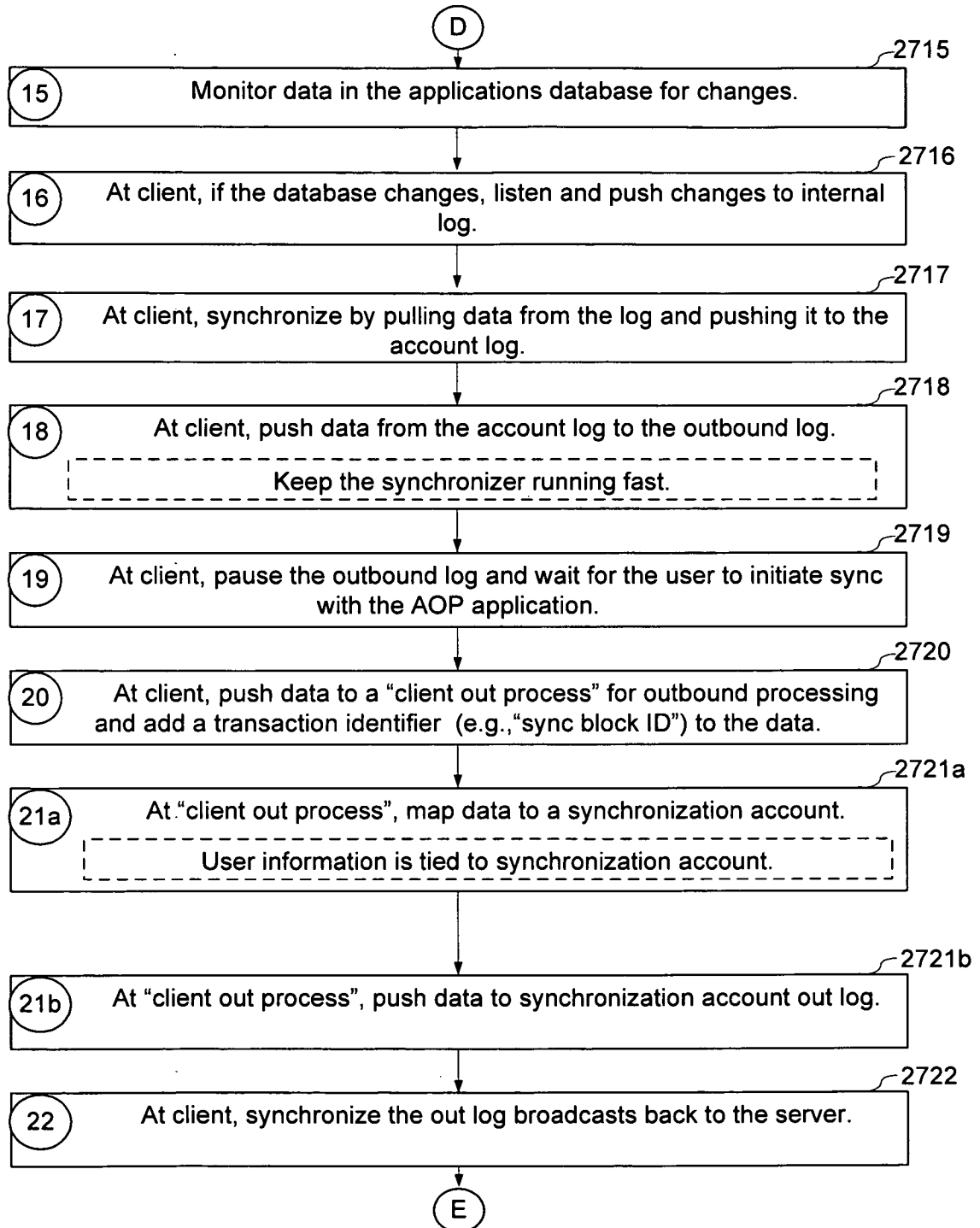


Figure 27D

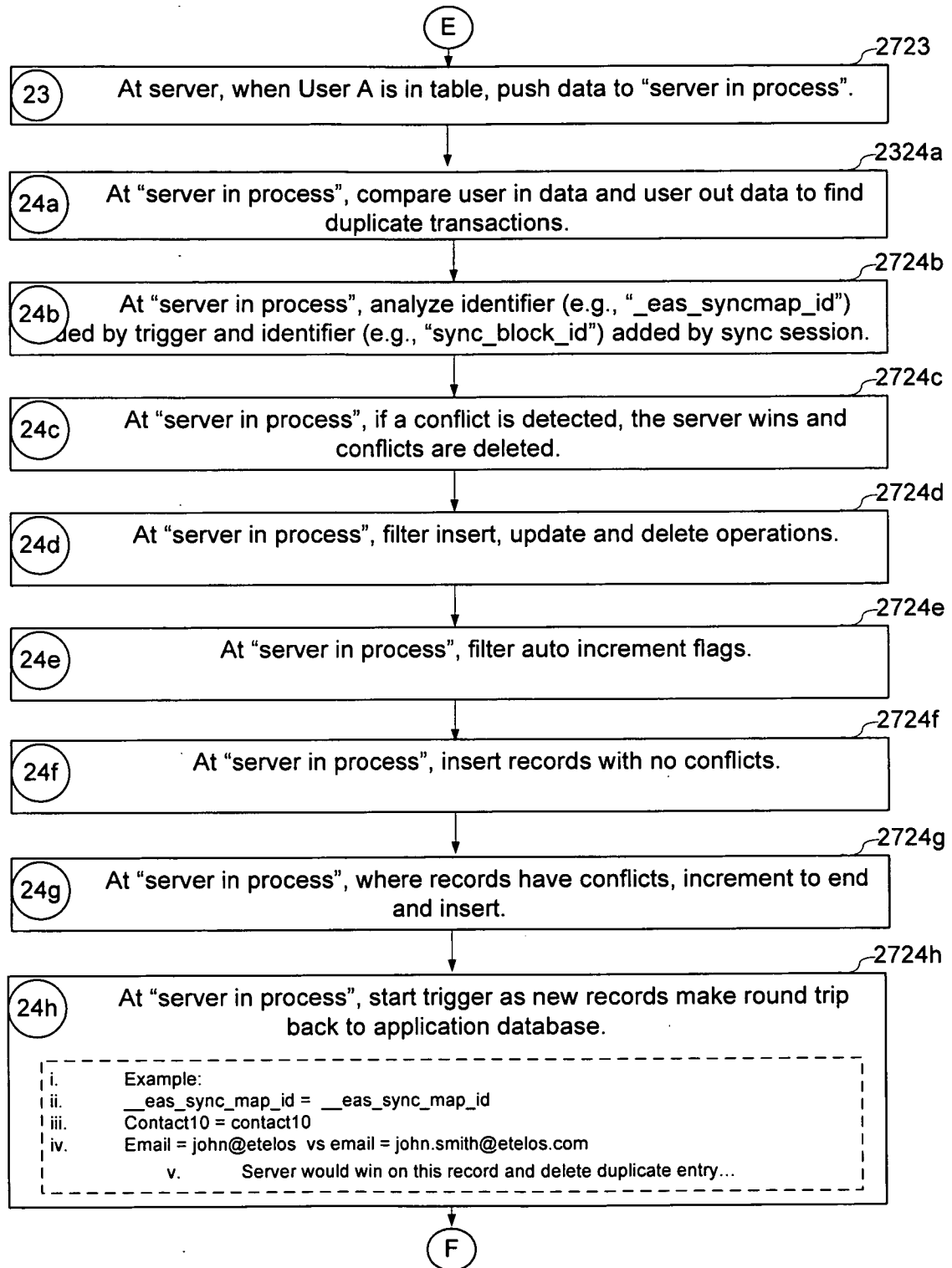


Figure 27E

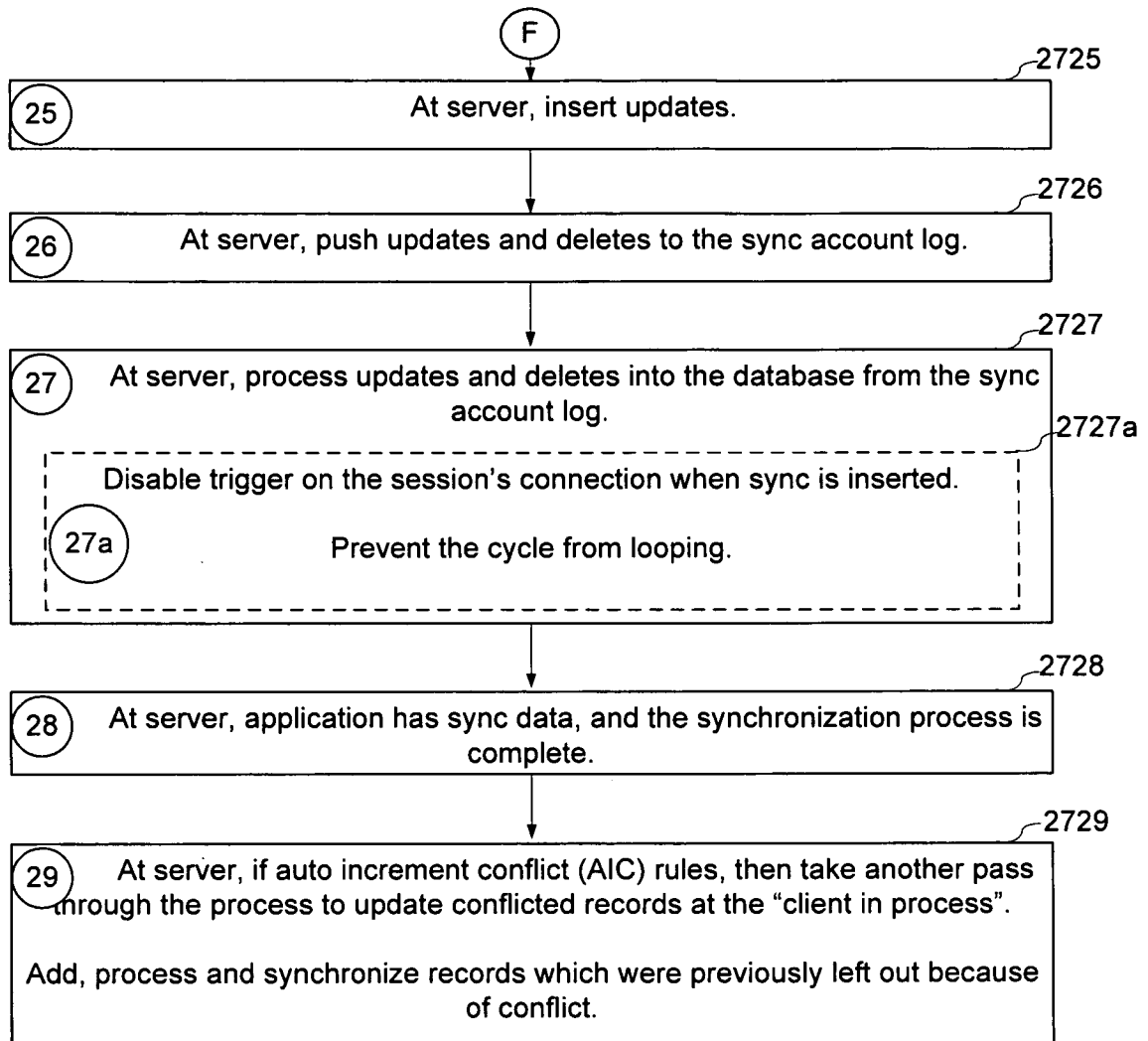


Figure 27F

2800

Auto Increment Conflict Resolution

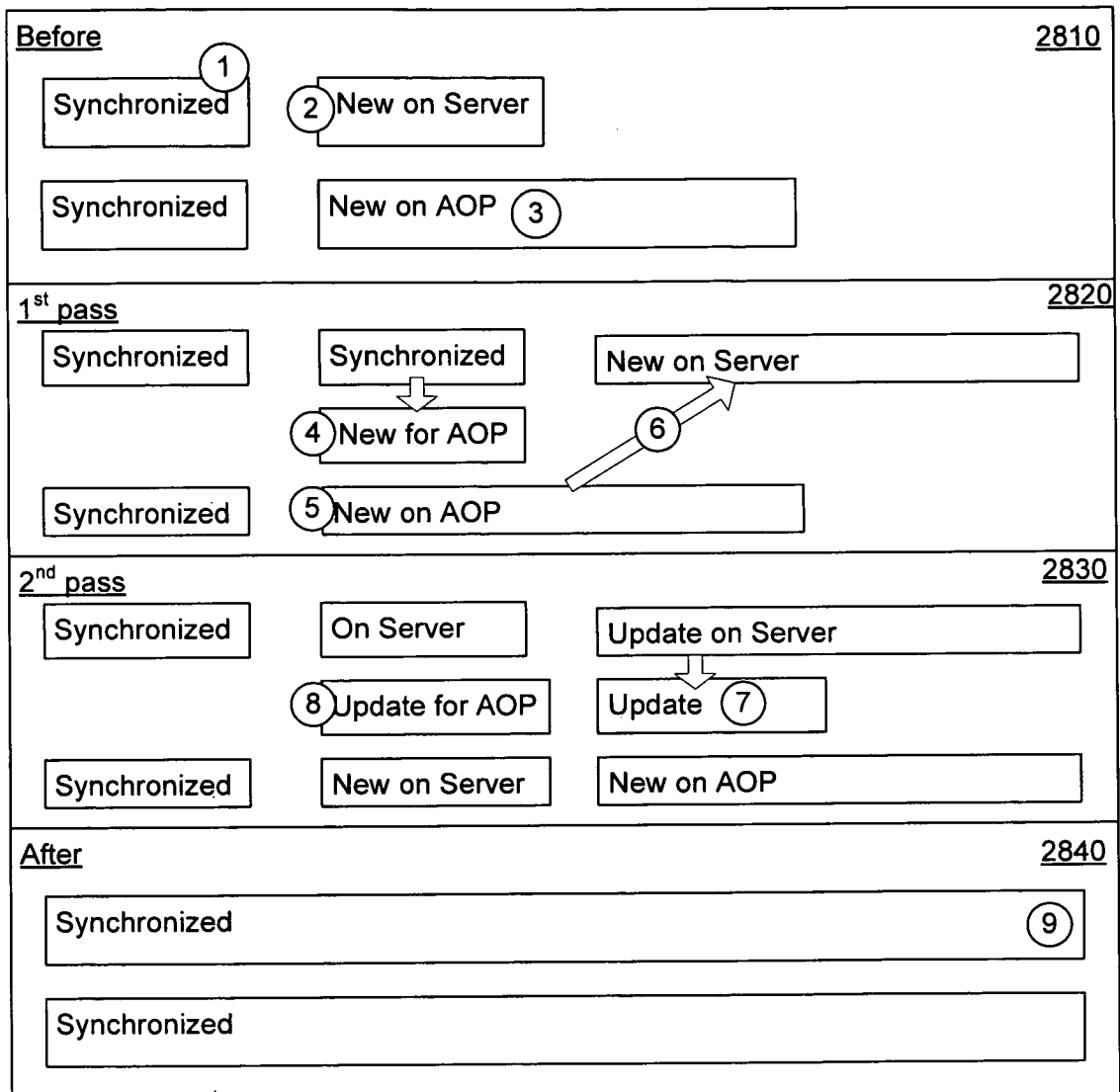


Figure 28

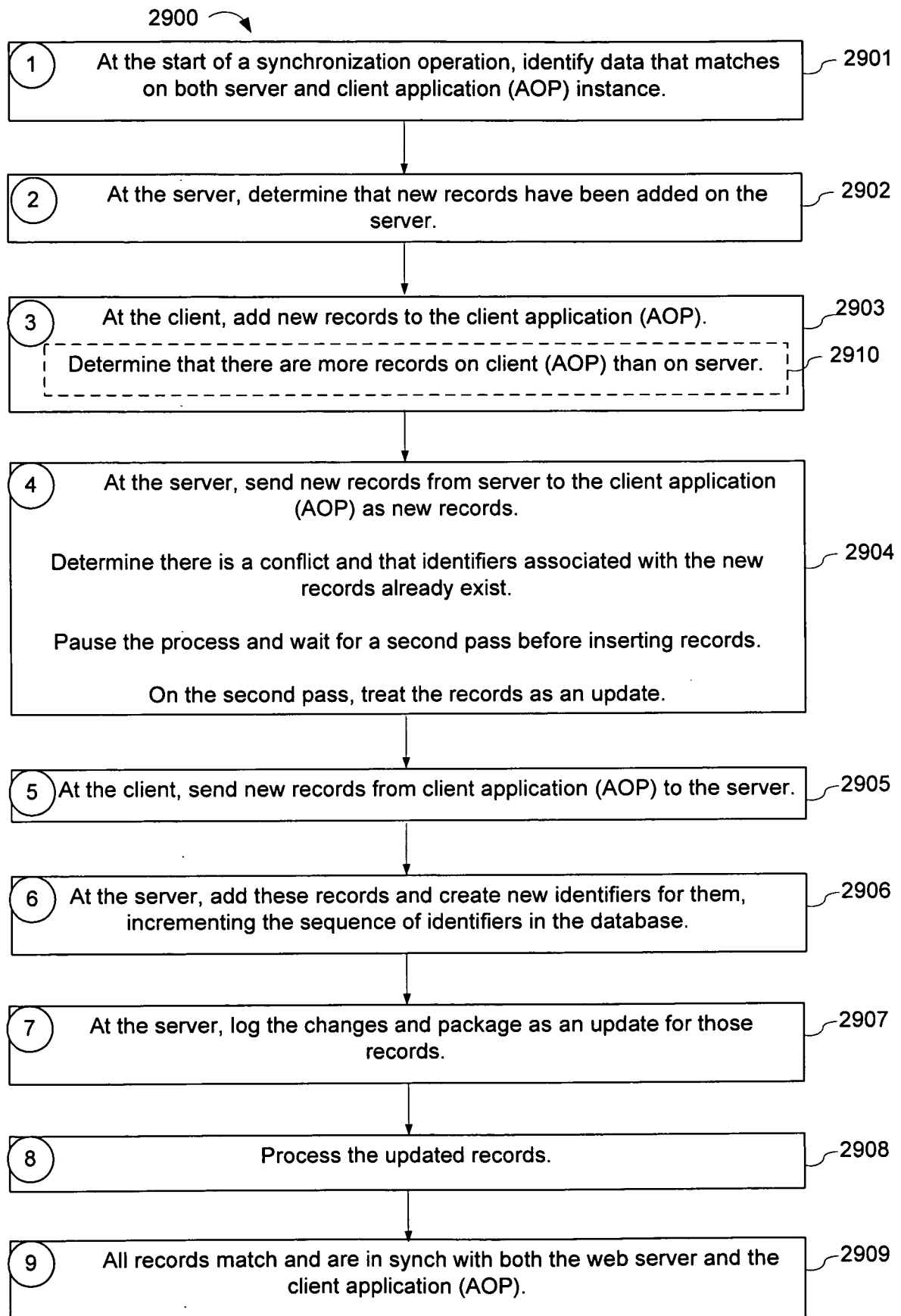


Figure 29

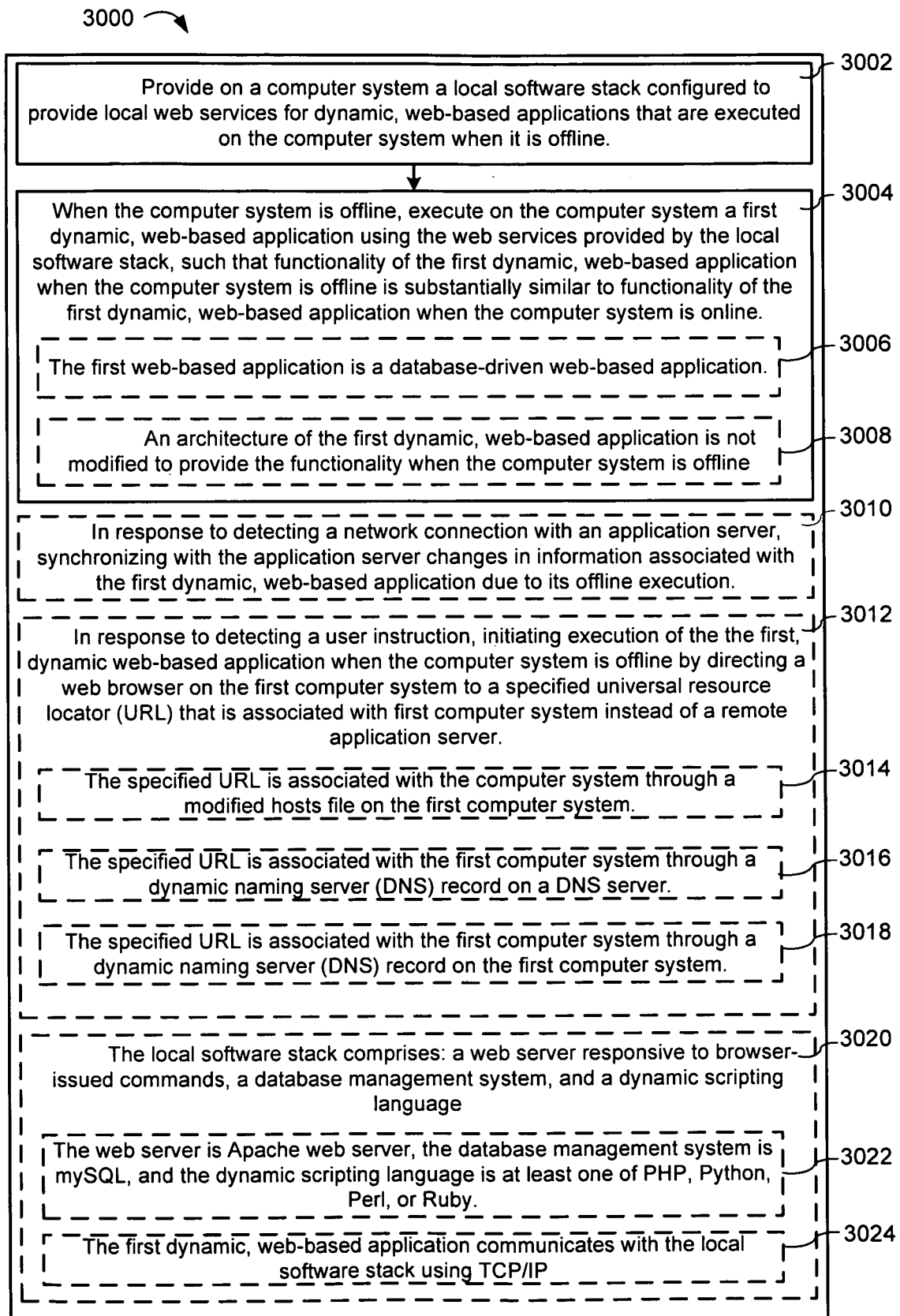


Figure 30

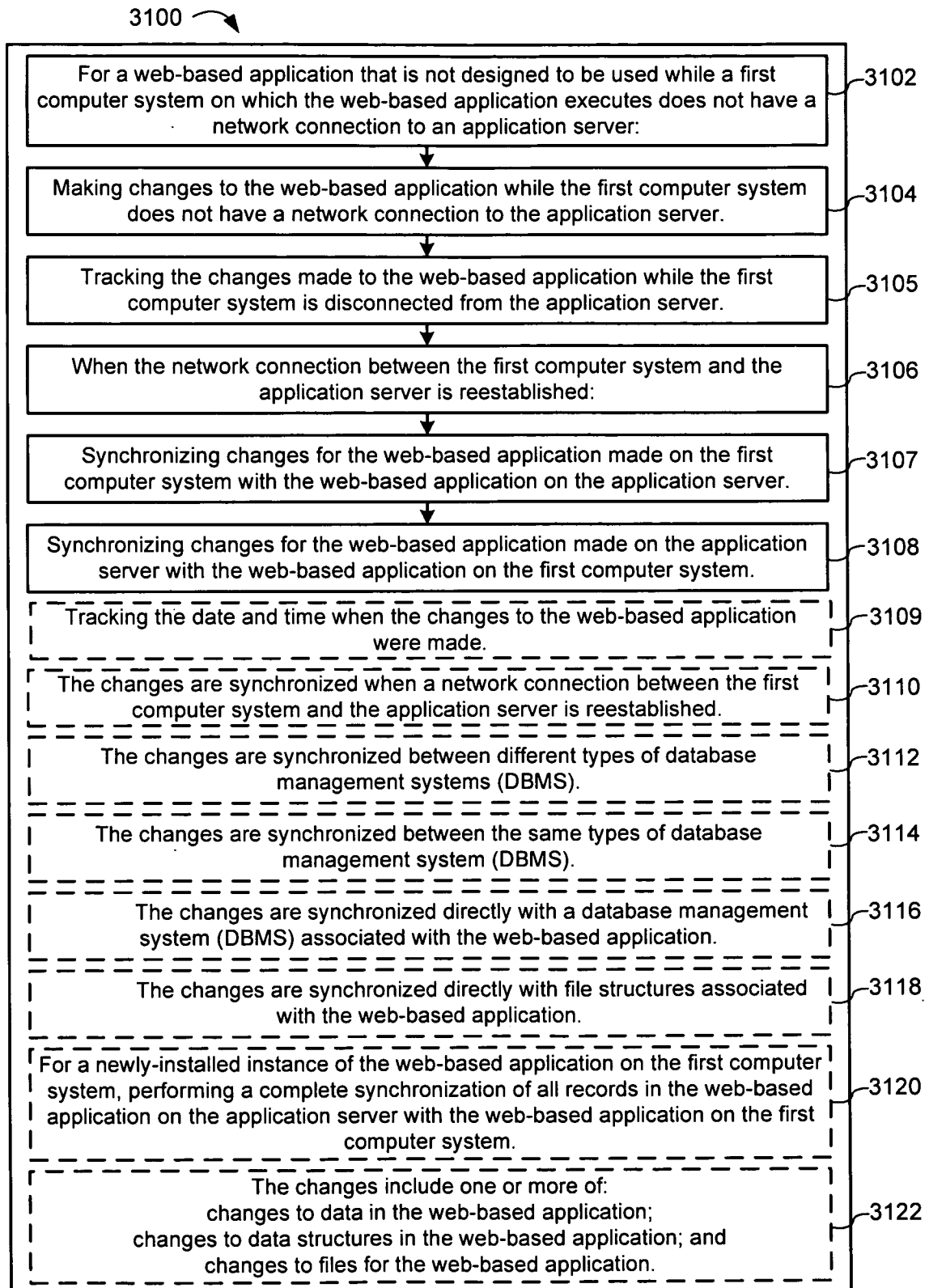


Figure 31

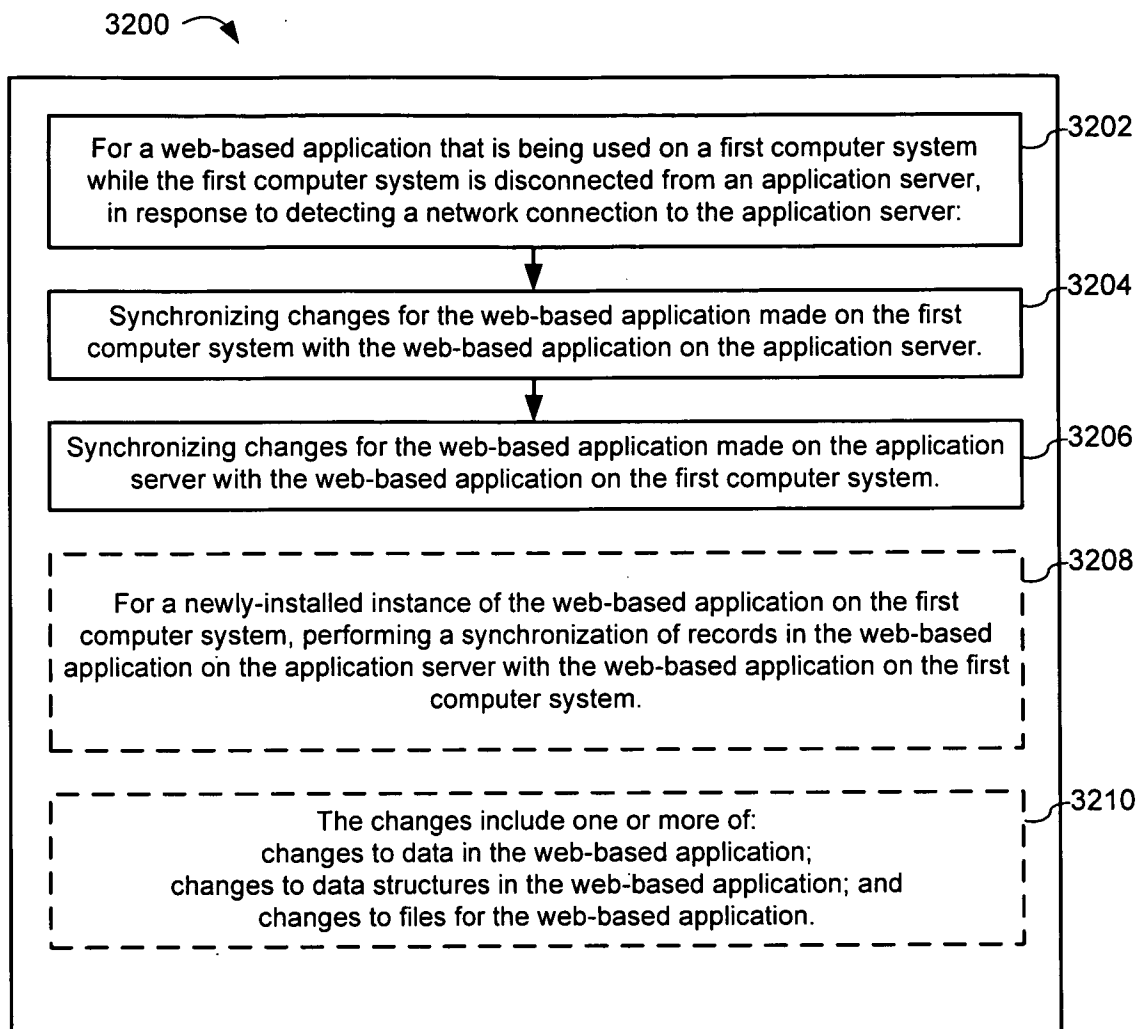


Figure 32

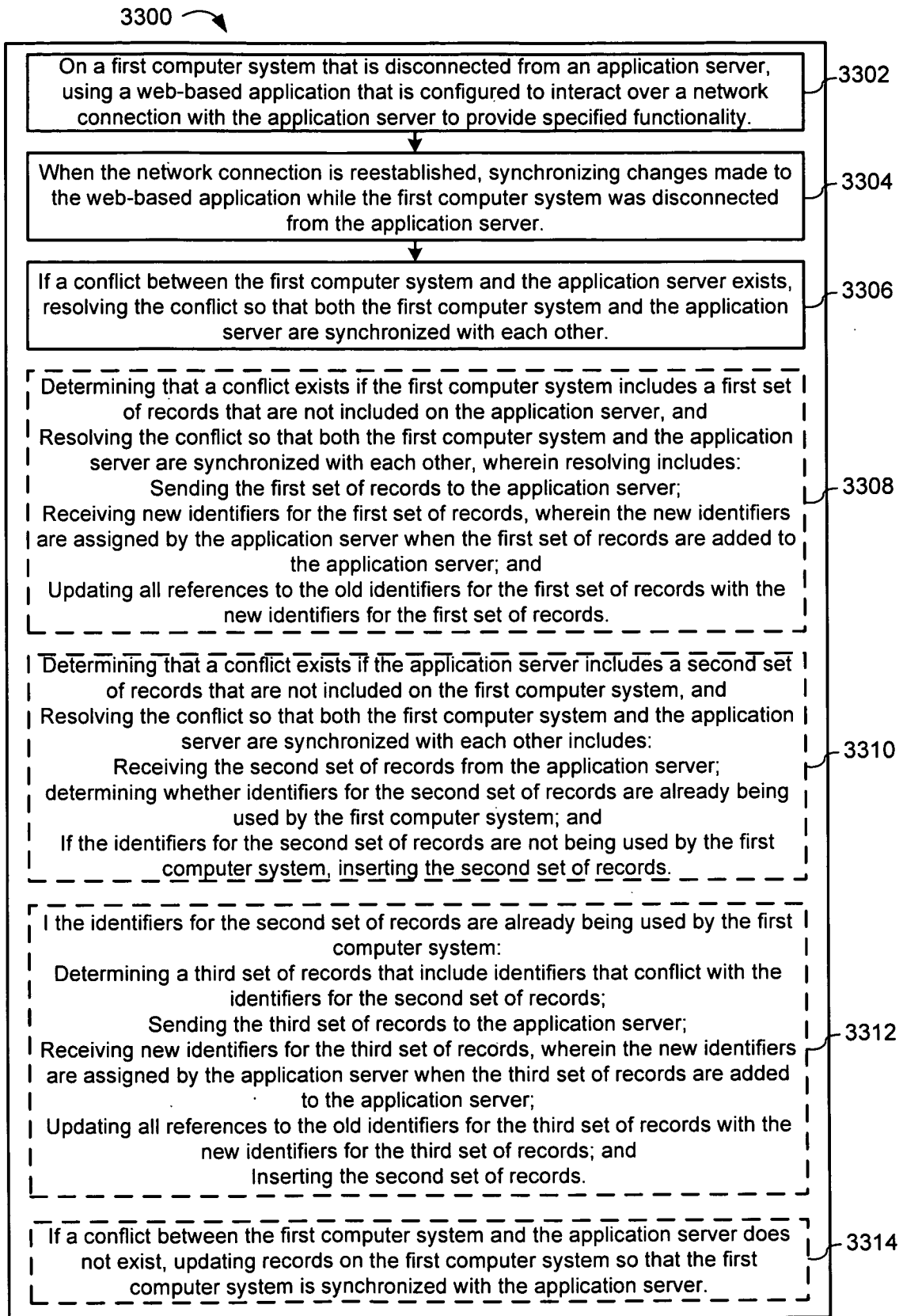


Figure 33

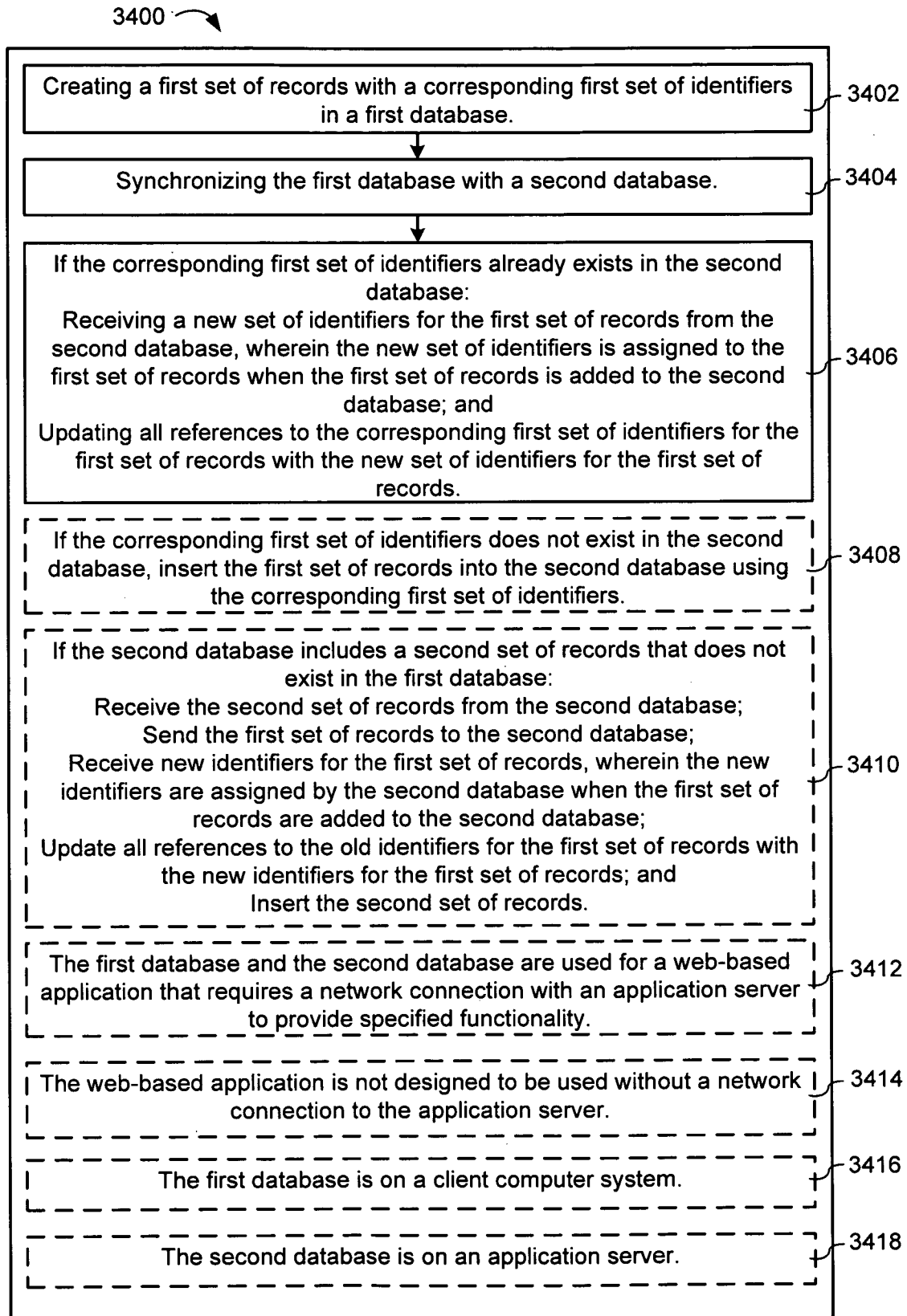


Figure 34

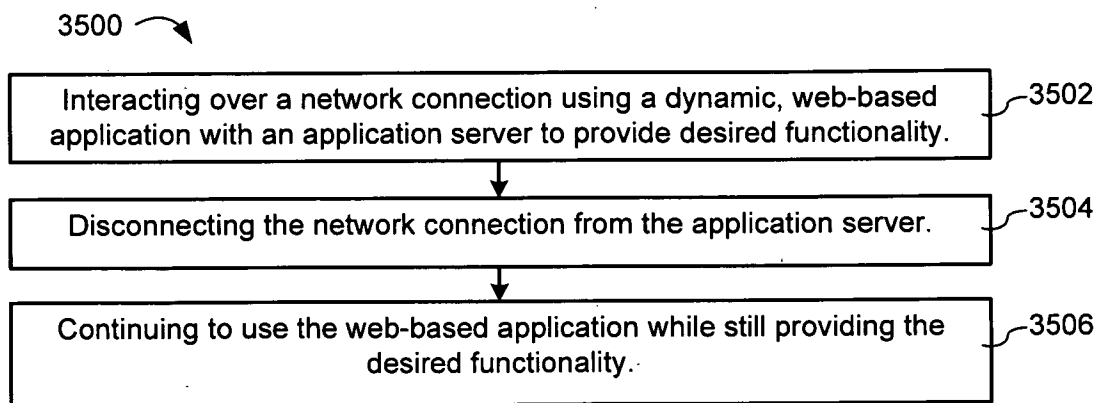


Figure 35

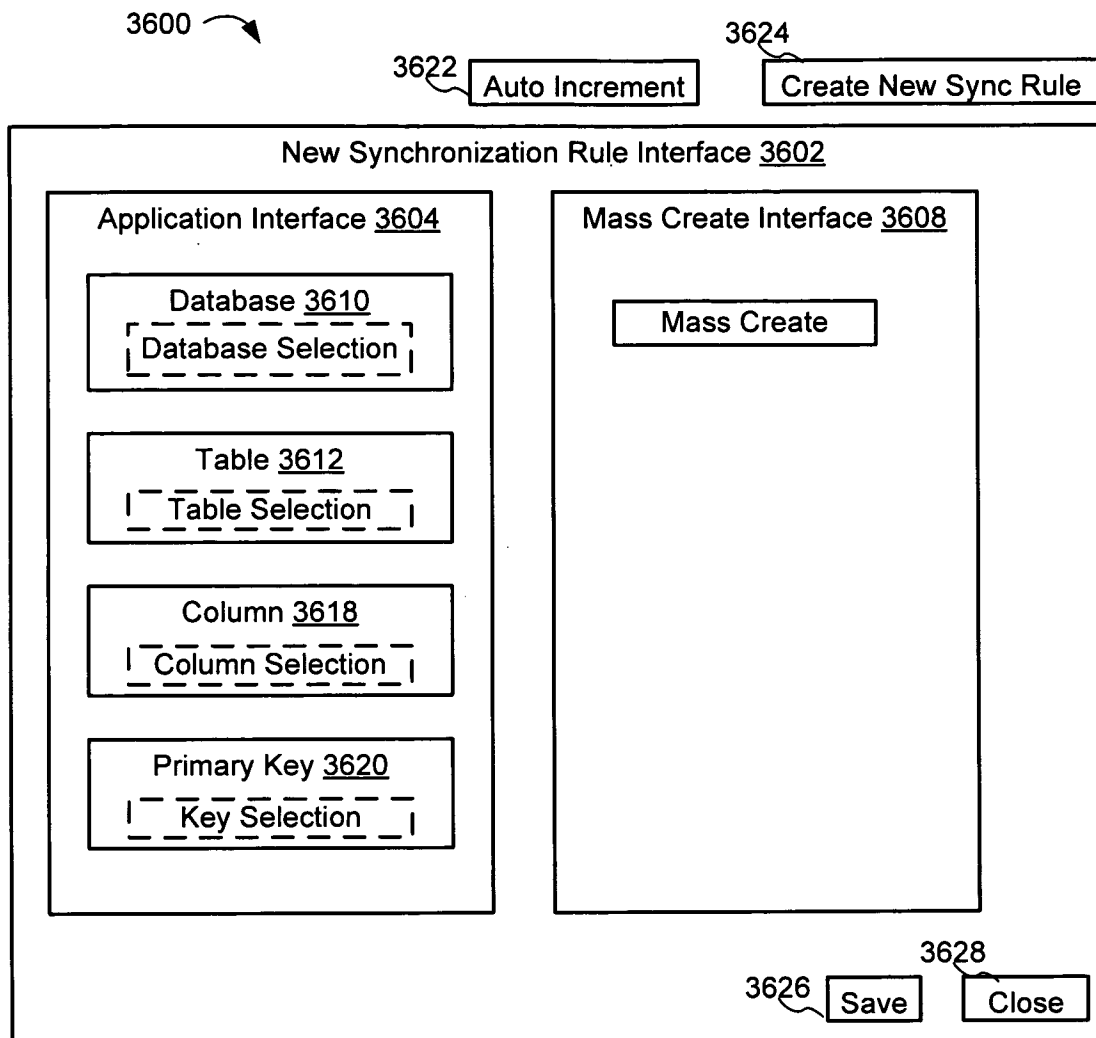


Figure 36

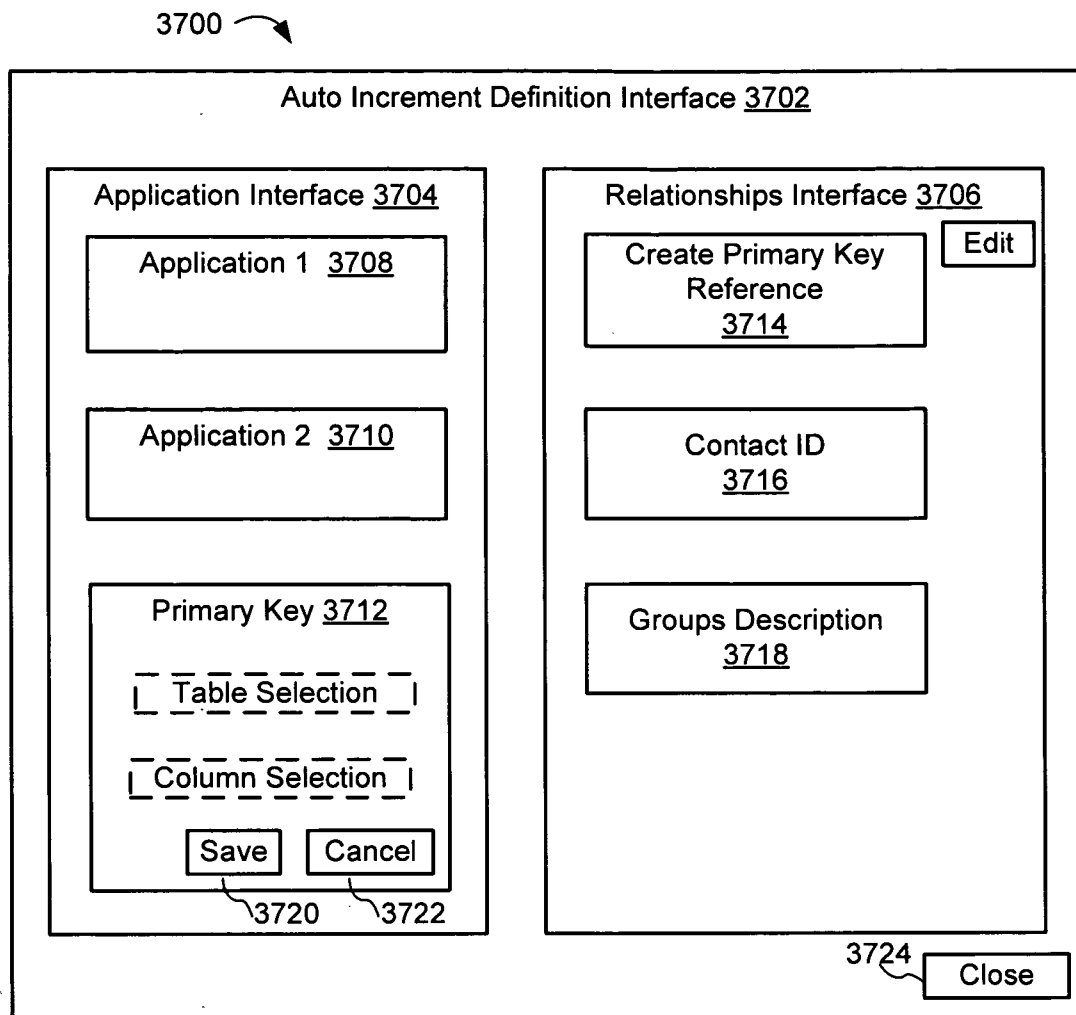


Figure 37

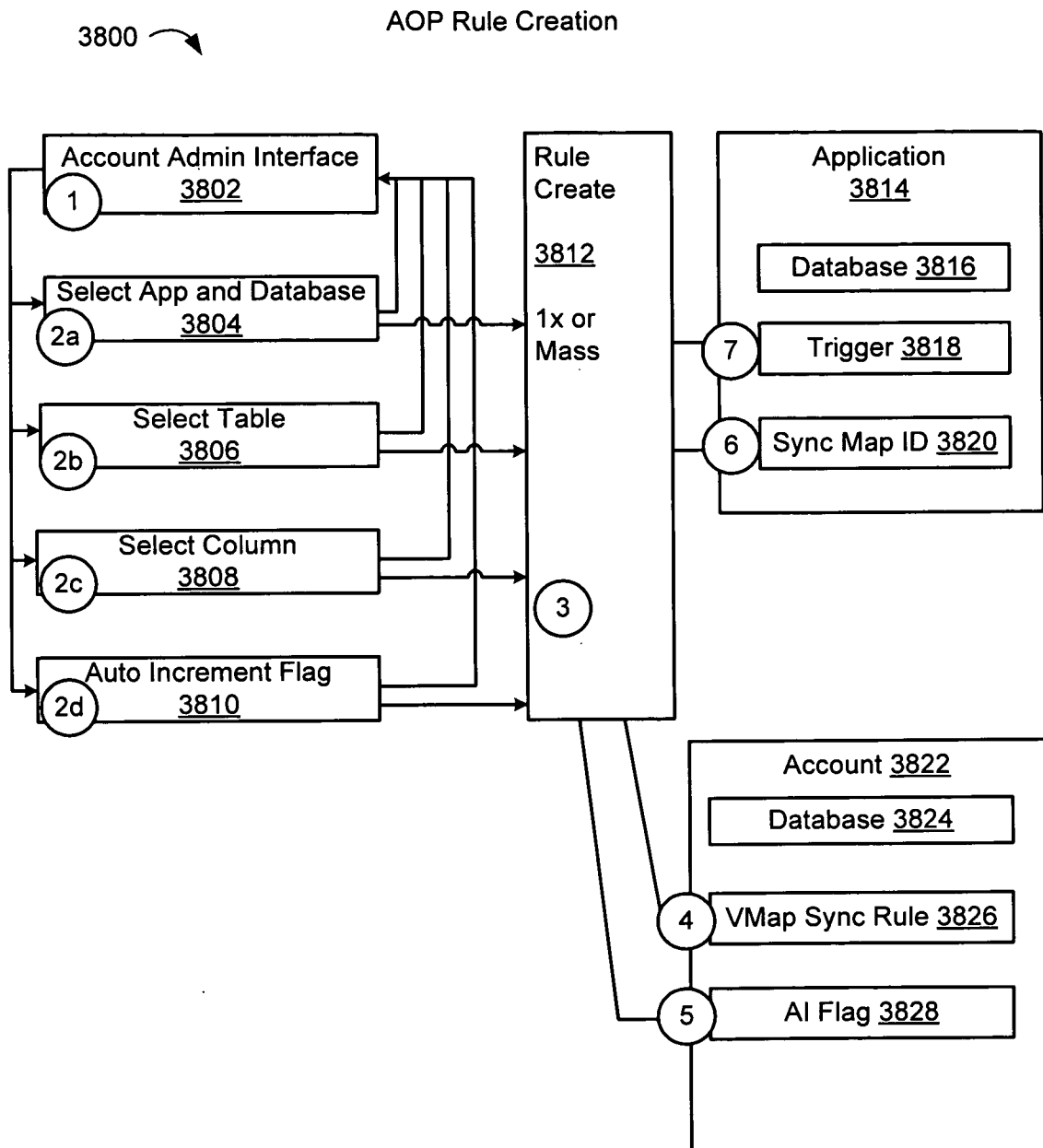


Figure 38

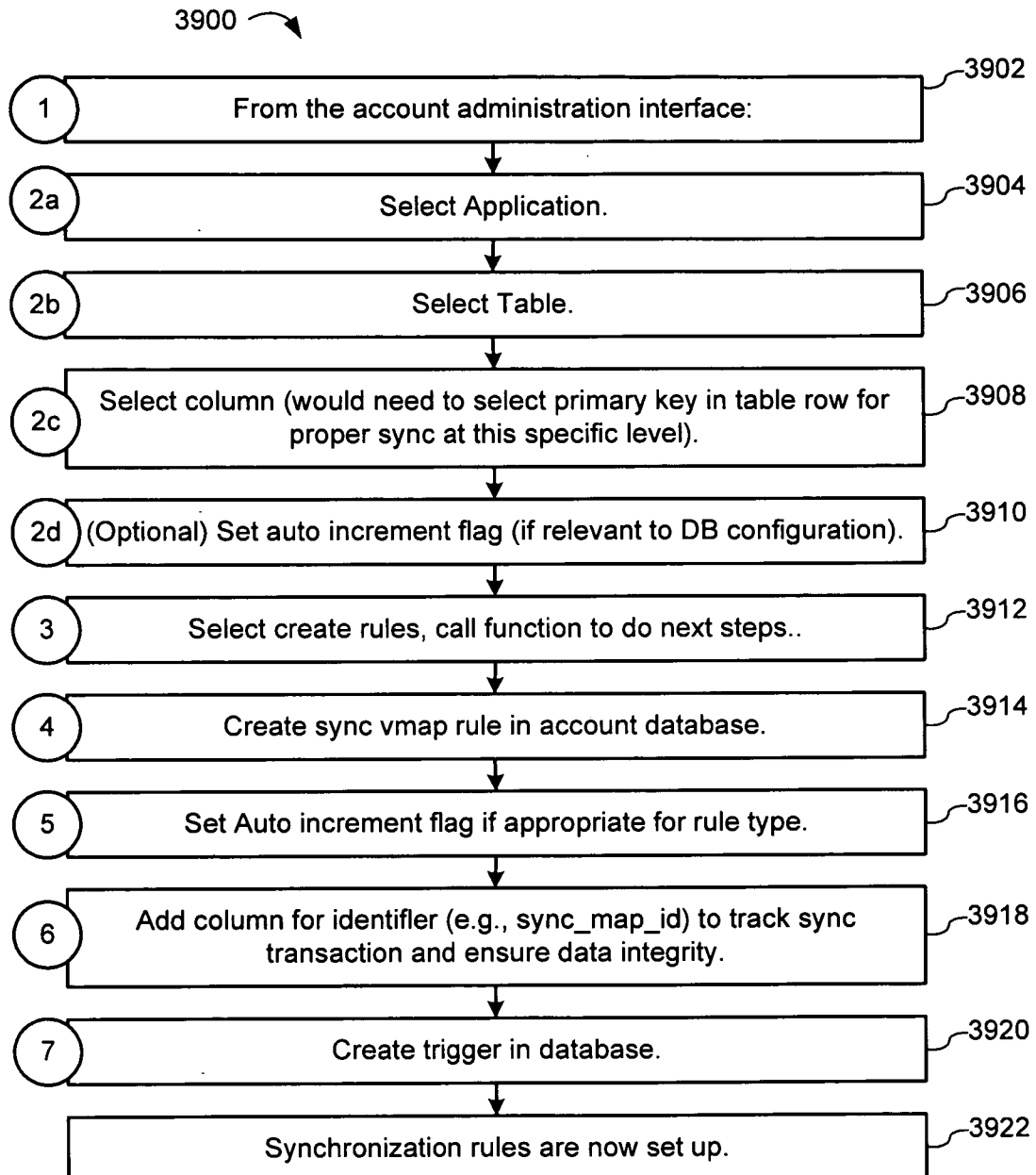


Figure 39

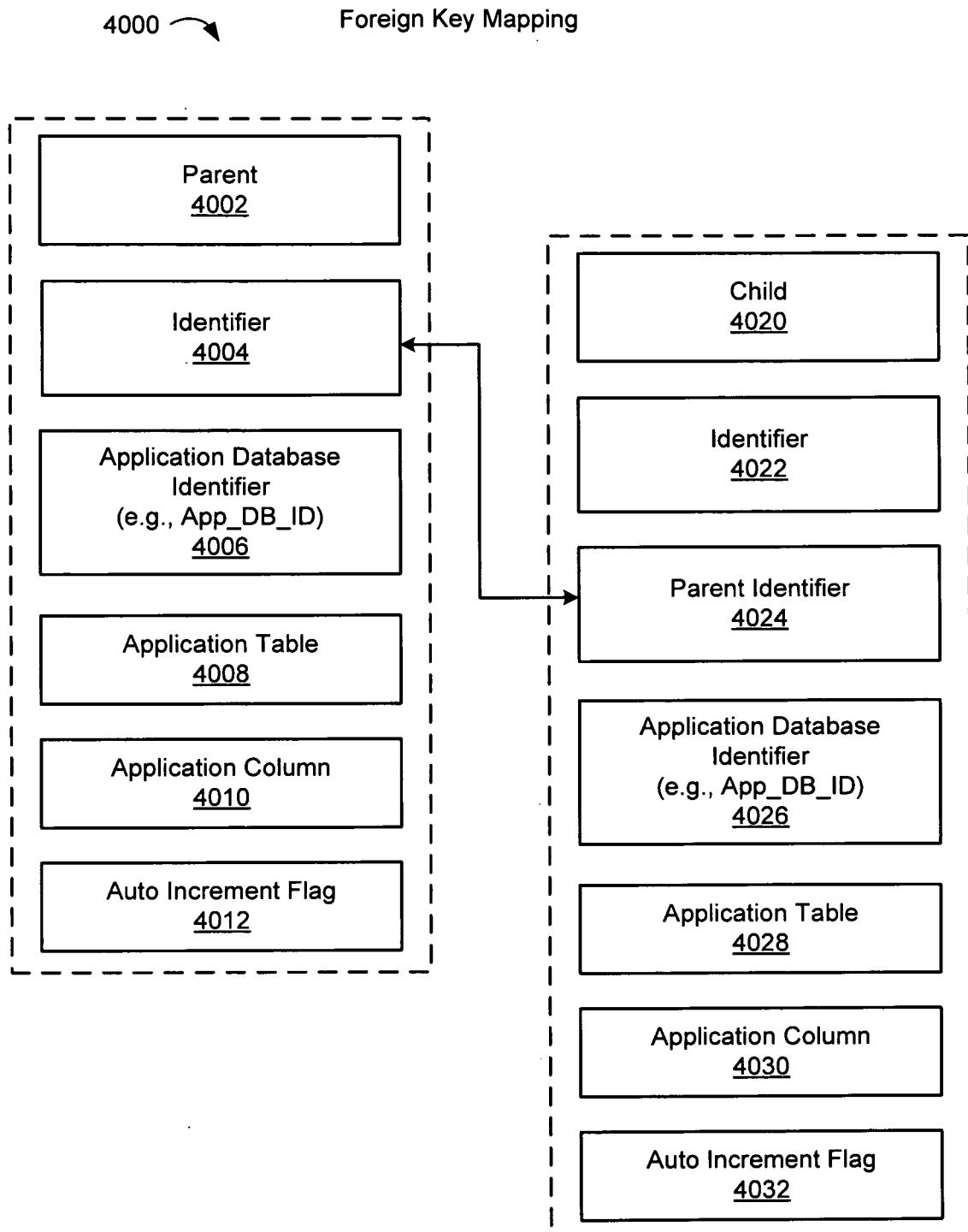


Figure 40

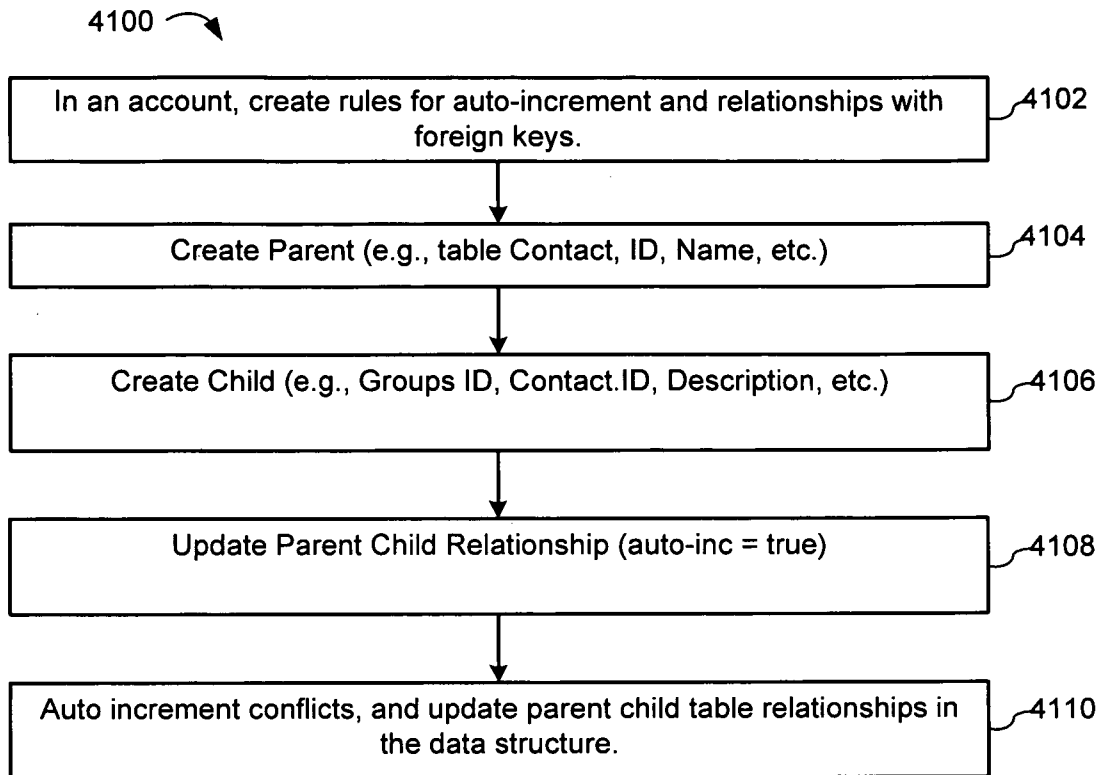


Figure 41

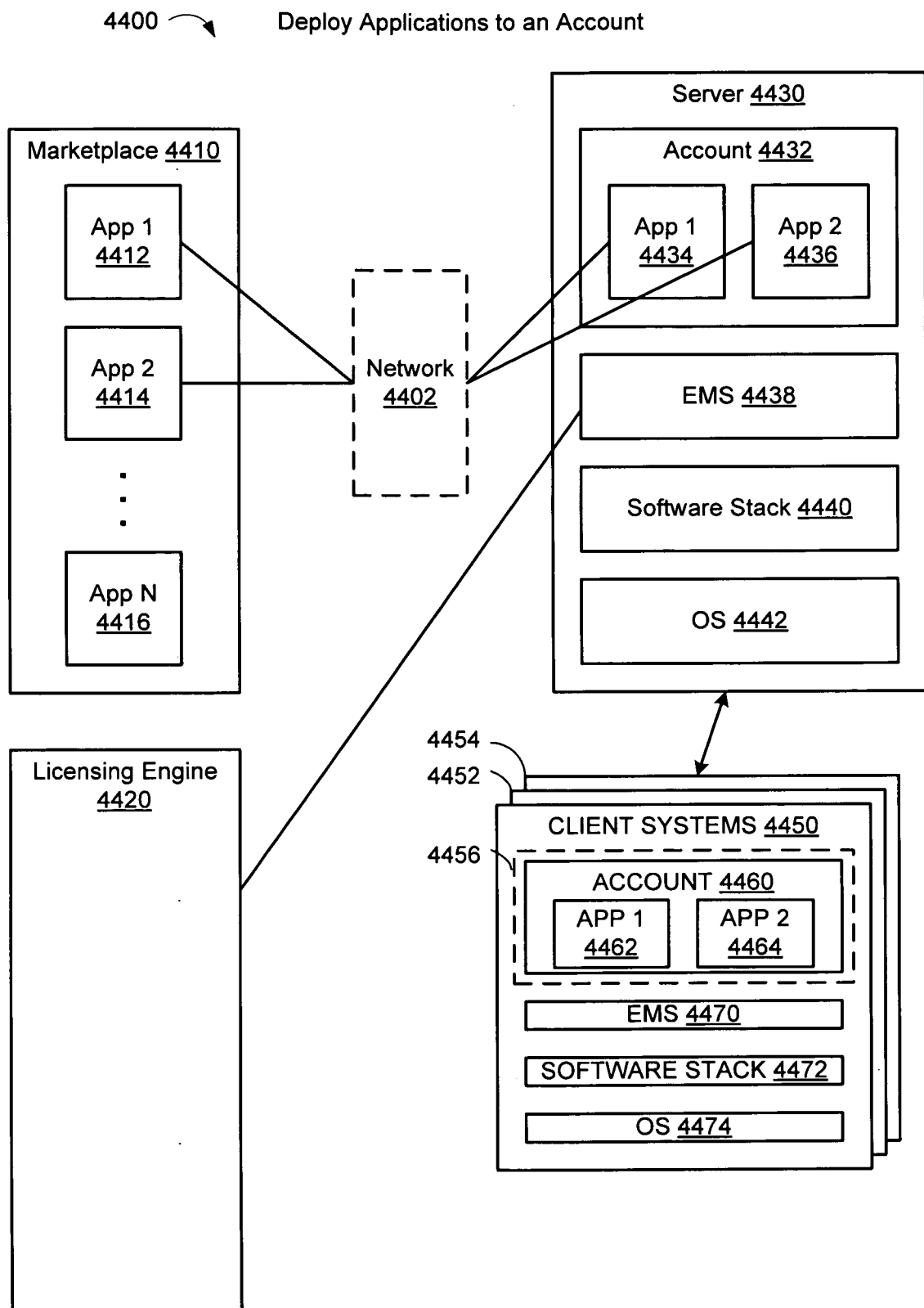


Figure 44

Multitenancy License Management

4500 ↗

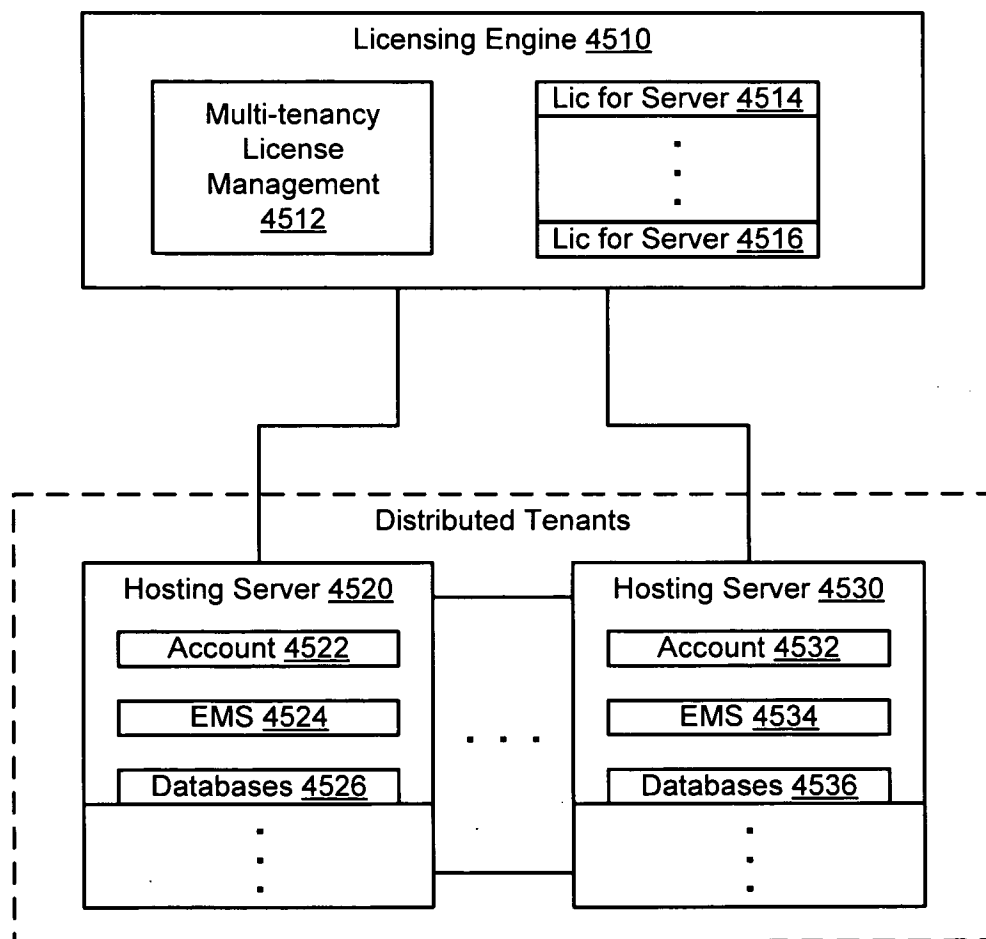


Figure 45

4600 → Web Application Marketplace and Hosting Infrastructure

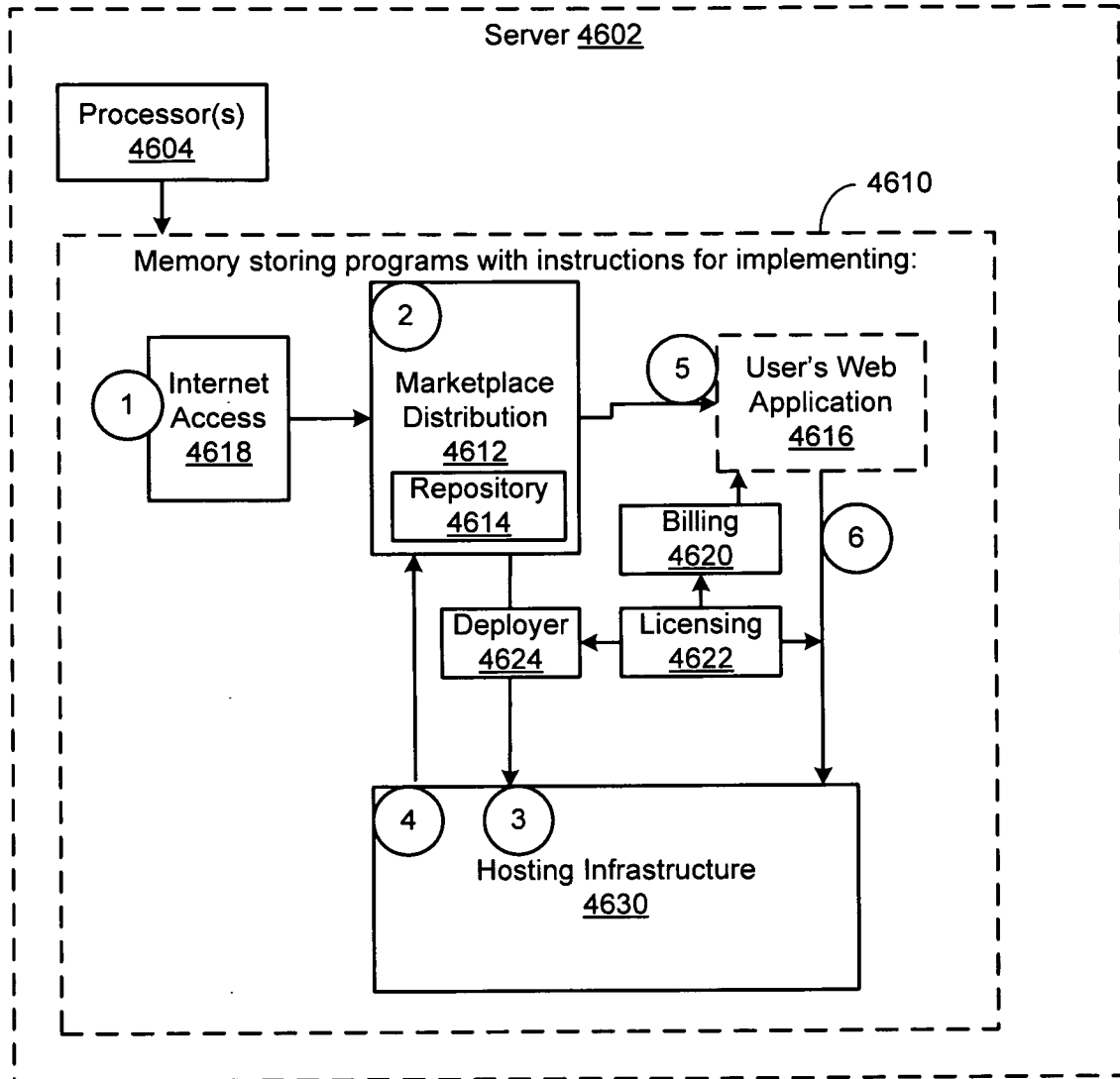


Figure 46

Web Application Marketplace and Hosting Infrastructure

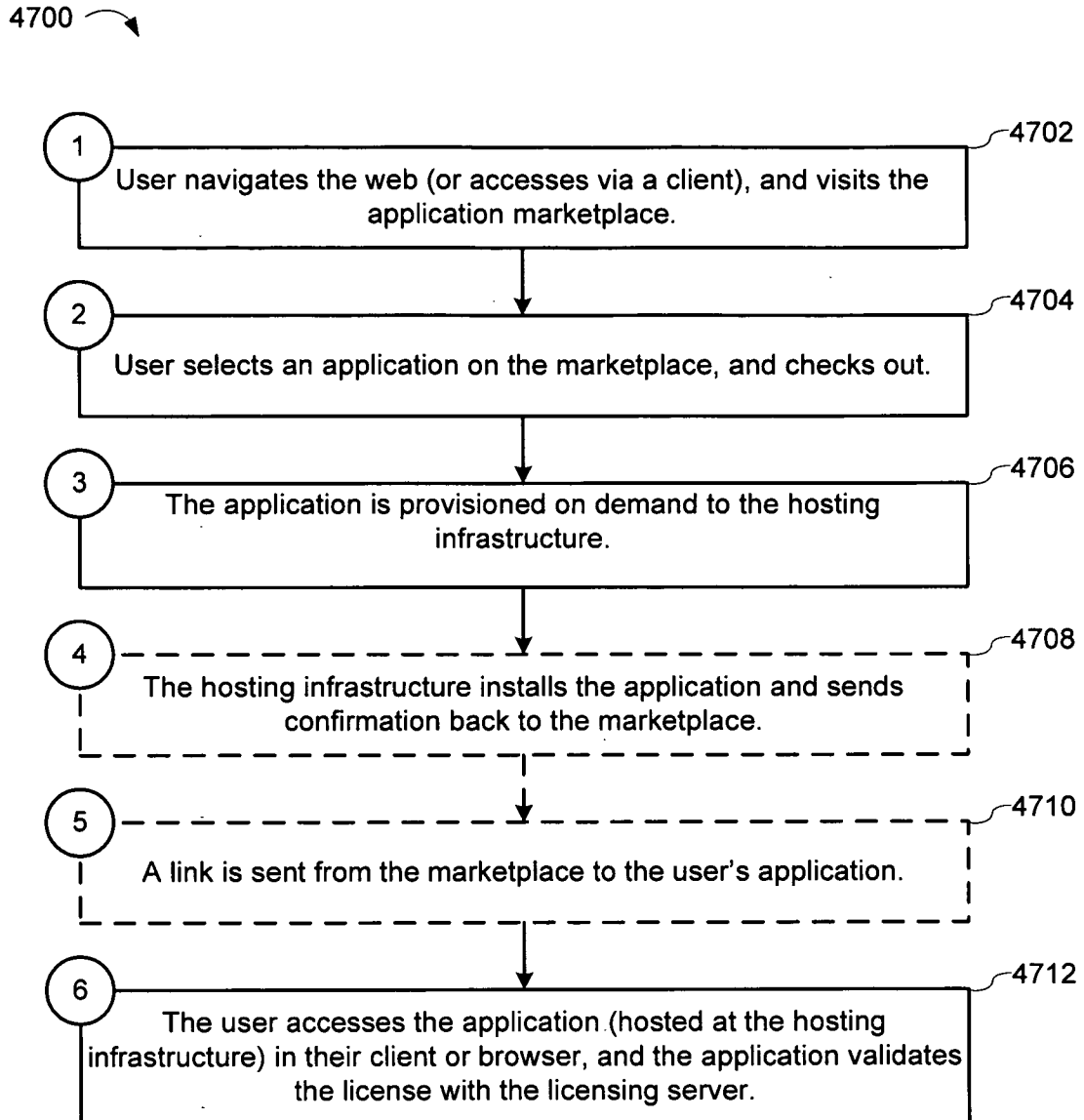


Figure 47

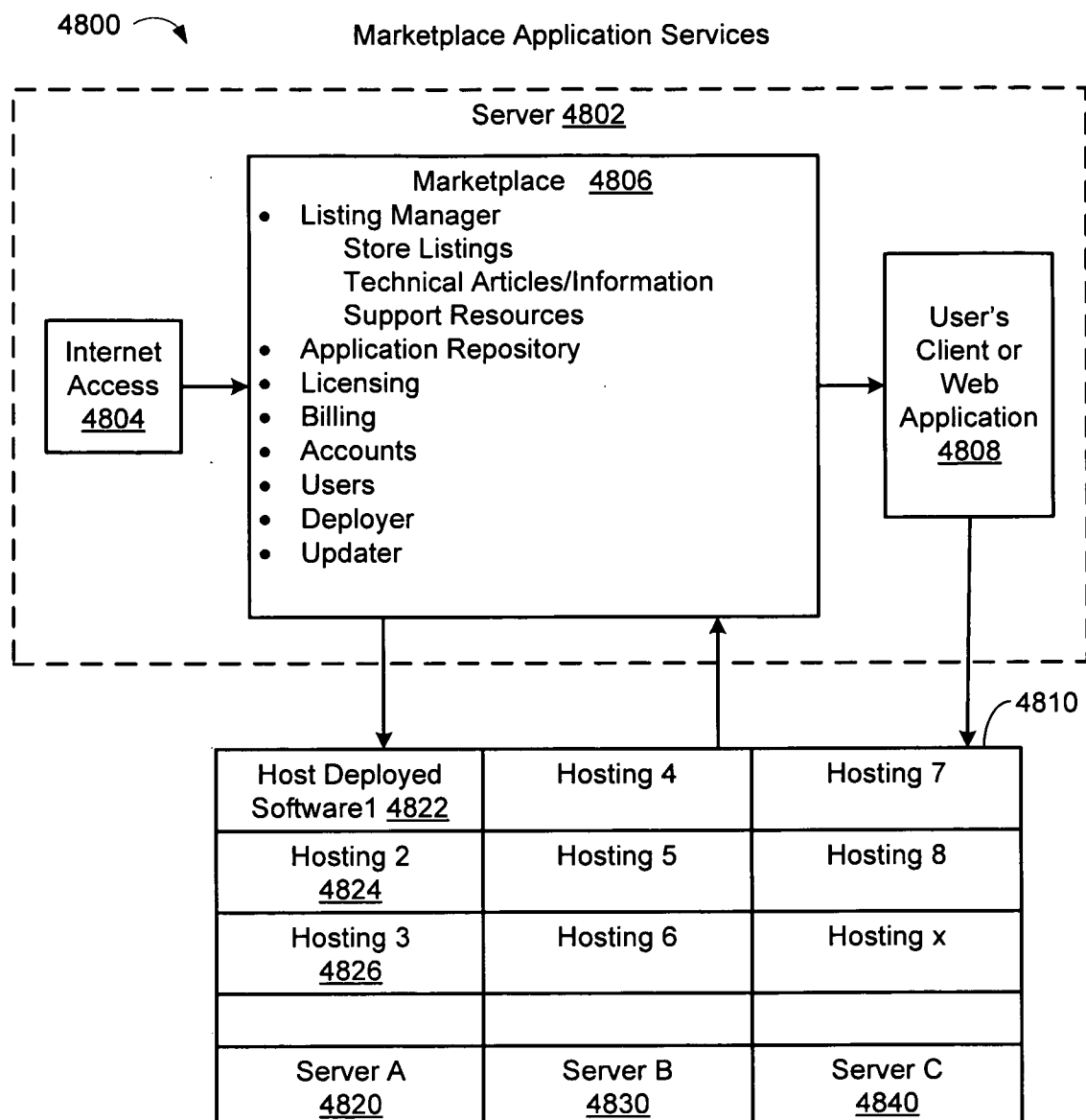


Figure 48
51/178

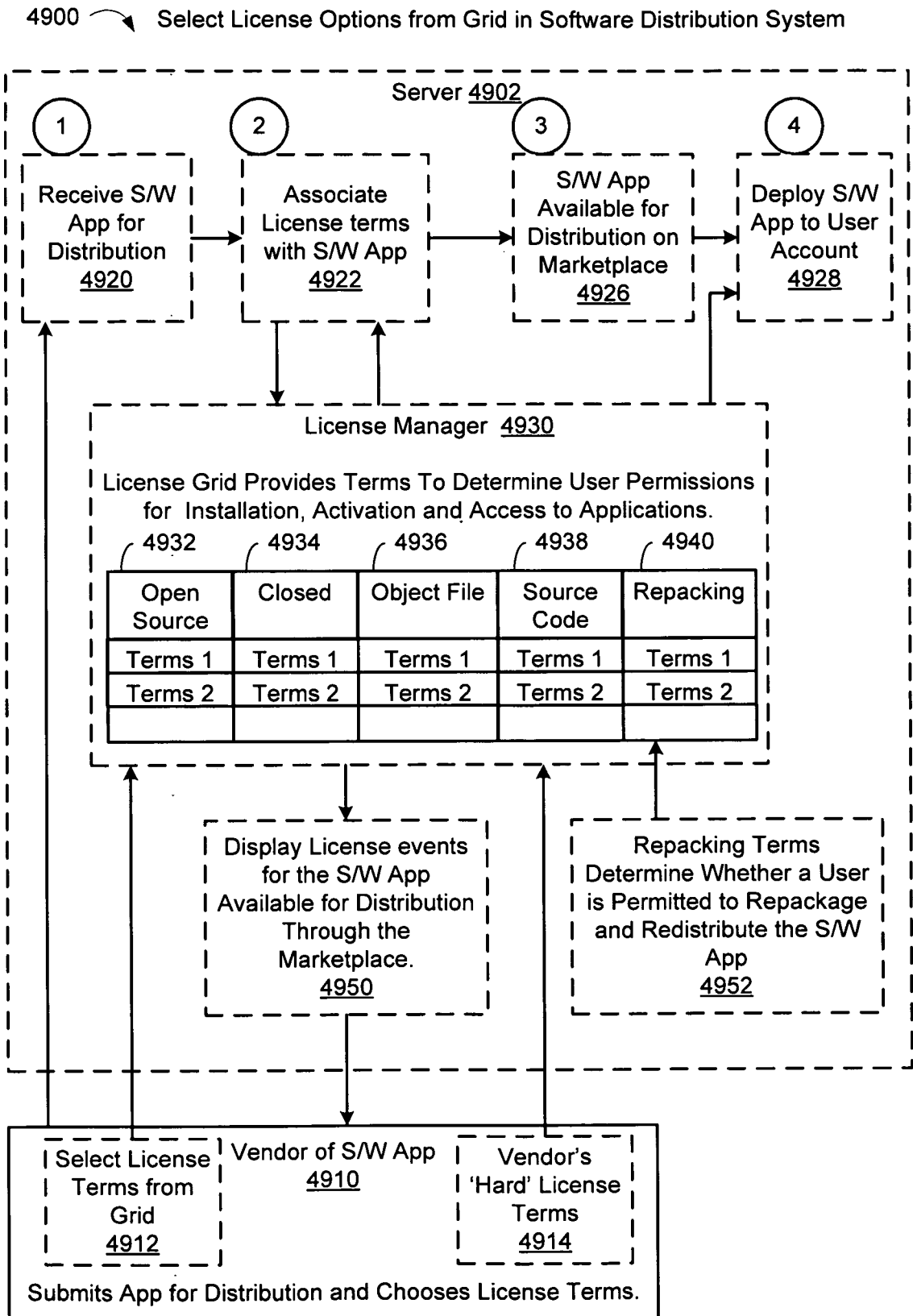


Figure 49

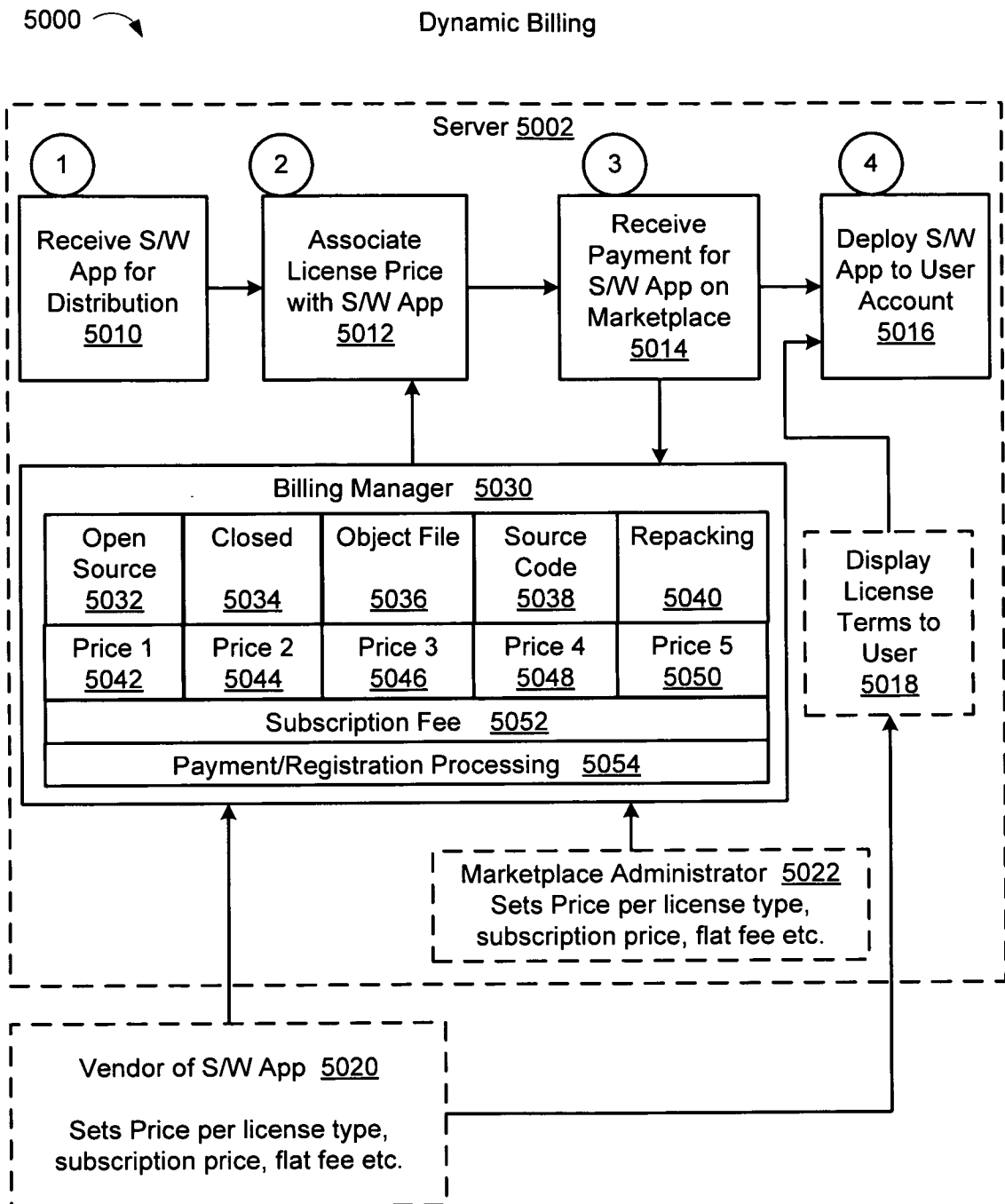


Figure 50

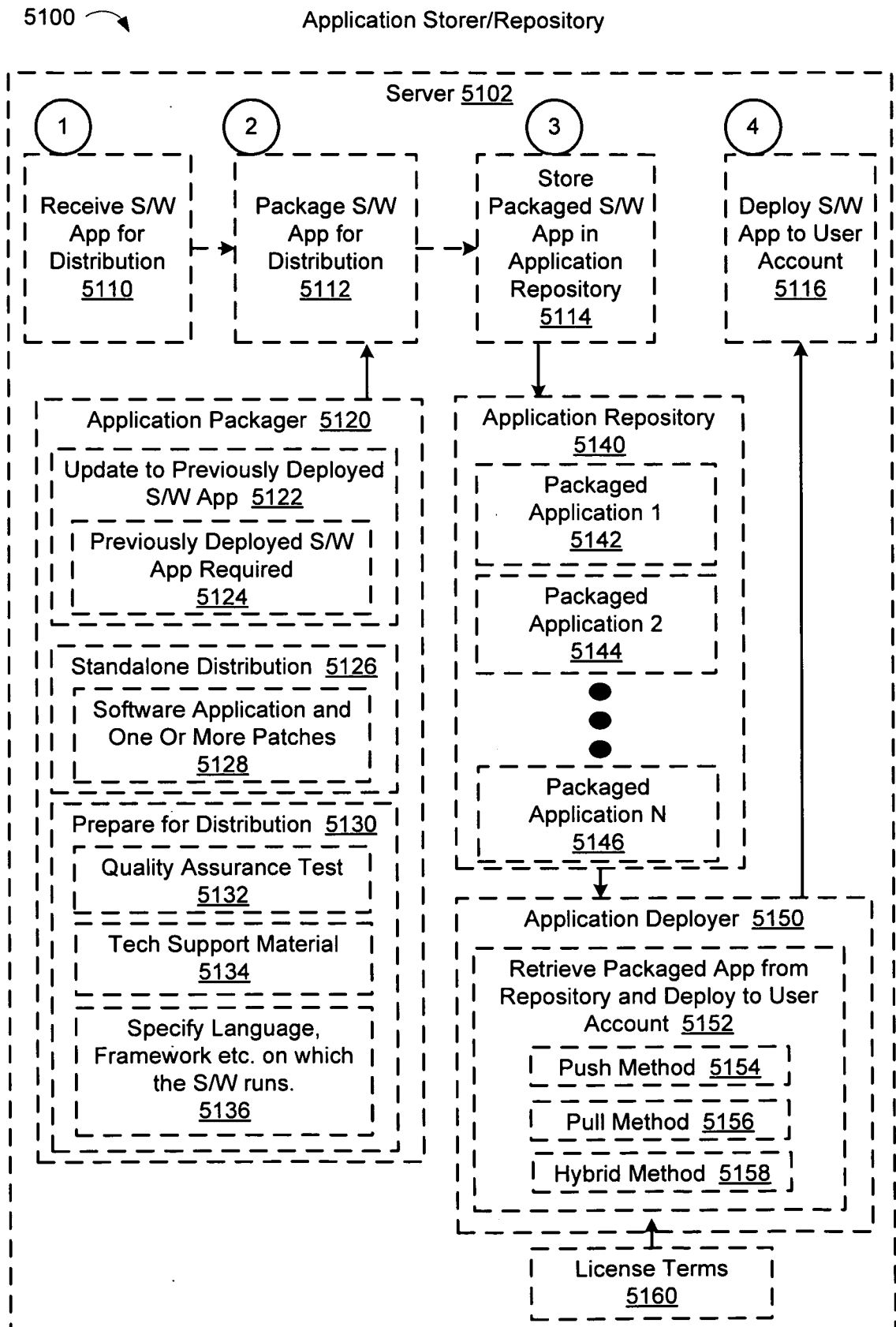


Figure 51

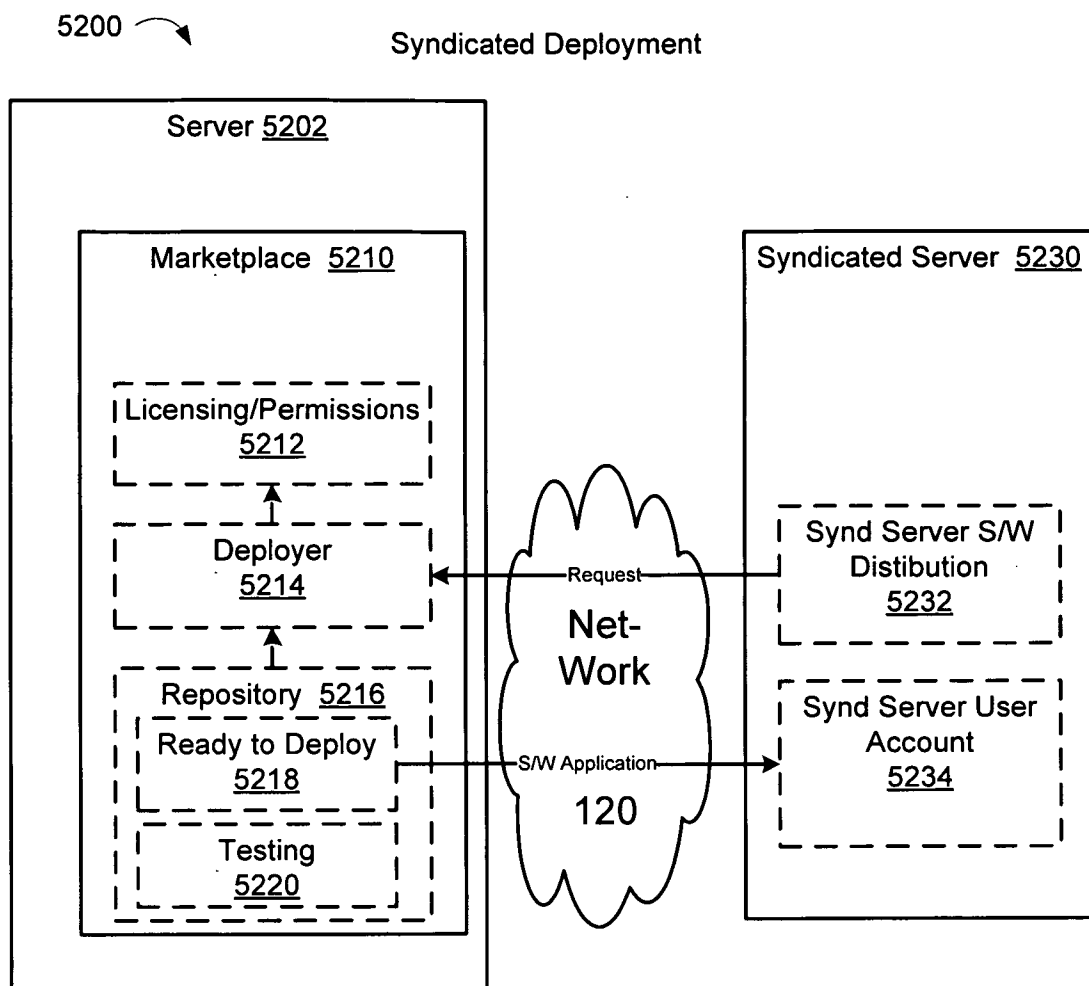


Figure 52

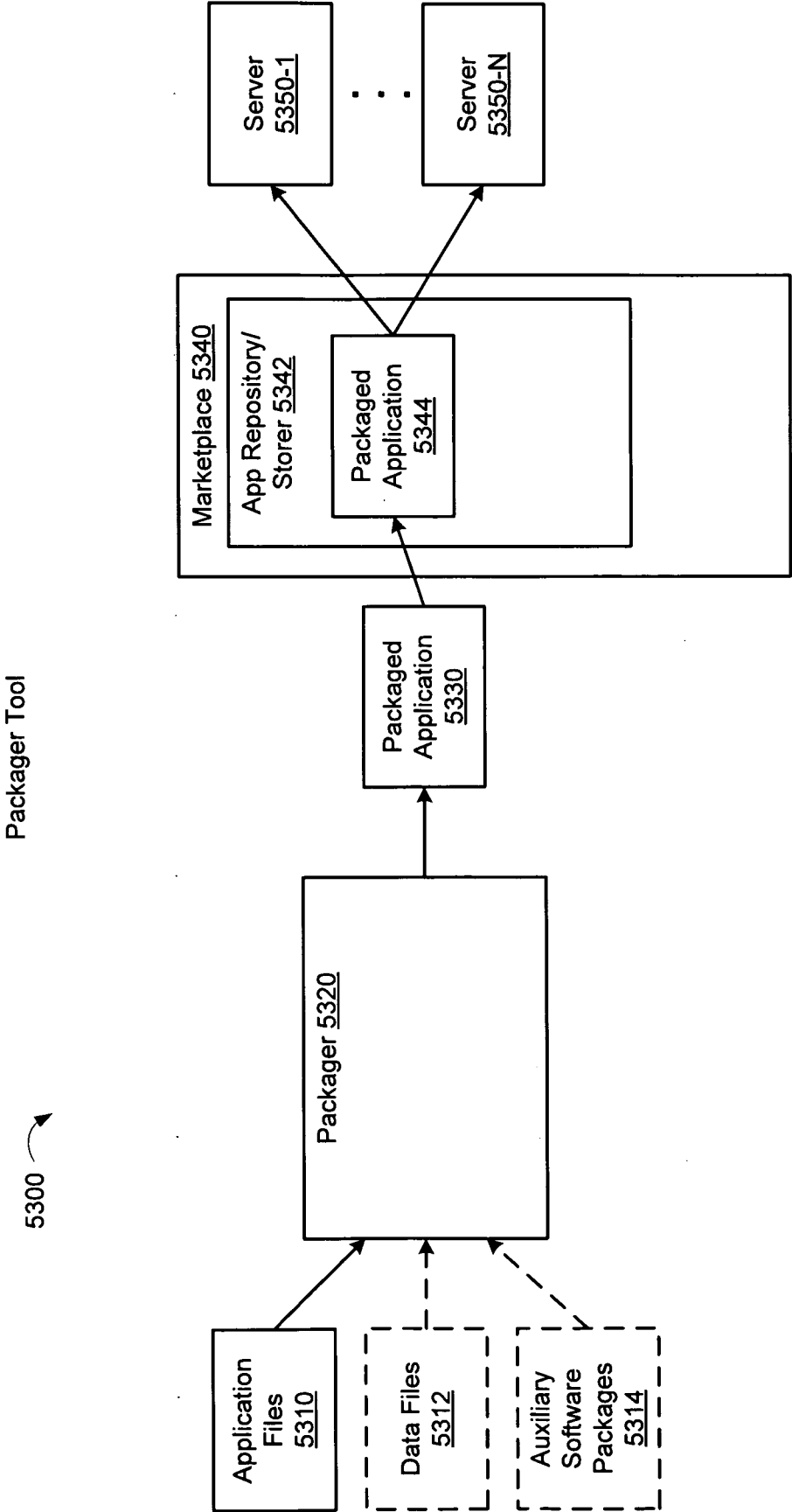


Figure 53

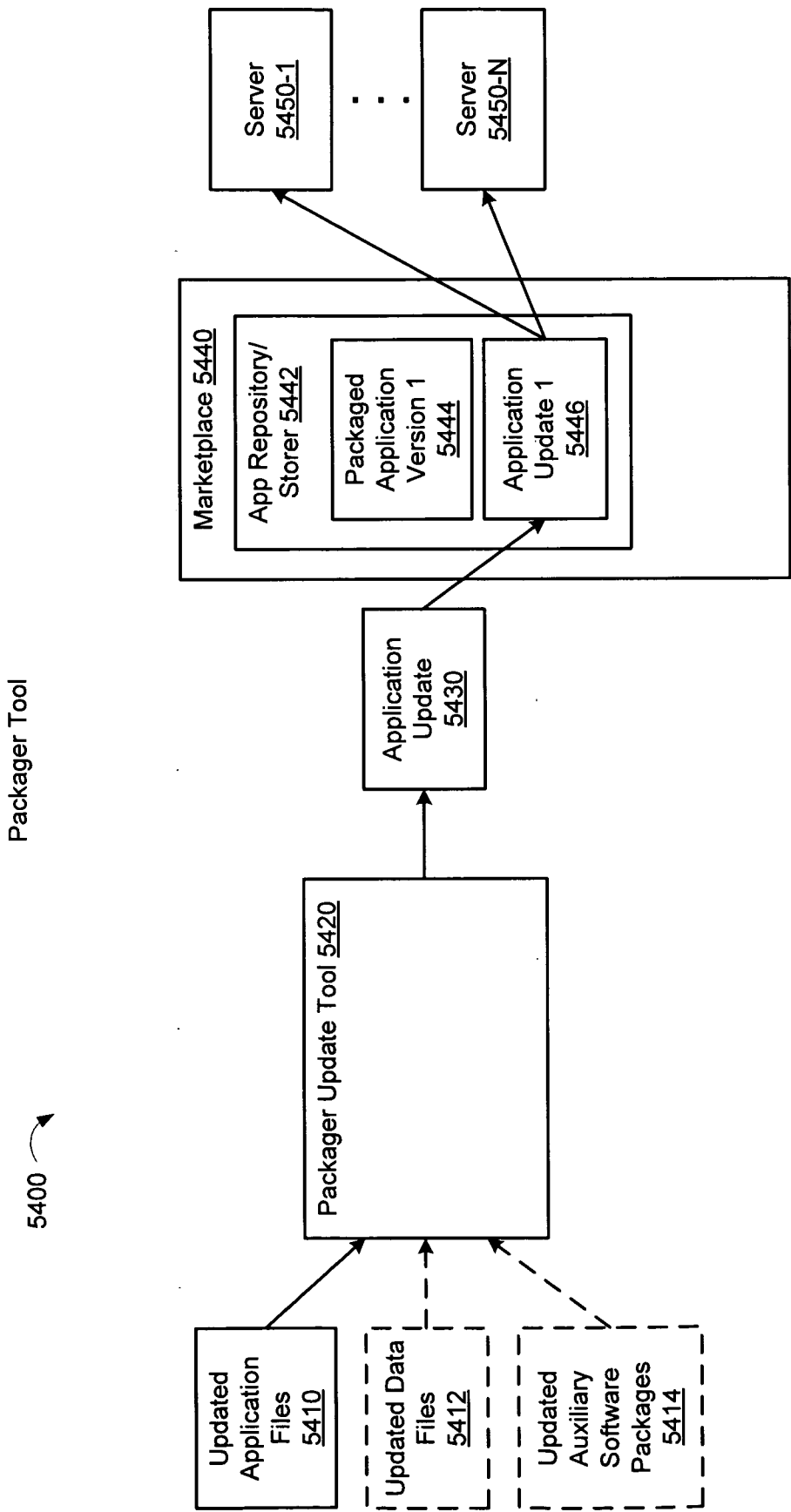


Figure 54

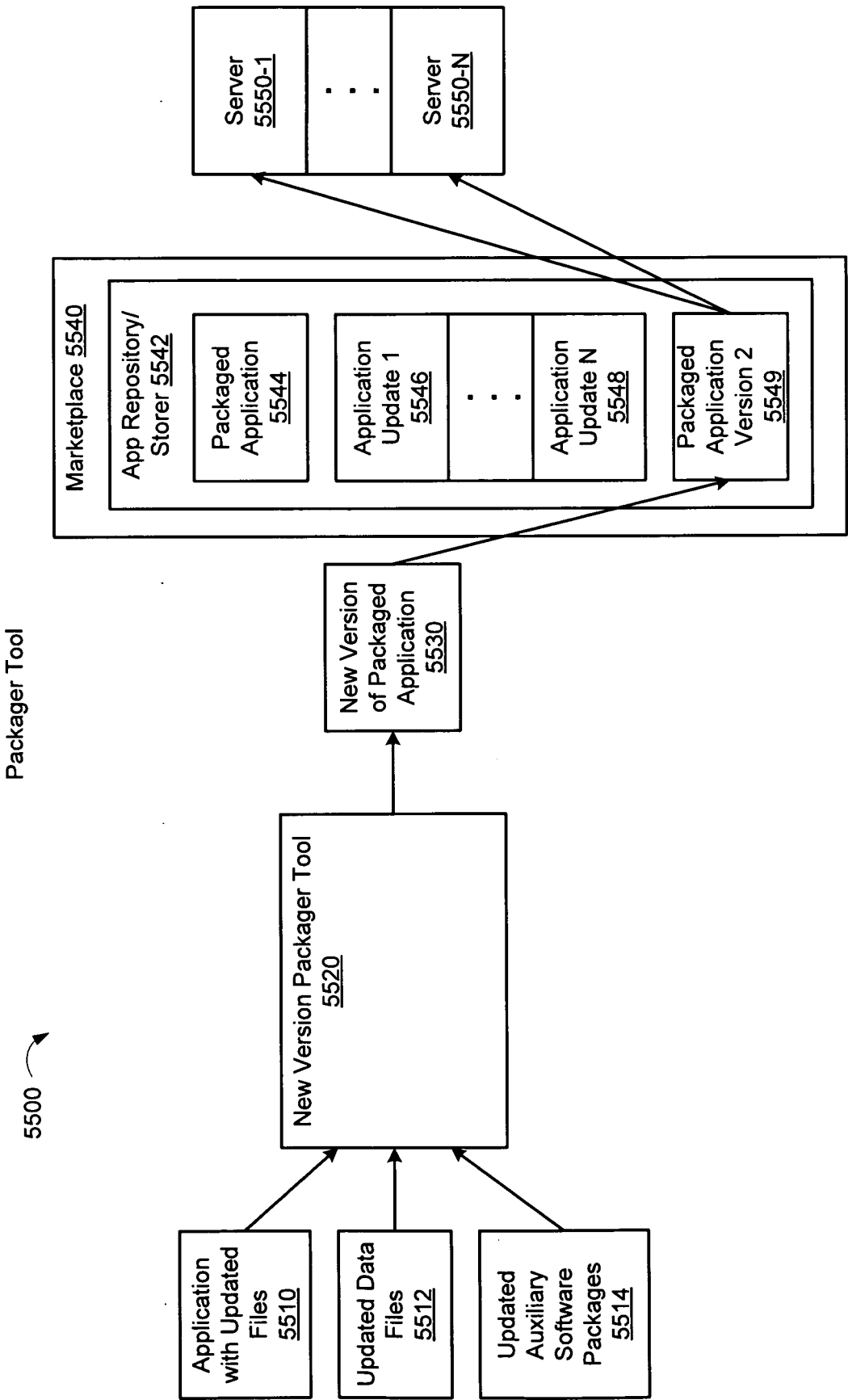


Figure 55

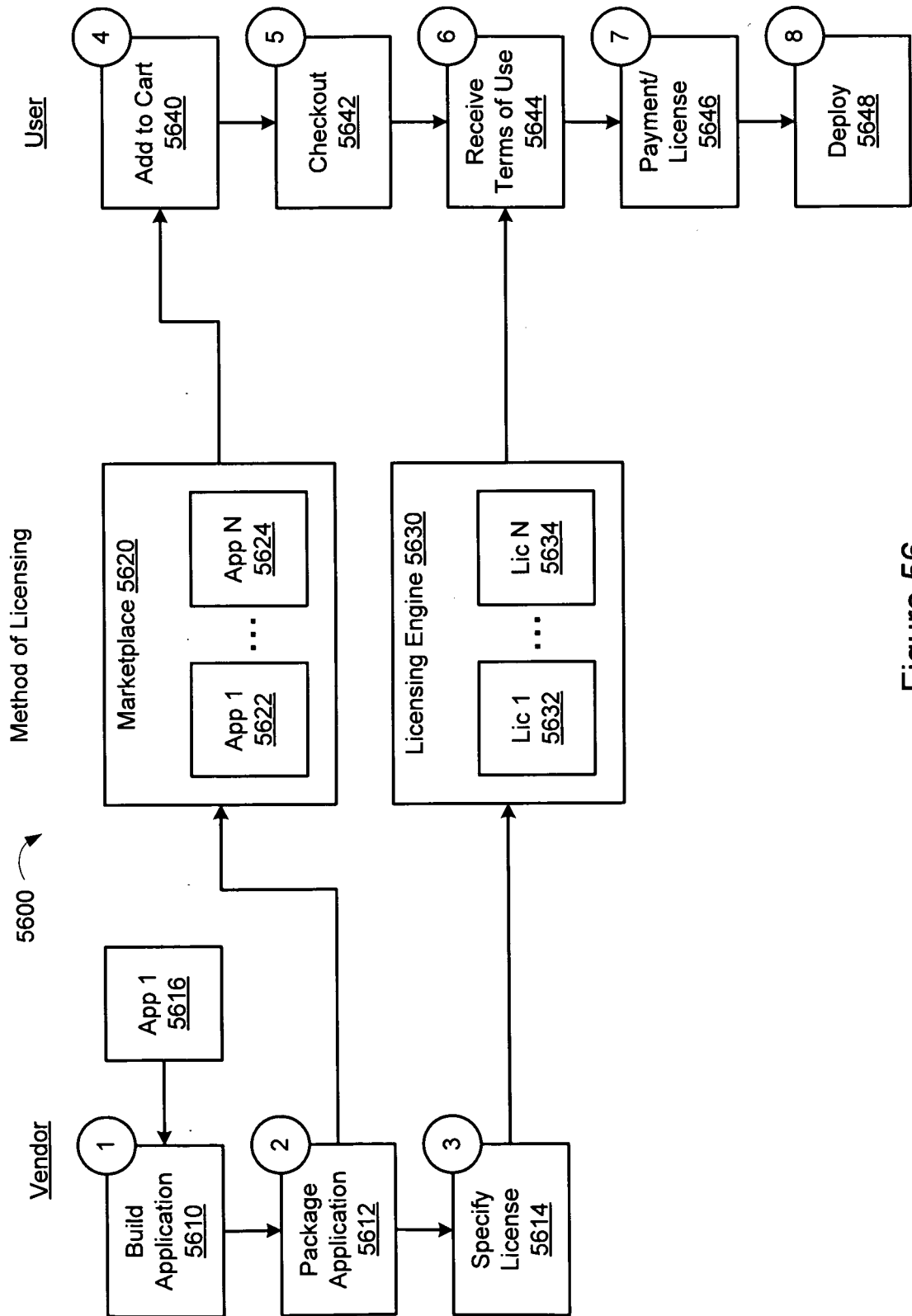


Figure 56

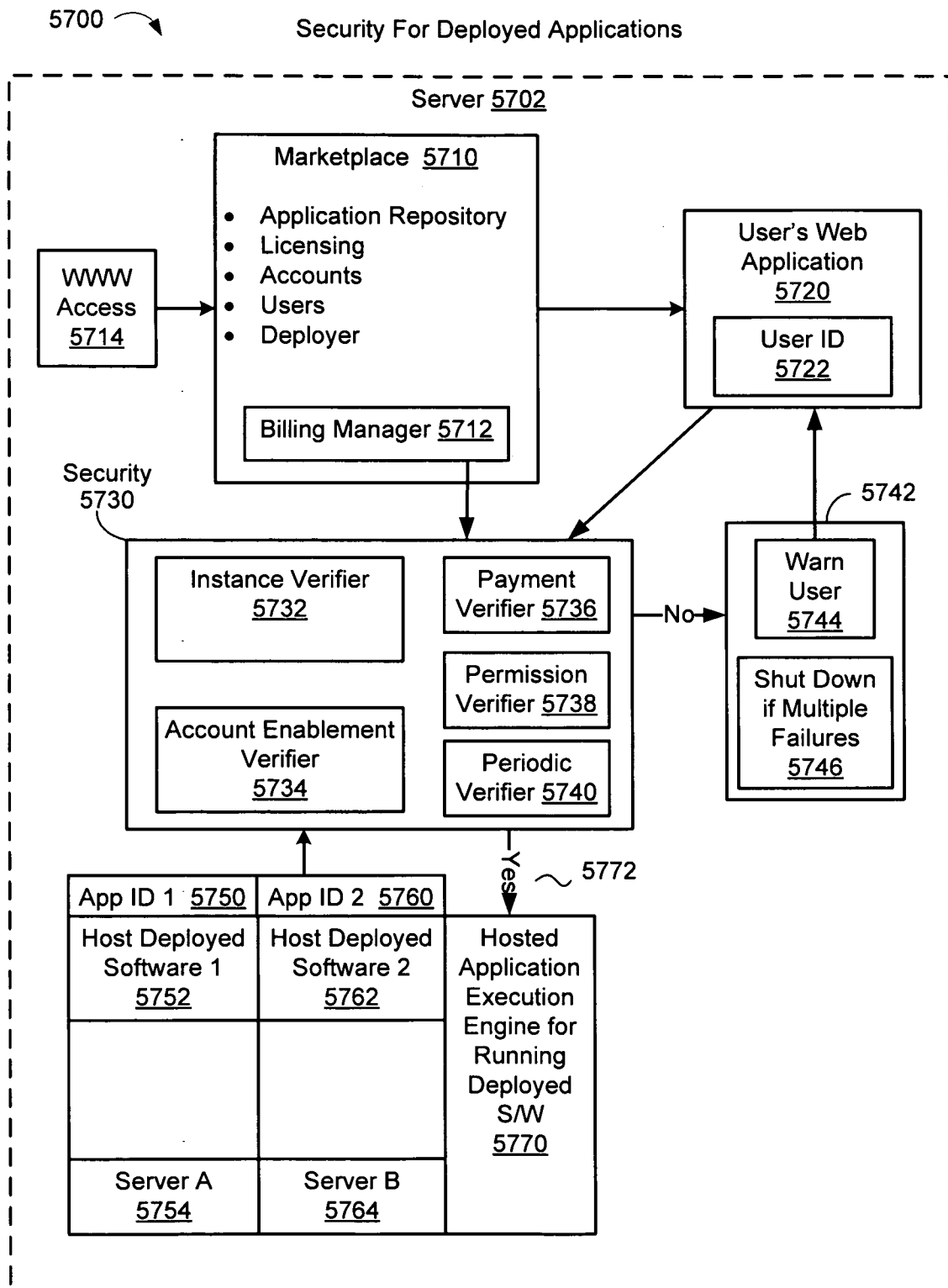


Figure 57

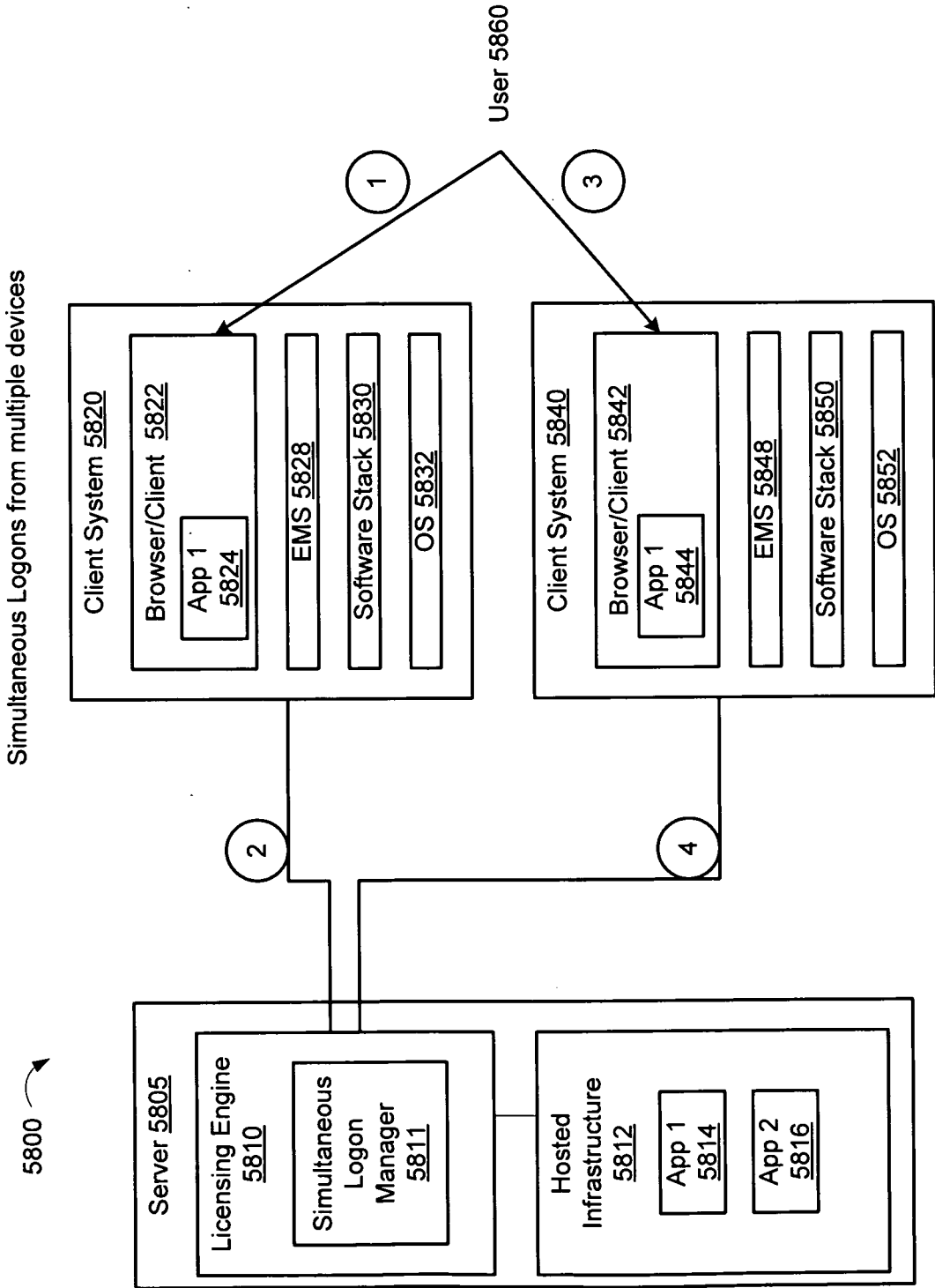


Figure 58

User Access Control Interface

5900 

Application <u>5902-1</u>	Status <u>5904</u>	Admin? <u>5906</u>	AOP <u>5908</u>	Source Code Access <u>5910</u>	Other Control <u>5912</u>
User <u>5920-1</u>	☒ Active	☐ Admin	☒ Enabled	☐ View ☐ Edit	...
User <u>5920-2</u>	☐ Active	☒ Admin	☐ Enabled	☐ View ☒ Edit	...
.	.	.	.	.	...
.	.	.	.	.	...
User <u>5920-N</u>	☒ Active	☐ Admin	☒ Enabled	☒ View ☐ Edit	...
.	.	.	.	.	...
.	.	.	.	.	...
Application <u>5932-N</u>	Status	Admin?	AOP	Source Code Access	...
User <u>5934-1</u>	☒ Active	☐ Admin	☒ Enabled	☐ View ☐ Edit	...
.	.	.	.	.	...
.	.	.	.	.	...
.	.	.	.	.	...

Figure 59

User Management Interface

	App 1 6002	...	App N 6004
User 6010-1	☒	...	☒
User 6010-2	☐	...	☐
...	...	...	...
User 6010-N	☒	...	☐

6000 ↗

Figure 60

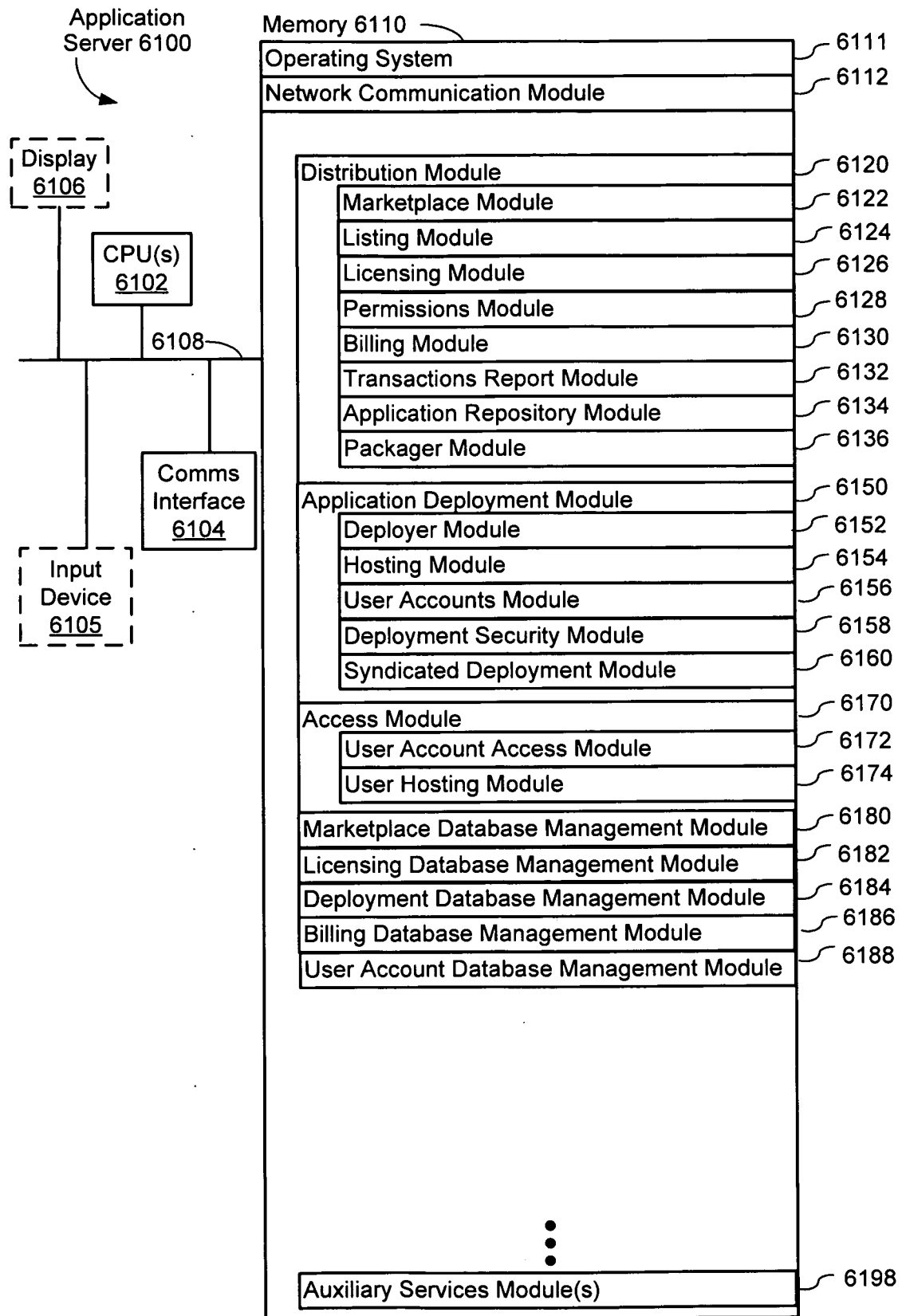


Figure 61

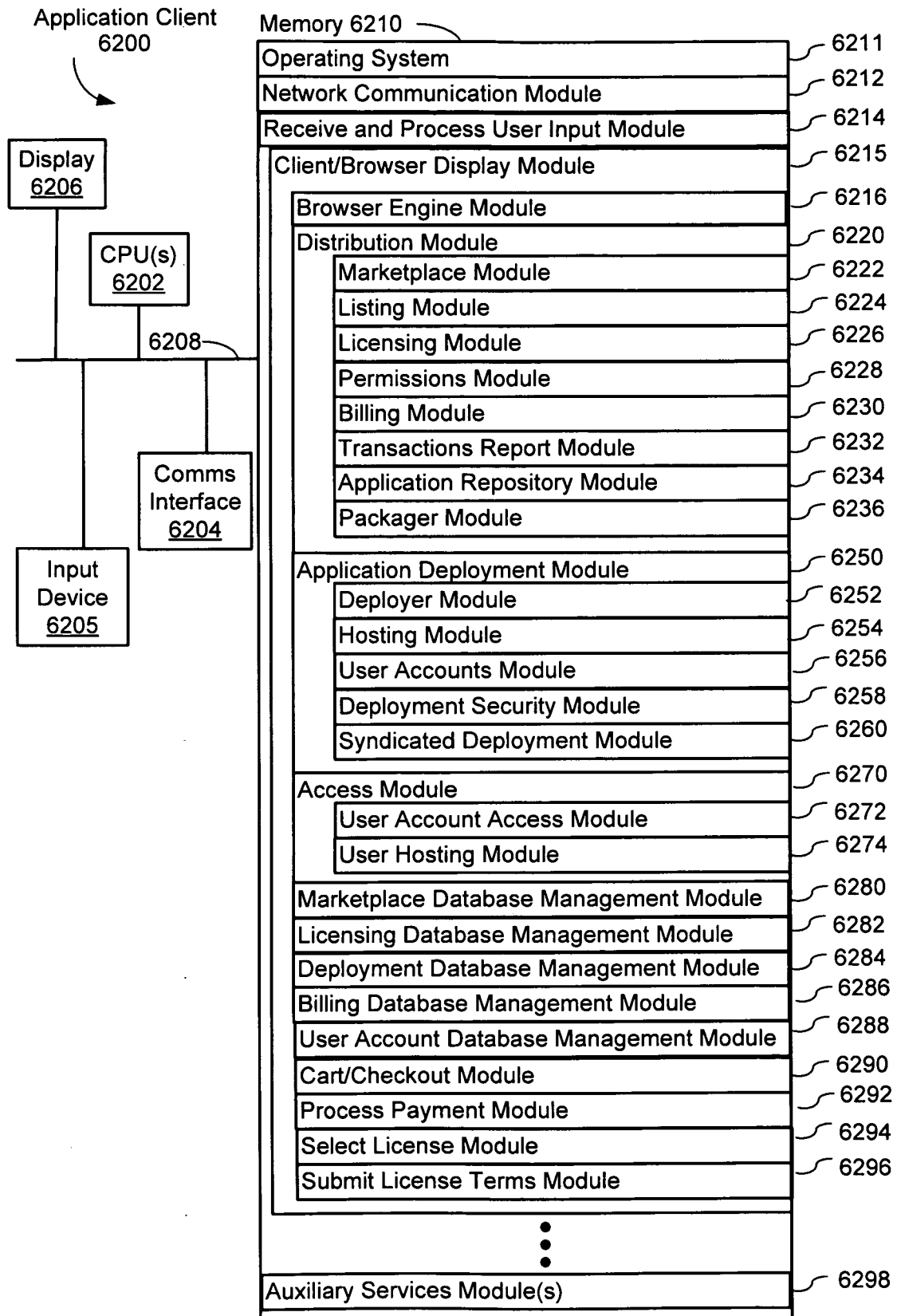


Figure 62

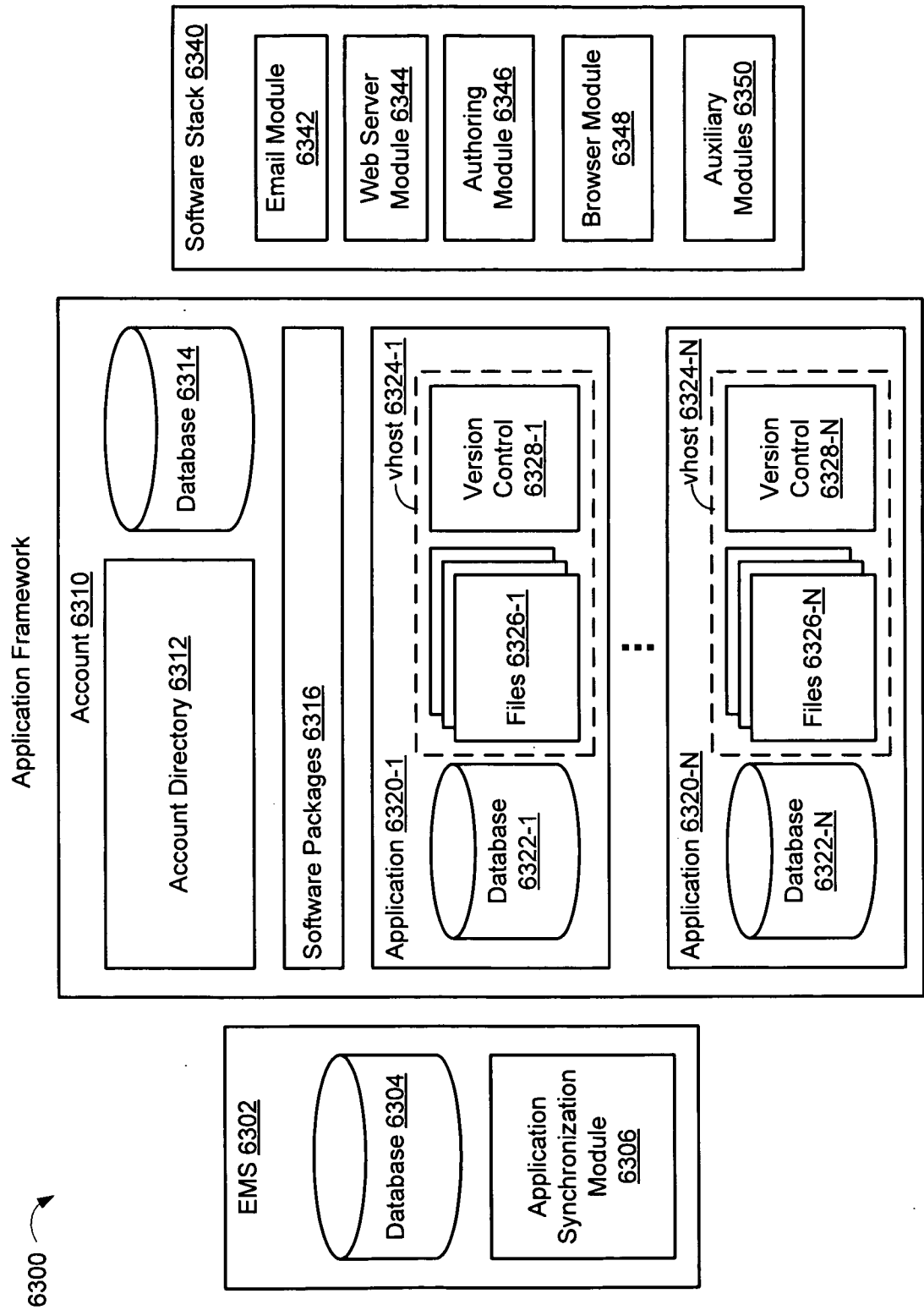


Figure 63

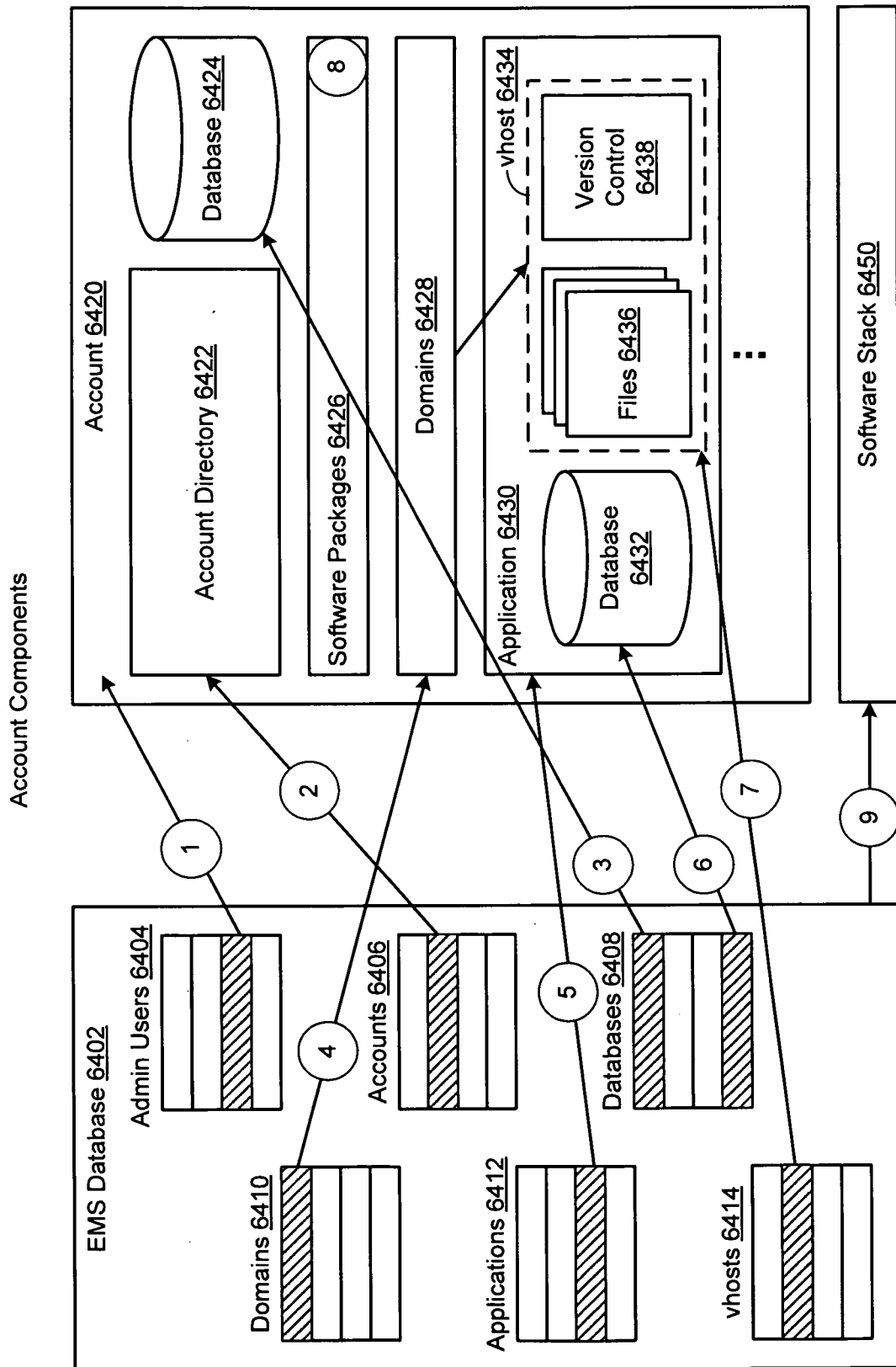


Figure 64A

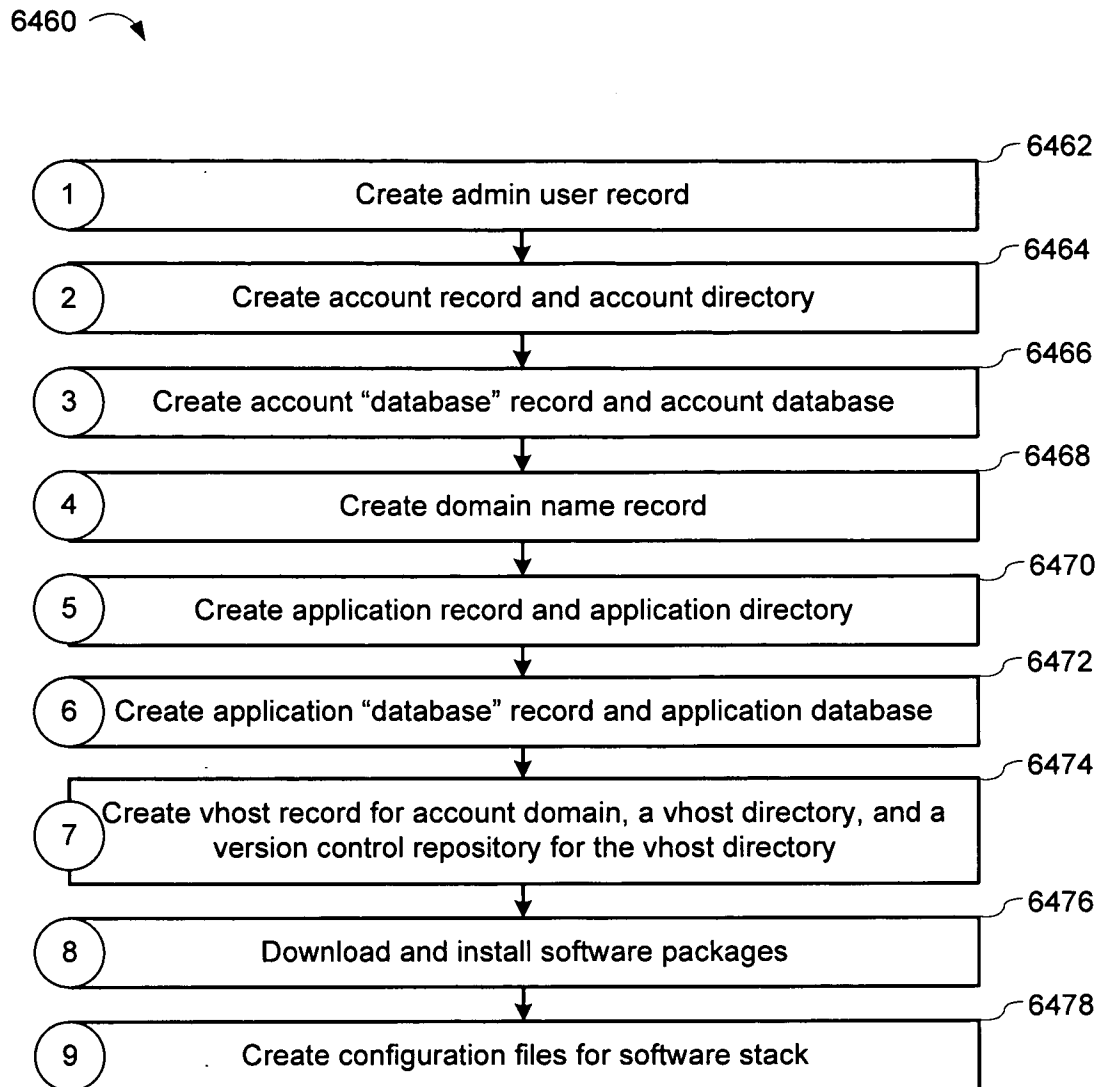


Figure 64B

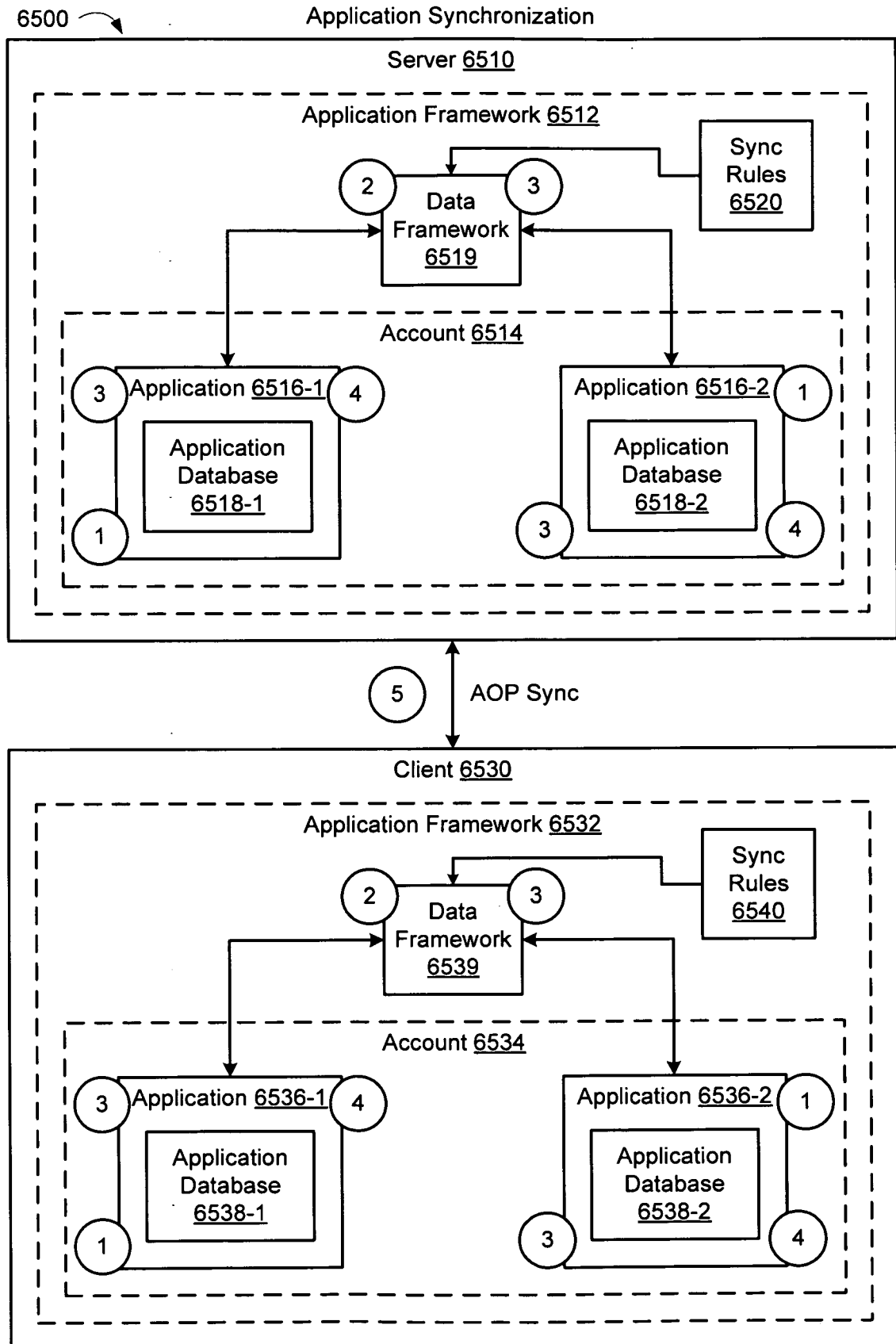


Figure 65

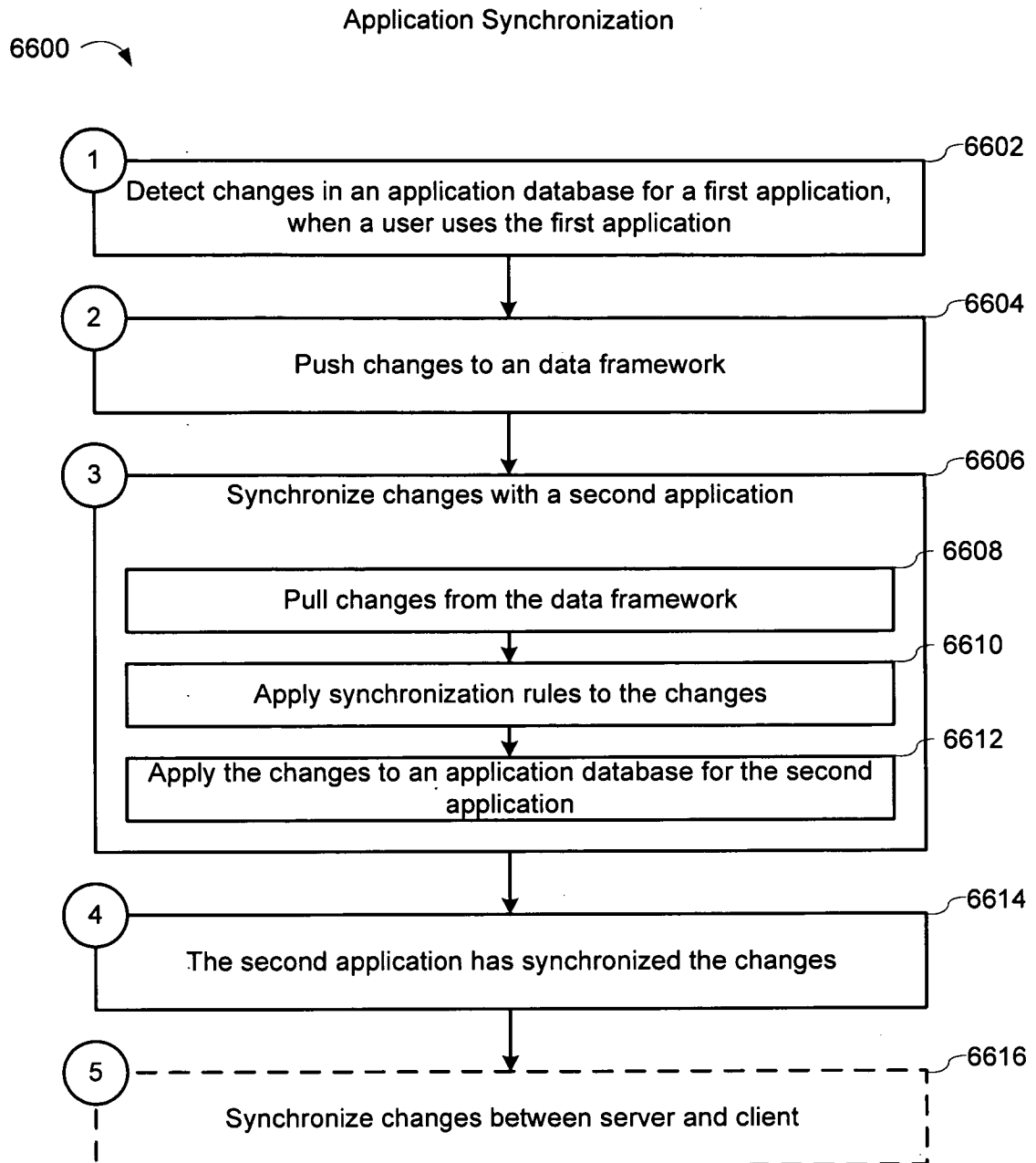


Figure 66

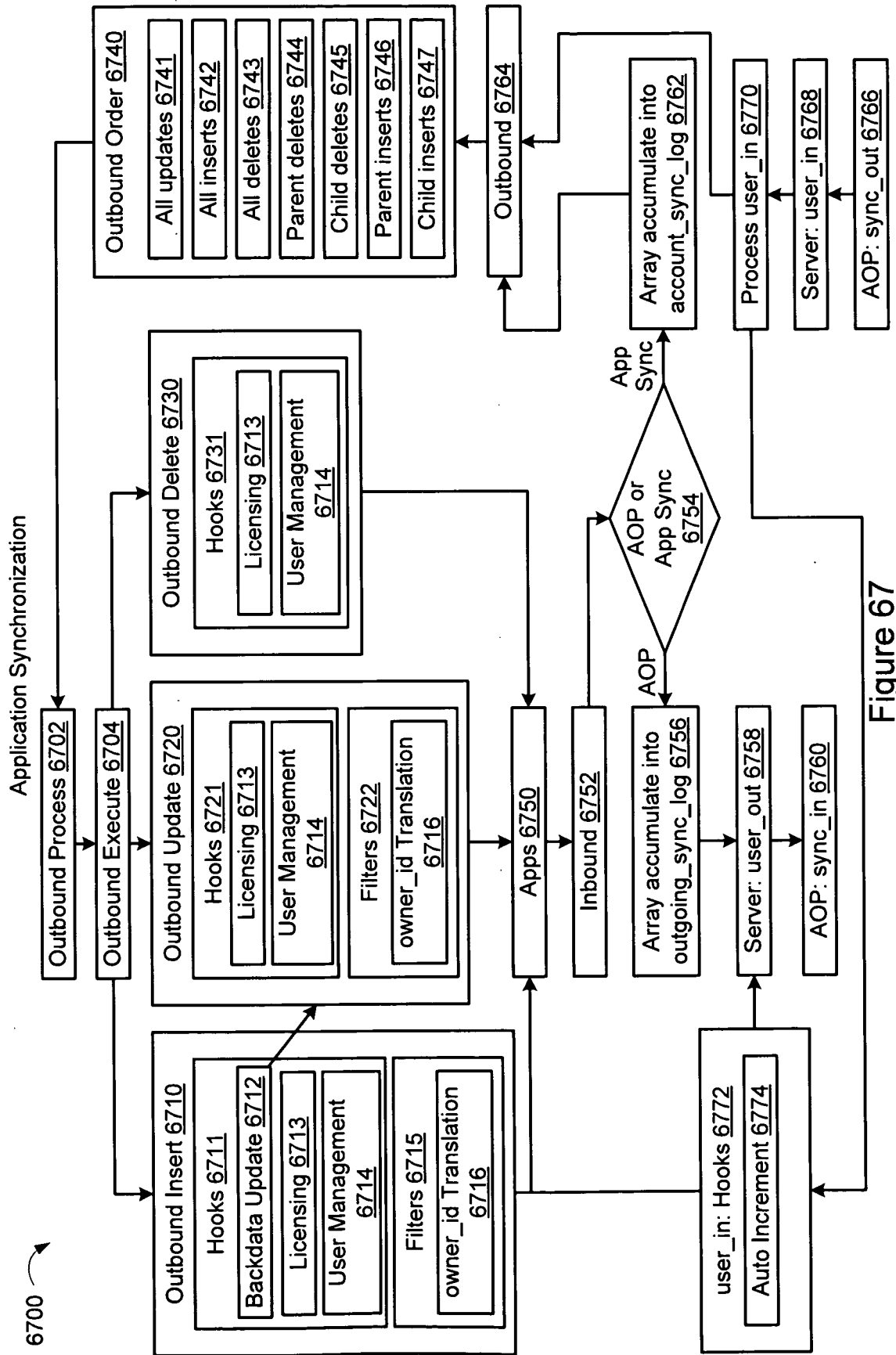


Figure 67

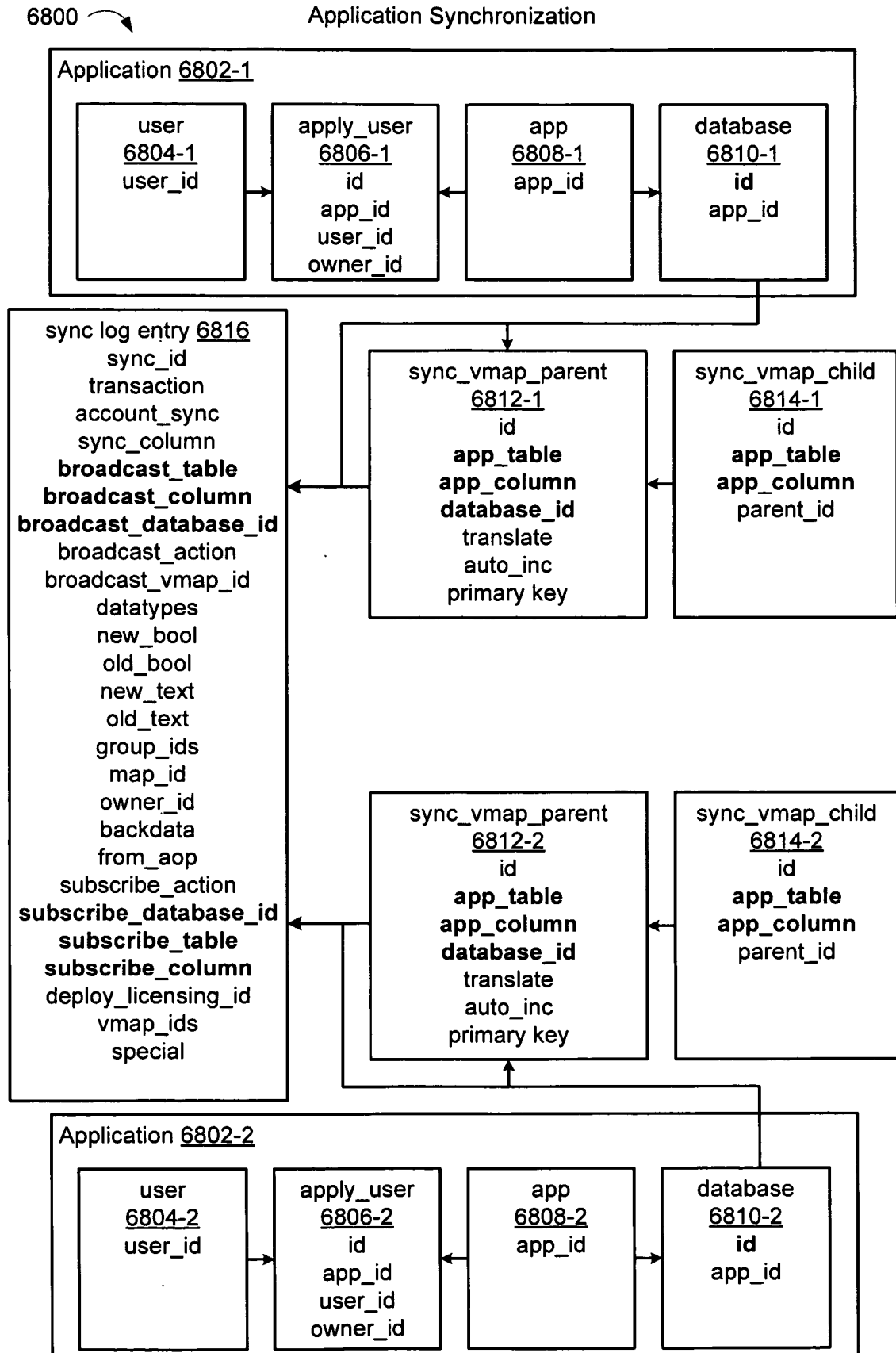


Figure 68

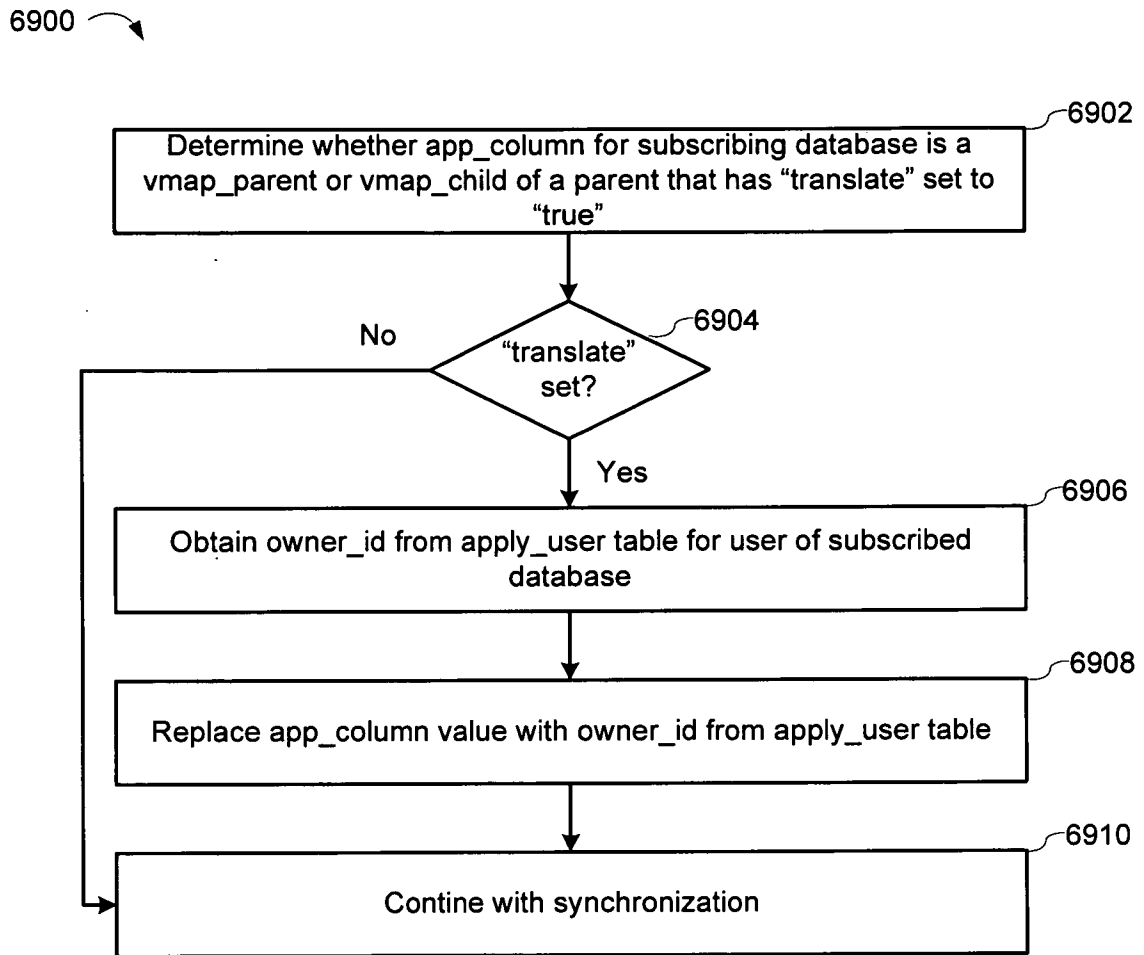


Figure 69

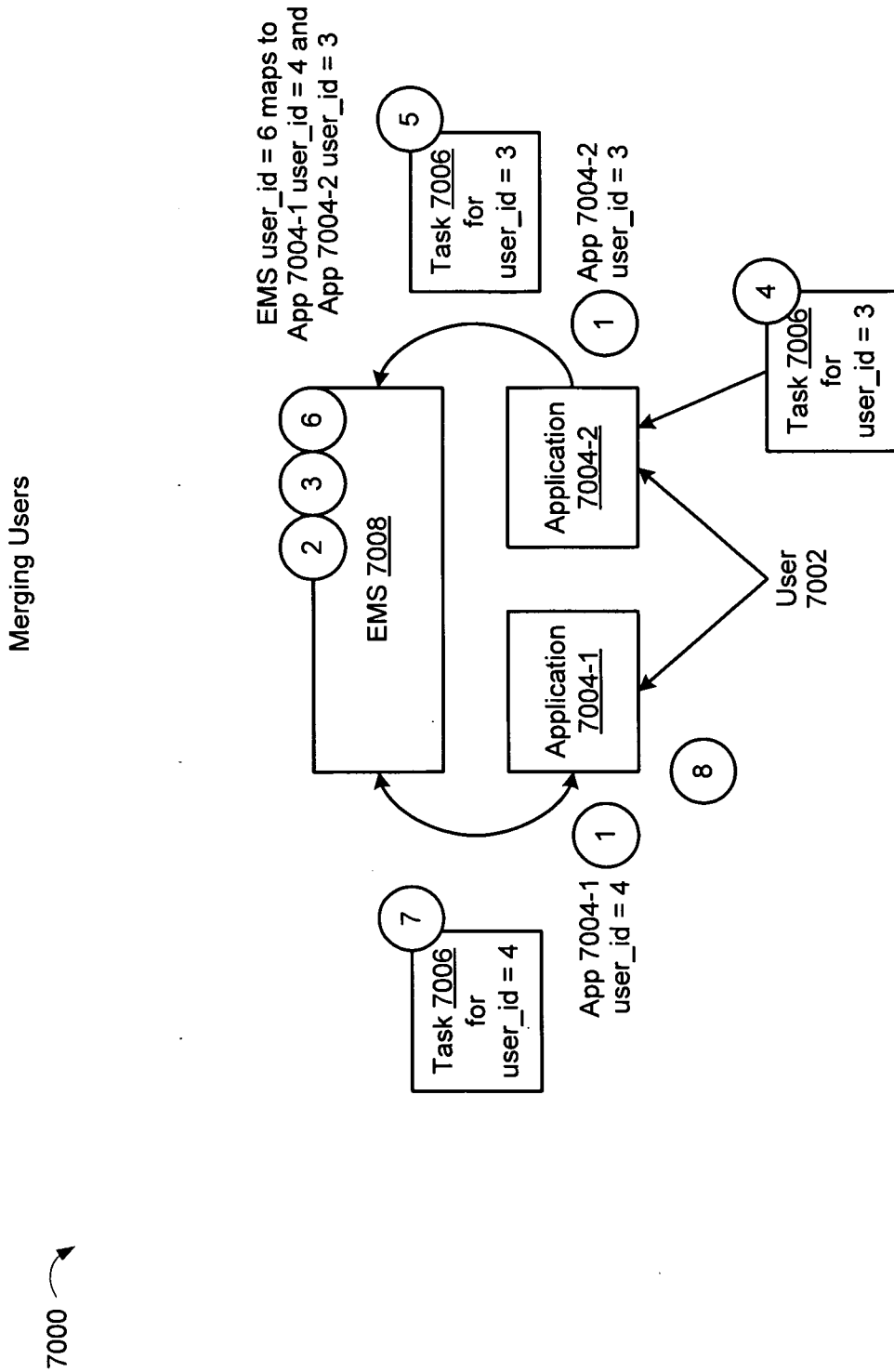
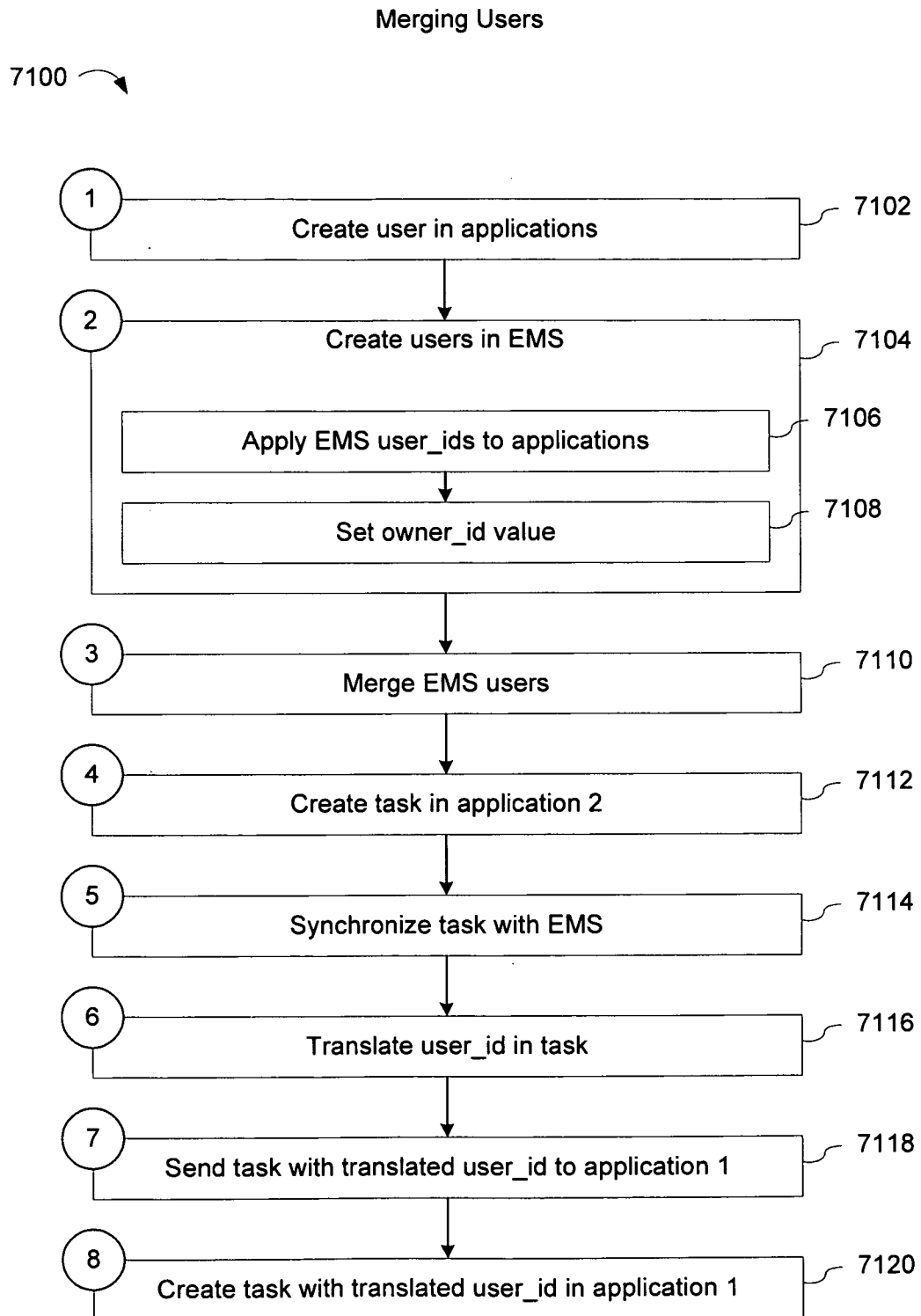


Figure 70

**Figure 71**

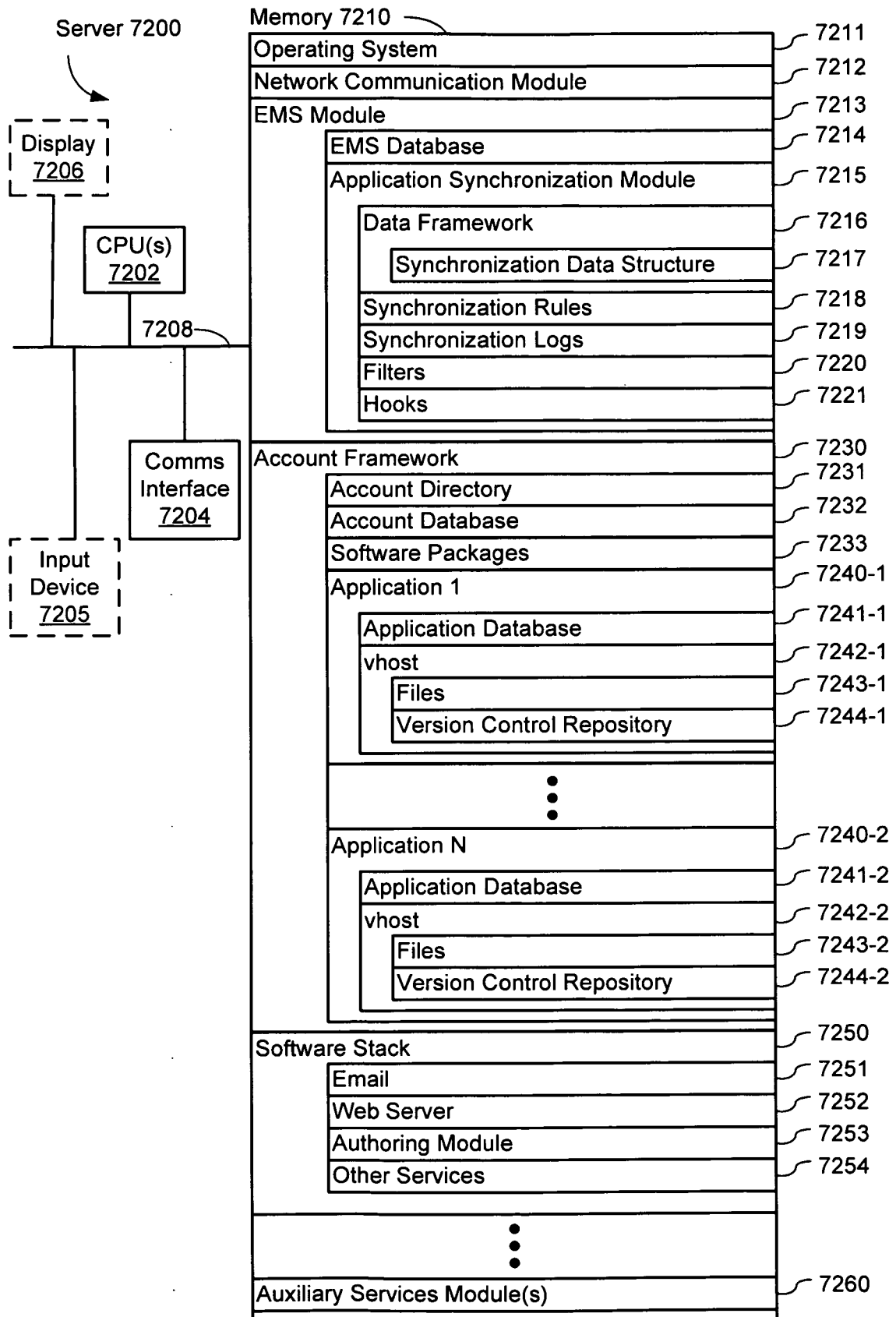


Figure 72

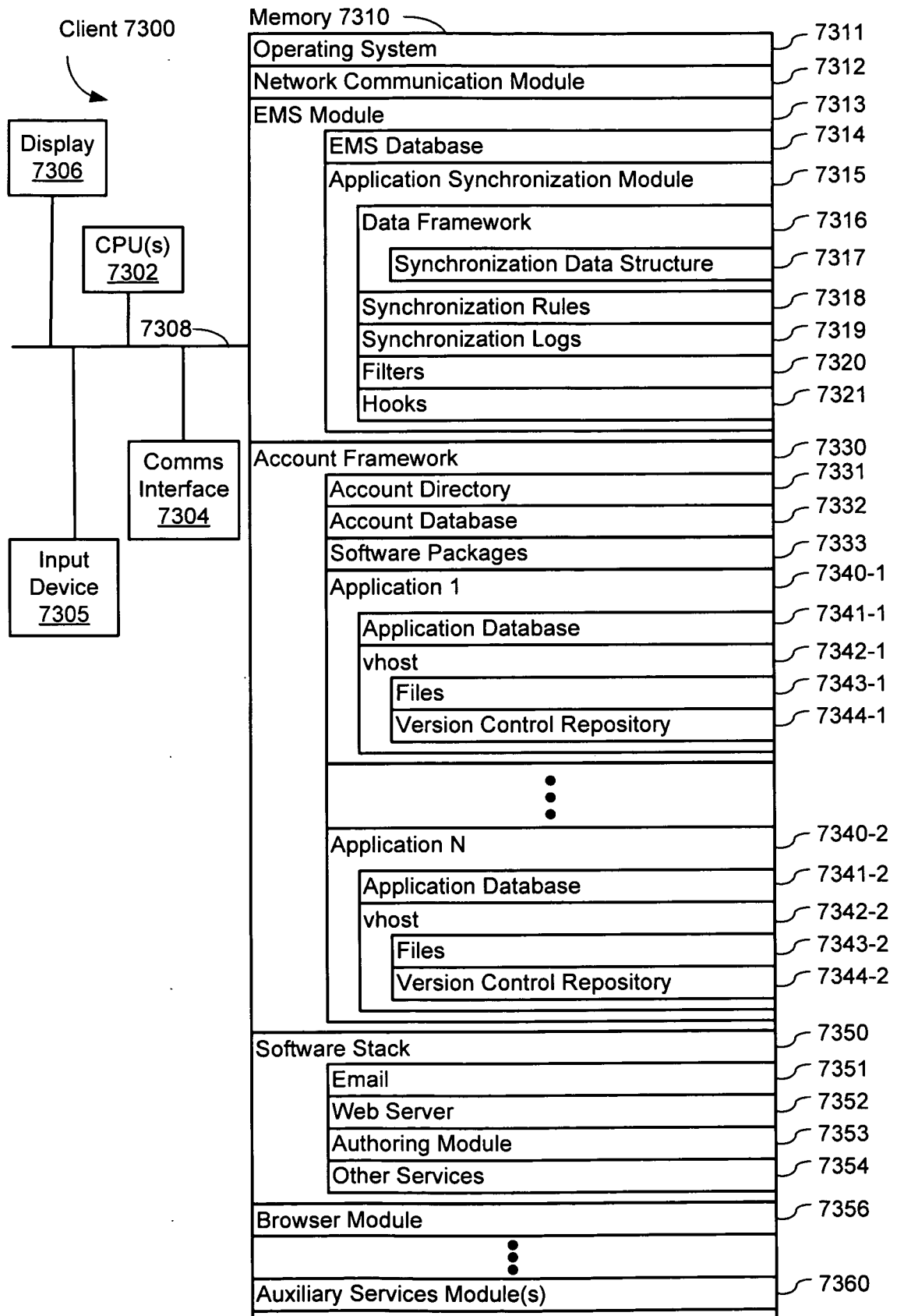


Figure 73

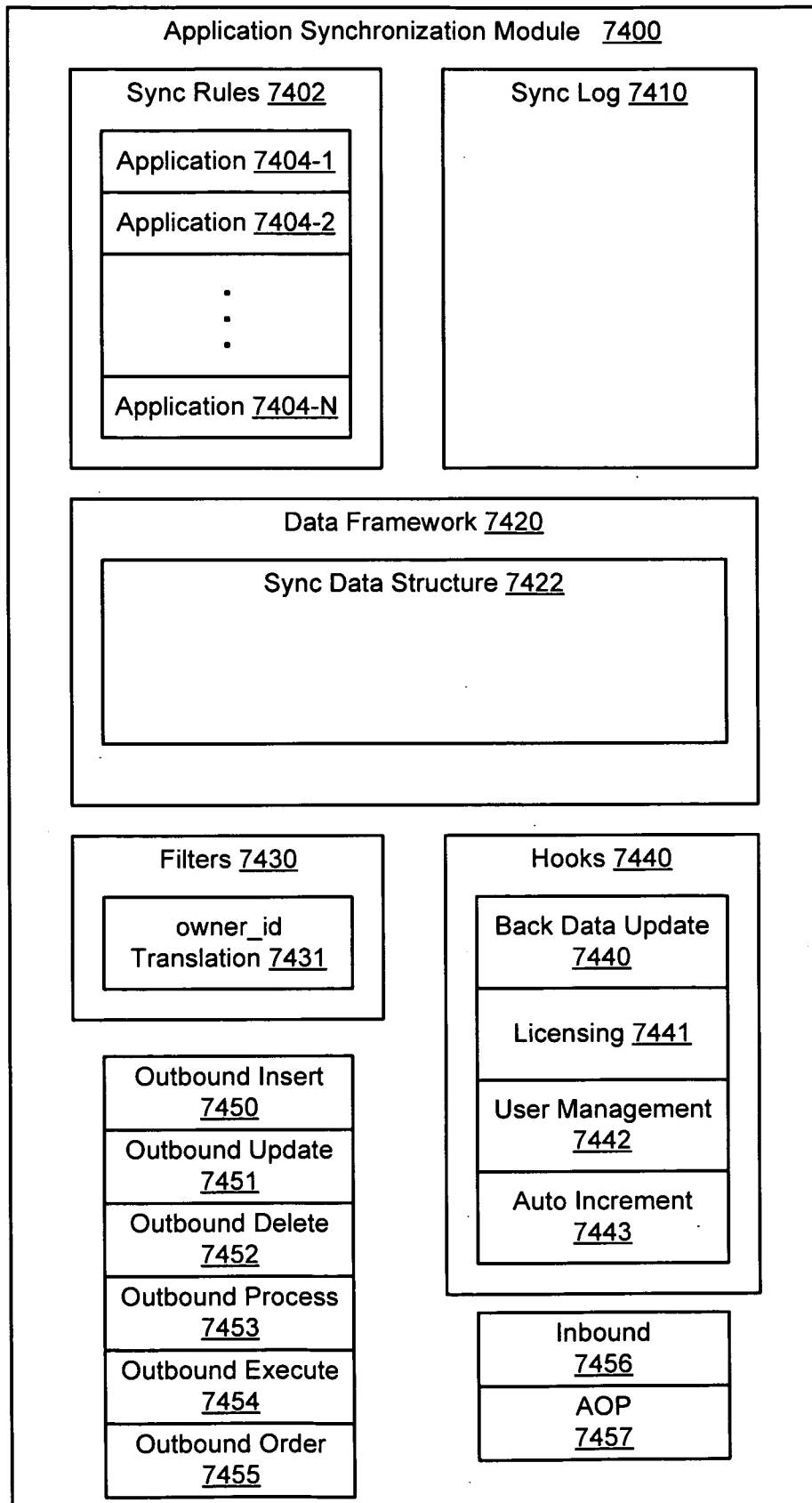


Figure 74
78/178

7500



eas_sync_app_log 7502
 created_on
 id
 transaction
 account_sync
 backdata
 publish
 vmap_id
 map_id
 action
 datatype
 app_table
 app_column
 sync_table
 sync_column
 old_bool
 new_bool
 old_text
 new_text
 group_ids
 owner_id
 get_children

inbound 7504
 created_on
 sync_id
 transaction
 account_sync
 database_id
 backdata
 map_id
 action
 datatype
 app_table
 app_column
 sync_table
 sync_column
 old_bool
 new_bool
 old_text
 new_text
 app_created_on
 backdata_database
 groups
 group_ids
 owner_id
 get_children
 from_aop
 broadcast_vmap_id

outbound 7506
 sync_id
 transaction
 account_sync
 sync_table
 sync_column
 broadcast_table
 broadcast_column
 broadcast_database_id
 broadcast_action
 broadcast_vmap_id
 datatypes
 new_bool
 old_bool
 new_text
 old_text
 group_ids
 map_id
 owner_id
 backdata
 from_aop
 subscribe_action
 subscribe_database_id
 subscribe_table
 subscribe_column
 deploy_licensing_id
 vmap_ids
 preserve_id
 special
 created_on

Figure 75

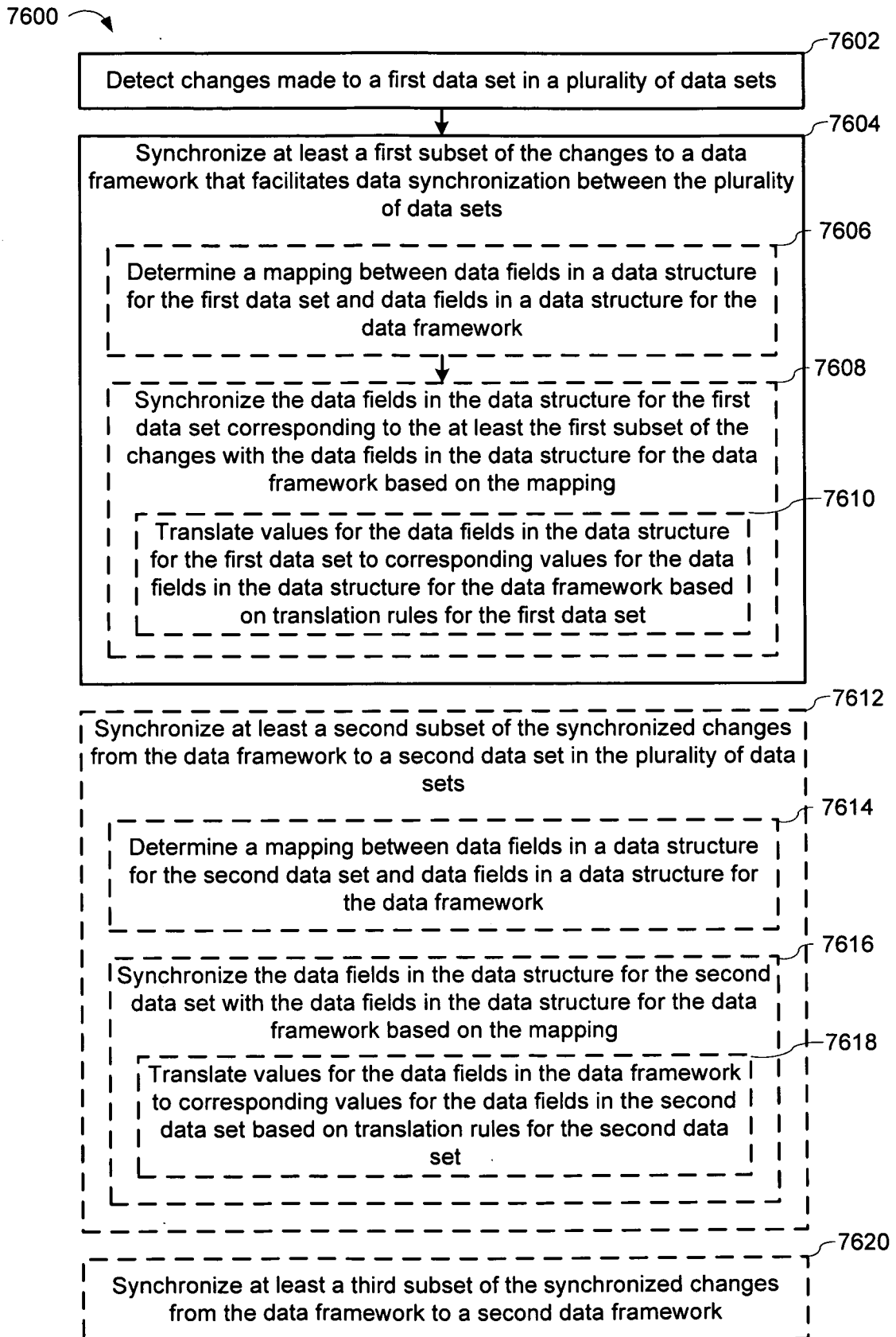


Figure 76

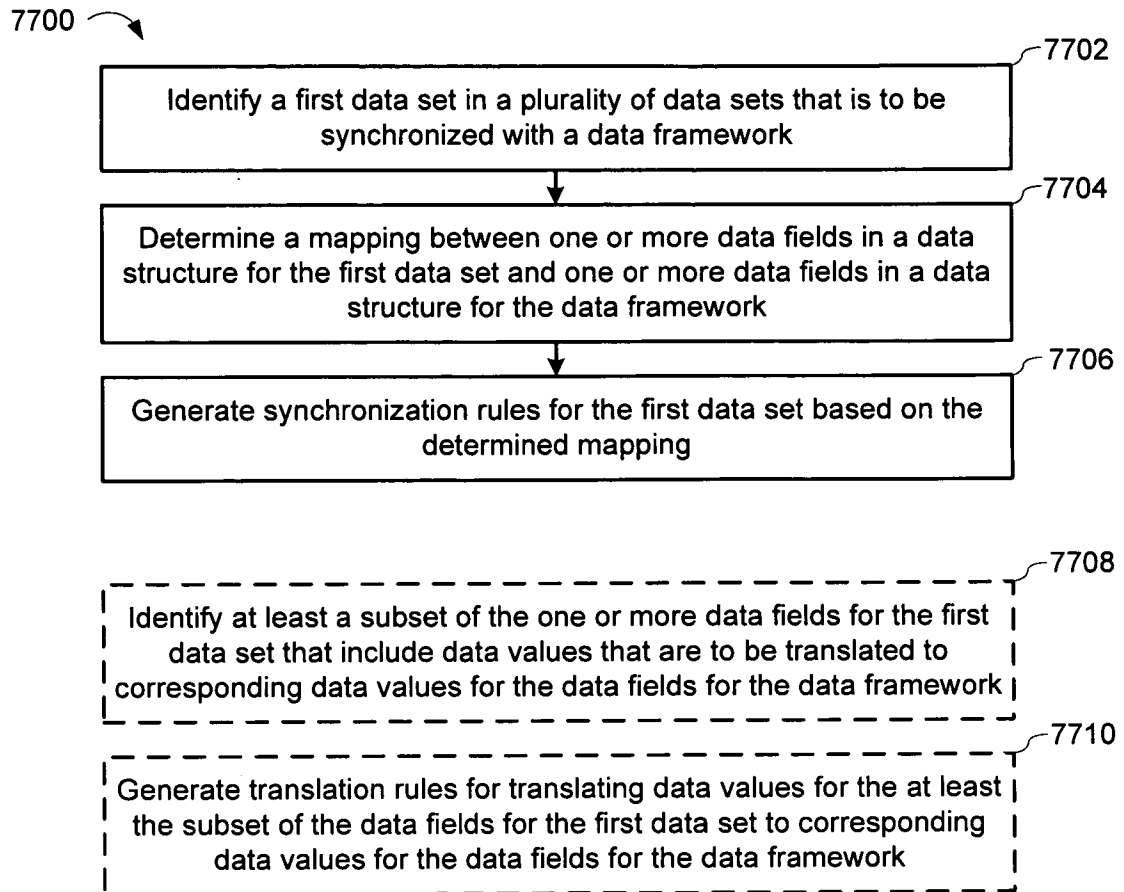


Figure 77

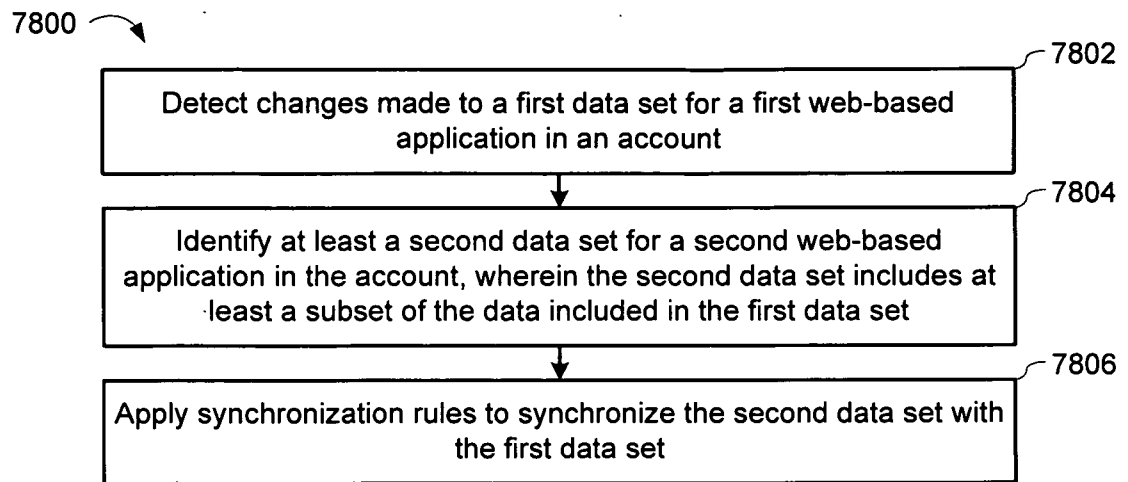


Figure 78

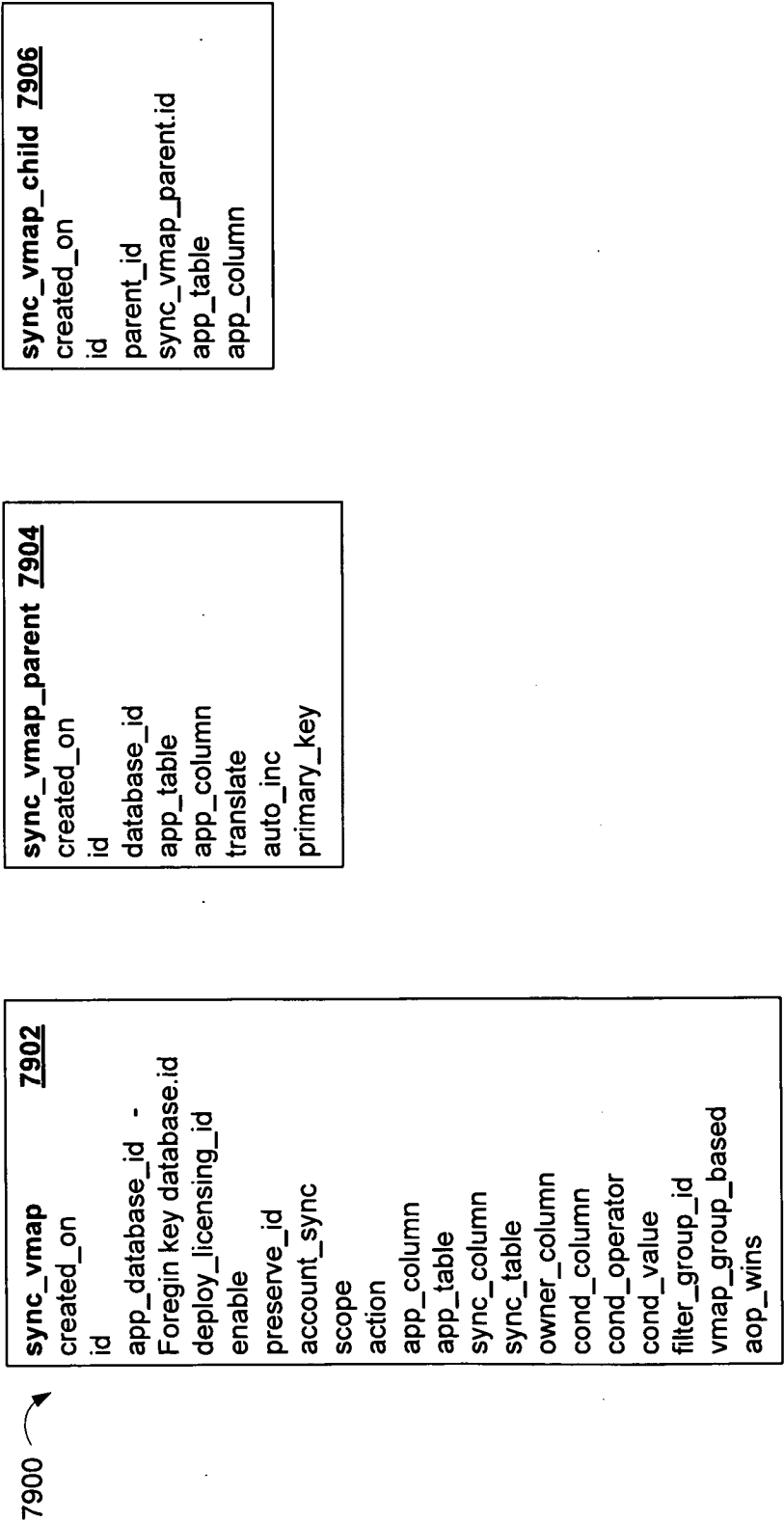


Figure 79

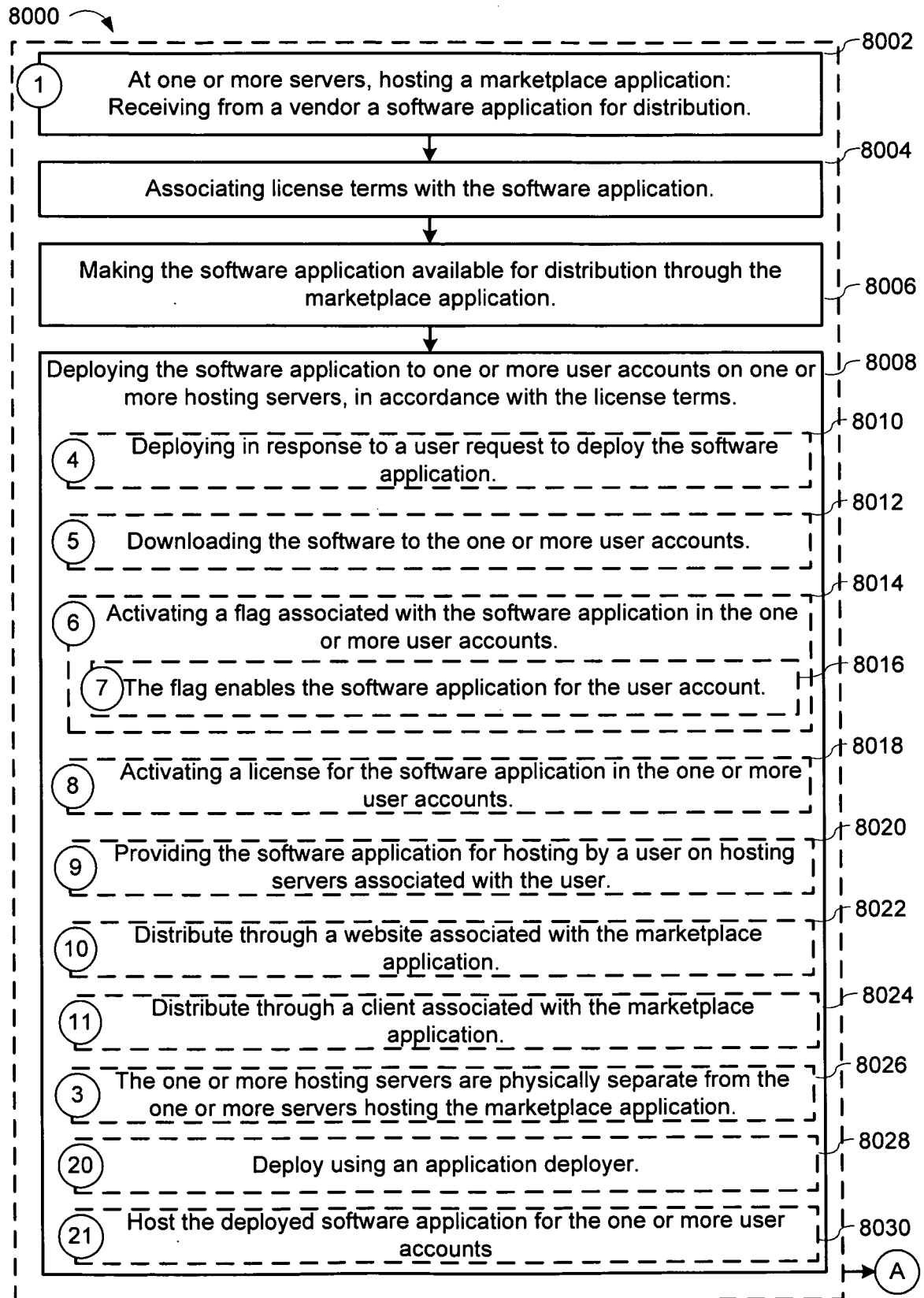


Figure 80

8100

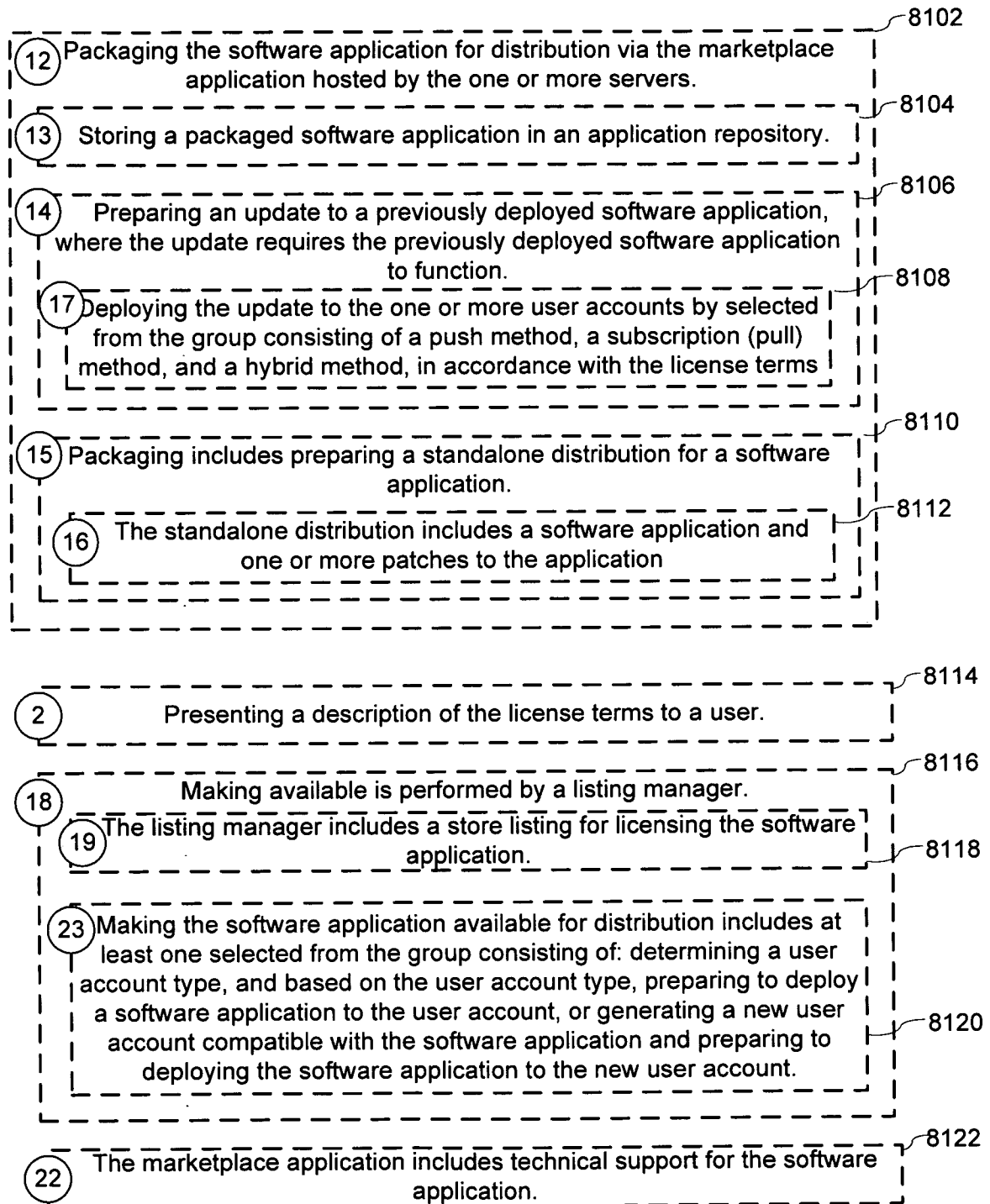


Figure 81
85/178

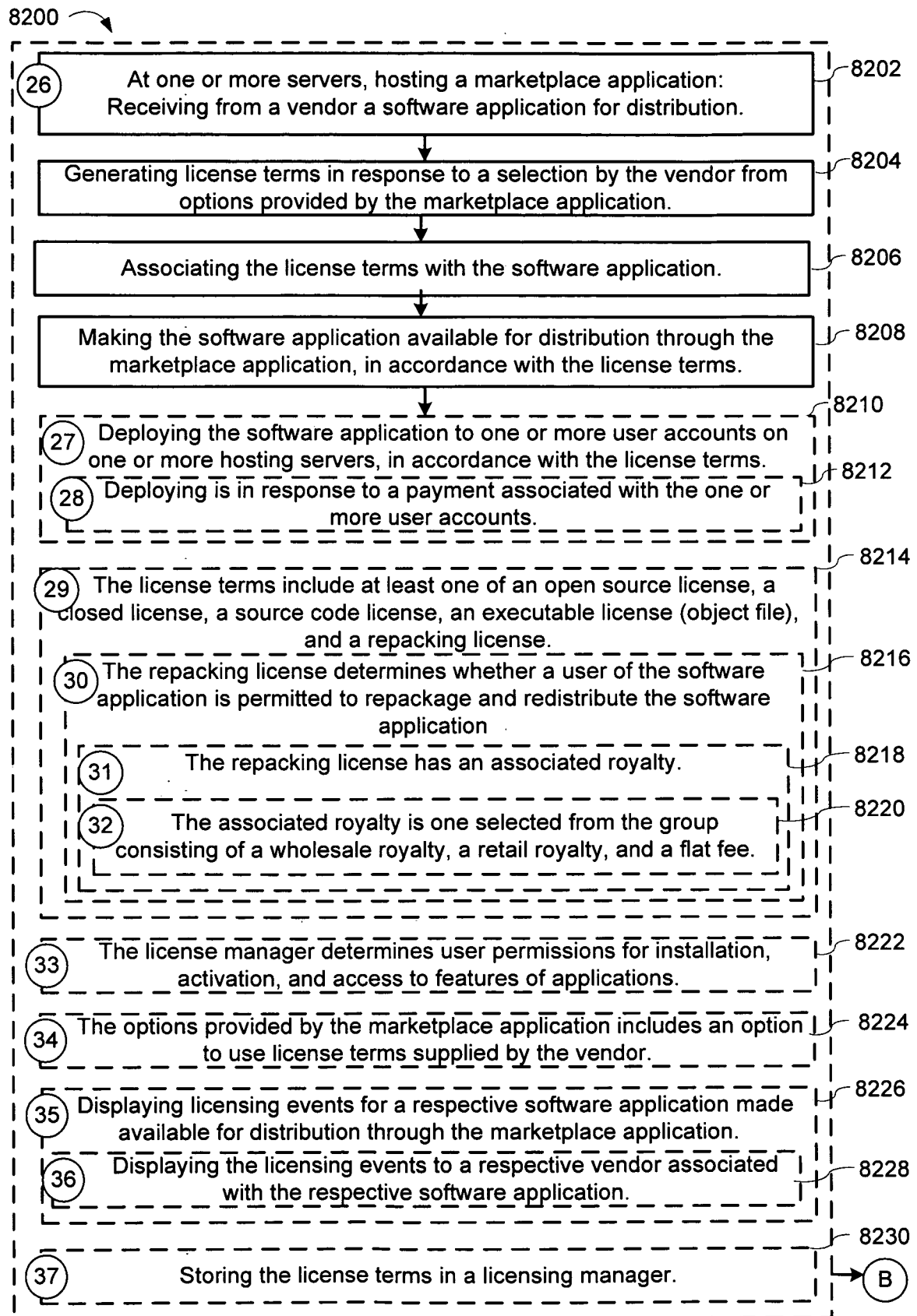


Figure 82

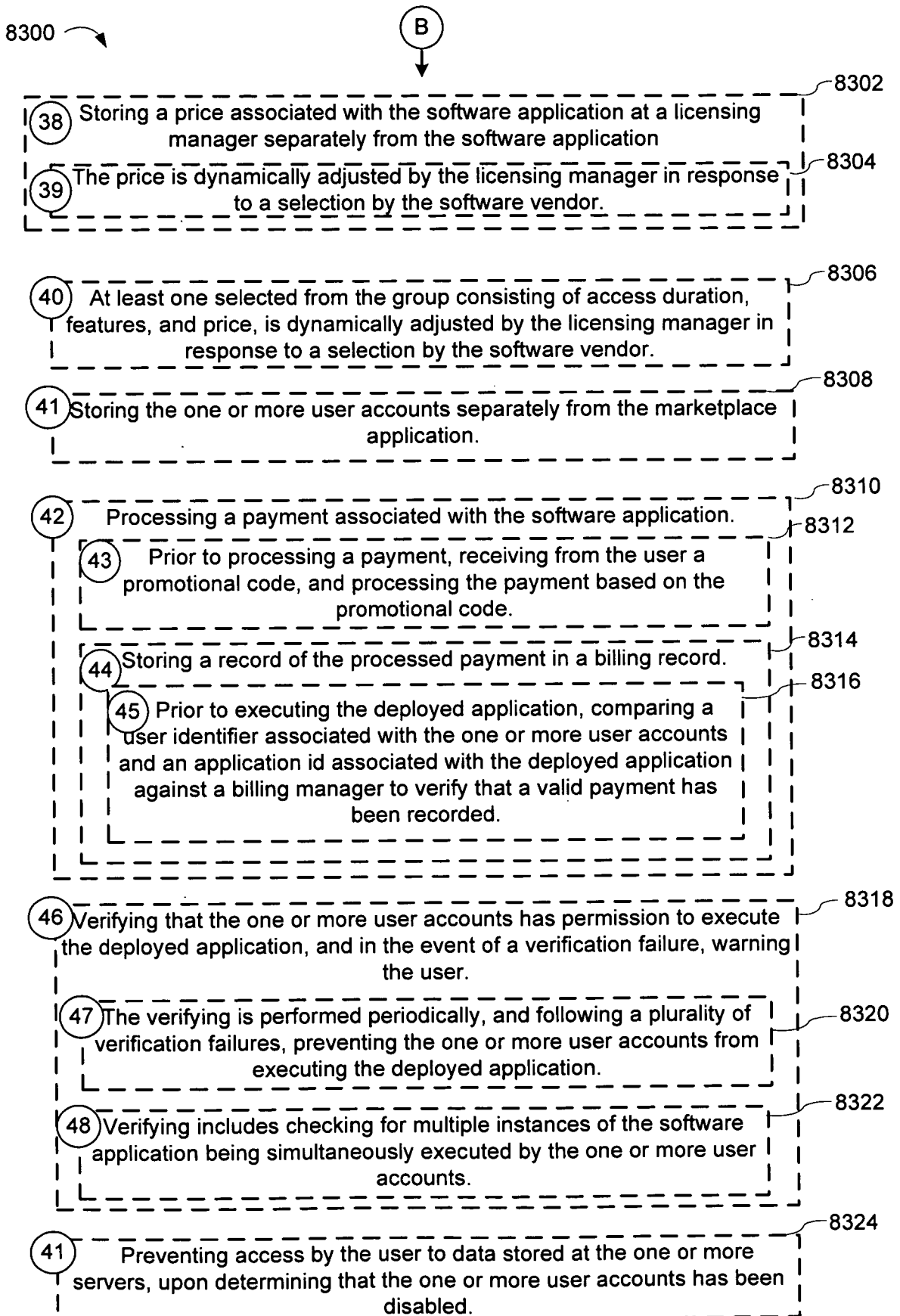


Figure 83

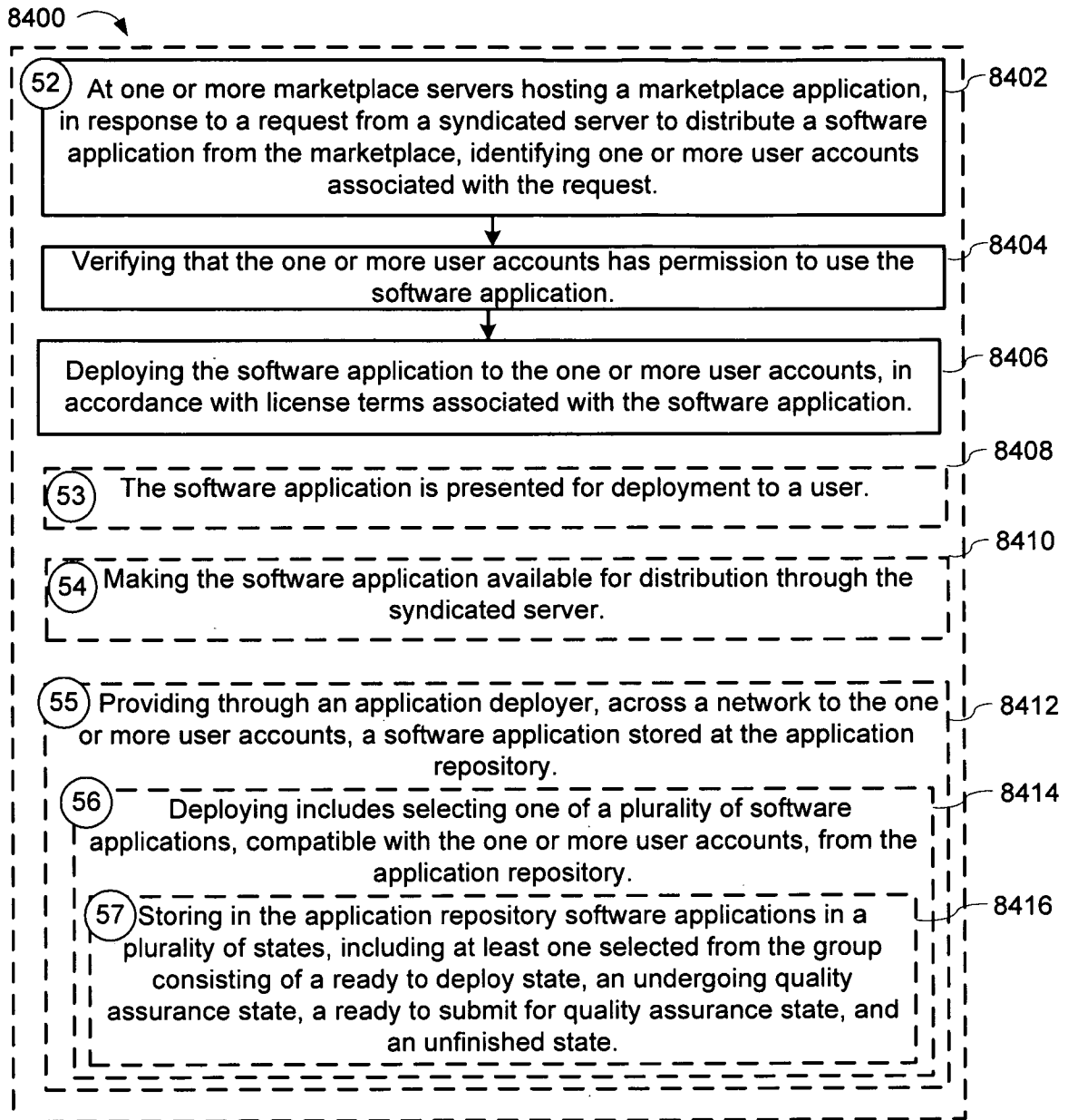


Figure 84

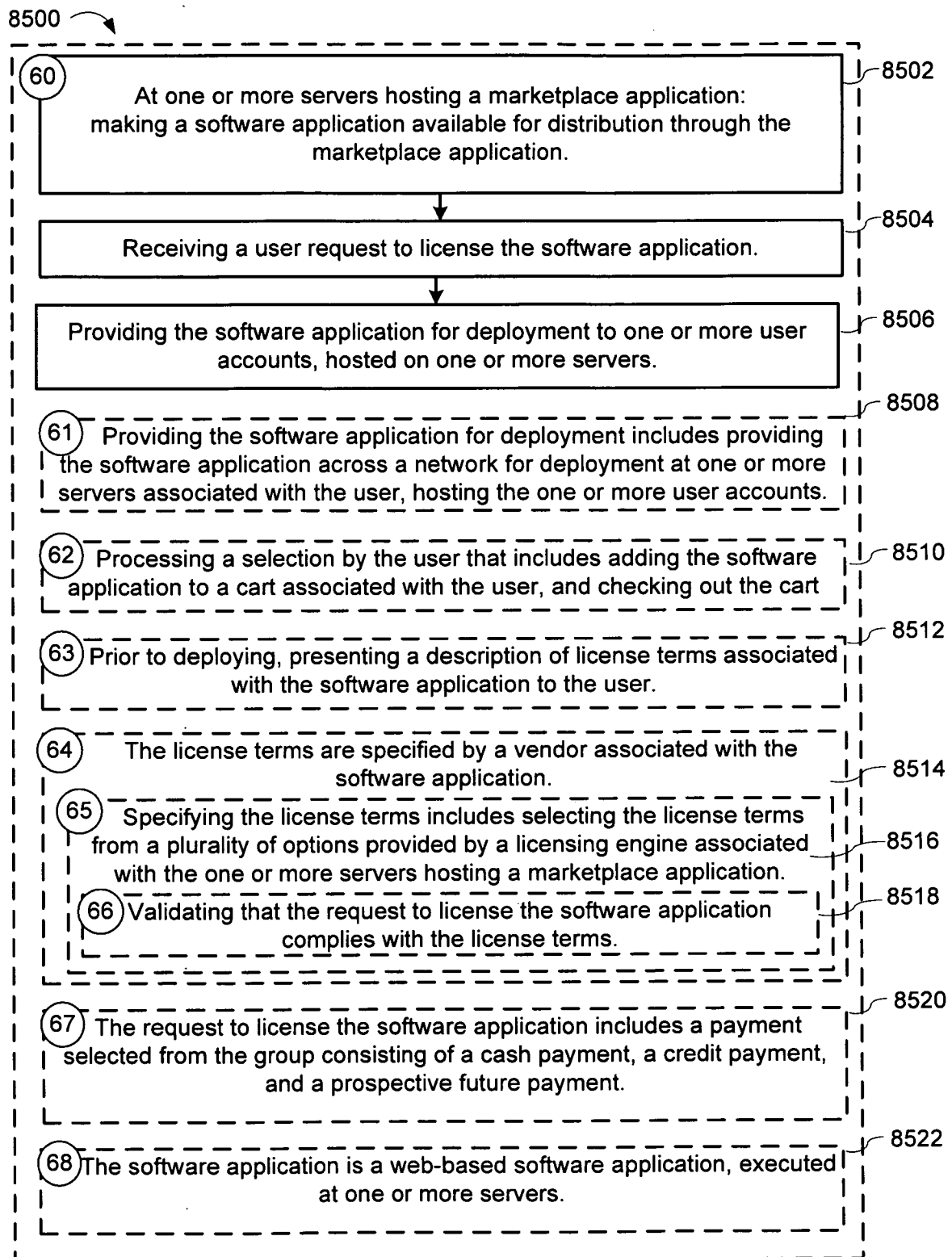
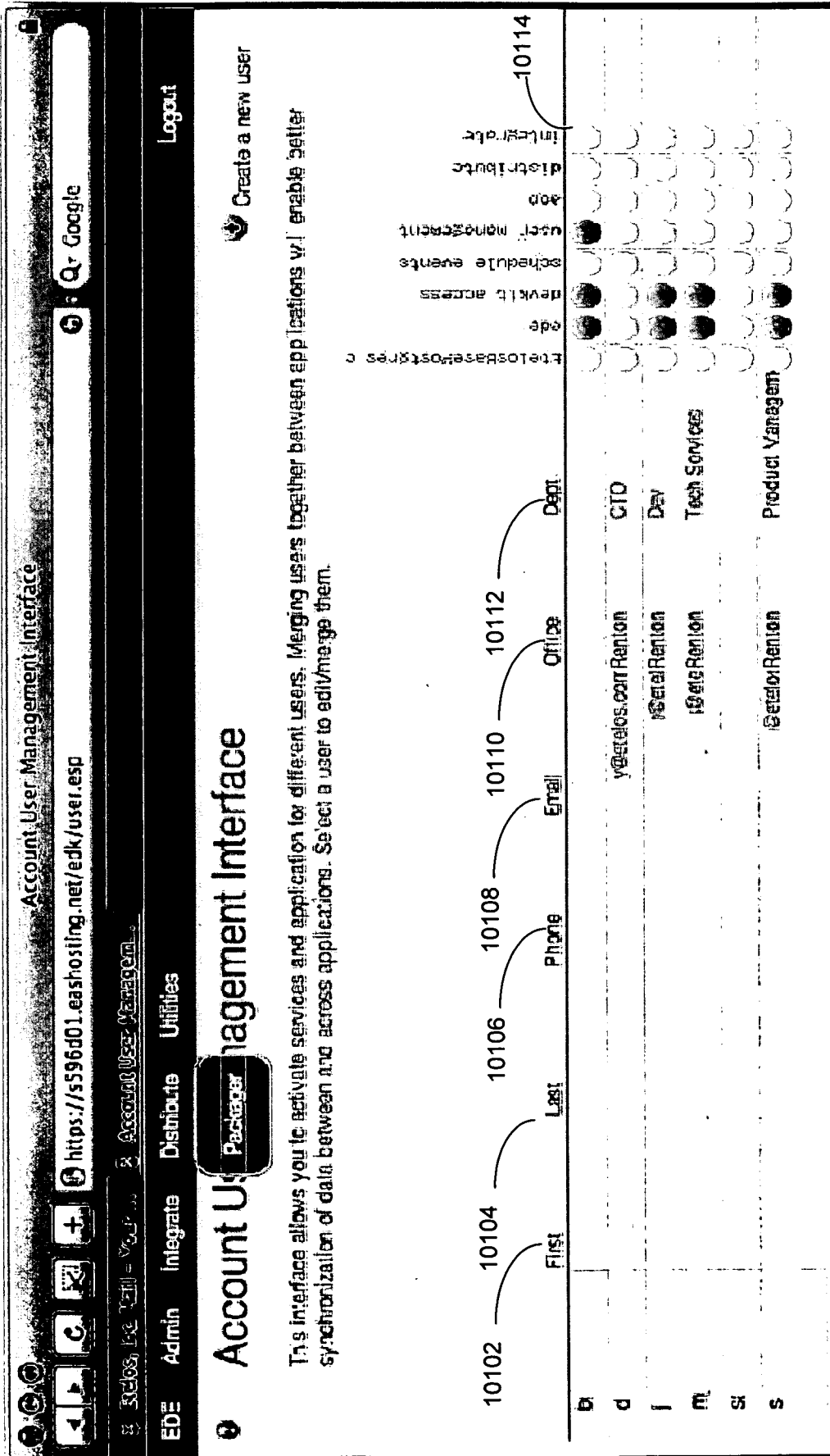


Figure 85



10100 →

Figure 101

Web Packager

https://ss36d01.eashosting.net:443/edk/webbuild/login.asp

Web Packager

EDE Admin Integrate Distribute Utilities Logout

Application Packager

This interface allows you to package up your application for distribution in the Marketplace and/or moving your application to another Etelos hosted solution.

Enter the credentials for your selected destination Etelos Store account

Marketplace Username:

Marketplace Password:

Note: Cookies and Javascript must both be enabled for this tool to function.

Figure 102

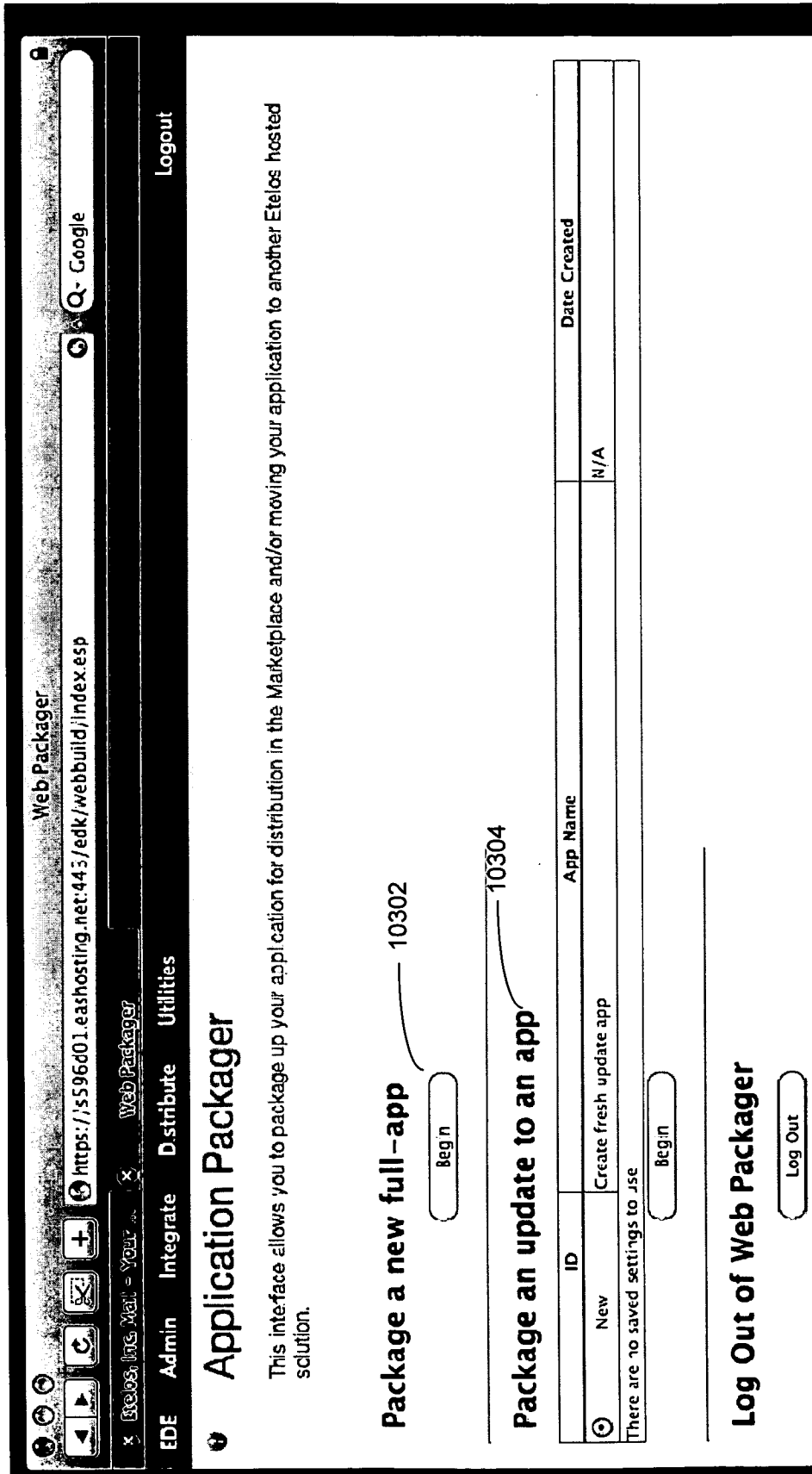


Figure 103

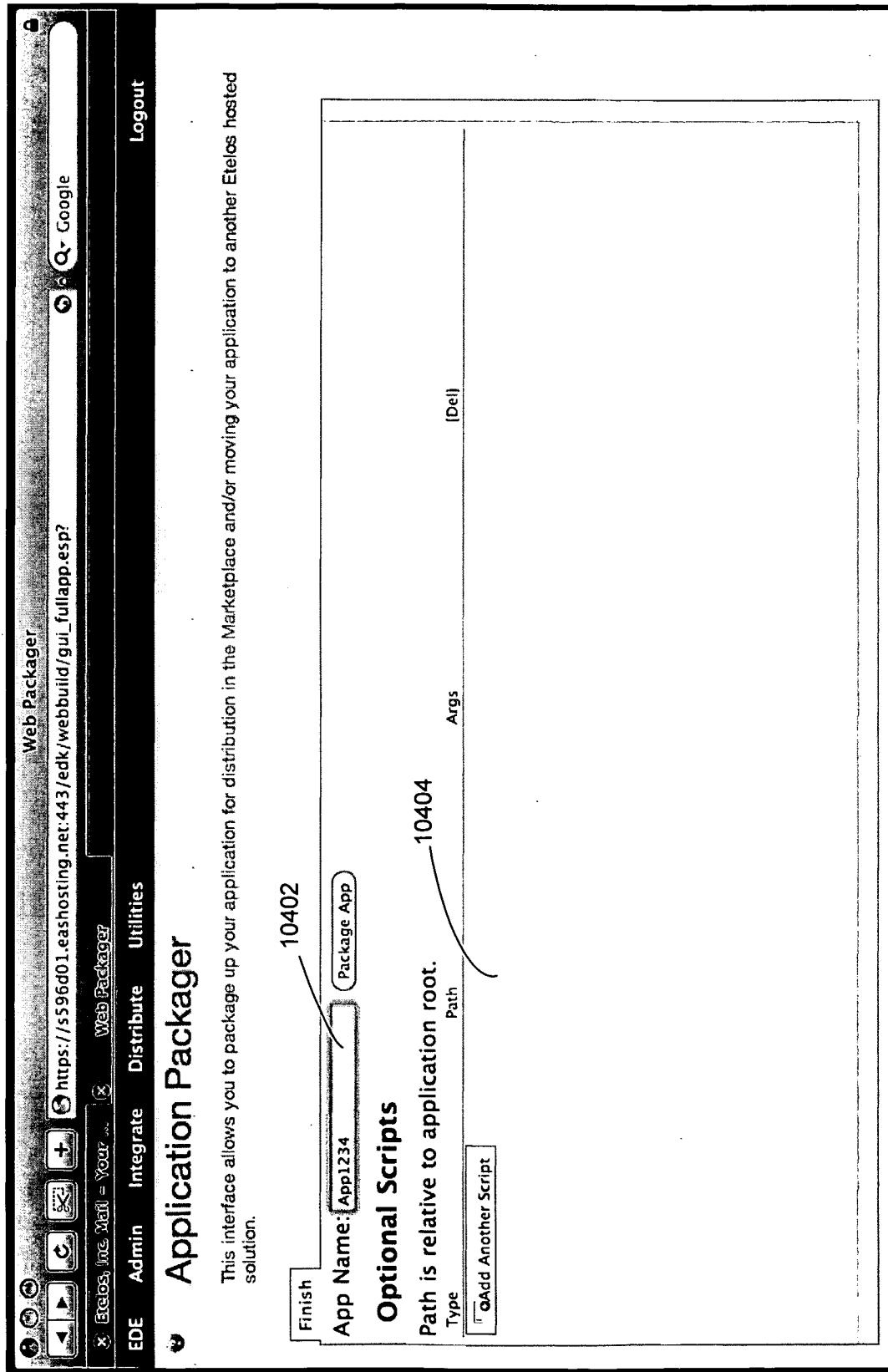


Figure 104

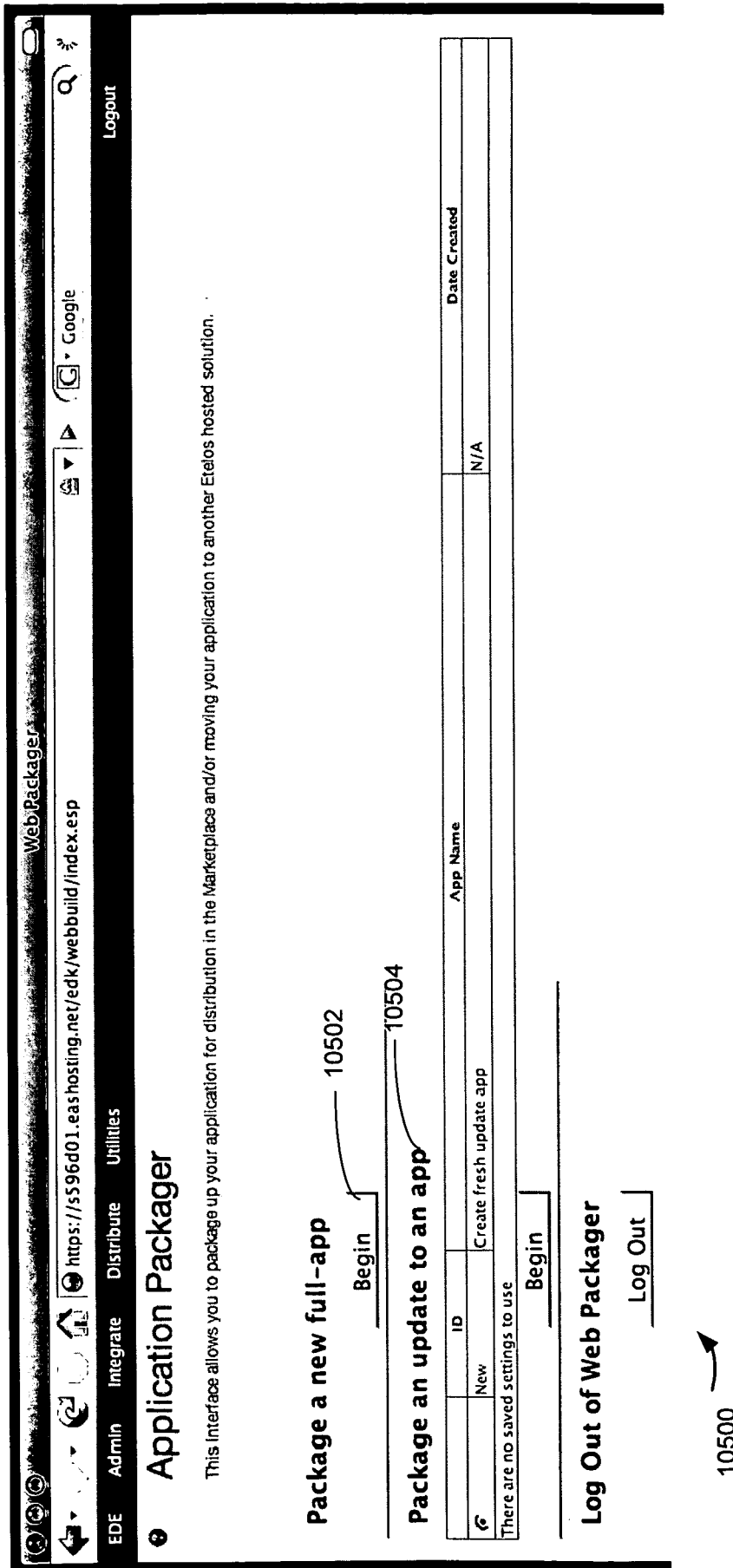


Figure 105

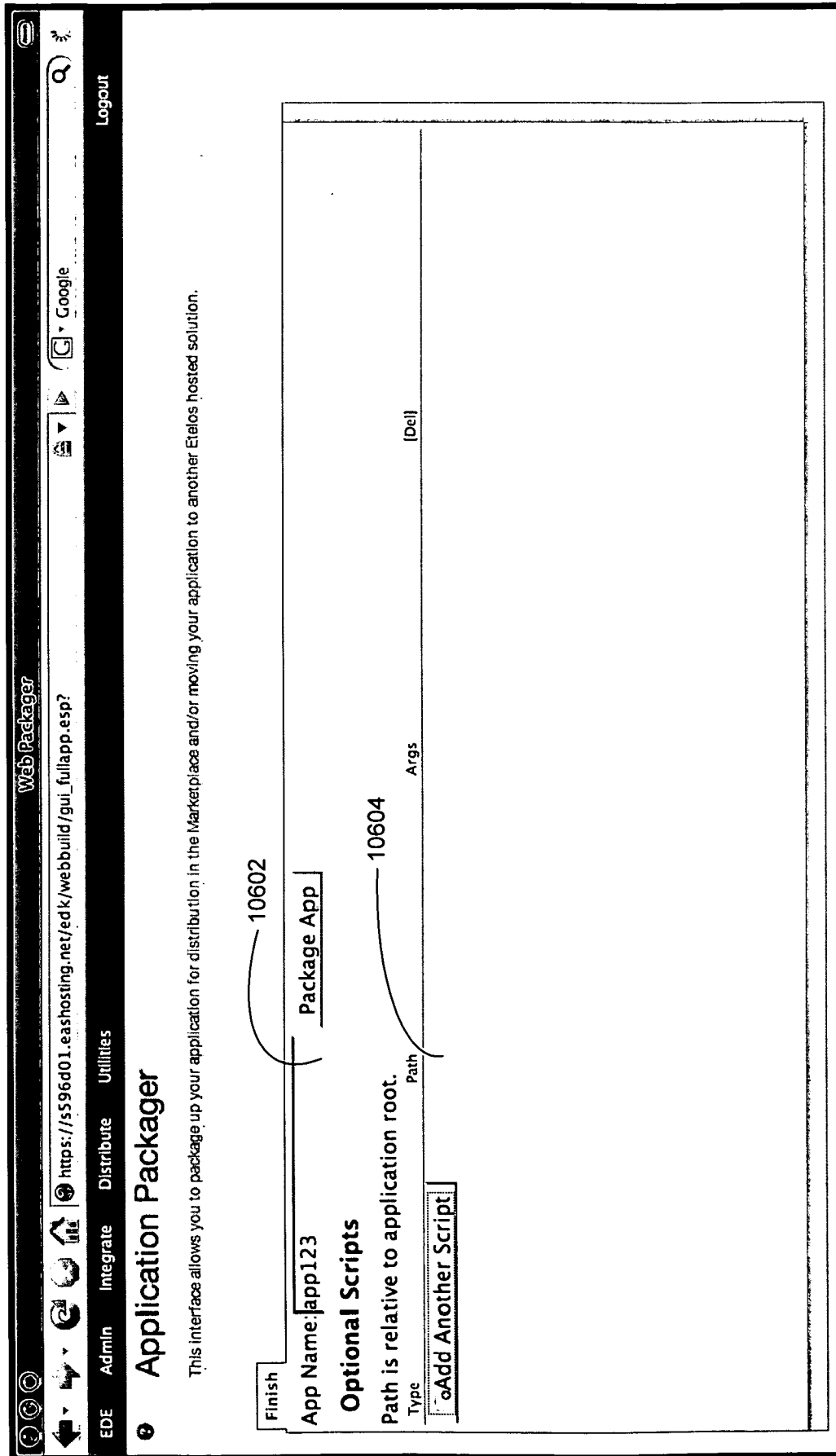


Figure 106

WebPackager

Logout

EDE Admin Integrate Distribute Utilities

Application Packager

This interface allows you to package up your application for distribution in the Marketplace and/or moving your application to another Etelos hosted solution.

Enter the credentials for your selected destination Etelos Store account

Marketplace Username:

Marketplace Password:

[Log In](#)

Note: Cookies and Javascript must both be enabled for this tool to function.

10700 →

Figure 107

EADL Admin Integrate Distribute Utilities Logout

Web Packager
https://ss596d01.eashtesting.net/sdk/webbuild/gui_miniapp.asp?build_id=0

Application Packager

This interface allows you to package up your application for distribution in the MarketPlace and/or moving your application to another Ekaos hosted solution.

10802

Introduction Files Web Email News Binder/Contact/Group Set Variables Schedule/Sync Finish		
Items		Meaning
☒	[Icon]	This resource will be part of your mini-app. If a container, all resources underneath will be included. Clicking will change it's status.
☐	[Icon]	This resource will not be included in your mini-app.
☐	[Icon]	Some of the items in this container will be included in your mini-app, but not all of them.
☒	[Icon]	This container can be expanded by clicking..
☒	[Icon]	This container is already expanded, and can be contracted by clicking..
☒	[Icon]	Working or waiting for server response.
☒	[Icon]	Normal file folder
☒	[Icon]	Folder of publications or issues
☒	[Icon]	Folder of binders or tables
☒ Highlighted Text	[Icon]	Highlighted items are those that have been created or changed since a saved build.

Done ss596d01.eashtesting.net 0.9125

Figure 108

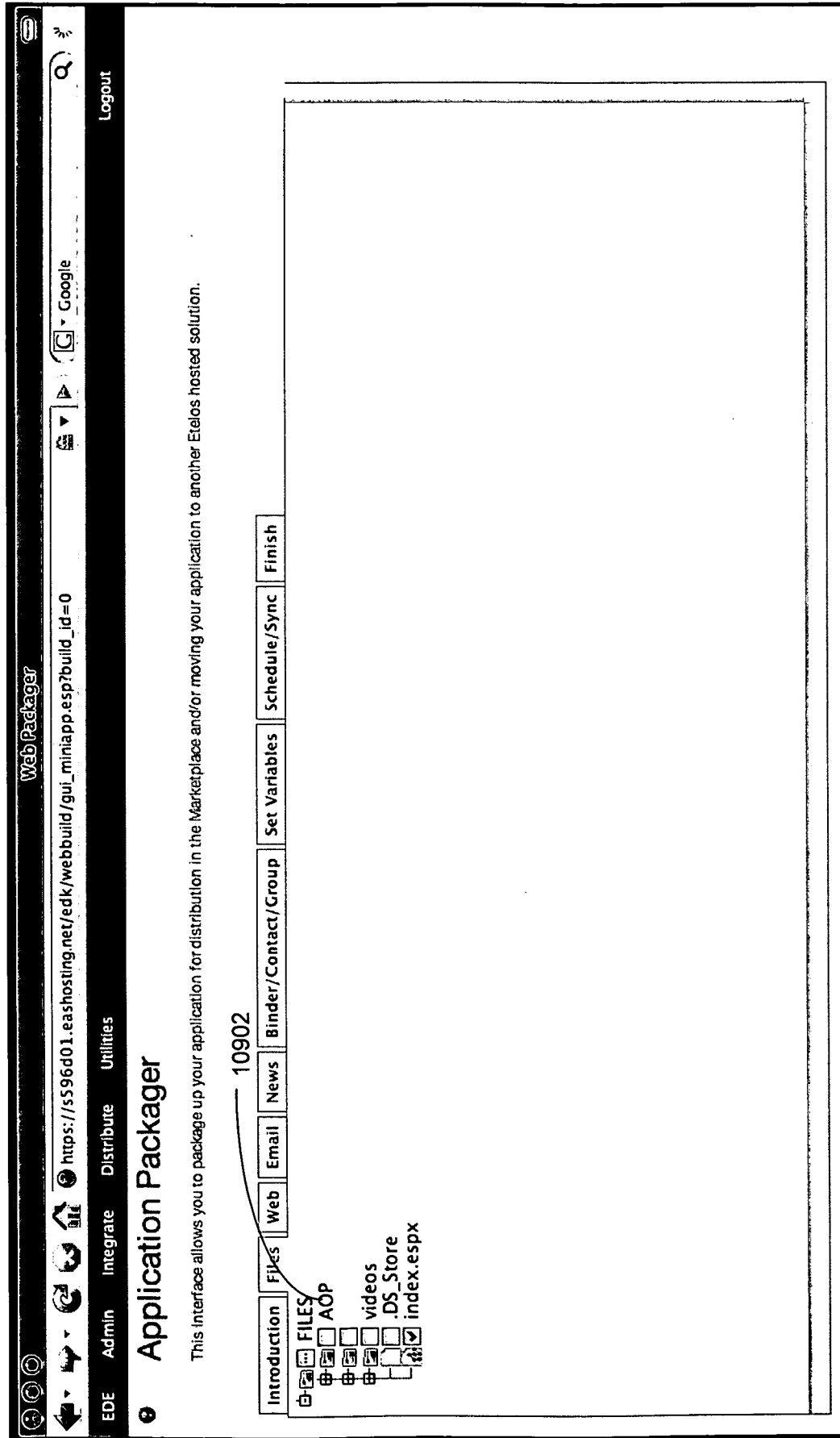


Figure 109

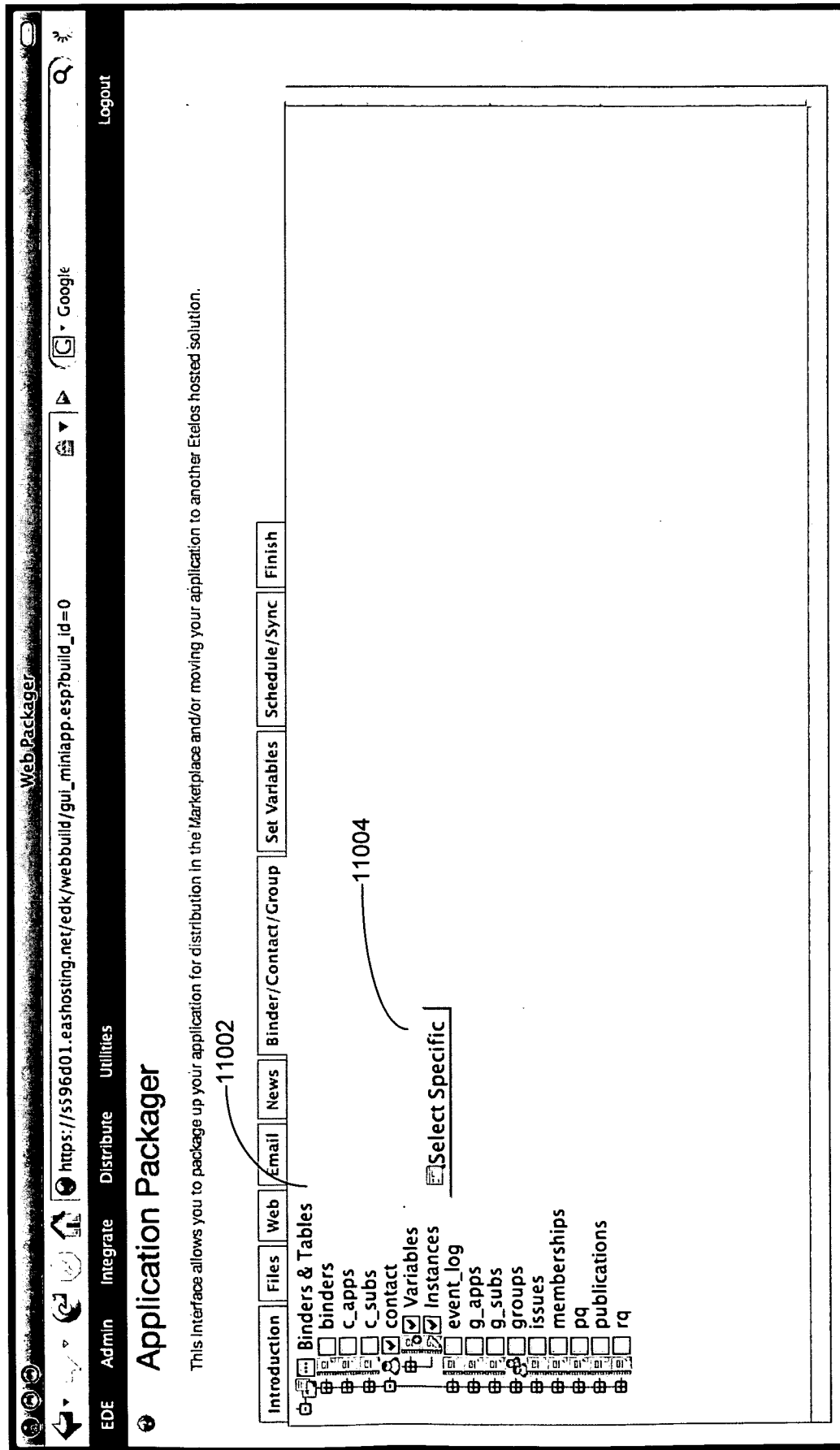
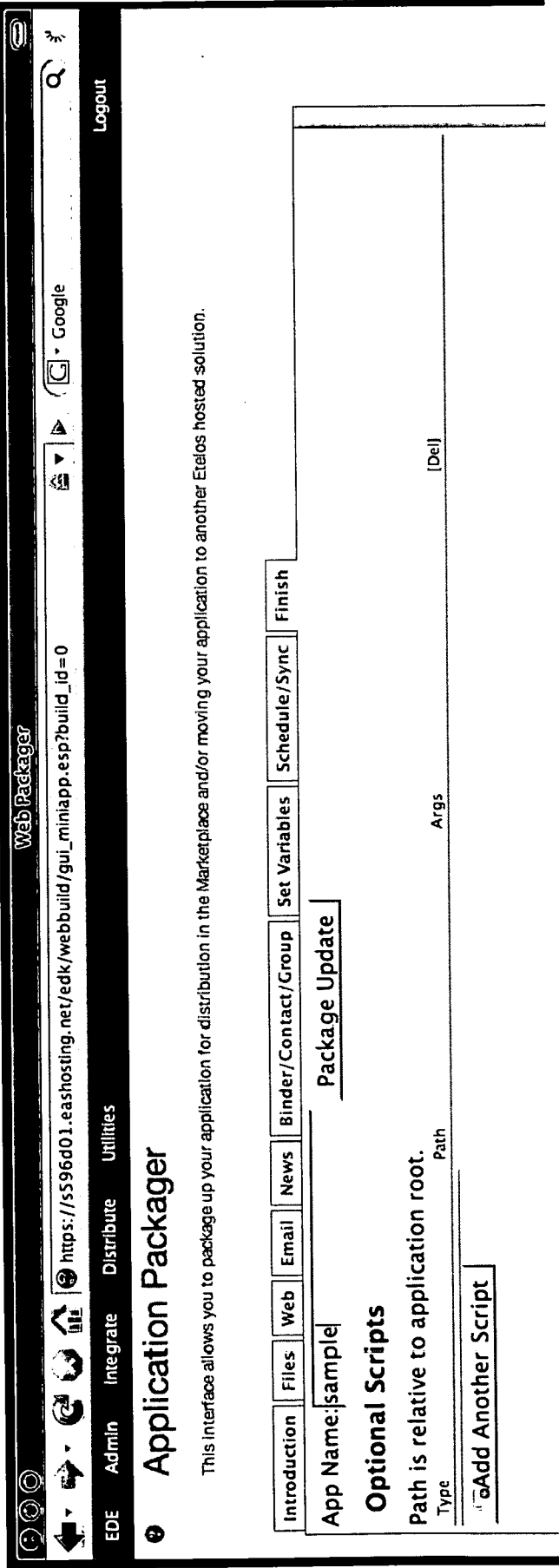


Figure 110

11000



11100 →

Figure 111

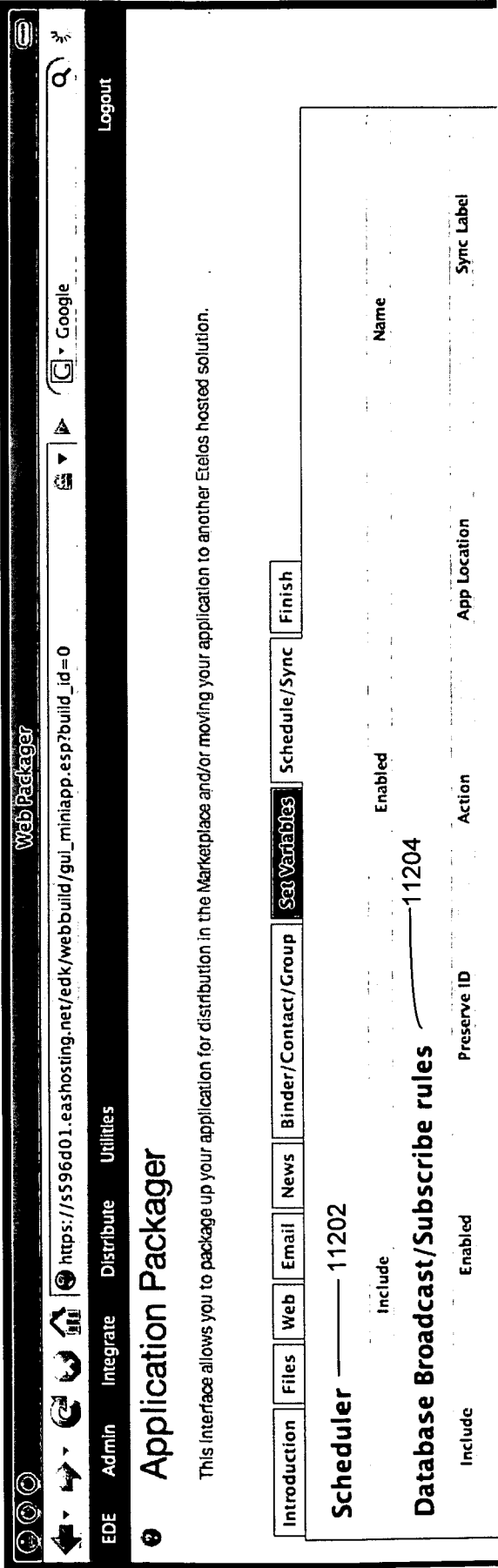


Figure 112

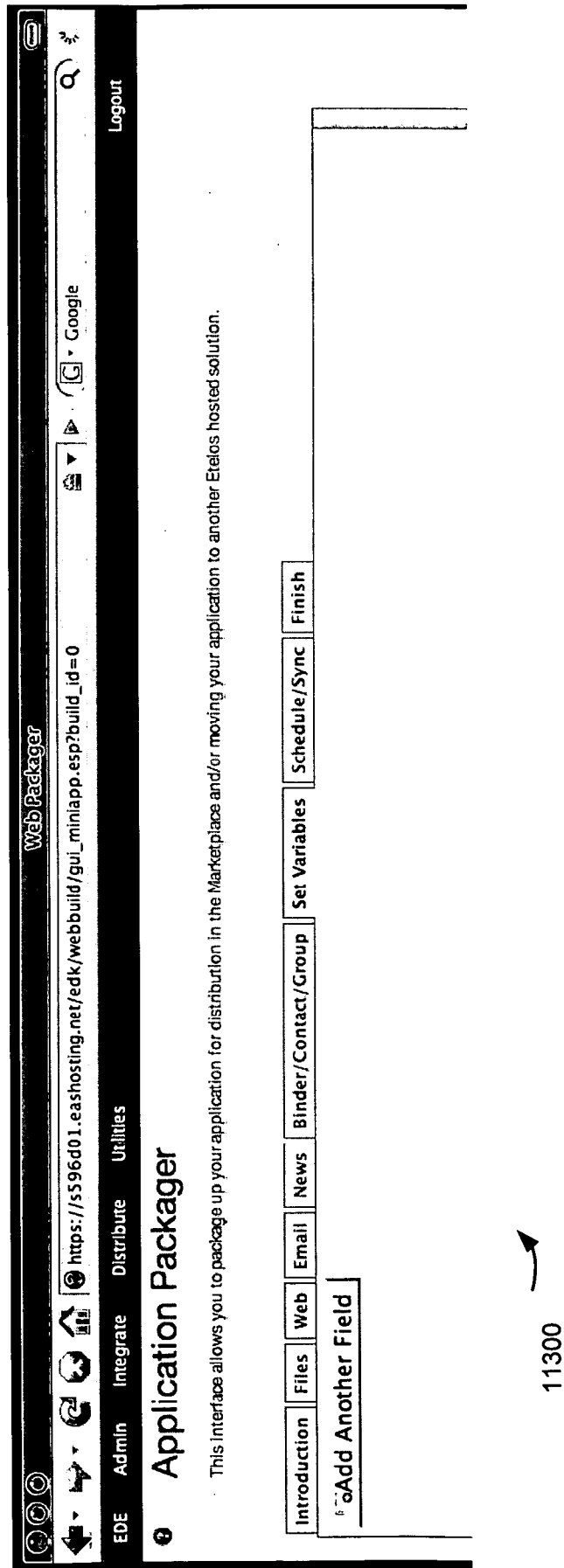


Figure 113

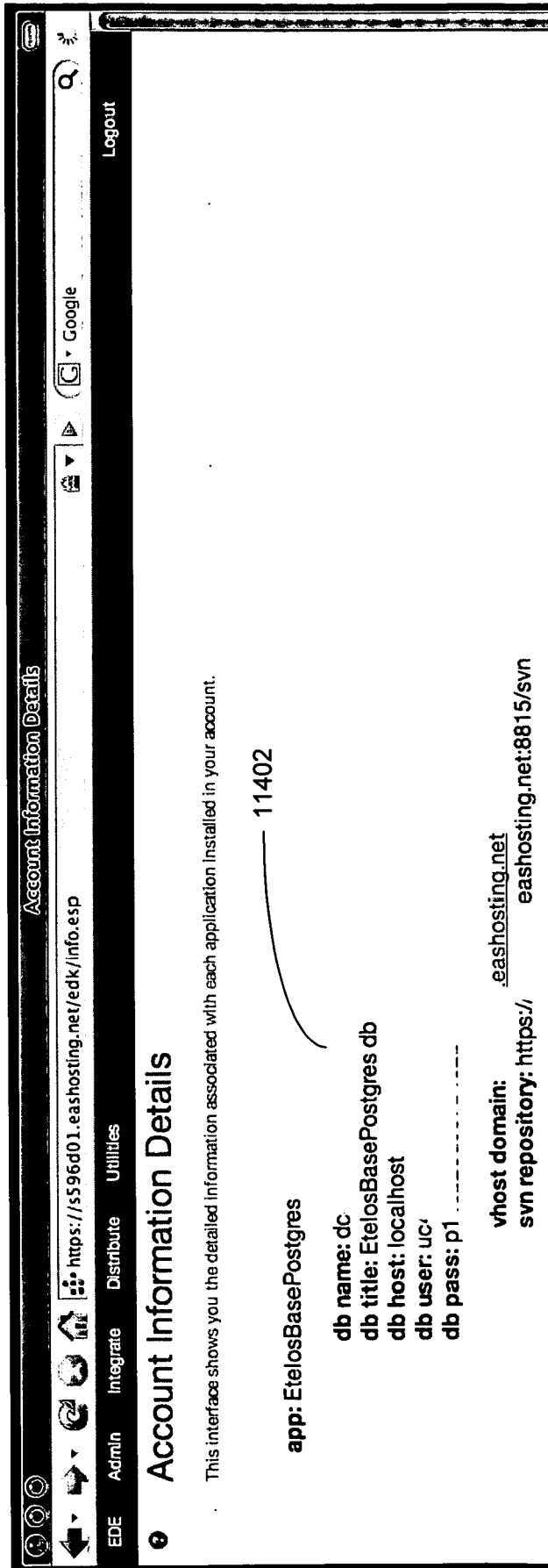


Figure 114

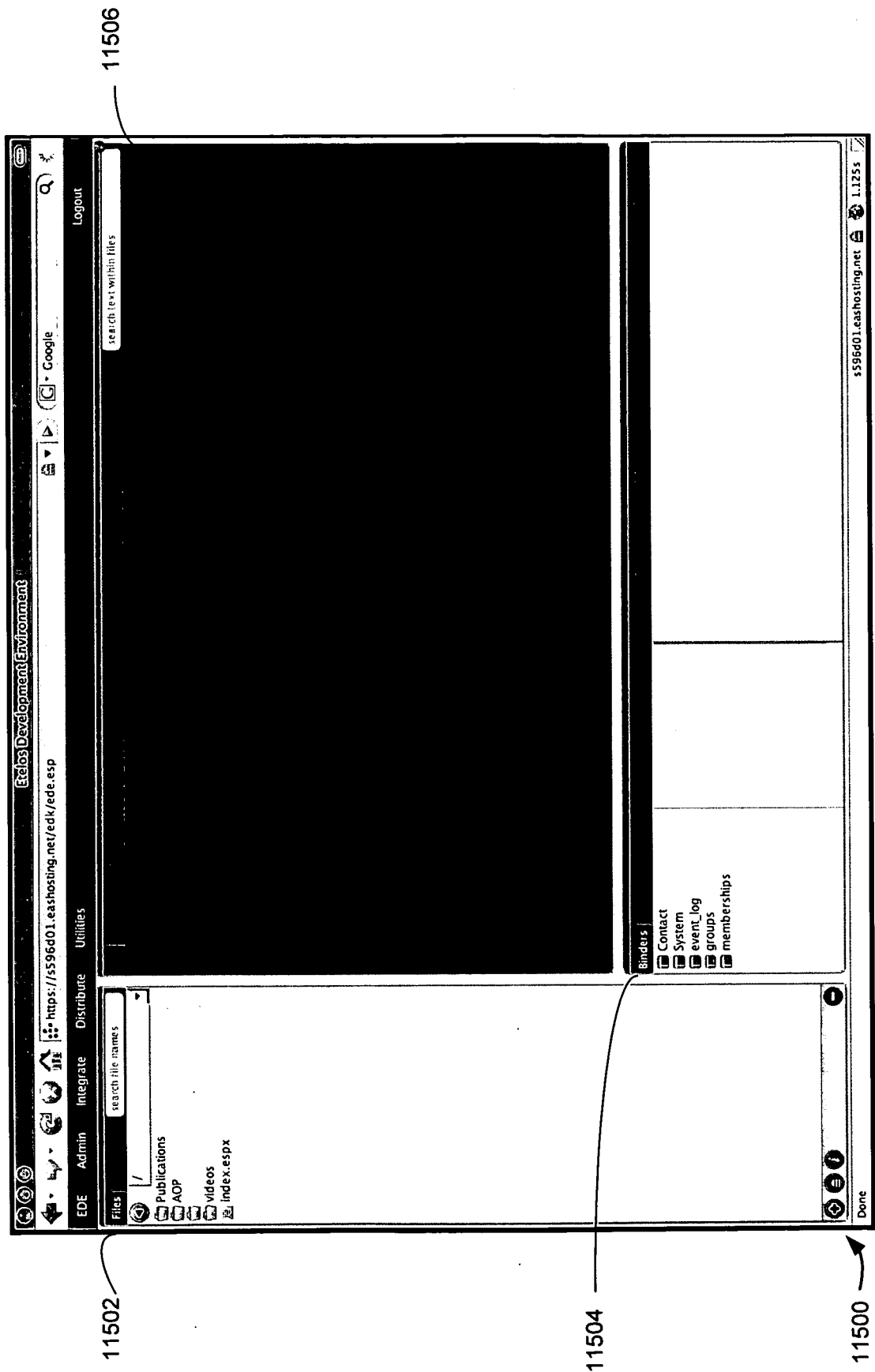


Figure 115

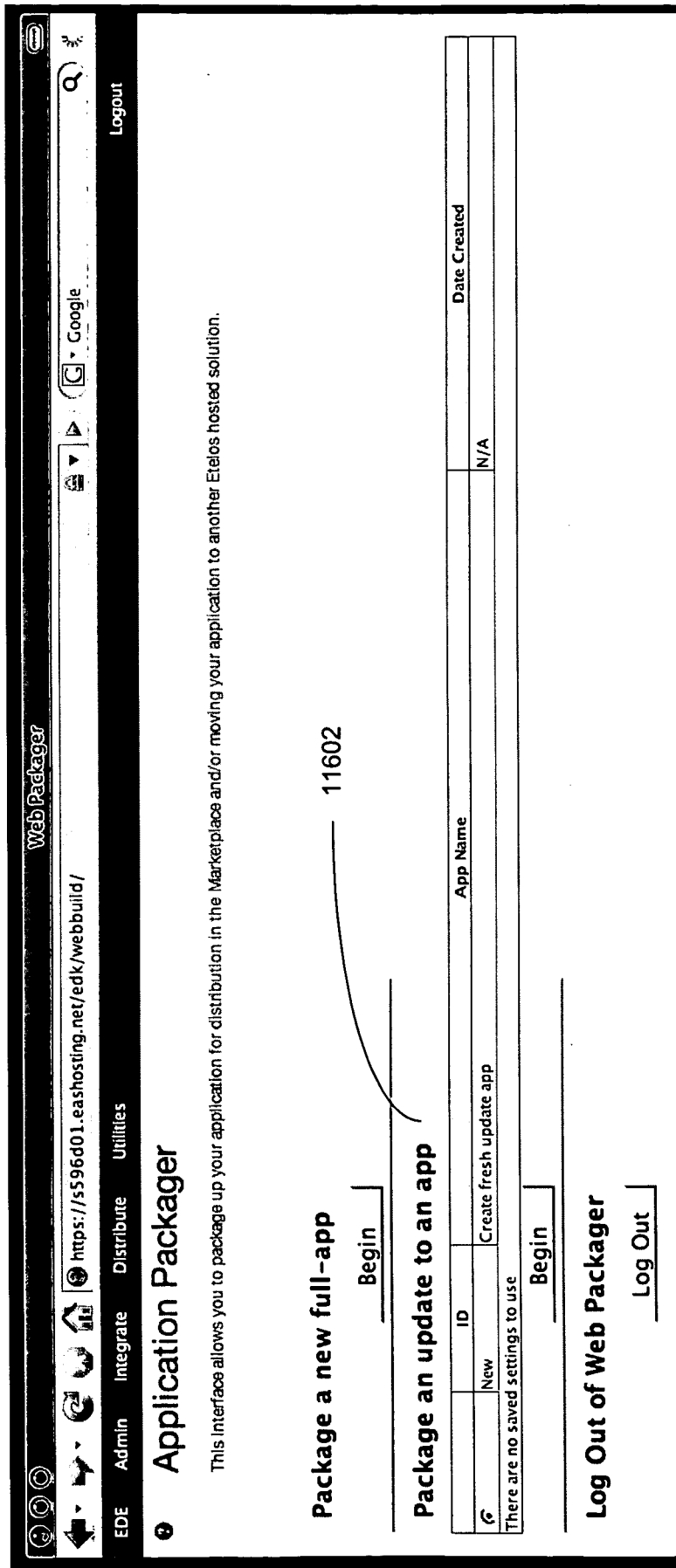


Figure 116

11702

11704

11706

11700

AOP Sync Rules Manager

This interface allows you to manage synchronization rules that apply to applications that are configured to work with AOP (Eteios Apps on a Plane).

11714 — Auto Increment — Create a new sync rule — 11716

11710 — 11712

Status	Code	App	App Table	App Column
	Sugar_v6_SYNC_1 db	calls_contacts		accept_status
	Sugar_v6_SYNC_1 db	calls_users		accept_status
	Sugar_v6_SYNC_1 db	meetings_contacts		accept_status
	Sugar_v6_SYNC_1 db	meetings_users		accept_status
	Sugar_v6_SYNC_1 db	ad_roles_actions		access_override
	Sugar_v6_SYNC_1 db	leads		account_description
	Sugar_v6_SYNC_1 db	accounts_bugs		account_id
	Sugar_v6_SYNC_1 db	accounts_cases		account_id
	Sugar_v6_SYNC_1 db	accounts_contacts		account_id
	Sugar_v6_SYNC_1 db	accounts_opportunities		account_id
	Sugar_v6_SYNC_1 db	cases		account_id
	Sugar_v6_SYNC_1 db	leads		account_id
	Sugar_v6_SYNC_1 db	projects_accounts		account_id
	Sugar_v6_SYNC_1 db	leads		account_name
	Sugar_v6_SYNC_1 db	prospects		account_name
	Sugar_v6_SYNC_1 db	accounts		account_type
	Sugar_v6_SYNC_1 db	ad_actions		adaccess
	Sugar_v6_SYNC_1 db	ad_actions		actype
	Sugar_v6_SYNC_1 db	tracker		action
	Sugar_v6_SYNC_1 db	ad_roles_actions		action_id

EDE Admin Integrate Distribute Utilities Logout

Figure 117

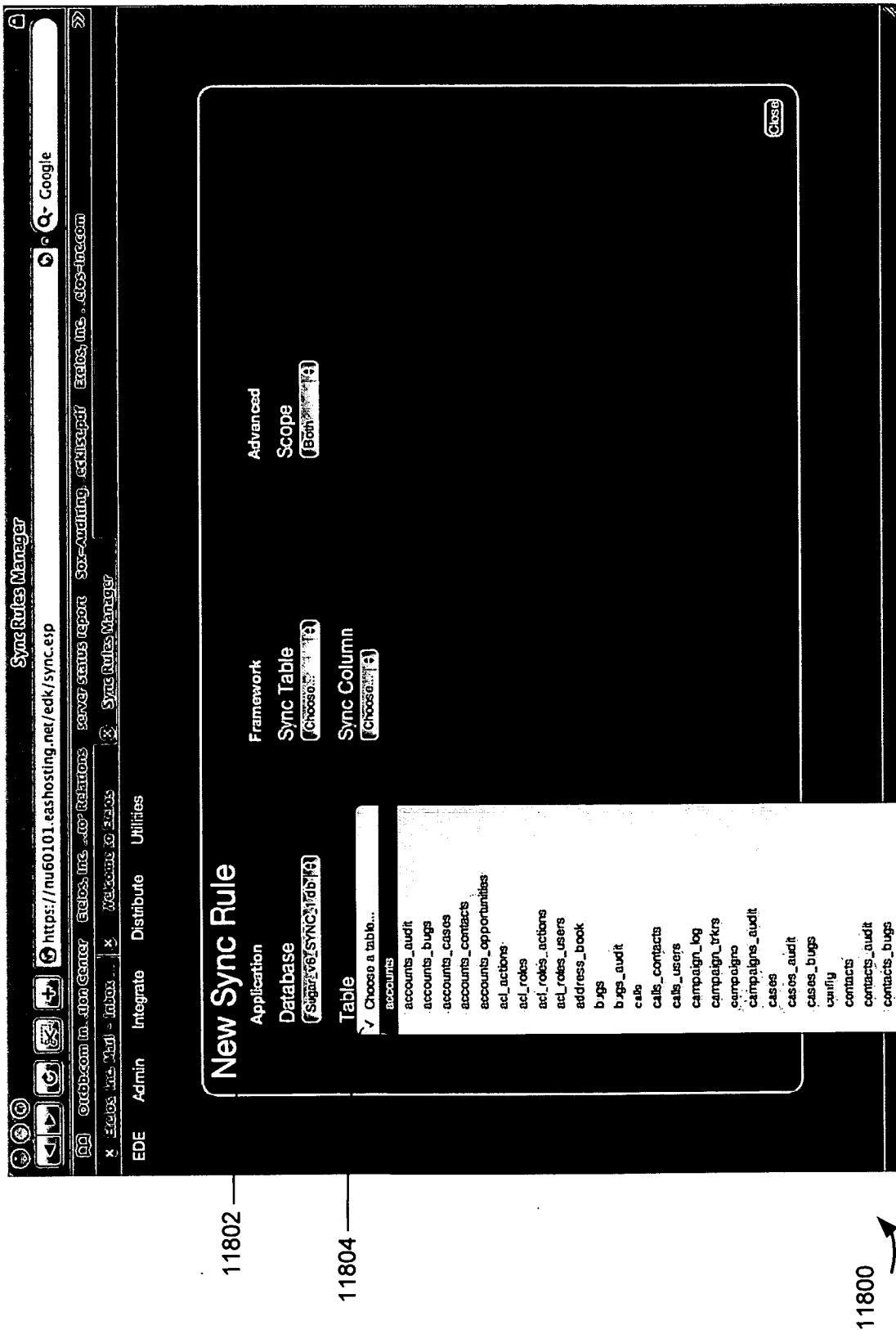


Figure 118

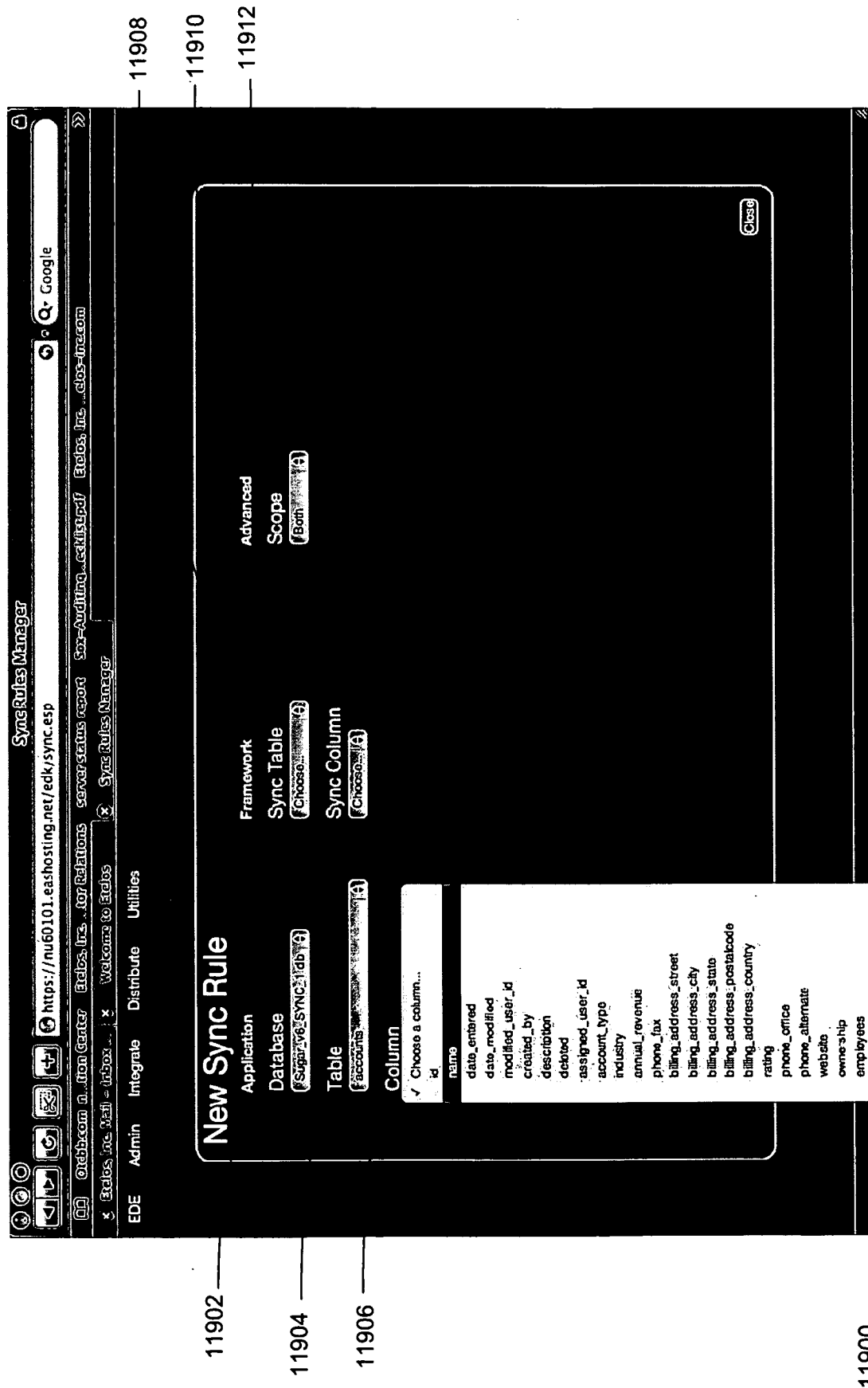


Figure 119

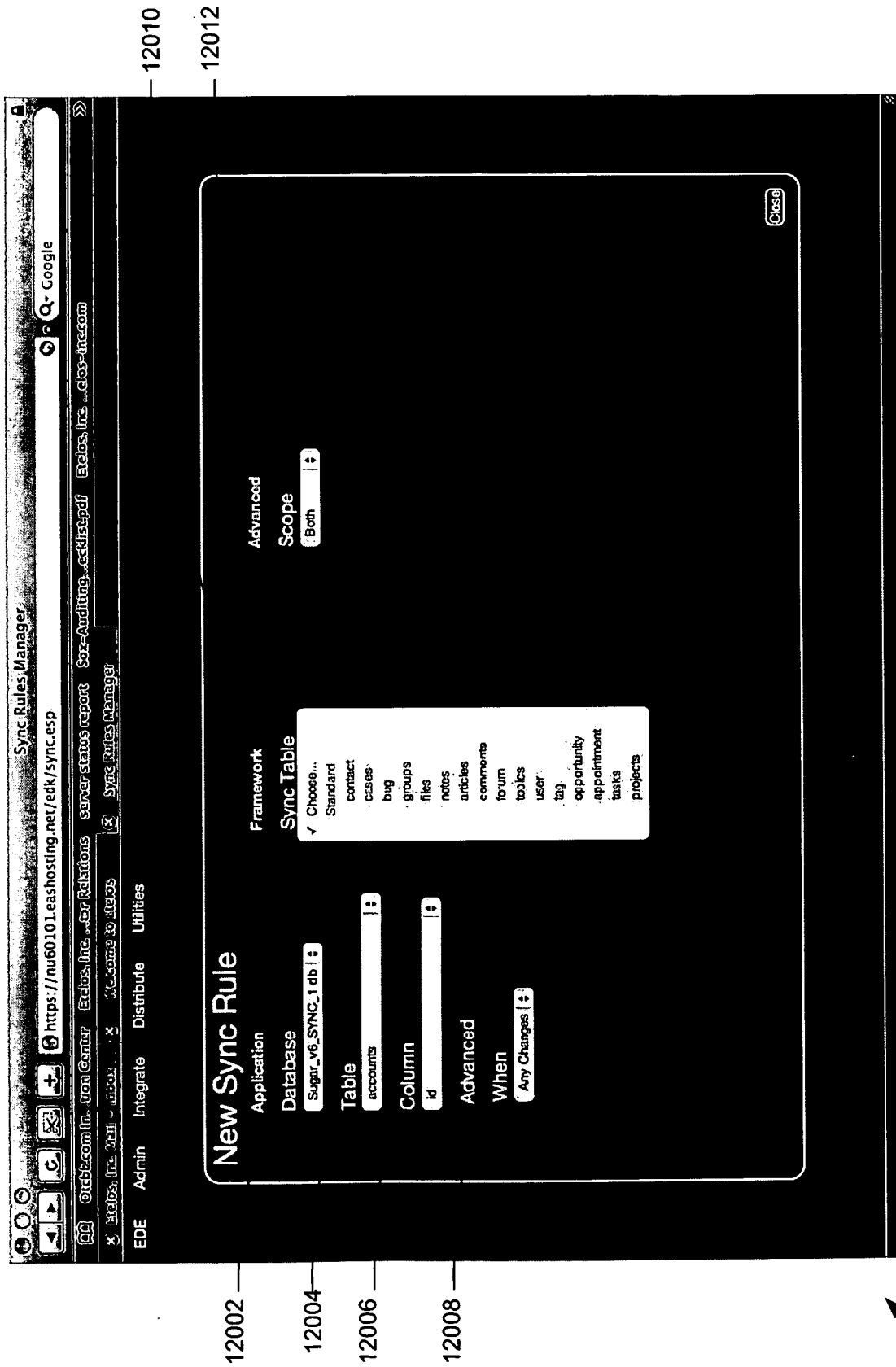


Figure 120

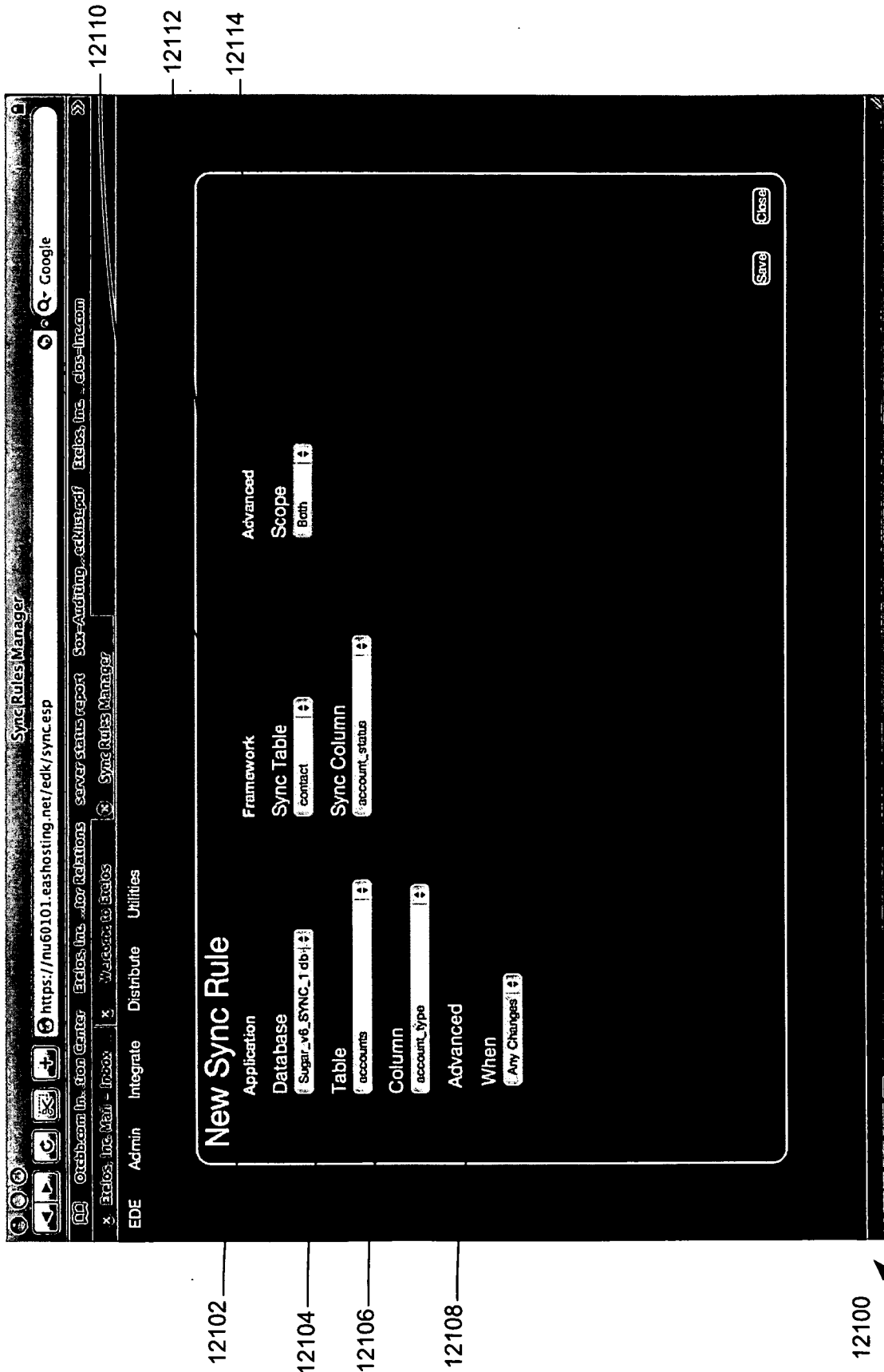


Figure 121

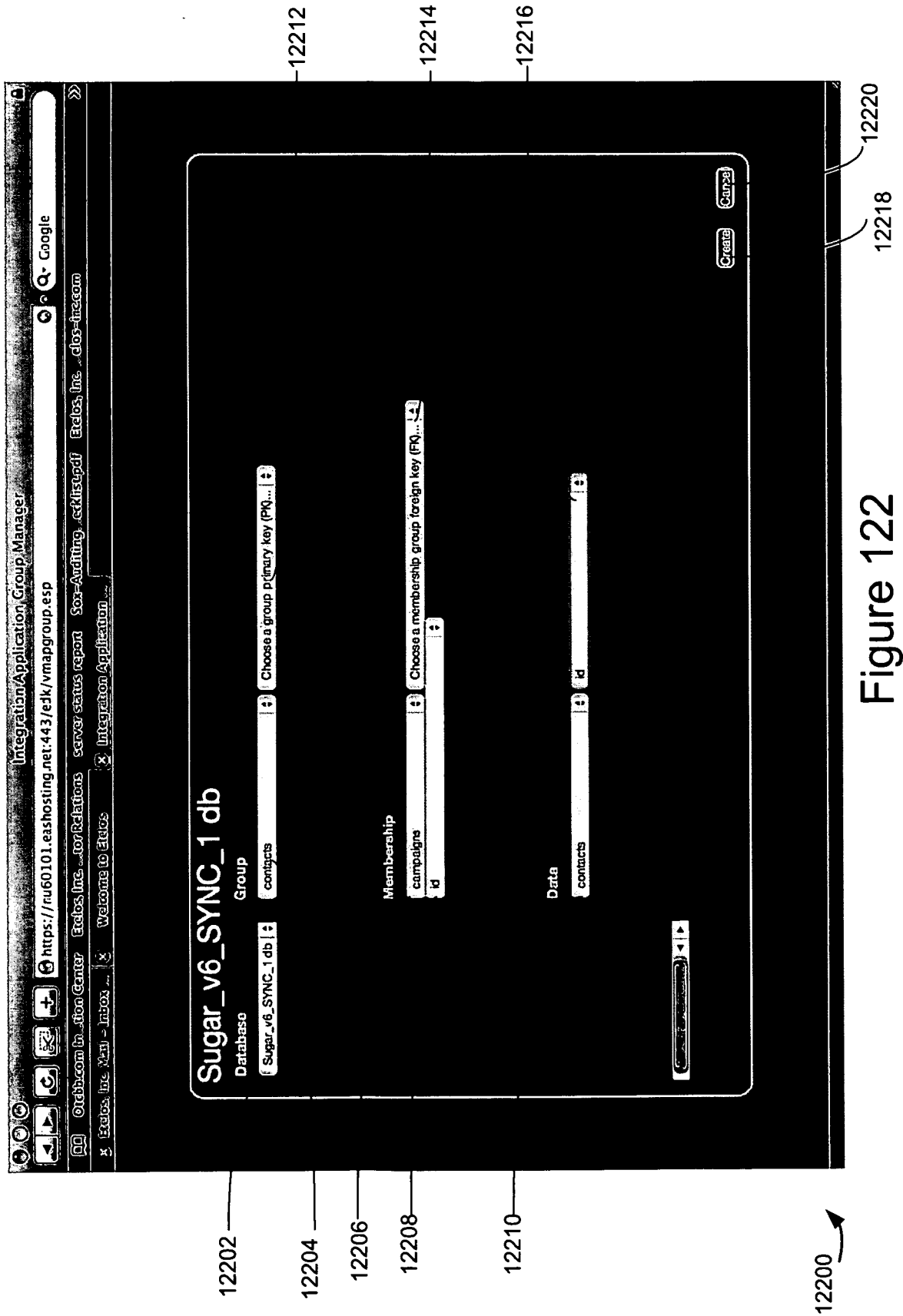


Figure 122

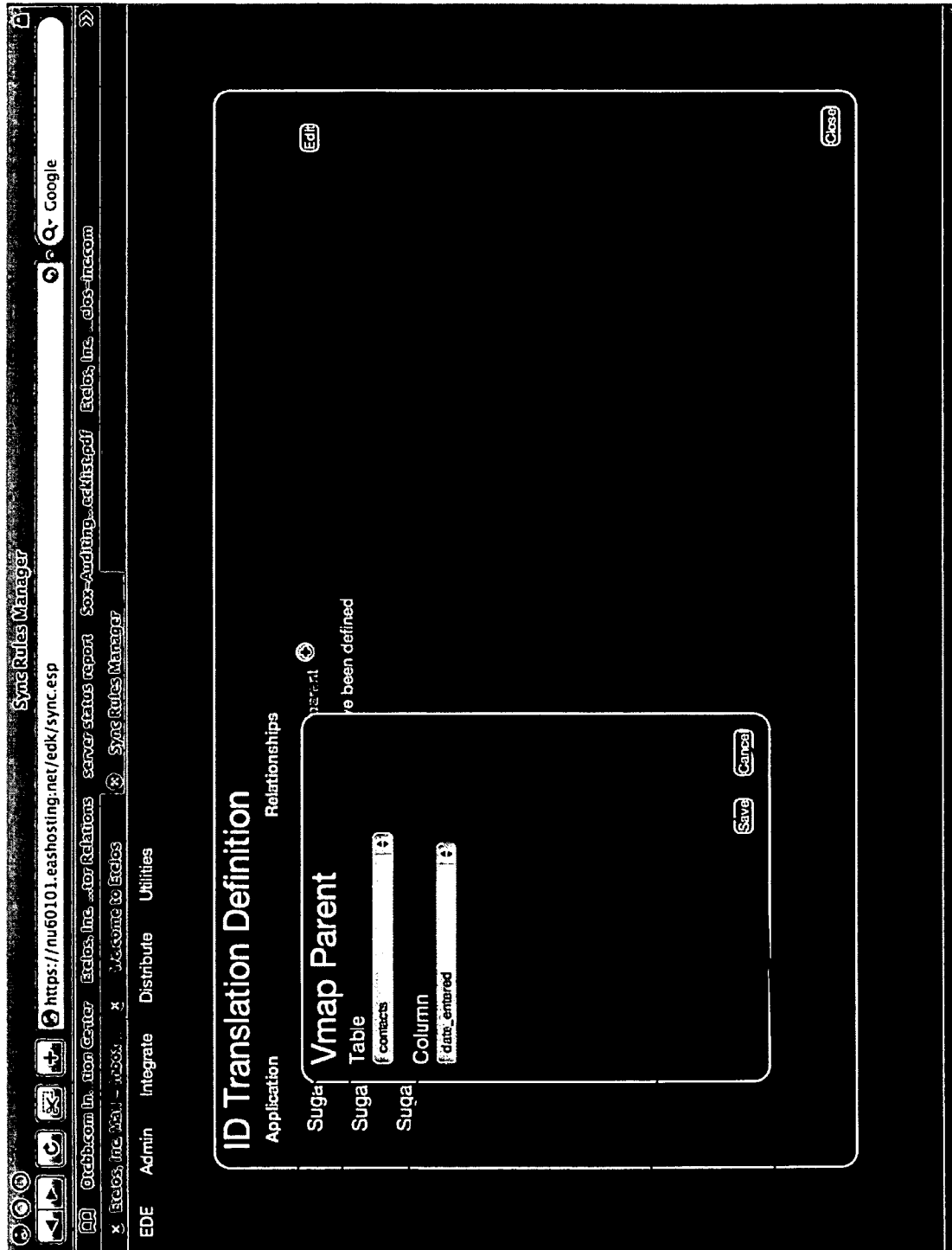


Figure 123

12302

12304

12306

12308

12310

12300



Figure 124

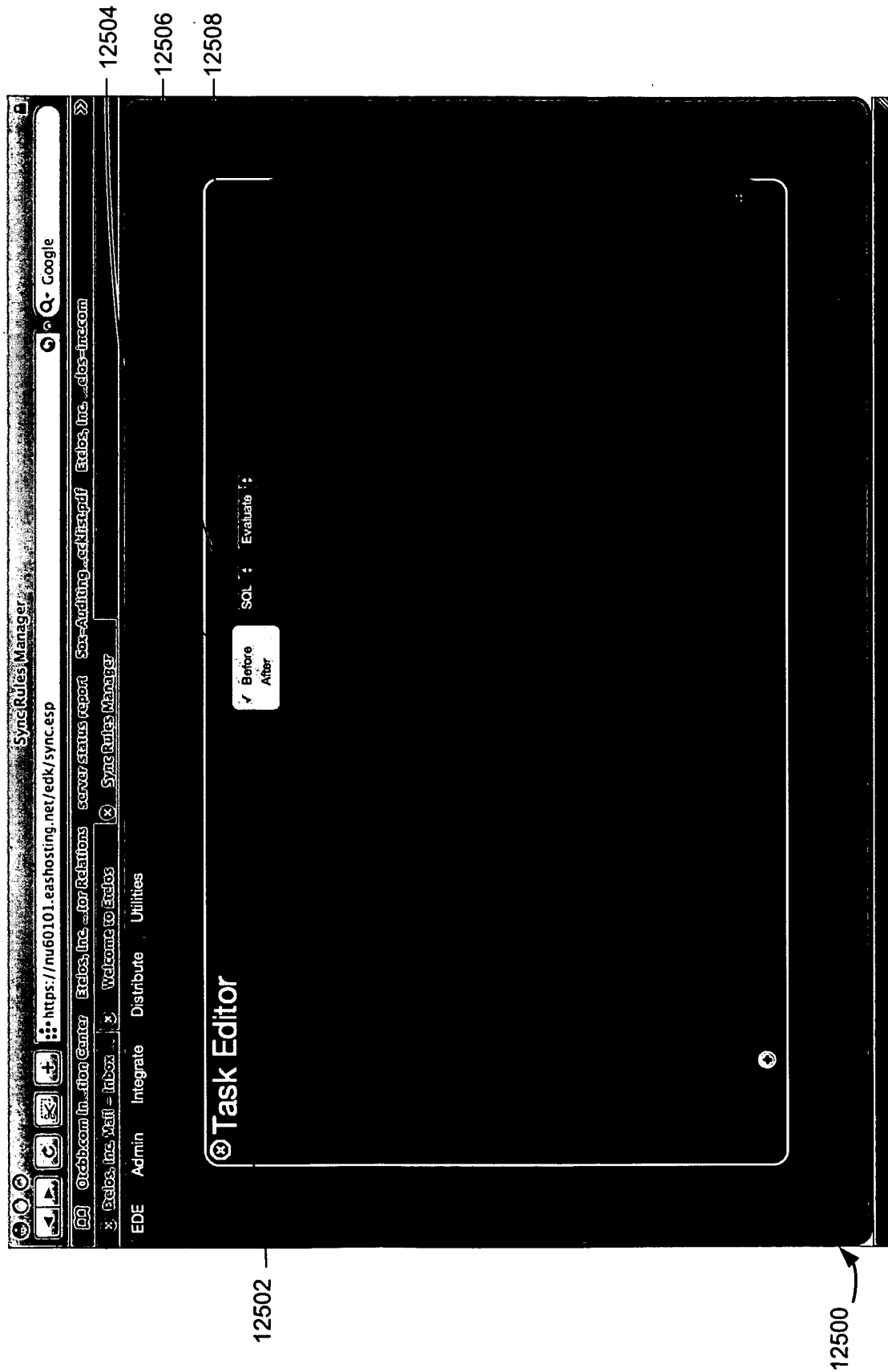


Figure 125

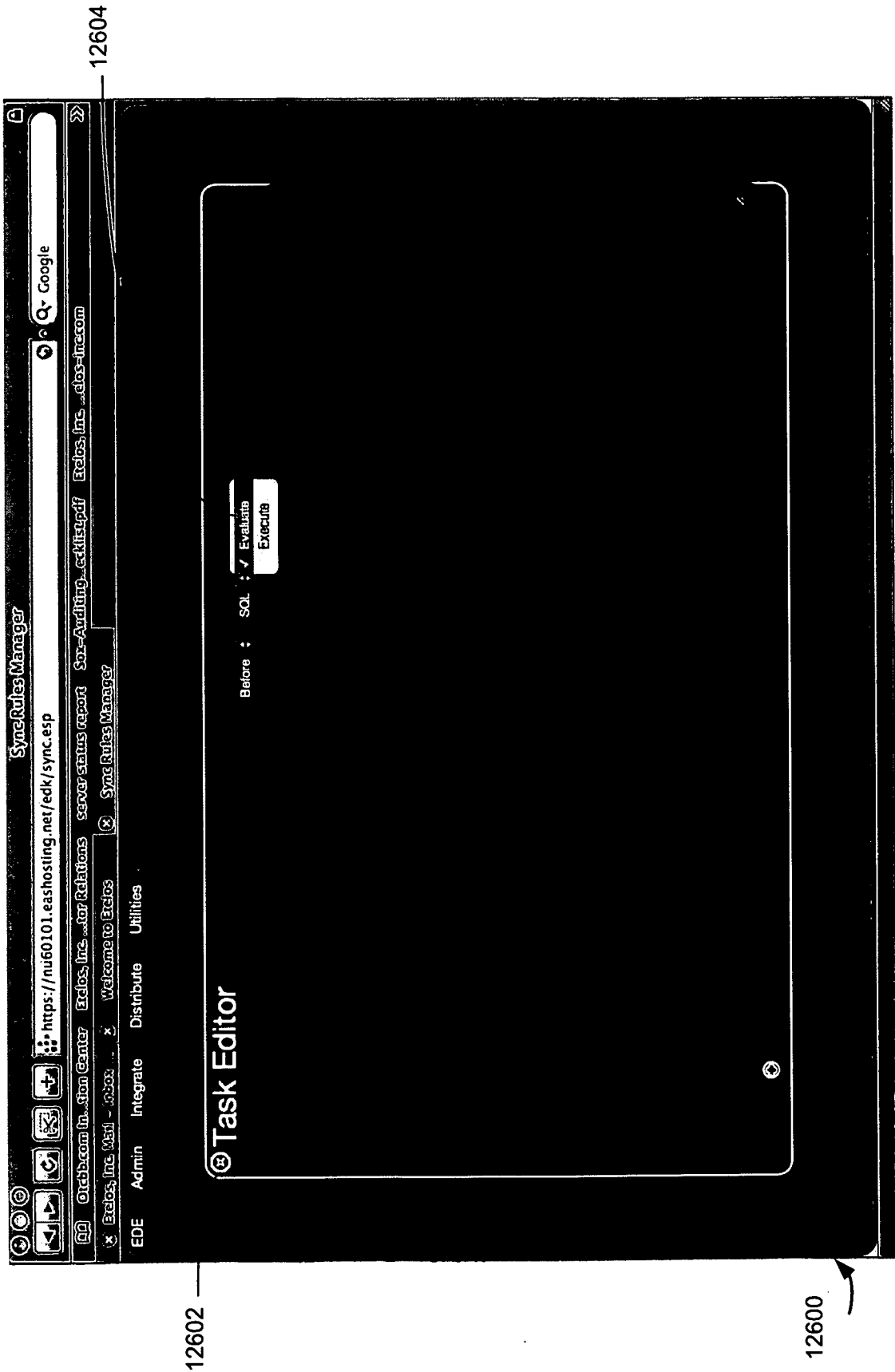


Figure 126

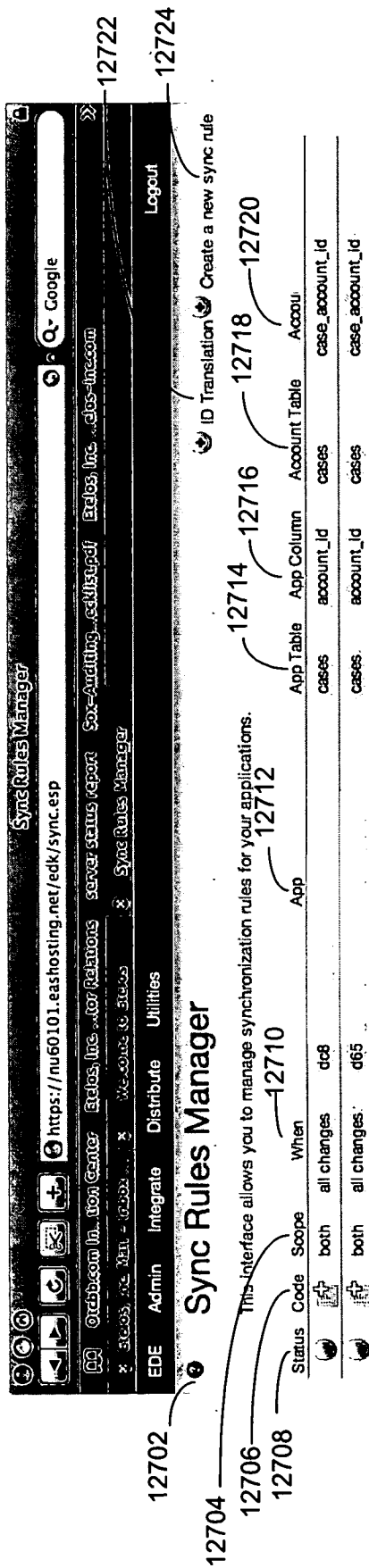
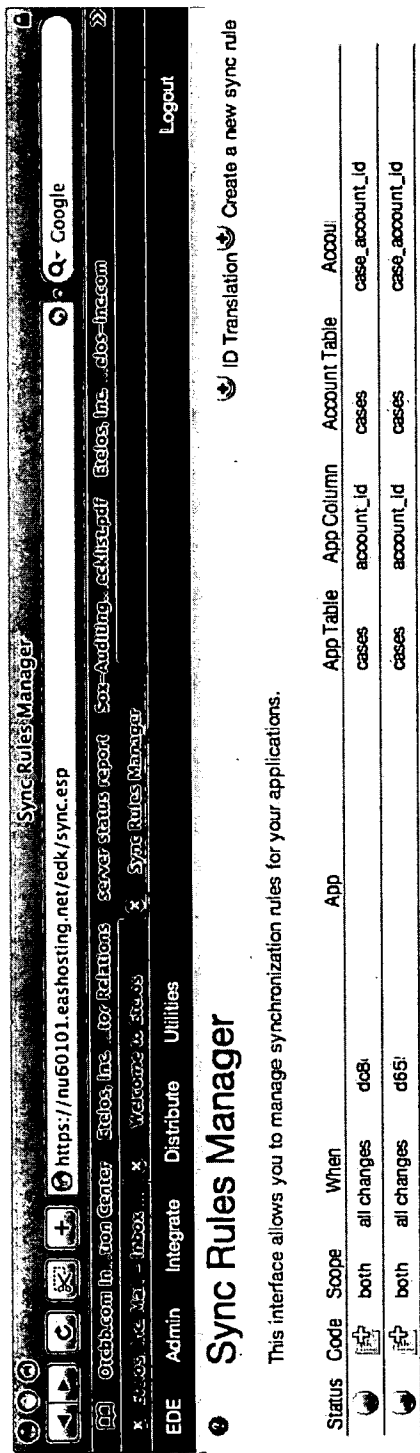


Figure 127



12800



Figure 128

Etelos, Inc.

Search:

Cart Account

Marketplace For Businesses For Developers Blogs About Support

Etelos Support

Support Home Getting Started Forums **Account** Etelos Ideas You are logged in as jstrid. (not you?)

My Account Info

- My Installations
- Billing History
- My Storefront
- My Storefront
- Store Reports
- My Apps
- My Accounts
- CHP Accounts
- Credit Card
- Profile

My Etelos Storefront

Use the following wizard to add products/services to the Etelos Marketplace. Need help? [Check out this video on posting your application.](#)

Store Listings/Link	Status	Type
Cert program test	Marketplace: Started Item: Not Ready	Application tools...
Demo app	Marketplace: Active_Online Item: Not Ready	Application tools...
Jers App Billing	Marketplace: Started Item: Not Ready	Billing Services tools...
Jers monthly mini	Marketplace: Declined Item: Not Ready	Mini-App tools...
test-hosting	Marketplace: Started Item: Not Ready	Hosting Services tools...
test web servke	Marketplace: Started Item: Not Ready	Web Services tools...
V35 Registration Website Manager	Marketplace: Started Item: Not Ready	Application tools...

page: 1

[+ Add Product](#)

Notification Email

Use this field to enter a comma separated list of emails that will receive notifications when someone purchases an item from your Etelos storefront. Note that you automatically receive notifications at the email address listed under your user profile so you do not need to include that email in the list.

[update](#)

Home | Partner | About Us | Contact | Press | Support | Logout | Staff | Jobs | Editor
[Terms of Use](#) | [Privacy Policy](#)
 All Rights Reserved

Done 1.240s YSlow 1.021s

12900

Figure 129

Welcome to Etelos

Search:

[Cart](#) [Account](#)

Marketplace **For Businesses** **For Developers** **Blogs** **About** **Support**

My Store

- Basic Info
- Choose App
- Licenses
- Product Info
- Marketing
- Features
- Demos
- Blogs
- FAQs
- Getting Started
- Support
- About Us
- Upgrades
- My Style
- Submit

Product/Service Basic Information
Update the basic information for your Marketplace listing.

Product Name: 13002 13004

Short Description: (192 character max)

Product Type: 13006

Item Status: (in the Marketplace):
 13008 13010

Store Item Details
(use this info for user feed and single sign on call)

Owner UUID: 632
Product ID: 1405
Product UUID: cbi

Product Type Definitions

Application - For distributing a packaged full application on the Etelos Marketplace. Etelos handles shopping cart, billing, licensing, provisioning and hosting.

Mini-App - For distributing a partial application. This app would be an update or add-on that is installed into an application that the user already owns.

Consulting Services - Sold in blocks of time. No app is installed. Etelos handles shopping cart and billing.

Web Services (no integrated billing) - This is a pass through type service. Customer clicks a link that redirects to a url you specify. There is also an option you to send the customer back to the Marketplace from your site so Etelos can handle billing.

Web Services (with integrated billing) - For applications and services hosted externally. Etelos handles the shopping cart, and billing, then passes the purchase information to a url you specify.

Application Access Key - For distributing a set of access keys that you setup. Etelos handles the shopping cart, billing and provisioning of the access keys. The user uses their key to access information at a url you specify.

<< Cancel Delete Next >>

13012

[Home](#) | [Partner](#) | [About Us](#) | [Contact](#) | [Press](#) | [Support](#) | [Logout](#) | [Staff](#) | [Jobs](#) | [Editor](#)
[Terms of Use](#) | [Privacy Policy](#)
All Rights Reserved.

Done 1.152s YSlow 1.147s

13000

Figure 130



Marketplace

Welcome to Etelos

Search:

[Cart](#) [Account](#)

Staging Your Application

[Blogs](#) [About](#) [Support](#)

Choose App File for "VBS Registration Website Manager"

Select the app file from the list below. This list contains all the apps you have posted to your support account from EDE (using the Build function.) You can change this app file at any time after placing your app on the store.

Application File	VBS_App (1864 WA=1)	13102
Application Size	Small-Mid Size Business App	13104
Estimated Install Time For this App	3 Minutes (If left blank the default estimated install time of 5 minutes will be used.)	13106

<< Cancel < Previous Next >

Home | Partner | About Us | Contact | Press | Support | Logoff | Staff | Jobs | Editor

[Terms of Use](#) | [Privacy Policy](#)

- All Rights Reserved.

My Store

Basic Info

Choose App

Licenses

Product Info

Marketing

Features

Demos

Blogs

FAQs

Getting Started

Support

About Us

Upgrades

My Style

Submit

Done

0.815s 0.816s 0.816s

13100

Figure 131

Welcome to Etelos

Search: [Cart](#) [Account](#)

Etelos

Marketplace For Businesses For Developers Blogs About Support

My Store

Basic Info
Choose App
Licenses
Product Info
Marketing
Features
Demos
Blogs
FAQs
Getting Started
Support
About Us
Upgrades
My Style
Submit

13202 Edit Licensing for "V8S Registration Website Manager"

You can customize the licensing options for your "Application" with the following tools.

What are Licenses?

1. Licenses are what users on the Etelos Store can purchase in order to use your "Application".
2. You can create multiple licenses for your application designed to accommodate different user types and marketing purposes.
3. Copies of Licenses are uniquely generated and associated with users to prevent abuse and keep track of distribution.
4. Licenses cannot be retroactively changed after purchases - the license sold is what the user has archived.
5. You can deactivate licenses if you decide to change your pricing model and this will remove those licenses from new distribution.

What are my Licensing Options for my "Application"?

- Per User (simple licensing) - An incremental license allowing you to charge a fee per user (ie. \$25 per user.) Use this license type if you have not set up custom licensing rules in your app. You can NOT limit the number of users added to the license.
- Per Usage (simple licensing) - A per usage based plan (ie. \$5 per project in one app.) Use this license type if you have not set up custom licensing rules in your app. You can NOT limit the usage count for this license.
- Per User(s) - An incremental license allowing you to charge a fee per block of users (ie. \$200 for 5 users.) Only use this license type if you have set up custom user rules in your app.
- Per Usage - A usage based plan. (ie. 3 projects at a time in one app, or, for hosting accounts, \$100 2GB database - 1GB transfer) Only use this license type if you have set up custom usage rules in your app.
- Per Account - A flat fee license per account using your Application. (ie. \$500 per Account)

Name	Description	Retail	Billing	Wholesale	Type	Count	Status
187 V8S Bundled Monthly Recurring	Monthly recurring billing bundled with Hosting Bundled hosting: Not Assigned	50	Monthly	40	Per Account	2	Off
180 test again	this is a test Bundled hosting: Not Assigned		Monthly		Per User	1	Off

[Add New License](#)

13206 Basics Info

13208 Install Type:

13210 Name:

13212 Billing Short Name:

13214 Description:

13216 Billing Description:

13218 License Version:

13220 Type:

13222 Qty/Count: Accounts

13224 Trial?:

13218 Part 2 - Billing/Hosting Information

13220 Billing Type:

13222 Bundled Hosting?

13224 Hosting Provider:

13224 Part 3 - Pricing and Resale Terms

13226 Retail Pricing:

13228 Resale Terms: [next](#)

13230 Wholesale Pricing: [next](#)

13230 Part 4 - Flags and License Status

13232 Source Code:

13234 Downloadable Source:

13236 App Sync Enabled:

13238 AOP Enabled:

13240 Allow Multiple Instances:

13242 External Product ID:

13244 Ad Server:

Status:

[Preview](#) [Cancel](#) [Save Changes](#) [Clone this License](#)

Link to use to purchase this license from your own website:
http://www.etelos.com/store/mystore/inoheader/purchase_now.aspx?storeid=1405&purchaseid=187
 (You can set up the style for this link on the Upgrades tab of this store entry.)

<< Cancel < Previous Next >>

Done 0.600s 2 Errors YSlow 2.448s

13200 →

Figure 132

13302

Welcome to Etelos

Etelos

Marketplace For Businesses For Developers Blogs About Support

Search: [Cart](#) [Account](#)

Setup Marketing Information

Product Preview & Store Categories

The following information is used to display your Application in the catalog and search results.

Banner Image: (recommended size 100px high by 350px wide at 72dpi) [\(More Info\)](#)



Product Image Thumbnail: (recommended size 225px high by 300px wide at 72dpi) [\(More Info\)](#)



Product Image: (upload a full size image that the user will see if they click on the Product thumbnail uploaded above.)

Desired Turnkey Rating:

How much setup is required for your Application? You may want to link getting started / consulting packages to this Application if appropriate.

Provided By Website Link (include http://)

13308

Categories:

- Office Solutions Staff / Employee Collaboration 13310
- Sales & Marketing Sales Force Automation
- Sales & Marketing Email & Newsletter Marketing
- Sales & Marketing Contact Management

Related Store Item: 13312

You can link to display with another store item.

Keywords: CRM Google Contact Management CRM developer Etelos CRM for Google Apps CRM for Google free trial

Enter keywords for searches. Note that product name and categories are also used in search results.

Hosting type: 13314

Hosted by Etelos 13316

This control what Hosting icon will show up next to your listing on the Marketplace.

Home | Partner | About Us | Contact | Press | Support | [Logout](#) | [Staff](#) | [Jobs](#) | [Editor](#)

[Terms of Use](#) | [Privacy Policy](#)

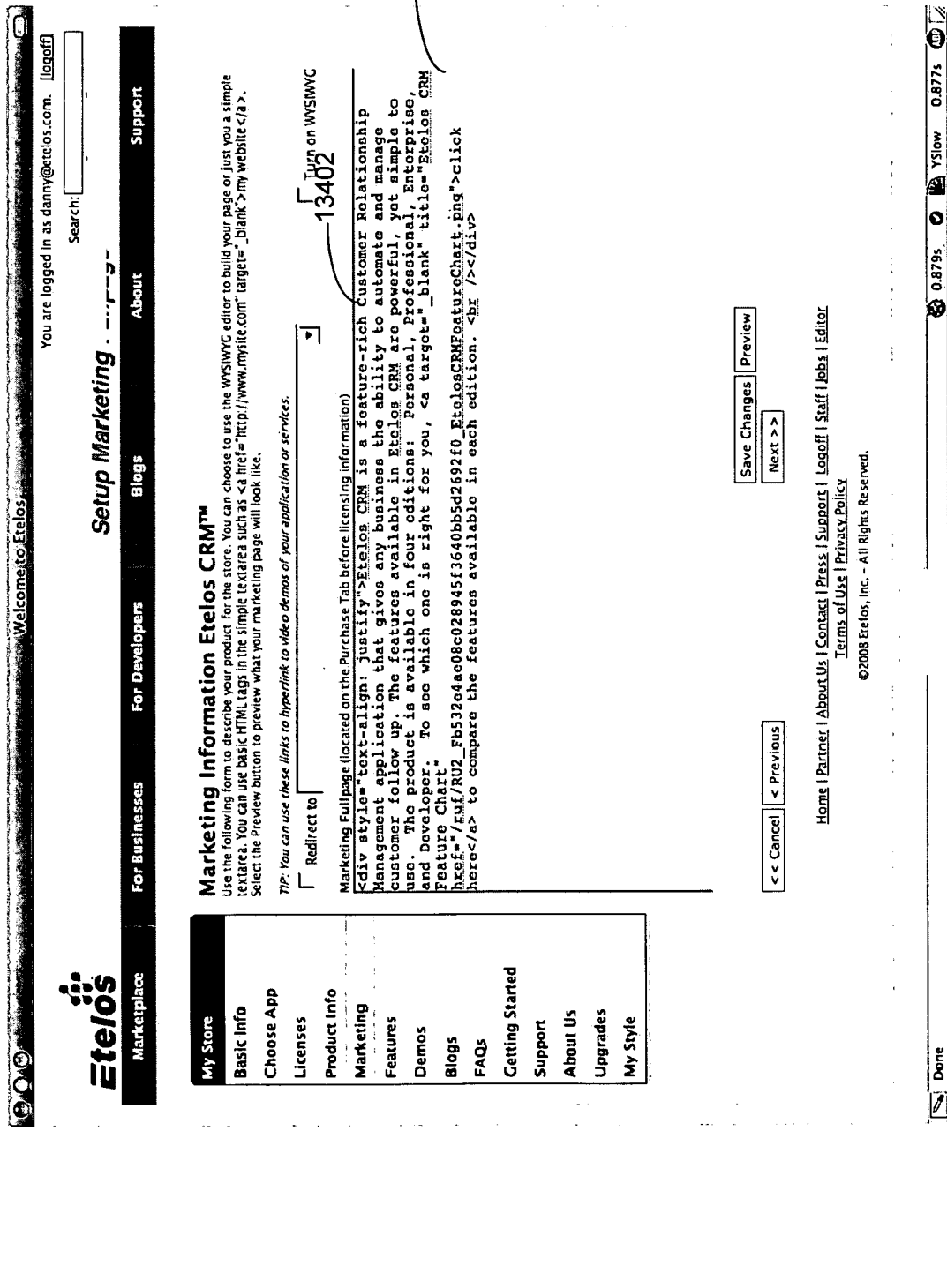
- All Rights Reserved.

13300

Done

1.3.21.5 0 YSlow 1.3.19.5

Figure 133



You are logged in as danny@etelos.com | Logout

Etelos

- Marketplace
- For Businesses
- For Developers
- Blogs
- About
- Support

Product Feature for Etelos CRM™

Use the following form to describe product features for the store. You can choose to use the WYSIWYG editor to build your page or just type a simple textarea. You can use basic HTML tags in the simple textarea such as
 or . Select the Preview button to preview what your feature page will look like.

Search:

Turn ON WYSIWYG

```
<p><br /></p>
```

<p>Etelos CRMtrade: is a feature-rich Customer Relationship Management application that gives any business, regardless of size, the ability to automate end manage customer follow-up more effectively. Etelos CRM is available in four editions: Personal, Professional, Enterprise, and Developer.
</p>

<p>After installation of any edition, each staff member can choose to use the application in whatever formats they prefer -- the Etelos portal, via their iPhone, or from their homepage on Google Apps, iGoogle, Microsoft Live, NetVibes, or Facebook. Adoption is easy when the app fits into their usual workflow, helping your sales staff quickly be more effective. </p>

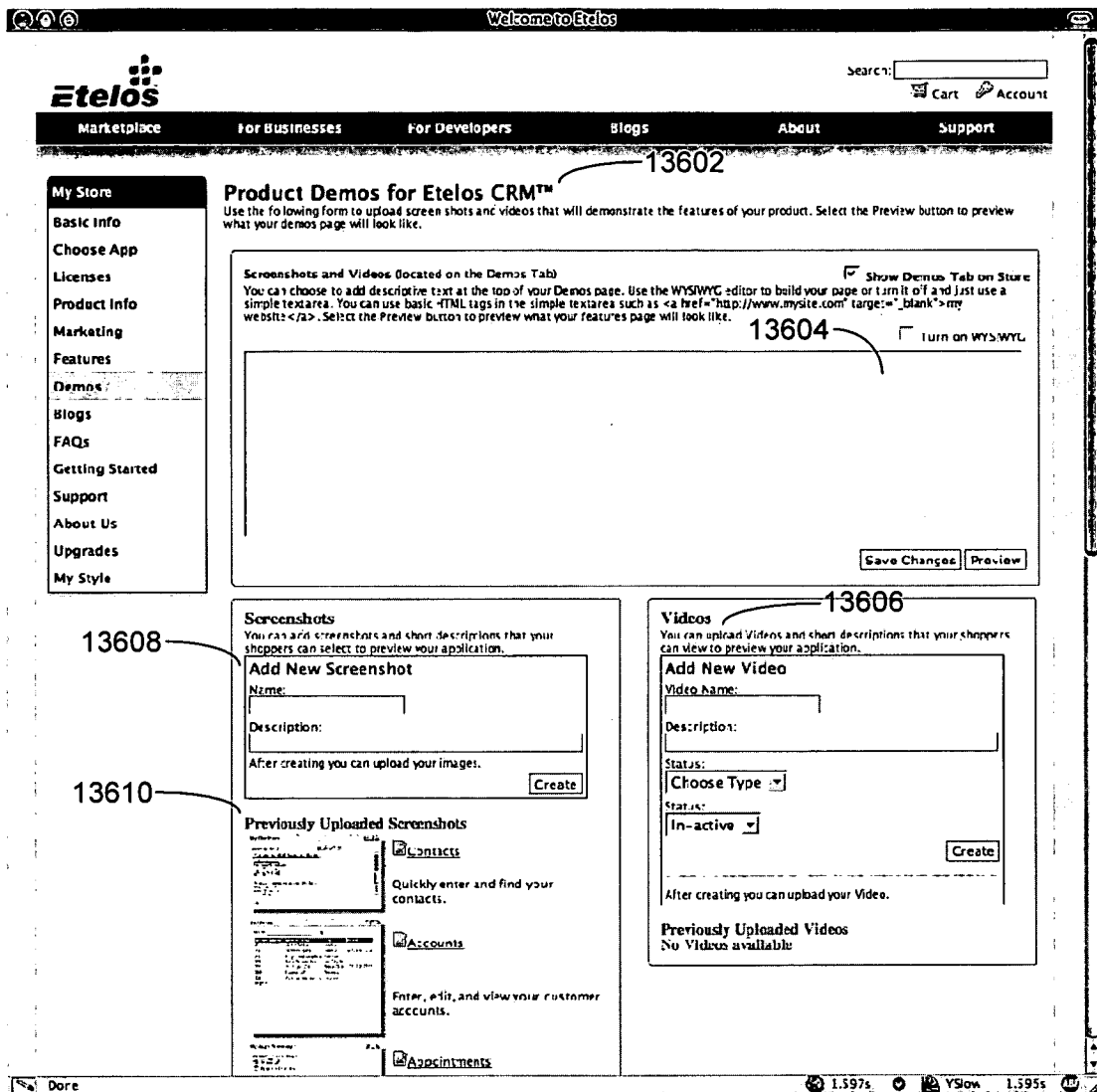
<p>Which Etelos CRM edition is right for your business?</p>

<p>The edition that's right for your business depends on your number of users, whether you want to use advanced features, or need access to the source code. The feature comparison chart and selected feature details listed below should provide all the information you need to pick the edition that's right for you.</p>

Save Changes Preview
Next: >>
<< Cancel Previous

Home | Partner | About Us | Contact | Press | Support | Logout | Staff Jobs | Editor Terms of Use Privacy Policy © 2008 Etelos, Inc. - All Rights Reserved.

Figure 135



13600 →

Figure 136



[Marketplace](#)
[For Businesses](#)
[For Developers](#)
[Blogs](#)
[About](#)
[Support](#)

You are logged in as [danny@etelos.com](#). [Logout](#)

Search: [Cart](#) [Account](#)

My Store

Basic Info

Choose App

Licenses

Product Info

Marketing

Features

Demos

Blogs

FAQs

Getting Started

Support

About Us

Upgrades

My Style

13702

13704

13706

13708

13710

13712

13714

Add a Blog Feed for "Etelos CRM™"

You can set up a blog feed for this store item. Just enter the url to your feed, set the feed status "On" and check the box for "Show Blog on My Store". If you don't currently have a blog for this product, you can go to create one with a blog tool purchased on the store. [Click Here](#) to search for blogging tools available on the store. (This link will open in a new window.)

Feed Link: (no http:// or feed://)

Feed Status: ☒ On ☐ (you can turn a feed off so it does not get updated with new posts)

☒ Show Blogs on My Store

(note: you must save changes before updating your feed)

[Home](#) | [Partner](#) | [About Us](#) | [Contact](#) | [Press](#) | [Support](#) | [Logout](#) | [Staff](#) | [Jobs](#) | [Editor](#)
[Terms of Use](#) | [Privacy Policy](#)

©2008 Etelos, Inc. - All Rights Reserved.

Figure 137

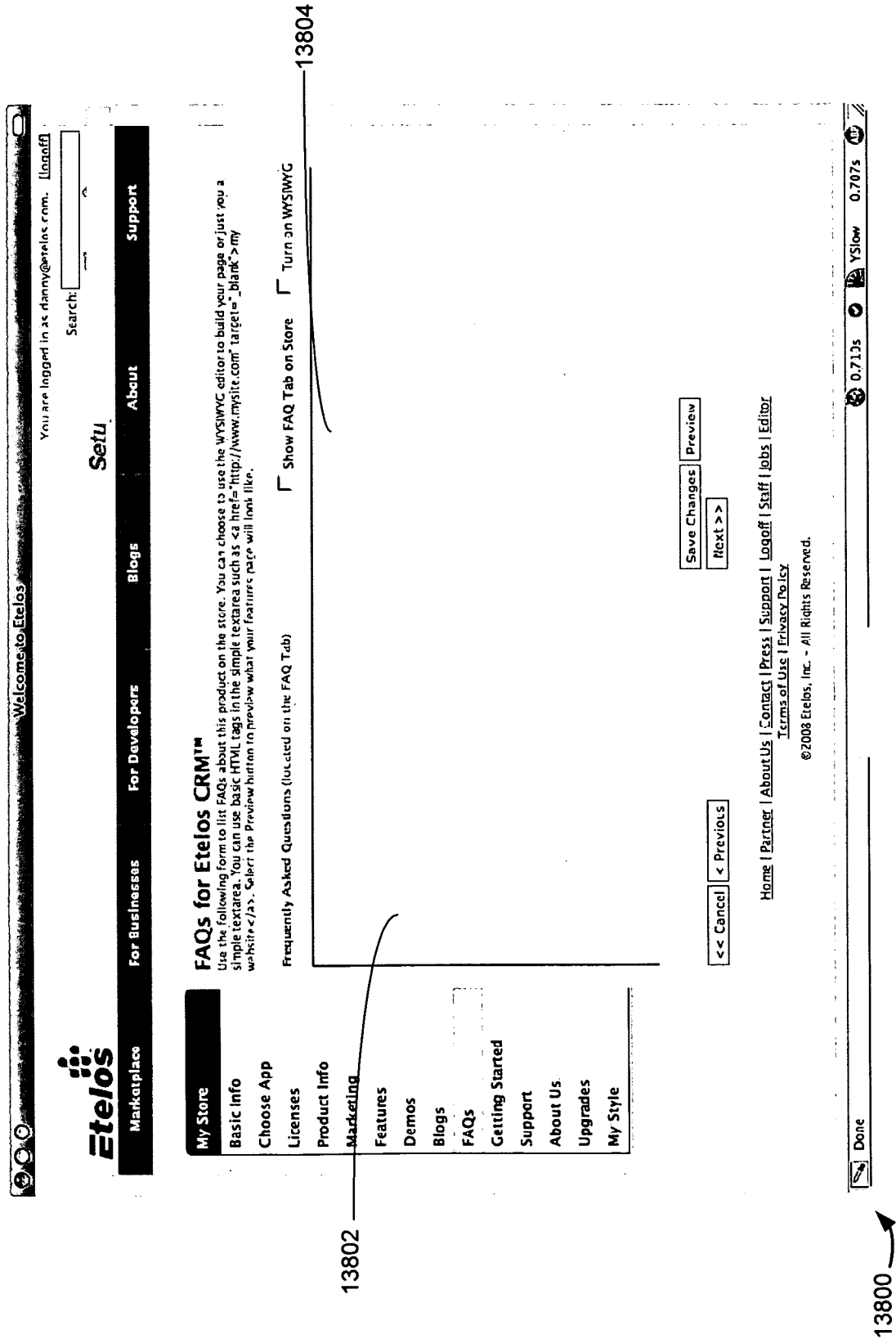


Figure 138

Welcome to Etelos

Etelos

Setup Getting Started Application

Search:

Cart Account

Marketplace For Businesses For Developers Blogs About Support

My Store

- Basic Info
- Choose App
- Licenses
- Product Info
- Marketing
- Features
- Demos
- Blogs
- FAQs
- Getting Started**
- Support
- About Us
- Upgrades
- My Style

Getting Started Information Etelos CRM™

Use the following form to edit the information that your contacts receive when "getting started" with your Application. The email will be sent upon completion of installation of the license (including trial). If you want to include the url to the client's app in your getting started page use this syntax: `http://<#|hosting.domain|>`

NOTE: There is a support page in addition to the getting started page

☒ Turn on WYSIWYG

Getting Started Page:

B I U ABC [Font family] [Font size]

Welcome

It's easy to get started. Click the link below to activate your account. Your username and password are the same as those for your Etelos Marketplace(tm) account.

[Click here to initialize your app and get started.](#)

If the above link does not work, you can try the link below.

`http://<#|hosting.domain|>/aesx.espx?eao=<#|contact.id|>&version=portal`

Path:

Save Changes Preview

Getting Started Email: (Text of email sent to user when there application is installed)

B I U ABC [Font family] [Font size]

Welcome

It's easy to get started. Click the link below to activate your account. Your username and password are the same as those for your Etelos Marketplace(tm) account.

[Click here to initialize your app and get started.](#)

If the above link does not work, you can try the link below.

`http://<#|hosting.domain|>/aesx.espx?eao=<#|contact.id|>&version=portal`

Path:

Save Changes Preview

Getting Started Page - Trial Only Content

The following content will only show up if the user has purchase a trial license.

B I U ABC [Font family] [Font size]

Welcome

It's easy to get started. Click the link below to activate your account. Your username and password are the same as those for your Etelos Marketplace(tm) account.

[Click here to initialize your app and get started.](#)

If the above link does not work, you can try the link below.

`http://<#|hosting.domain|>/aesx.espx?eao=<#|contact.id|>&version=portal`

Path:

Save Changes Preview

<< Cancel < Previous Next >>

Home | Partner | About Us | Contact | Press | Support | Logout | Staff | Jobs | Editor
Terms of Use | Privacy Policy

- All Rights Reserved.

Transferring data from www.etelos.com...

0.370s YSlow 1.253s

13900

Figure 139

[illegible]

Figure 140

Welcome to Etelos

Etelos 14102 Search: [Cart](#) [Account](#)

[Marketplace](#) [For Businesses](#) [For Developers](#) [Blogs](#) [About](#) [Support](#)

14104

My Store

- Basic Info
- Choose App
- Licenses
- Product Info
- Marketing
- Features
- Demos
- Blogs
- FAQs
- Getting Started
- Support
- About Us
- Upgrades
- My Style

About Us

You can set up an About Us page for this product on the store. You can choose to use the WYSIWYG editor to build your page or just you a simple textarea. You can use basic HTML tags in the simple textarea such as: `<a href="http://www.mysite.com" target="_blank">my website</a>`. Select the Preview button to preview what your features page will look like.

☐ Show About Us Tab on Store

Company Name 14106

Company Website Link (include http://) 14108

Contact Email (also used as the 'reply to' address for emails sent from this store listing) 14110

About Us Fullpage ☐ Turn on WYSIWYG 14112

[Home](#) | [Partner](#) | [About Us](#) | [Contact](#) | [Press](#) | [Support](#) | [Logoff](#) | [Staff](#) | [Jobs](#) | [Editor](#)
[Terms of Use](#) | [Privacy Policy](#)
- All Rights Reserved.

Done 0.250s 0.849s

14100

Figure 141

Welcome to Etelos

14202 Search: [Cart](#) [Account](#)

Etelos

Marketplace For Businesses For Developers Blogs About Support

14204

My Store

- Basic Info
- Choose App
- Licenses
- Product Info
- Marketing
- Features
- Demos
- Blogs
- FAQs
- Getting Started
- Support
- About Us
- Upgrades
- My Style

Product Upgrades

Use this page to set up your Product Upgrades Pages. On this page you will link licenses for this product with an upgrade license. This upgrade license must be associated with another store entry that you set up. The upgrade store entry will be linked to a mini-app which will be installed when the user chooses to upgrade to the new license. Your mini app can be as simple as one text file, or it could be a complex conglomerate of pages and tables.

You will save two upgrade page options. The first is the standard upgrade page that a user will see by clicking on the "Upgrade" tab on your store listing. You can choose to hide/unhide this tab using the check box below. The Second upgrade option is to link straight to the Upgrade Cart with a link in your app or from another website. You can setup the look and feel of your Upgrade Cart on the My Style page.

14206 ☒ Show Upgrades Tab on Store

Upgrades

Add New Upgrade

Name: 14208

Description: 14210

License to Upgrade To: 14212 ☒

Status: 14214 ☒ Active 14216

Create your upgrade license using the form above, then add licenses that can be upgraded using the drop down below (note: the drop down will only be visible after you create the upgrade record.)

Current List of Licenses that can be upgraded

No Licenses Added

Current upgrades for this item 14218

Etelos CRM Personal to Professional [\[edit\]](#)

Upgrade from Etelos CRM Personal Edition to Professional Edition

Upgrade to license: Upgrade to Professional

Upgrade link you would use from your app:

➔ http://www.etelos.com/store/mystore/noheader/upgrade_now.aspx?domain=1#&system.domain=1#&upgraded=44

Etelos CRM Pro to Personal [\[edit\]](#)

Downgrade to Personal Edition

Upgrade to license: Downgrade to Personal Edition

Upgrade link you would use from your app:

➔ http://www.etelos.com/store/mystore/noheader/upgrade_now.aspx?domain=1#&system.domain=1#&upgraded=45

Etelos CRM Upgrade to Enterprise Edition [\[edit\]](#)

Upgrade to Enterprise Edition

Upgrade to license: Upgrade to Enterprise Edition

Upgrade link you would use from your app:

➔ http://www.etelos.com/store/mystore/noheader/upgrade_now.aspx?domain=1#&system.domain=1#&upgraded=46

[Home](#) | [Partner](#) | [About Us](#) | [Contact](#) | [Press](#) | [Support](#) | [Logoff](#) | [Staff](#) | [Jobs](#) | [Editor](#)

[Terms of Use](#) | [Privacy Policy](#)

All Rights Reserved.

Done 1.540s 1.54s

14200 →

Figure 142

Welcome to Etelos

Search:

[Cart](#) [Account](#)

Marketplace **For Businesses** **For Developers** **Blogs** **About** **Support**

My Store

Basic Info
Choose App
Licenses
Product Info
Marketing
Features
Demos
Blogs
FAQs
Getting Started
Support
About Us
Upgrades
My Style

My Store Style

Use this page to set up the look and feel of your syndicated store pages. You can link to this syndicated store from another website for upgrades (see the upgrades tab) and new purchases using the link listed at the bottom of the licensing page (choose a license to see the link for that license.)

Use this area to setup the look and feel of your store page.

Page Back Ground Color: Choose Color Body Font Color: #000000 Choose Color

Page Headings Font Color: #000000 Choose Color Font:

☐ Use this style for upgrades only (If you check this box the default Etelos Marketplace style will be applied.)

Page Header
You can use basic HTML tags in the simple textarea such as my website. Select the Preview button to preview what your features page will look like.

```
<div class="content" style="background-color: #FFFFFF;">
<p></p>
<div style="width: 100%; height: 30px; background-image:
url(http://www.etelos.com/images/img/orangealice.jpg); background-repeat: repeat-x;">
</div>
</div>
```

Page Footer

```
<div style="border-width: 1px 0px 0px 0px; border-style: solid; border-color:
#996600;">
</div>
```

Save Changes Preview Cart

<< Cancel < Previous Next >>

[Home](#) | [Partner](#) | [About Us](#) | [Contact](#) | [Press](#) | [Support](#) | [Logoff](#) | [Staff](#) | [Jobs](#) | [Editor](#)
[Terms of Use](#) | [Privacy Policy](#)
All Rights Reserved.

http://www.etelos.com/article.aspx?show=11444 0.912s YSlow 0.91s

14300

Figure 143

Welcome to Etelos

Search:

Cart Account

Marketplace For Businesses For Developers Blogs About Support

My Store

- Basic Info
- Product Info
- Marketing
- Pricing Grid**
- Features
- Demos
- Blogs
- FAQs
- Support
- About Us
- My Cart
- My Style
- Submit

Pricing Grid and Purchase Link Setup 14402

Set up the items that will appear in the pricing grid on the Marketing page for this product. When the user clicks the "Purchase Now" link they will be redirected to the url you specify.

Add a New Pricing Item 14404

Name:

Description:

Price (example: \$19.95 per month) 14406

Redirect URL: (This is where the client should redirect to when they click the Buy Now link.) 14408

Link Text: (text the user will see on the Marketing page)
 Buy it Now 14410

Sort Order: 14412

Status: ☒ Active 14414

14416

List of Current Pricing

No Results Found

<< Cancel < Previous

[Home](#) | [Partner](#) | [About Us](#) | [Contact](#) | [Press](#) | [Support](#) | [Logoff](#) | [Staff](#) | [Jobs](#) | [Editor](#)
[Terms of Use](#) | [Privacy Policy](#)
All Rights Reserved.

Done 0.874s YSlow 0.871s

14400

Figure 144

Welcome to Etelos

Search:

Cart Account

Marketplace For Businesses For Developers Blogs About Support

My Store

- Basic Info
- Licenses
- Product Info
- Marketing
- Features **14504**
- Demos
- Blogs
- FAQs
- Getting Started
- Support
- About Us
- Web Service Post
- My Style
- Submit

Web Services "Post Data" Setup

Use this page to set up the url that purchase information will be posted to at checkout.

Postback URL
After the users card is processed, what url should we post their purchase information to?
 (Include http:// or https:// in your url)

Choose what format you want to receive the posted data:
☒ JSON ☐ XML

Sample of the code you will need to receive at the url you specified above.

JSON Sample:

```
{
  "store_name": "etelos", "listing_id": "2448", "customer_id": "03573b3ffc1d66a92767dc67e4770e29", "first_name": "Jerrrod", "last_name": "inc.", "email": "jerrrod.srid@etelos-inc.com", "phone": "425 999 3888", "address1": "230 Mill Ave S", "address2": "Suite 400", "city": "Kenton", "state": "WA", "zip": "98055", "country": "US", "company_website": "", "order_id": "b005uccdbb39b3uccdb3b1sc7aad", "App Install Main Item": "unit_price": "12.00", "item_id": "6a2b-y788-76y8-21c9-876g", "product_id": "393", "quantity": "2", "item_description": "My App", "Install Sub Item": "unit_price": "50.00"}]
```

XML Sample:

```
<?xml version="1.0"?>
<message>
  <store_name>#etelos</store_name>
  <listing_id>2448</listing_id>
  <customer_id>03573b3ffc1d66a92767dc67e4770e29</customer_id>
  <first_name>Jerrrod</first_name>
  <last_name>St id</last_name>
  <company>Etelos inc.</company>
  <email>jerrrod.srid@etelos-inc.com</email>
  <phone>425 999 3888</phone>
  <address1>230 Mill Ave S</address1>
  <address2>Suite 400</address2>
  <city>Kenton</city>
  <state>WA</state>
  <zip>98055</zip>
  <country>US</country>
  <company_website></company_website>
  <order_id>b005uccdbb39b3uccdb3b1sc7aad</order_id>
  <App Install Main Item>unit_price:12.00,item_id:6a2b-y788-76y8-21c9-876g,product_id:393,quantity:2,item_description:My App,Install Sub Item:unit_price:50.00</App Install Main Item>
</message>
```

What your page should return
Your page should return the following (just print it on your page):

- <status>success</status>
- <status>failed</status> if there is an error on your end

There are three variables that you can pass back in order to print them on the users getting started page. This would normally be some unique information that the user will need in order to continue:

- <username>username</username>
- <password>password</password>
- <url>http://www.yoururl.com</url>

You can include whatever information you want in these variable. As an example instead of a password you might include an activation code.

To include this value on your getting started page include the following code on that page:
<#subscriptions_username#> <#subscriptions_password#> <#subscriptions_url#>

Add-on items
Once a user has purchased a subscription under this Marketplace listing, you can send them back to the marketplace to add on more items. To do this just add a link from your site to http://www.etelos.com/store/nostyle/web_listing.aspx. You need to include the users customer_id and the listing_id either as url variables or post variable. (These will be sent to you using the example code above, after the original purchase.)

Add-On Example:
http://www.etelos.com/store/nostyle/web_listing.aspx?customer_id=86290e68f8c6bb75304fc13953347ak&listing_id=2448

This will land the user on the listing page for this marketplace item where they can add items to their cart.

Cancelling a Subscription
You can also cancel a recurring subscription for one of your users with a url call. To do this call https://www.etelos.com/store/nostyle/web_cancellation.asp. You need to include the users customer_id, the item_id you are canceling, your username and your password (used to authenticate that you are allowed to cancel the item). You can pass these either as url variables or post variable.

Cancellation Example:
https://www.etelos.com/store/nostyle/web_cancellation.asp?customer_id=86290e68f8c6bb75304fc13953347ak&item_id=597862740ef896b884826edf9185e671&owneren=yourusername&ownerpw=yourpassword

The page will respond with either "success" or "failed"

<< Cancel Previous Next >>

Home | Partner | About Us | Contact | Press | Support | Logout | Staff | Jobs | Editor
Terms of Use | Privacy Policy
All Rights Reserved.

<http://www.etelos.com/article.aspx?show=12237> 3.833s YSlow 0.829s

14500

Figure 145

The screenshot shows the Etelos Support page. The main header includes the Etelos logo, a search bar, and navigation links: Marketplace, For Businesses, For Developers, Blogs, About, and Support. A large black banner with the text "Etelos Support" is prominently displayed. Below this, a navigation bar contains links for Support Home, Getting Started, Forums, Account, and Etelos Ideas. A message indicates the user is logged in as "jstrie" (not you).

The "My Account Info" sidebar on the left lists various account management options. The main content area, titled "My Etelos Storefront", provides instructions on adding products and services. It features a table of "Store Listings/Link" with columns for the link, status, type, and a "tools..." link. Annotations 14602, 14604, 14606, and 14608 point to specific rows in this table.

Store Listings/Link	Status	Type	tools...
Cert program test	Marketplace: Started Item: Not Ready	Application	tools...
Demo app	Marketplace: Active, Online Item: Not Ready	Application Access Key	tools...
Iers App Billing	Marketplace: Started Item: Not Ready	Billing Services	tools...
Iers monthly mini	Marketplace: Declined Item: Not Ready	Mini-App	tools...
Iers Hosting	Marketplace: Started Item: Not Ready	Hosting Services	tools...
Iers web service	Marketplace: Started Item: Not Ready	Web Services	tools...
VBS Registration Website Manager	Marketplace: Started Item: Not Ready	Application	tools...

Below the table, there is a section for "Add Product" and a "Notification Email" section with a text input field and an "update" button. A "Tools" dropdown menu is visible on the right, showing options like "Preview", "User Count", "user count (csv)", and "Upgrade".

The footer contains links for Home, Partner, About Us, Contact, Press, Support, Logout, Staff, Jobs, and Editor, along with Terms of Use and Privacy Policy. A copyright notice "All Rights Reserved" is also present.

14600 →

Figure 146

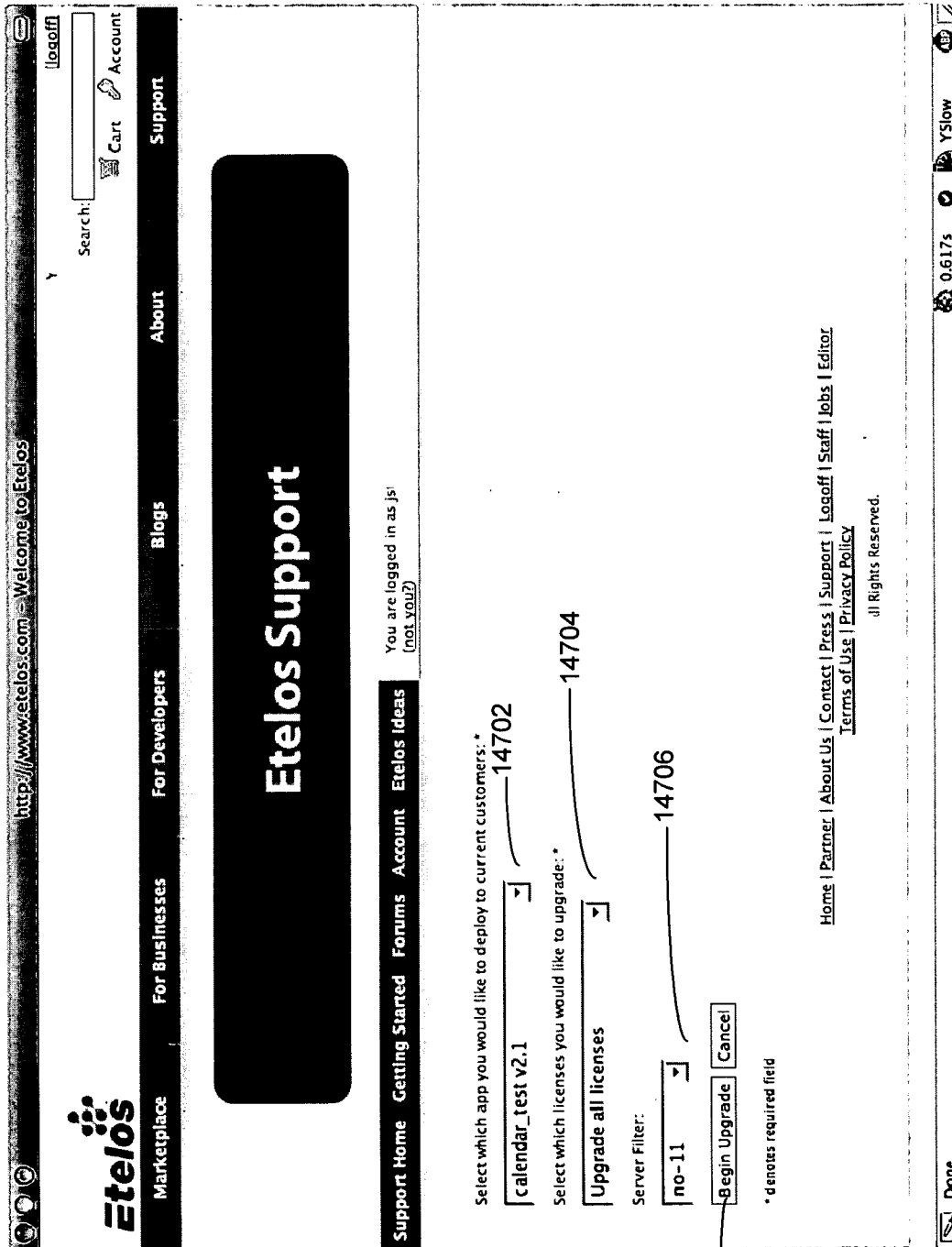


Figure 147

14708

14700

The screenshot shows the Etelos Support page. At the top, there is a navigation bar with links: Marketplace, For Businesses, For Developers, Blogs, About, and Support. A large black banner with white text reads "Etelos Support". Below this, a sub-navigation bar includes: Support Home, Getting Started, Forums, Account, and Etelos Ideas. A user is logged in as EricB. (not you?).

On the left, a sidebar menu lists: My Account Info, My Installations, Billing History, My Storefront, My Storefront, My Apps, My Accounts, CHP Accounts, Credit Card, and Profile.

The main content area has a section for "My Account Info" with fields for Account Name, Domain, UID, Username, Password, Internal IP, and External IP. A bracket labeled "14802" groups the Domain, UID, Username, Password, and Internal IP fields. Below these fields is a note: "Internal and External IPs are required if you want add new domains to this account" and a "Create" button.

Below the account info is a section titled "My Developer Accounts". It contains a table of accounts with columns for Name, Default Domain, Status, and Alternate Domains. A bracket labeled "14804" points to the "Normal" deployment type. A bracket labeled "14806" points to the "Broadcast (strict id) deployment" note. A bracket labeled "14808" points to the "Add Domain" button. A bracket labeled "14810" points to the "Add Domain" button. The table lists several accounts, including crms1340, crms2685, crms3427, training112, and trial145.

At the bottom, there is a footer with links: Home, Partner, About Us, Contact, Press, Support, Logout, Staff, Jobs, Editor, Terms of Use, and Privacy Policy. It also states "All Rights Reserved."

14800 →

Figure 148

Welcome to Etelos

You are logged in as [logoff]

Search: [Cart](#) [Account](#)

[Marketplace](#) [For Businesses](#) [For Developers](#) [Blogs](#) [About](#) [Support](#)

Etelos Support

[Support Home](#) [Getting Started](#) [Forums](#) [Account](#) [Etelos Ideas](#) [\(not you?\)](#)

You are logged in as EricB.

Create a Sub Domain For crms1340.etelosservice.com

Enter your new domain in the box below and then click submit

Do main: [14904](#)

[Home](#) | [Partner](#) | [About Us](#) | [Contact](#) | [Press](#) | [Support](#) | [Logoff](#) | [Staff](#) | [Jobs](#) | [Editor](#)

[Terms of Use](#) | [Privacy Policy](#)

All Rights Reserved.

Done 0.770s 0.755s YSlow

Figure 149

Store Product List [reports list]				
Click on the store listing you want to view a summary report for.				
Product Name	Store Owner	Company	Status	CSV
ABC Blogset	Danny Kolke	Etelos	Active_Online / Ready	[export to csv]
ADAM Promotional Version	Da	Etelos	Active_Online / Not Ready	[export to csv]
AdSense Dev	Ml	Etelos Systems, Inc.	Active_Online / Not Ready	[export to csv]
Africa Test Product	Da	Etelos	Active_Online / Not Ready	[export to csv]
Basic Consulting	Da	Etelos	Active_Online / Not Ready	[export to csv]
BIFX CRM	Biz	BizExchange	Active_Online / Not Ready	[export to csv]
BizCamp Etelos CRM for Google Apps	Da	Et	Active_Online / Not Ready	[export to csv]
Bullseye Incoming Fax Manager	Da	Etelos	Active_Online / Not Ready	[export to csv]
Bullseye Outgoing Fax Service	Da	Etelos	Active_Online / Not Ready	[export to csv]
Catalog Manager	Da	Etelos	Active_Online / Not Ready	[export to csv]
Community 101	Da	Etelos	Active_Online / Ready	[export to csv]
CRM/SEM/CMS by Rivalda	Jov	Emrecore	Active_Online / Not Ready	[export to csv]
CRM Test Listing ** For Testing Only**	Da	Et	Active_Online / Not Ready	[export to csv]
Custom Dev	Da	Etelos	Active_Online / Not Ready	[export to csv]
Demo app	Jer	Etelos Inc.	Active_Online / Not Ready	[export to csv]
EAS	D	Etelos	Active_Online / Not Ready	[export to csv]
EASX	Da	Etelos	Active_Online / Not Ready	[export to csv]
E-commerce Application	Da	Etelos	Active_Online / Not Ready	[export to csv]
E-commerce Payment Portal SM-Pro add on	Da	Etelos	Active_Online / Not Ready	[export to csv]
EDE 5.0	Da	Etelos	Active_Online / Not Ready	[export to csv]
EDE Lite	Da	Etelos	Active_Online / Not Ready	[export to csv]
Empressr	Bh	Fusebox	Active_Online / Ready	[export to csv]
Etelos Ad Serve™	Da	Et	Active_Online / Ready	[export to csv]
Etelos Application Hosting	Et	Etelos App Hosting	Active_Online / Not Ready	[export to csv]
Etelos Apps on a Plane™ Beta Signup	Danny Kolke	Etelos	Active_Online / Ready	[export to csv]
Etelos Conference App for iPhone	Da	Etelos	Active_Online / Not Ready	[export to csv]
Etelos CRM™ for Google Apps	Da	Etelos	Active_Online / Ready	[export to csv]
Etelos CRM for Google Apps (V3)	Da	Etelos	Active_Online / Not Ready	[export to csv]
Etelos CRM for Google Apps v6 SYNC	Da	Etelos	Active_Online / Not Ready	[export to csv]
Etelos CRM™ for iPhone	Da	Etelos	Active_Online / Ready	[export to csv]
Etelos CRM™ for Microsoft Live	Da	Etelos	Active_Online / Ready	[export to csv]
Etelos CRM for Microsoft Live (v5)	Da	Etelos	Active_Online / Not Ready	[export to csv]
Etelos CRM™ for Netvibes	Da	Etelos	Active_Online / Ready	[export to csv]

Figure 150

15000

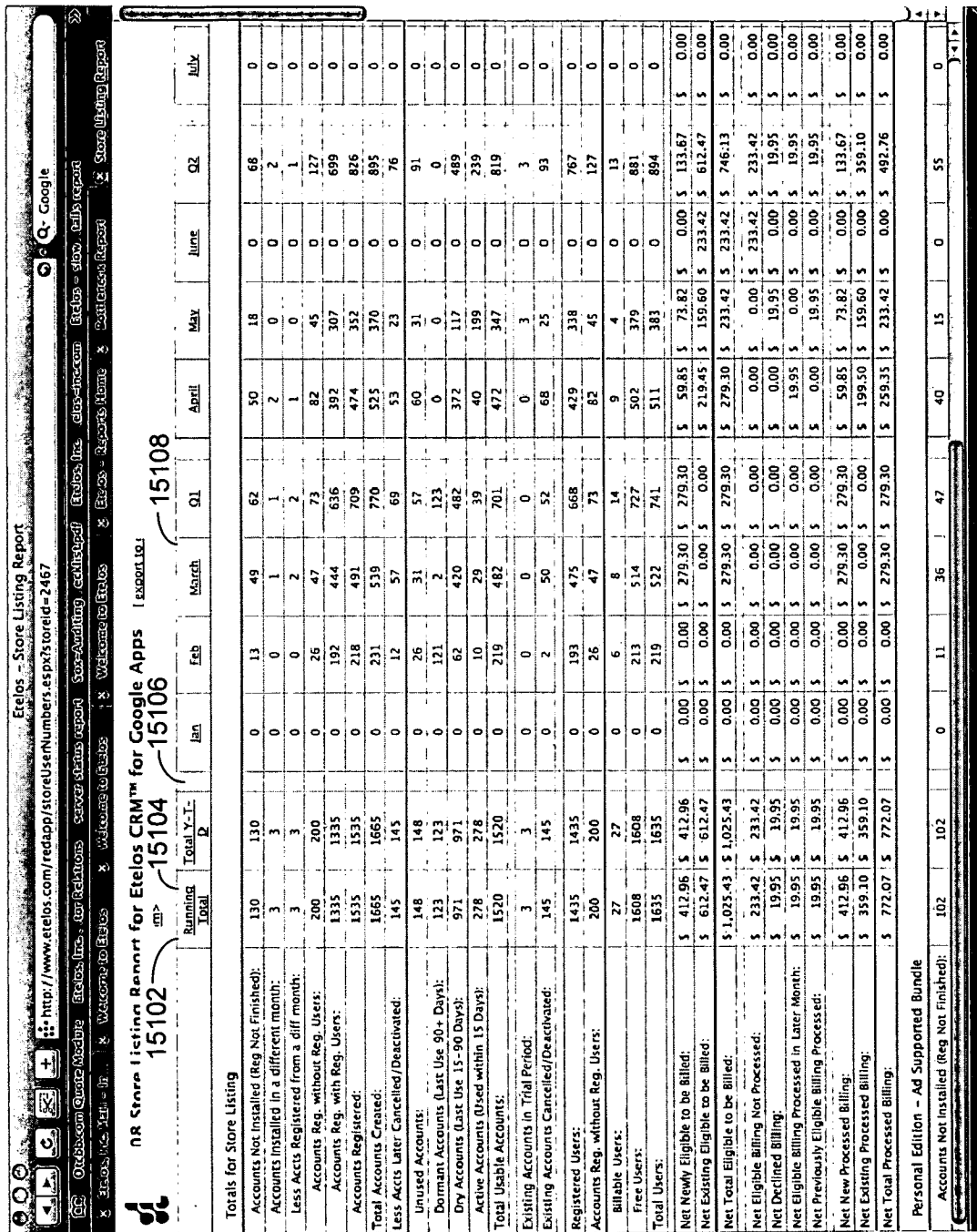


Figure 151

Etelos - Store Listing Report

<http://www.etelos.com/redapp/storeUserNumbers.aspx?storeId=2467>

Google

Etalos, Inc. - User Relations Server Status Report Size-Auditing etalapp Etalos, Inc. - dms-incom Etalos = slow bills report

Welcome to Etalos Welcome to Etalos Reports Home Softwarch Report Store Listing Report

Store Listing Report for Etalos CRM™ for Google Apps

[export to csv]

	Running Total	Total Y-T-D	Jan	Feb	March	Q1	April	May	June	Q2	July
Totals for Store Listing											
Accounts Not Installed (Reg Not Finished):	130	130	0	13	49	62	50	18	0	68	0
Accounts Installed In a different month:	3	3	0	0	1	1	2	0	0	2	0
Less Accts Registered from a diff month:	3	3	0	0	2	2	0	0	0	1	0
Accounts Reg. without Reg. Users:	200	200	0	26	47	73	82	45	0	127	0
Accounts Reg. with Reg. Users:	1335	1335	0	192	444	636	392	307	0	699	0
Accounts Registered:	1535	1535	0	218	491	709	474	352	0	826	0
Total Accounts Created:	1665	1665	0	231	539	770	525	370	0	895	0
Less Accts Later Cancelled/Deactivated:	145	145	0	12	57	69	53	23	0	76	0
Unused Accounts:	148	148	0	26	31	57	60	31	0	91	0
Dormant Accounts (Last Use 90+ Days):	123	123	0	121	2	123	0	0	0	0	0
Dry Accounts (Last Use 15-90 Days):	971	971	0	62	420	482	372	117	0	489	0
Active Accounts (Used within 15 Days):	278	278	0	10	29	39	40	199	0	239	0
Total Usable Accounts:	1520	1520	0	219	482	701	472	347	0	819	0
Existing Accounts In Trial Period:	3	3	0	0	0	0	0	3	0	3	0
Existing Accounts Cancelled/Deactivated:	145	145	0	2	50	52	68	25	0	93	0
Registered Users:	1435	1435	0	193	475	668	429	338	0	767	0
Accounts Reg. without Reg. Users:	200	200	0	26	47	73	82	45	0	127	0
Billable Users:	27	27	0	6	8	14	9	4	0	13	0
Free Users:	1608	1608	0	213	514	727	502	379	0	881	0
Total Users:	1635	1635	0	219	522	741	511	383	0	894	0
Net Newly Eligible to be Billed:	\$ 412.96	\$ 412.96	\$ 0.00	\$ 0.00	\$ 279.30	\$ 279.30	\$ 59.85	\$ 73.82	\$ 0.00	\$ 133.67	\$ 0
Net Existing Eligible to be Billed:	\$ 612.47	\$ 612.47	\$ 0.00	\$ 0.00	\$ 0.00	\$ 0.00	\$ 219.45	\$ 159.60	\$ 233.42	\$ 612.47	\$ 0
Net Total Eligible to be Billed:	\$ 1,025.43	\$ 1,025.43	\$ 0.00	\$ 0.00	\$ 279.30	\$ 279.30	\$ 279.30	\$ 233.42	\$ 233.42	\$ 746.13	\$ 0
Net Eligible Billing Not Processed:	\$ 233.42	\$ 233.42	\$ 0.00	\$ 0.00	\$ 0.00	\$ 0.00	\$ 0.00	\$ 0.00	\$ 233.42	\$ 233.42	\$ 0
Net Declined Billing:	\$ 19.95	\$ 19.95	\$ 0.00	\$ 0.00	\$ 0.00	\$ 0.00	\$ 0.00	\$ 19.95	\$ 0.00	\$ 19.95	\$ 0
Net Eligible Billing Processed in Later Month:	\$ 19.95	\$ 19.95	\$ 0.00	\$ 0.00	\$ 0.00	\$ 0.00	\$ 19.95	\$ 0.00	\$ 0.00	\$ 19.95	\$ 0
Net Previously Eligible Billing Processed:	\$ 19.95	\$ 19.95	\$ 0.00	\$ 0.00	\$ 0.00	\$ 0.00	\$ 0.00	\$ 19.95	\$ 0.00	\$ 19.95	\$ 0
Net New Processed Billing:	\$ 412.96	\$ 412.96	\$ 0.00	\$ 0.00	\$ 279.30	\$ 279.30	\$ 59.85	\$ 73.82	\$ 0.00	\$ 133.67	\$ 0
Net Existing Processed Billing:	\$ 359.10	\$ 359.10	\$ 0.00	\$ 0.00	\$ 0.00	\$ 0.00	\$ 199.50	\$ 159.60	\$ 0.00	\$ 359.10	\$ 0
Net Total Processed Billing:	\$ 772.07	\$ 772.07	\$ 0.00	\$ 0.00	\$ 279.30	\$ 279.30	\$ 259.35	\$ 233.42	\$ 0.00	\$ 492.76	\$ 0

Personal Edition - Ad Supported Bundle

Figure 152

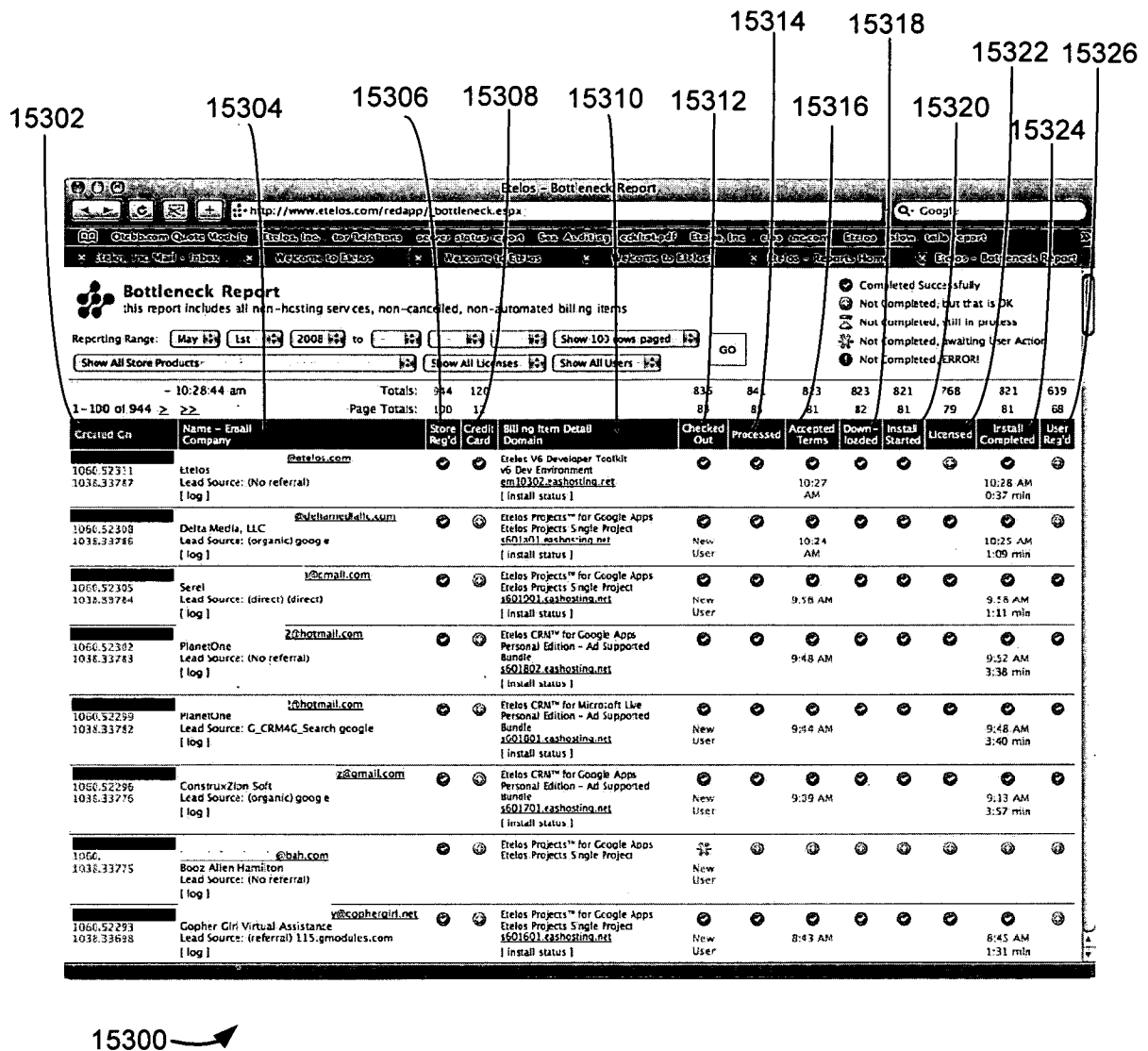


Figure 153

The screenshot shows the Etelos Support page. At the top, there is a navigation bar with links: Marketplace, For Businesses, For Developers, Blogs, About, and Support. A large black banner with the text "Etelos Support" is prominently displayed. Below this, a secondary navigation bar includes links: Support Home, Getting Started, Forums, Account, and Etelos Ideas. A user status message indicates "You are logged in (not you?)".

On the left side, there is a "My Account Info" sidebar with links: My Installations, Billing History, My Storefront, My Storefront, Store Reports, My Apps, My Accounts, CHP Accounts, Credit Card, and Profile.

The main content area features an "Edit App Info" form (labeled 15402) for the app "test_contactmini1". The form includes fields for App Name, Type (Update), Group (New Group), Version (2.0), and Description ("This is a test update so you can see this."). Below the form are "Update" and "Cancel" buttons (labeled 15404).

Below the form is the "My Packaged Apps" section (labeled 15410). It contains a list of apps with columns: Name, Deploy, App Type, Created, Group, Version, and Archive. The list includes apps like "test_contactmini1", "calendar_test", "MailManager2_1", "training_and_certification", "for test for deploy3", "V6 Blank App", "v6 CRM Full", "v6 mini app test7", and "Casey License Final".

Annotations with arrows point to specific elements in the app list:

- 15406 points to the "Name" column header.
- 15408 points to the "Deploy" column header.
- 15412 points to the "App Type" column header.
- 15414 points to the "Group" column header.
- 15416 points to the "Version" column header.
- 15418 points to the "Archive" column header.

At the bottom of the page, there is a status bar showing "Done" and a timer "1.352s".

15400 →

Figure 154

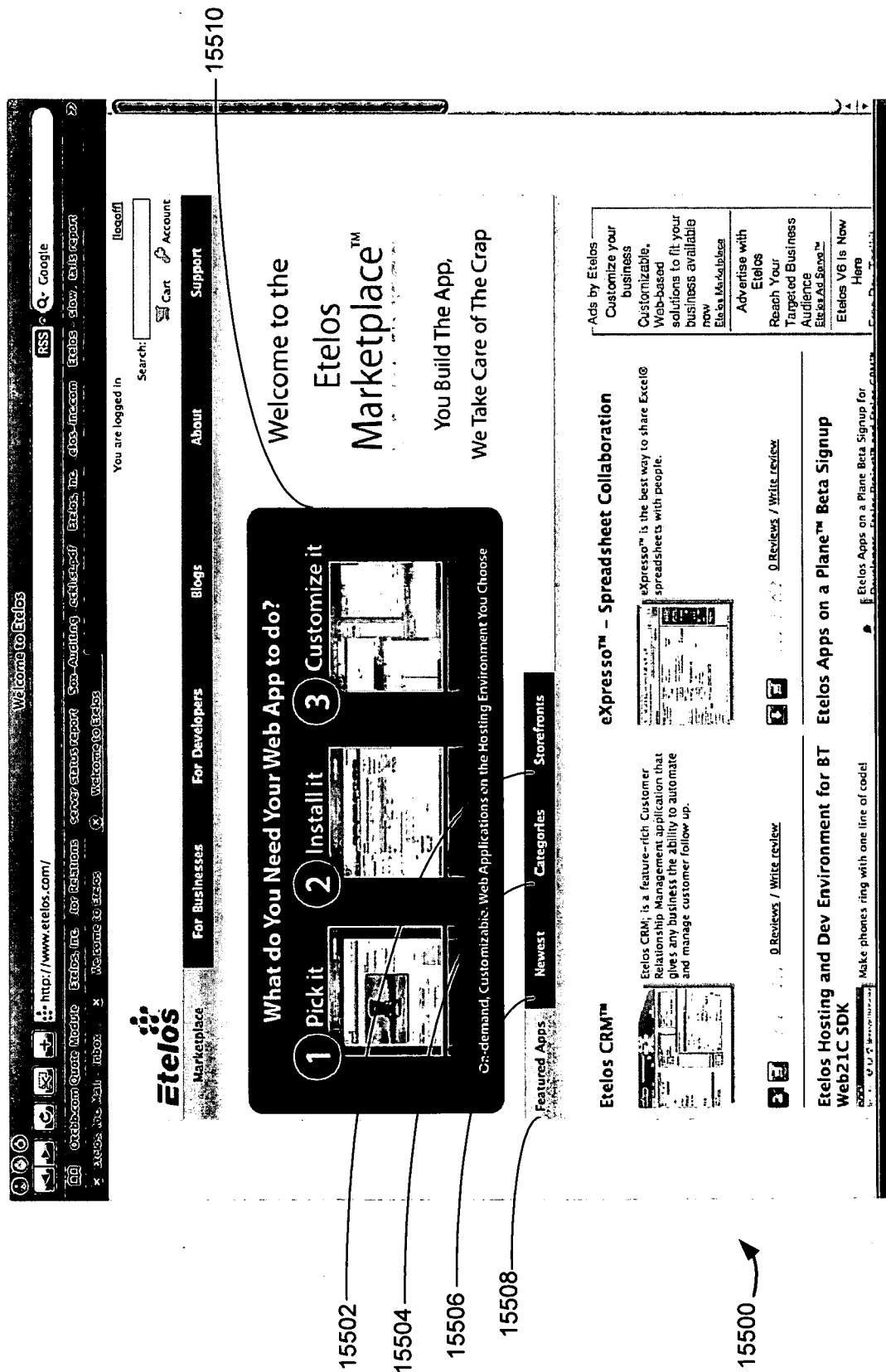


Figure 155

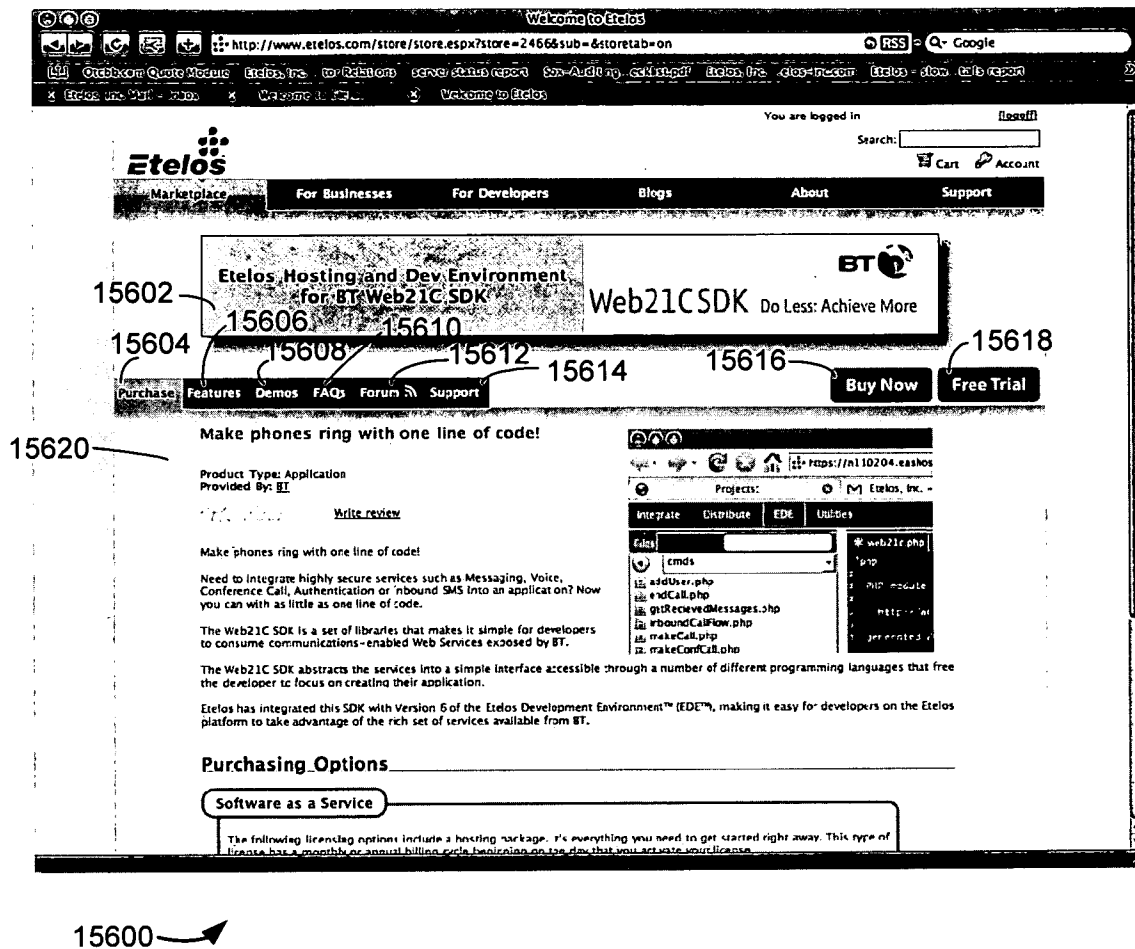
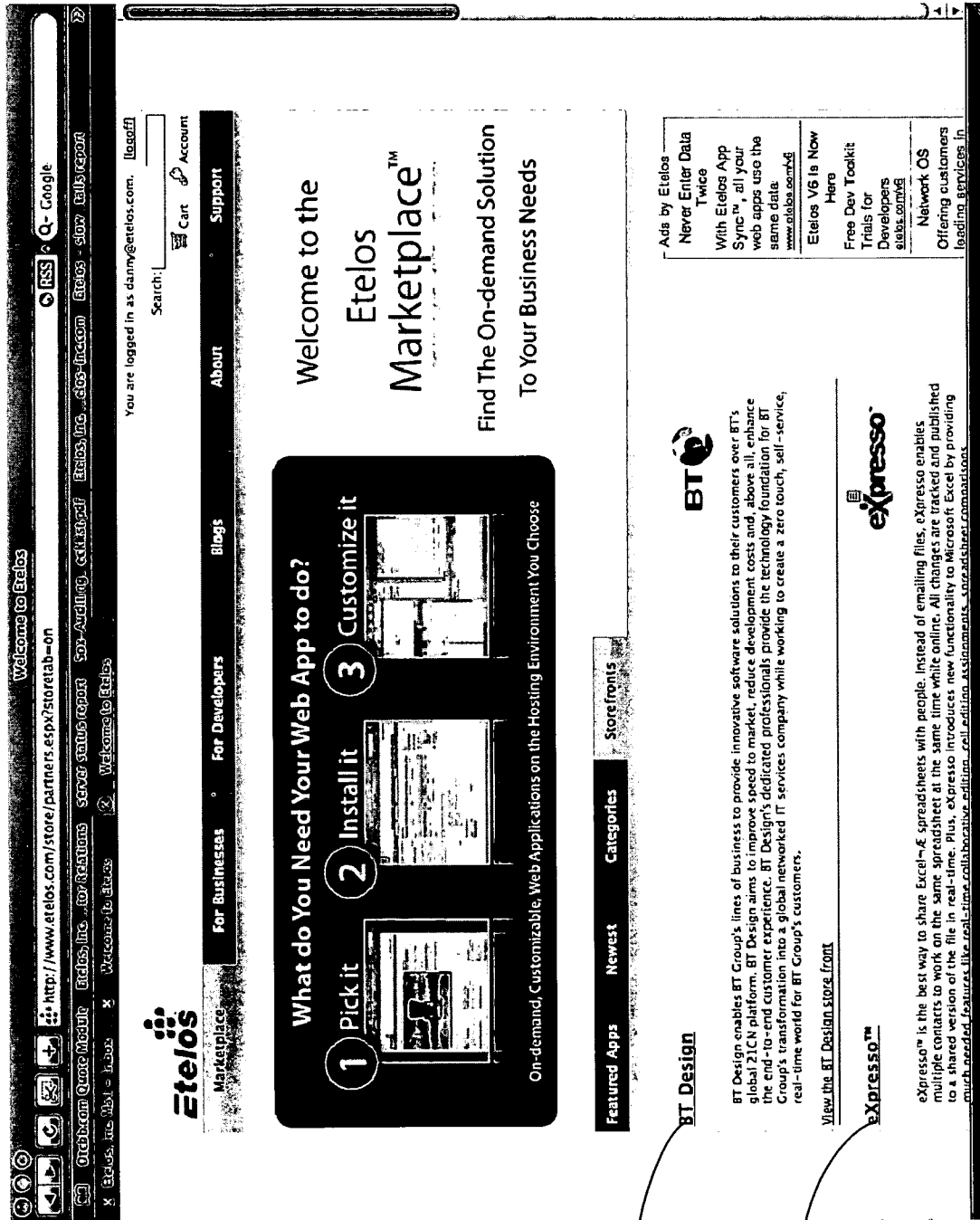


Figure 156



15702

15704

15700

Figure 157

Marketplace For Businesses For Developers Blogs About

Support Home Getting Started Forums Account Etelos Ideas

You are logged in as [danny@etelos.com](#). (not you?)

15902

15904

15900

15902

15904

15900

15906

15908 15910 15912 15914

Billing History

Your account has no pending transactions.

Date	Order ID	Name	Description	Amount Paid
	0408_24295	Etelos Projects™ for Google Apps	Credit - Professional -	\$ -19.95
	0408_24295		Total	\$ -19.95
	0308_24011	Etelos Projects™ for Google Apps	Credit - Personal -	\$ -19.95
	0308_24011		Total	\$ -19.95
	0408_24291	Etelos Projects™ for Google Apps	Credit - Personal -	\$ -9.95
	0408_24291		Total	\$ -9.95
	0408_24288	Etelos Projects for Google Apps V6	Credit - Personal -	\$ -9.95
	0408_24288		Total	\$ -9.95
	0408_24077	Etelos Projects for Google Apps V6	Personal -	\$ 9.95
	0408_24077		Total	\$ 9.95

Figure 159

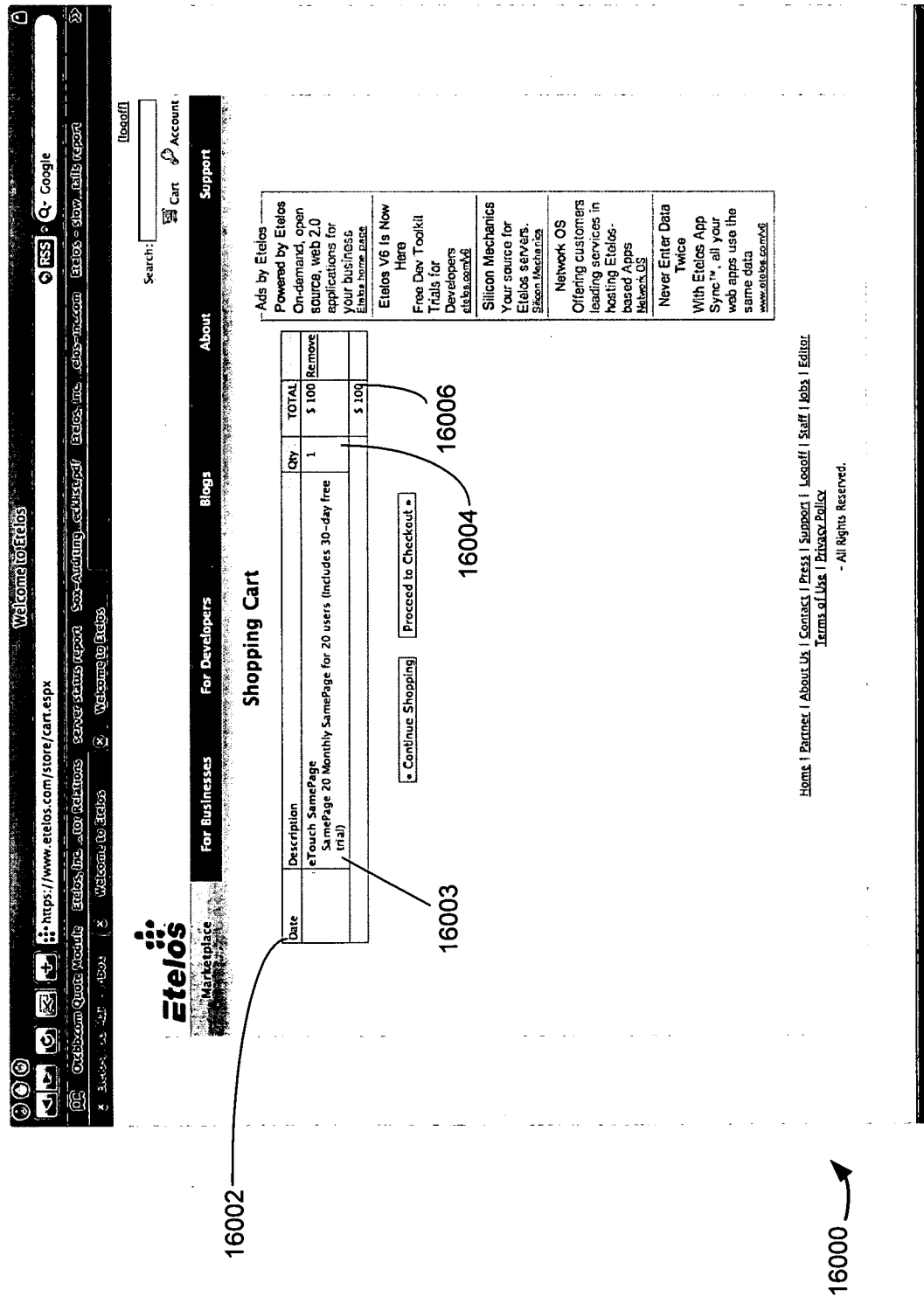
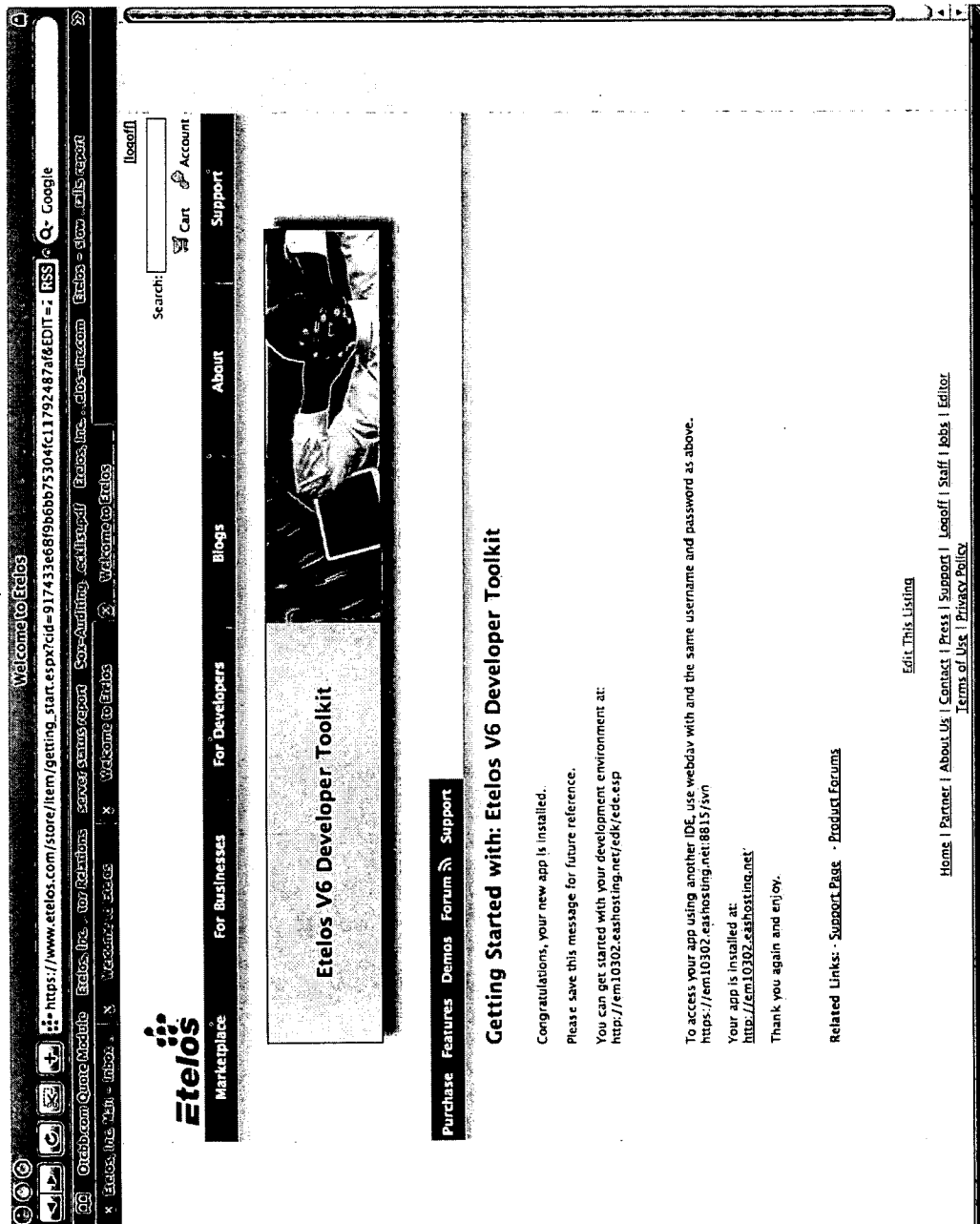


Figure 160

The screenshot shows the Etelos CRM website. The browser address bar displays the URL: https://www.etelos.com/store/item/getting_start.aspx?id=917433e68f9b6bb75304c11792487a&EDIT=RSS. The page features a navigation menu with links: [Marketplace](#), [For Businesses](#), [For Developers](#), [Blogs](#), [About](#), and [Support](#). A search bar is located at the top right, with a "Search:" label and a "Go" button. The main content area is titled "CRM Test Listing ** For Testing Only**" and includes a "Welcome" section. The welcome message states: "It's easy to get started. Click the link below to activate your account. Your username and password is the same as your Etelos Marketplace account." It then provides a link to "Click here to initialize your app and get started." and a URL: <http://n281420.eashosting.net/boess.aspx?token=917433e68f9b6bb75304c11792487a&version=portal>. The page also includes a "Related Links" section with links to [Support Page](#) and [Product Forums](#).

Figure 161





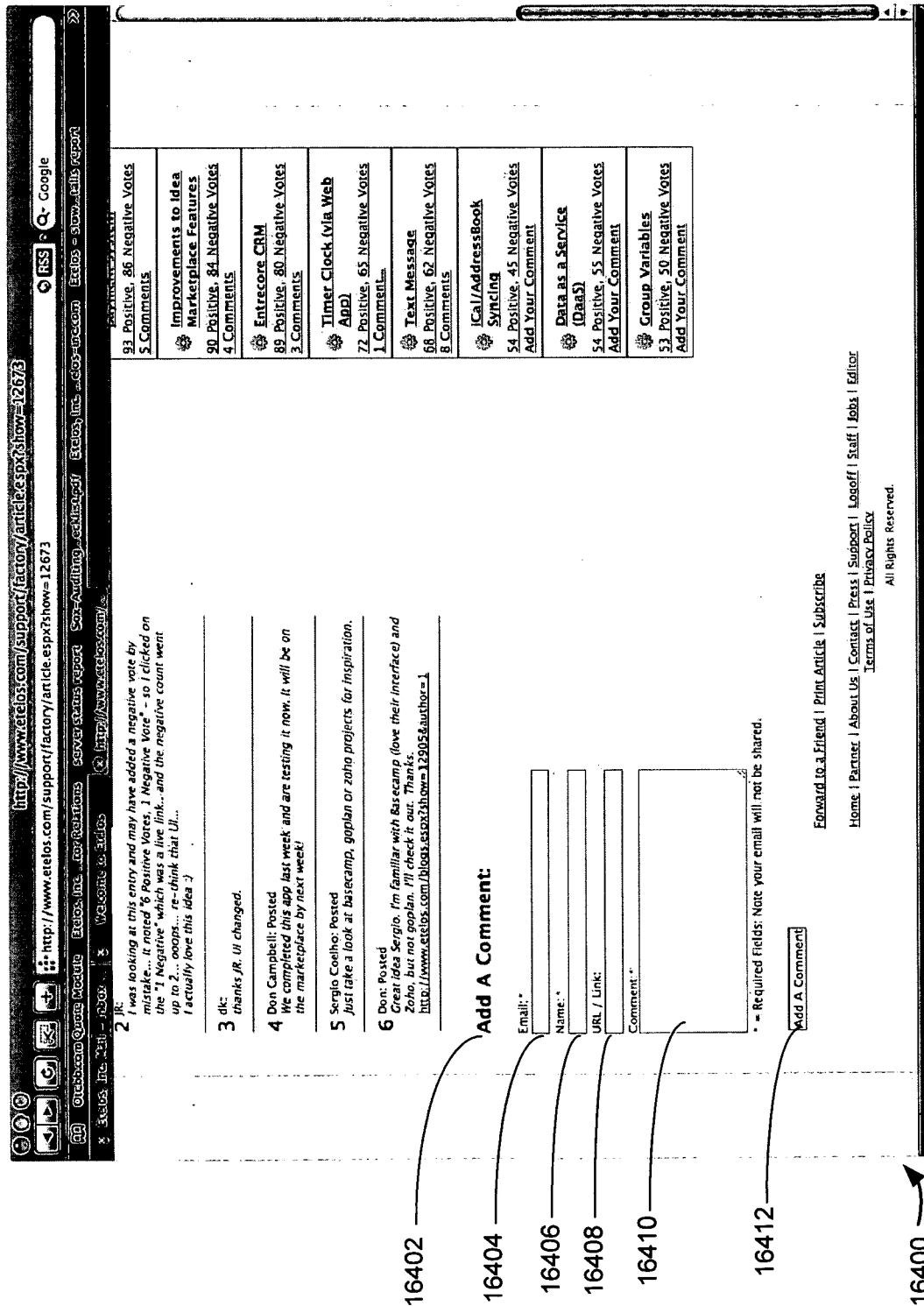


Figure 164

The screenshot shows the Etelos Idea Factory website. At the top, there is a navigation bar with links: [Marketplace](#), [For Businesses](#), [For Developers](#), [Blogs](#), and [About](#). Below this is a large black banner with the text "Etelos Support" in white. To the right of the banner, there is a sidebar with links: [Support Home](#), [Getting Started](#), [Forums](#), and [Account](#). The main content area displays a list of ideas, with the first one being "Start a New Idea" (ID 16502) and the second being "Most Popular Ideas" (ID 16504). The "Most Popular Ideas" section lists several ideas, including "Syndicated Project Management App" (ID 129), "Concept of Private Ideas" (ID 120), and "Create a Turnkey App for Accelerating Google Apps Dev" (ID 120). A large black banner with the text "Etelos Support" is positioned above the list of ideas. The bottom of the page features a footer with the text "© 2008 Etelos, Inc. All Rights Reserved." and a link to "Privacy Policy".

16502 — [Start a New Idea](#)

16504 — [Most Popular Ideas](#)

16500 — [Create a Turnkey App for Accelerating Google Apps Dev](#)

Etelos Support

Etelos

Marketplace For Businesses For Developers Blogs About

Support Home Getting Started Forums Account

You are logged in (not you?)

[Most Recent Ideas](#) [Most Popular Ideas](#) [Most Talked About](#) [My Ideas](#)

[Syndicated Project Management App](#)
by
An on-demand Project Management App that can surface through Google and Google Apps. Round trip integration with Google Spreadsheets for real-time dashboard and team meetings. [More...](#)

129 Positive, 108 Negative Votes | 6 Comments

[Concept of Private Ideas](#)
by
What about the ability to create "private" or "invite only" ideas - this would be for an idea that you have but are willing to share only with a smaller group of people. [More...](#)

120 Positive, 112 Negative Votes | 6 Comments

[Create a Turnkey App for Accelerating Google Apps Dev](#)
by
Starter app to include tools for managing users, accounts, licensing, support, etc. and even basic samples that show how to do gadgets and apps integration. [More...](#)

© 2008 Etelos, Inc. All Rights Reserved. [Privacy Policy](#)

Figure 165

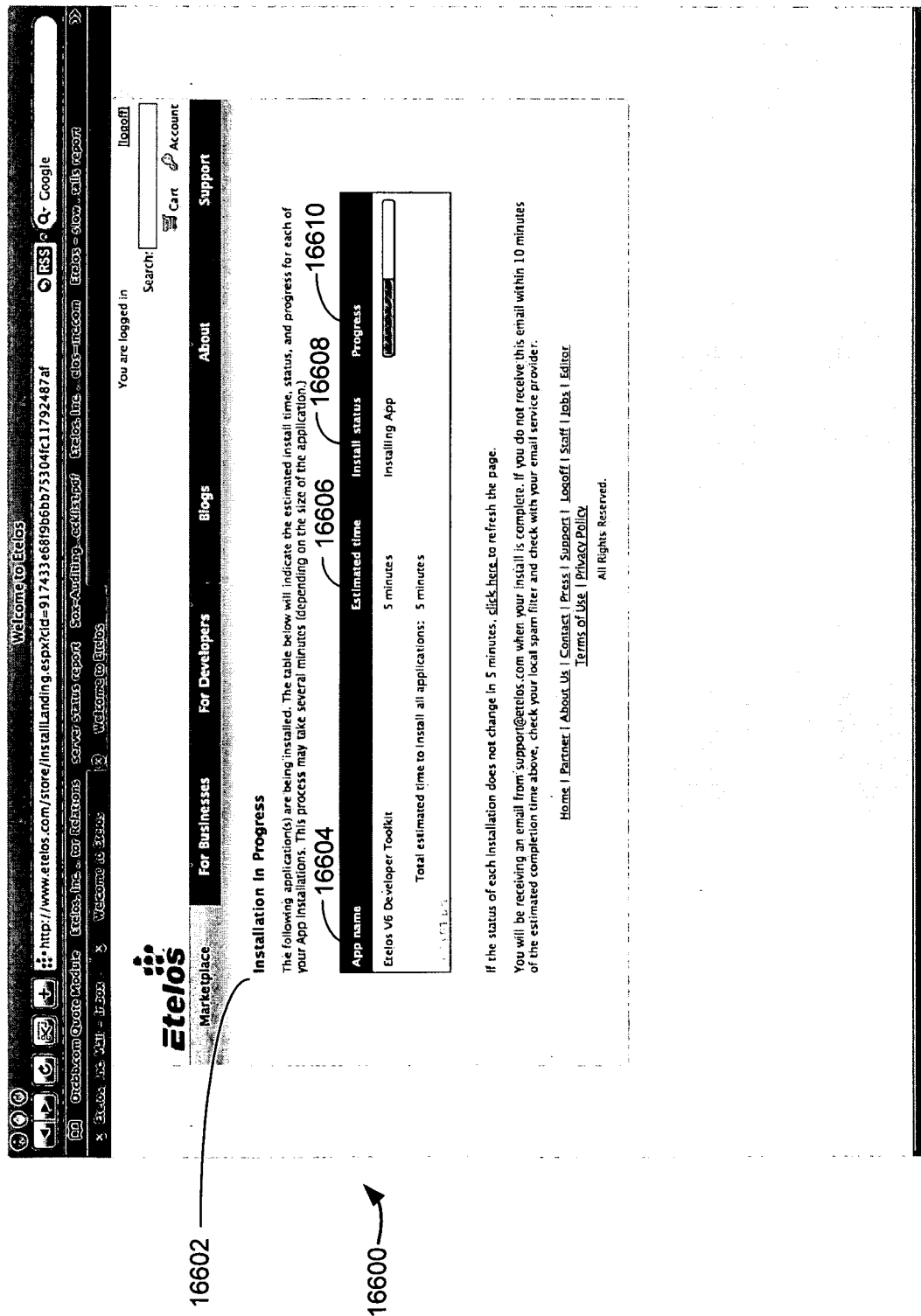


Figure 166

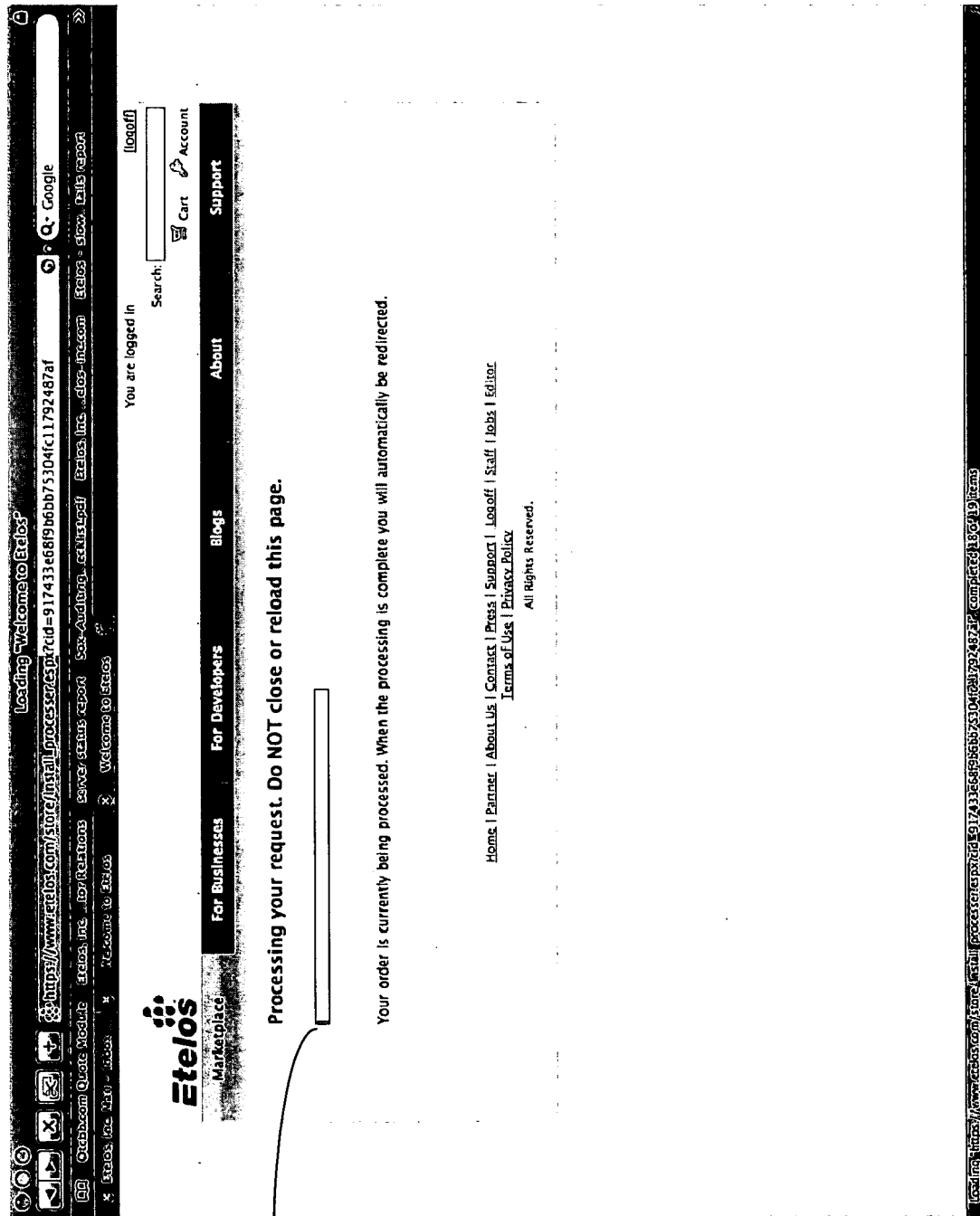
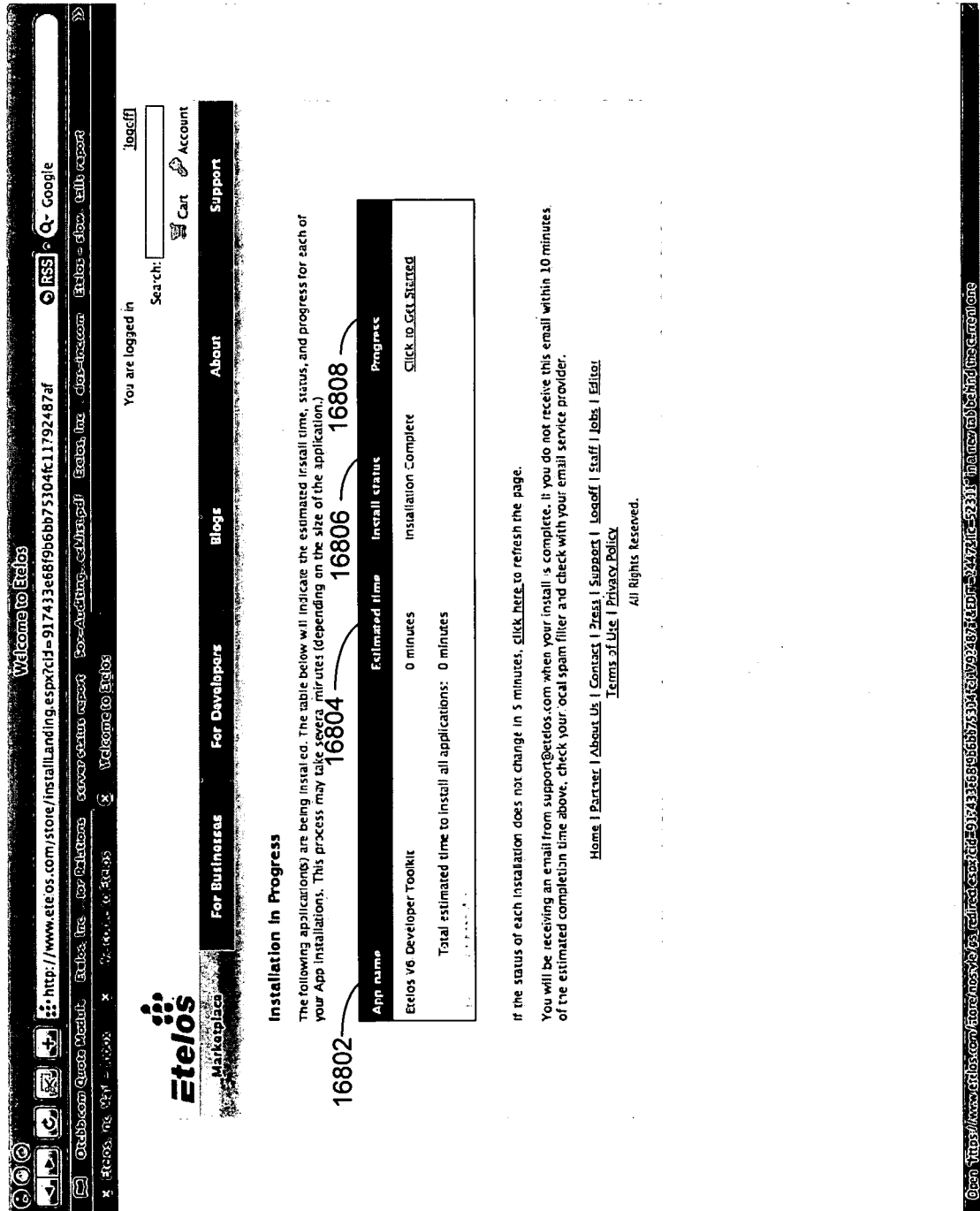


Figure 167

16702

16700



16900

16902

16904

16906

16908

Etelos Support

My Developer Accounts

Select an Account Name to edit it's properties, or click "Add Account" to manually enter an account. Please Note: The accounts below are either accounts you have purchased thru the store for hosting your applications, or accounts from your own EAS server. If you need to purchase a hosting account click [here](#). If you add an alternate domain, you must point the domain to ns3.etelos.com and ns4.etelos.com.

Name	Default Domain	Status	Action
Normal	demo130.etelos.com	Q Status/History	Build an App
3servepath 130	Alternate Domains:		Add Domain
A Demo Account1	demo100.etelos.com	Q Status/History	Build an App
	Alternate Domains:		Add Domain
A Demo Account2	demo101.etelos.com	Q Status/History	Build an App
	Alternate Domains:		Add Domain
A Demo Account3	demo102.etelos.com	Q Status/History	Build an App
	Alternate Domains:		Add Domain
Aheadads Nuoz CRM test	no2100.eashosting.net	Q Status/History	Build an App
	Alternate Domains:		Add Domain
Becky Agent CRM	qa100.etelos.com	Q Status/History	Build an App
	Alternate Domains:		Add Domain
Beta Insurance	qa151.etelos.com	Q Status/History	Build an App

Figure 169

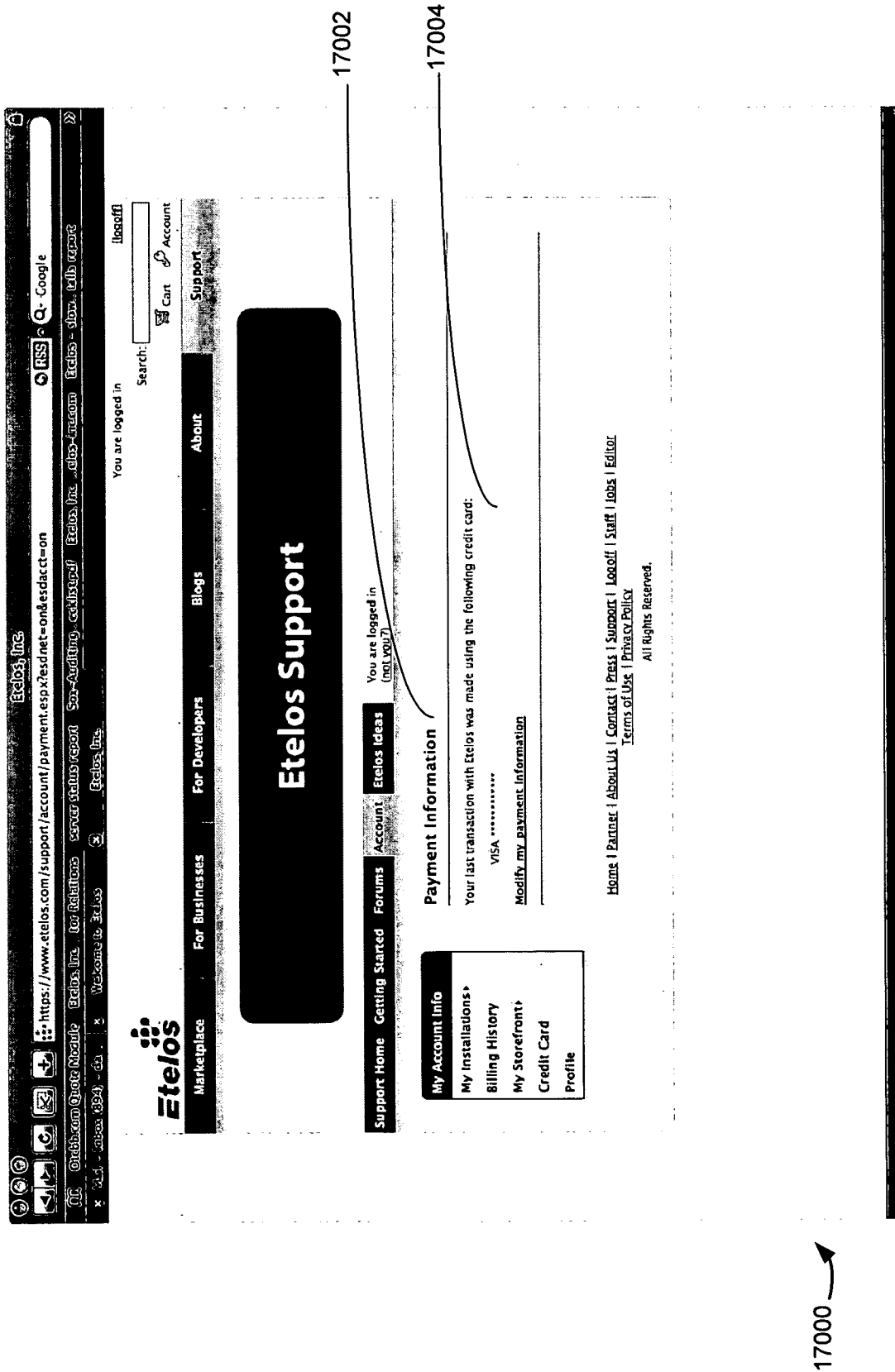


Figure 170

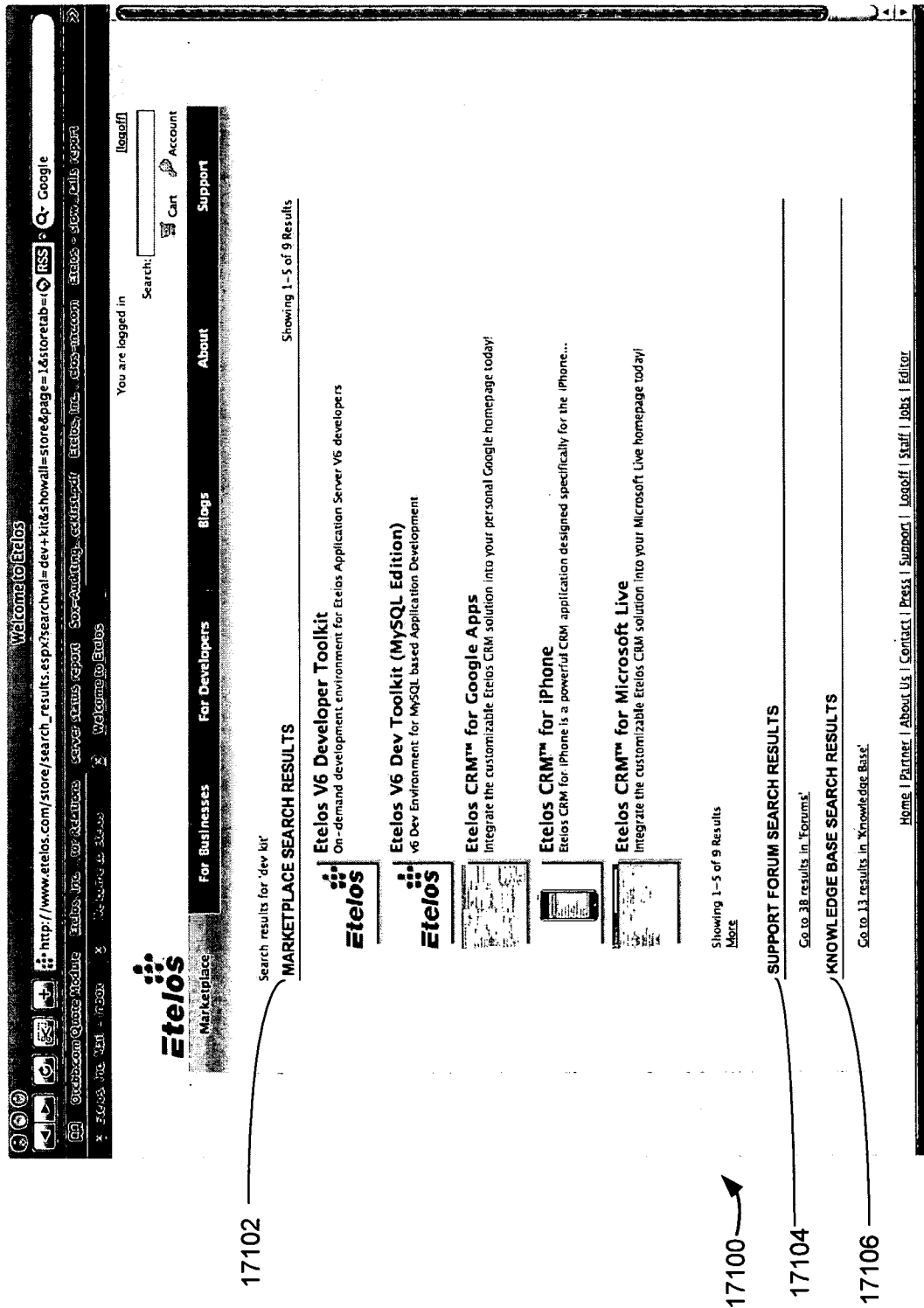


Figure 171

[http://www.etelos.com/store/cart.aspx?aid=4&subid=](#)
[RSS](#)
[Google](#)

[Welcome to Etelos](#)

[Etelos.com Quote Module](#)
[Etelos, Inc. - By Citigroup](#)
[Service status report](#)
[SaaS-Audit.org](#)
[etelostatof](#)
[Etelos, Inc. - cloy-inc.com](#)
[Etelos - now](#)
[Etelos report](#)

[Etelos, Inc. 2008](#)
[Welcome to Etelos](#)

You are logged in [Logout](#)

Search:

[Cart](#) [Account](#)

[Marketplace](#)
[For Businesses](#)
[For Developers](#)
[Blogs](#)
[About](#)
[Support](#)

Shopping Cart

Date	Description	Qty	TOTAL
	Etelos V6 Developer Toolkit	1	\$ 0
	v6 Dev Environment Dev 30 Day Trial for v6 Environment		\$ 0

[Continue Shopping](#)
[Proceed to Checkout](#)

17202

17204

17206

17200

[Home](#) |
 [Partner](#) |
 [About Us](#) |
 [Contact](#) |
 [Press](#) |
 [Support](#) |
 [Logout](#) |
 [Staff](#) |
 [Jobs](#) |
 [Editor](#)
[Terms of Use](#) |
 [Privacy Policy](#)
 All Rights Reserved.

Figure 172

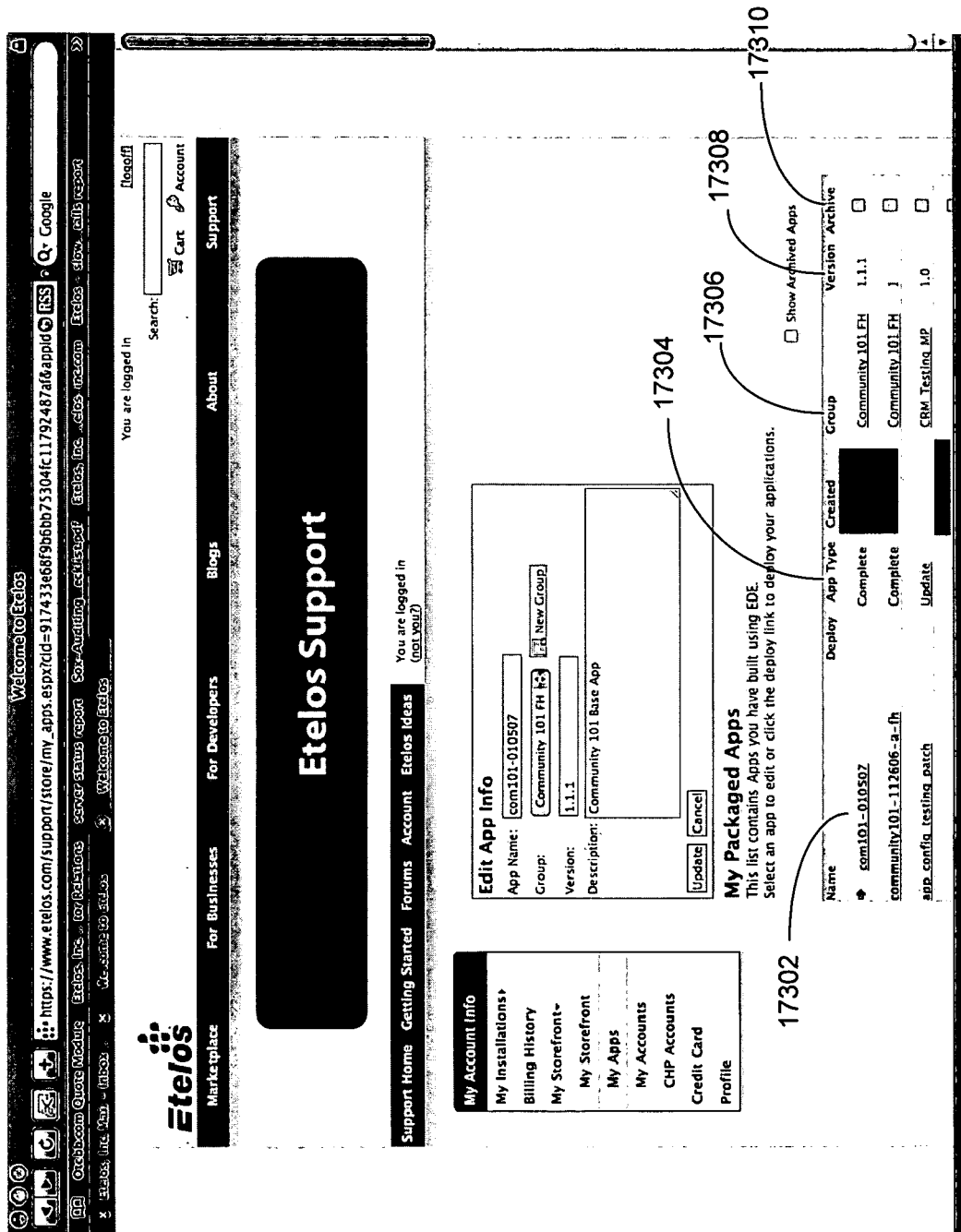


Figure 173

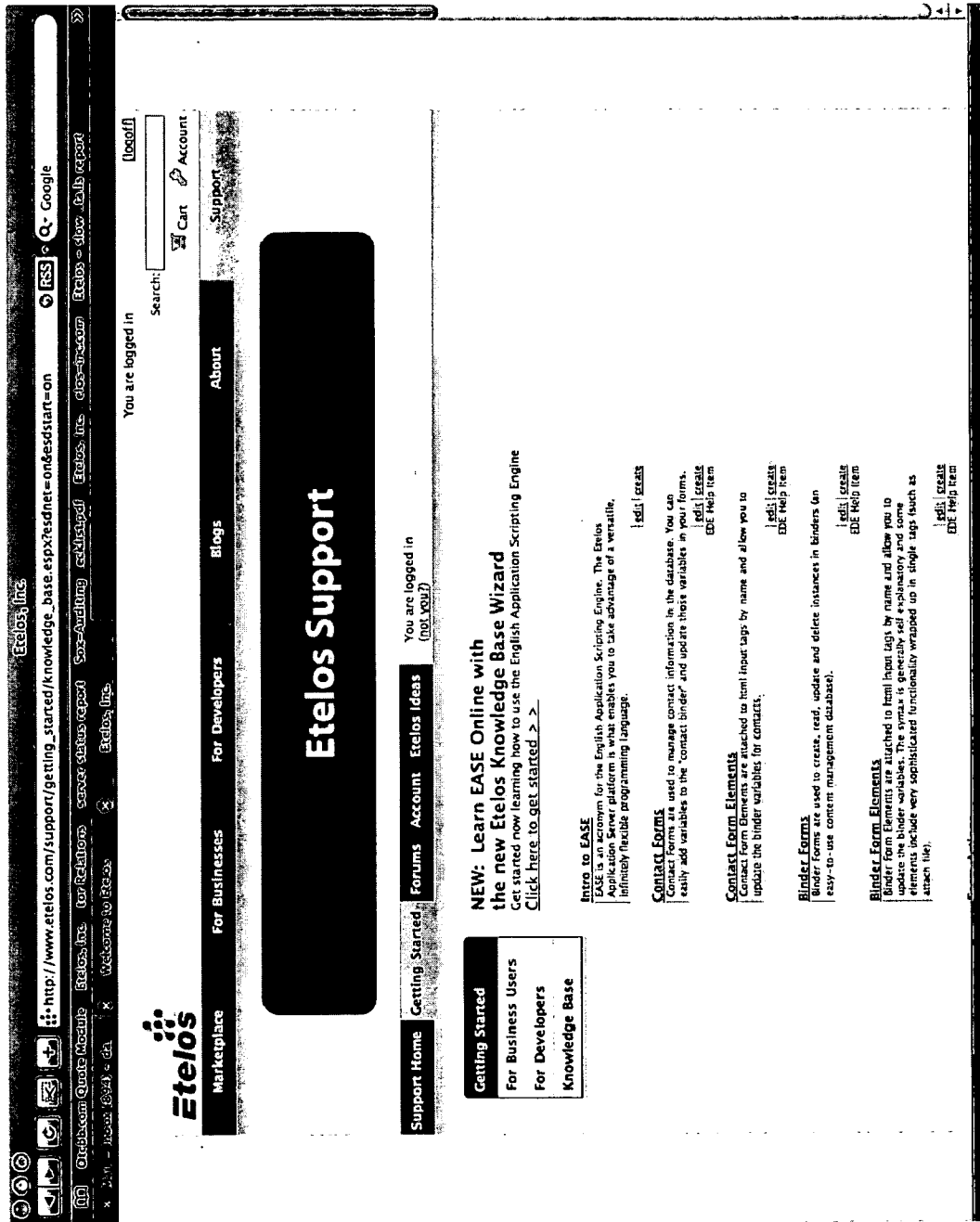


Figure 174

17500

You are logged in

Search:

[Cart](#) [Account](#)

Marketplace

For Businesses

For Developers

Blogs

About

[Support](#)

Etelos Support

[Support Home](#)

[Getting Started](#)

[Forums](#)

[Account](#)

[Etelos Ideas](#)

You are logged in
(not you?)

My Account Info

[My Installations](#)

[My Installations](#)

[Pending Installs](#)

[Downloads](#)

[Ready to Install](#)

[View All](#)

[Billing History](#)

[My Storefront](#)

[Credit Card](#)

[Profile](#)

My Installations

The following installations are in your Etelos account. Click the "Getting Started" button and the "Support Information" button to read the available information for each installation.

Install Date	Product Info	Licensing	Getting Started/Support
17502	CRM Test Listing ** For Testing Only** Application Personal Edition - Ad Supported Bundle more...	Per User: 1	Getting Started Support n281470@xbshosting.net Cancel
17504	Etelos CRM for Google Apps (V5) Application Personal Edition - Ad Supported Bundle more...	Per User: 1	Getting Started Support neil.brianlane.com.9443 Reinstall
Expiration Date:	Sugar CRM Application Free 10 Day Trial more...	Per Account	Getting Started Support temo137@eteloserve.com Cancel
Expiration Date:	EDE Lite Etelos EDE EDE Lite Trial more...	Per User:	Getting Started Support Reinstall

17500 17502 17504 17506 17508 17510

Figure 175

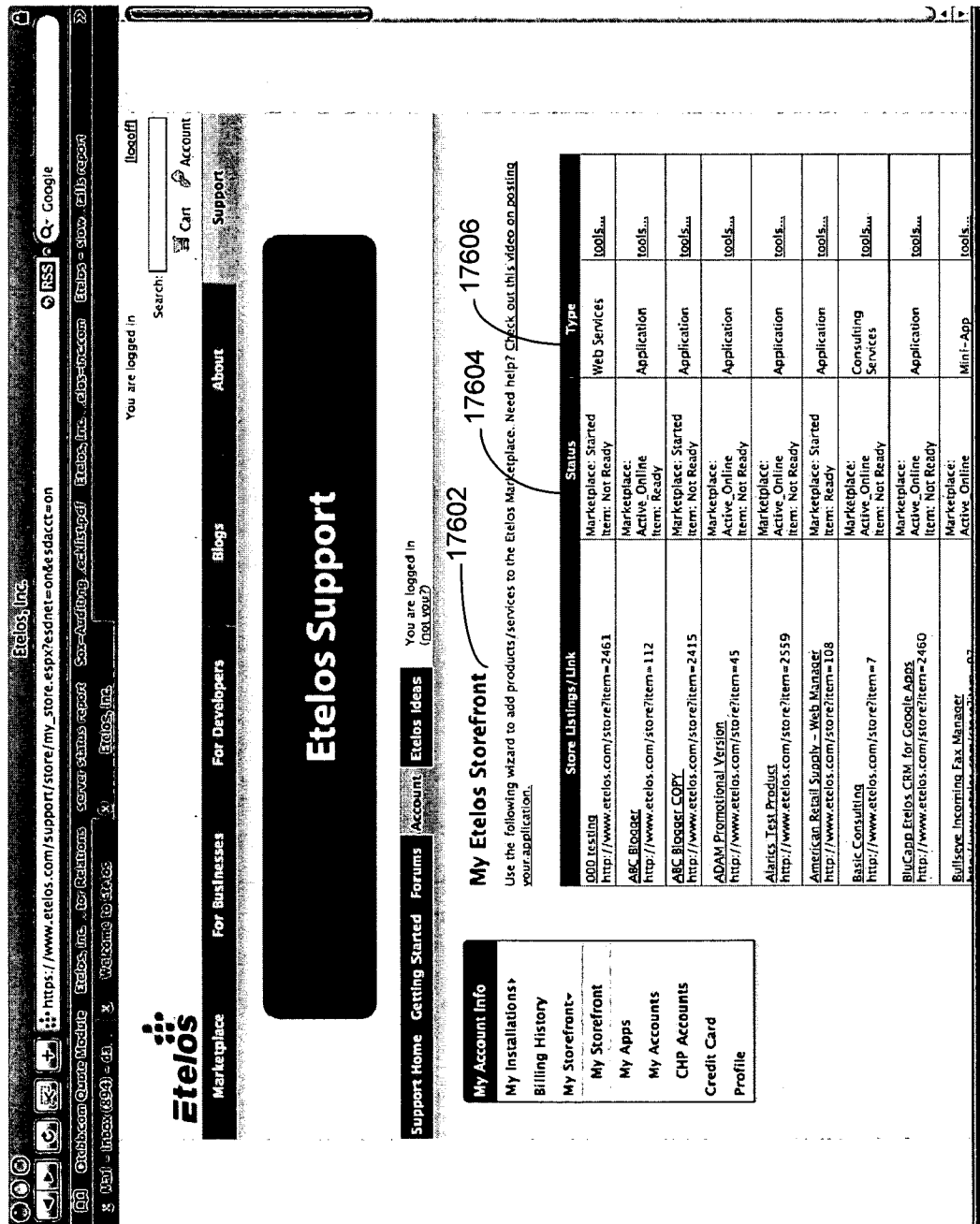


Figure 176

Etelos

Marketplace For Businesses For Developers Blogs About Support

Welcome to Etelos

http://www.etelos.com/support/showforum.aspx?searchby=topic_only&include=on&esdforum=on&son= RSS - Q: Google

Search:

You are logged in (not you?)

17702 17700

17708 17710 17712

17706 17704

Discussion Topics (listed in order of most recent update)

Thread	Last Post	Replies	Views	Subscribe
Is it possible to modify the Etelos/Google CRM app code?		4	18	☐
CRM Installation 2s		15	118	☐
Block Publishing via Groups		4	17	☐
apps not loading		2	17	☐
Comparing with Basecamp		5	18	☐
Sessions and Login/logout/locks in v6		5	30	☐
File attached: Click here to download Edit/Delete		2	13	☐
Edit Staus Groups problem: Netvibes CRM		NEW	14	☐
Calendar Time: Netvibes CRM				☐

Figure 177

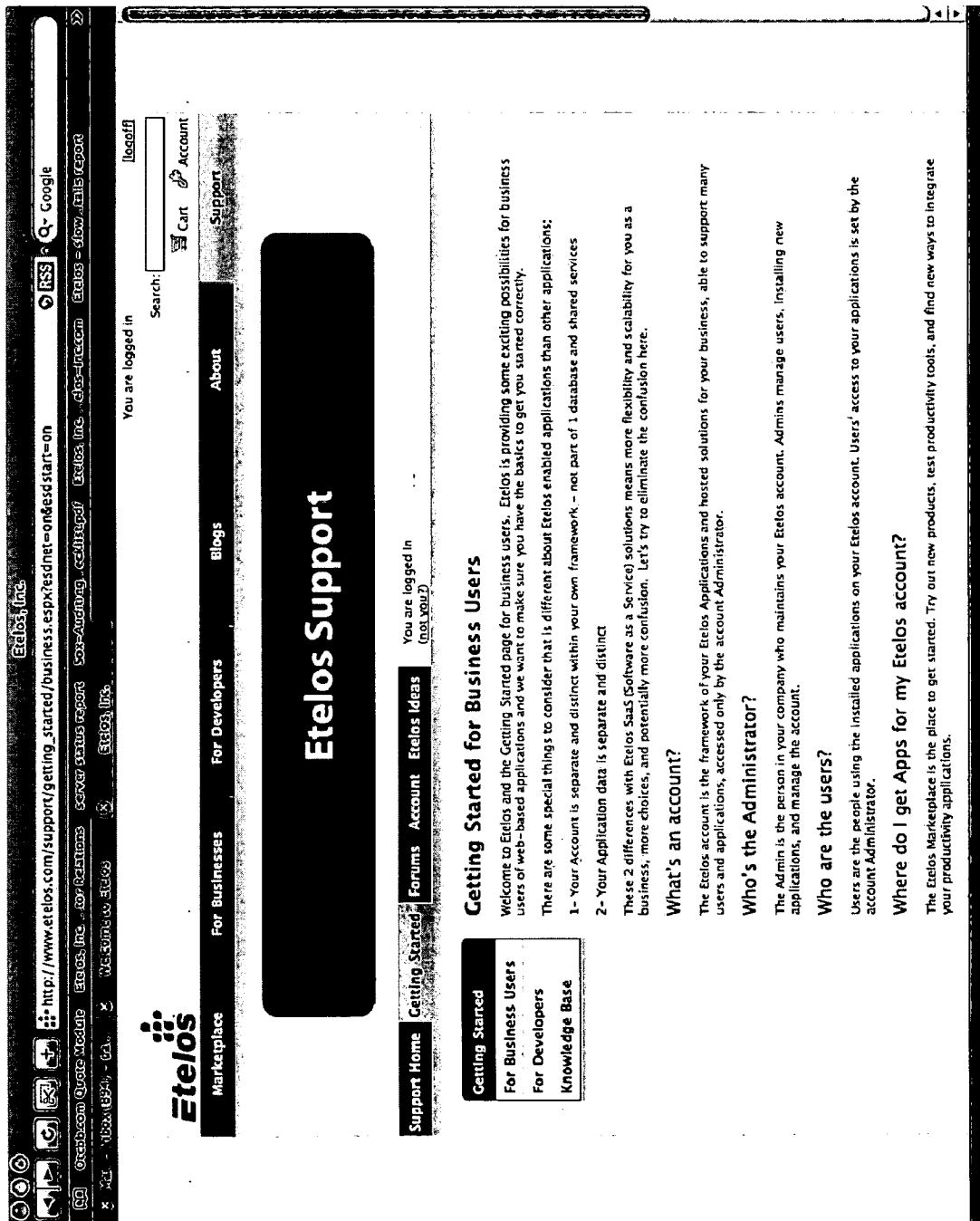


Figure 178

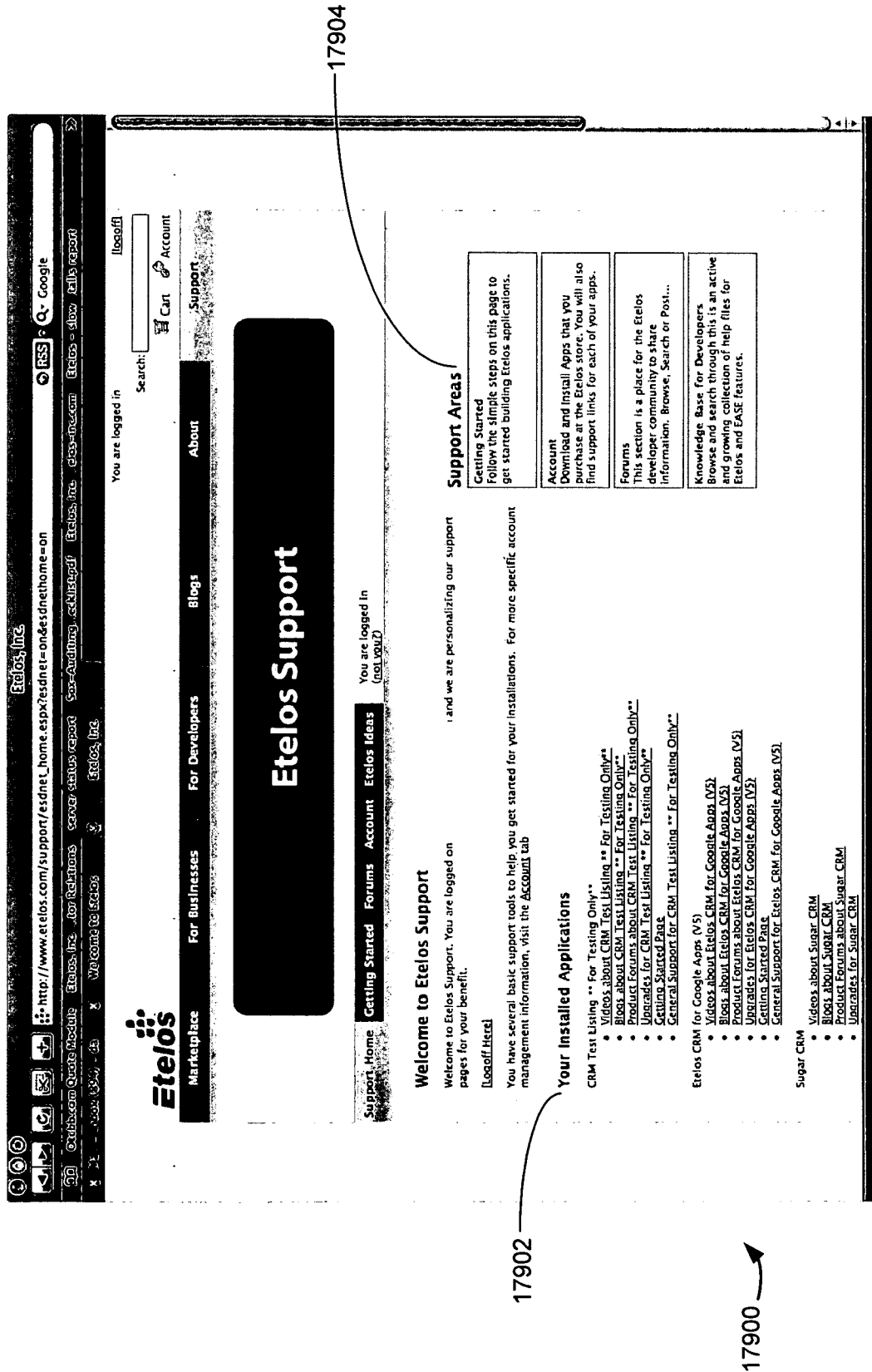


Figure 179

Welcome to Etelos

Search: [Account](#) [Cart](#) [Support](#)

Subcategories - ()

[About](#) [Blogs](#) [For Developers](#) [For Businesses](#) [Marketplace](#)

Etelos Support

You are logged in (not you?)

[Support Home](#) [Getting Started](#) [Forums](#) [Account](#) [Etelos Ideas](#)

Introducing EASE

EASE is an acronym for English Application Scripting Engine. The Etelos Application Server platform is what enables you to take advantage of programming with EASE. In the next few pages we are going to show how easy it is to program your Web application with EASE.

EASE uses an English-based syntax for control over sophisticated scripting engines.

```
<# start form;
for contact (#[contactid]#);
when updating add to group (123);
when updating publish (1234) to contact ([subject.id]);
when finished redirect to ("IPACE_URL_45j#");
.##
```

ex1. Starting Contact form syntax with formactions

Select next below to start learning how easy EASE is to use...

[<< Previous](#) [Next >>](#)

[Edit / Create](#)

Getting Help
EASE syntax help is readily available in EDC by clicking on the [E](#) icon or by 'right clicking' inside any text editor window. EDC also has a [popup window](#) where you can browse and select from EASE help syntax to be inserted into your pages, emails, etc. [Edit / Create](#)

Intro to EASE
EASE is an acronym for English Application Scripting Engine. The Etelos Application Server platform is what enables you to take advantage of programming with EASE, and in the next few pages we are going to introduce you to how easy it is to program your web application with EASE. [Edit / Create](#)

Contact Variables and Webpages
Basics on how to display contact variables on a webpage. [Edit / Create](#)

Load Contact and Reference
Basics on how to display contact variables on a webpage. [Edit / Create](#)

Binder, Variables and Webpages
Binder Variable Basics and displaying them in a webpage or email. [Edit / Create](#)

Basic EASE Applications
With EASE all web application development boils down to two fundamental engines: forms & lists. [Edit / Create](#)

Types of Forms
Basic definitions of types of forms [Edit / Create](#)

Figure 180

Etelos Support

Update Your Profile

This information is used to send all physical and electronic correspondence including invoices, email notifications and newsletters.

First Name:
 Last Name:
 Company: Etelos
 Title:
 Company Web Address: http://
 Phone:
 Email:
 Address1:
 Address2:
 City:
 State/Prov:
 Zip:
 Country: United States of America
 Username:
 Password:

Figure 181

The screenshot displays the Etelos Marketplace website. At the top, there is a navigation bar with links for "BT Design Storefront", "Etelos Marketplace", and a search bar. Below the navigation bar, a large banner features the text "Use your Web apps Offline, Online, Anytime" and "Sign up for Etelos AOP™ Beta today". To the right of the banner, there are links for "Featured Apps", "Newest", "Categories", "Storefronts", and "Sign Up Now". Below the banner, there is a section titled "Etelos Hosting and Dev Environment for BT Web21C SDK" with a sub-header "Make phones ring with one line of code!". To the right of this section, there is a "Read more" button. Below the "Read more" button, there is a "0 Reviews / Write review" link. At the bottom right, there is a "Sign Up Now" button. The website also includes a footer with the text "Open This: http://www.etelos.com/marketplace/storefront/etelos-apps/etelos-aop-beta/etelos-aop-beta-the-current-one".

Figure 182

18200 →

18300 →

18300 →

172/178

Figure 183

18400 →

The screenshot displays the Etelos Web21CSDK website. The header includes a search bar, login options, and navigation links. The main content area features the product title 'Etelos Hosting and Dev Environment for BT Web21C SDK' with a 'Do Less: Achieve More' tagline. Below this, there are buttons for 'Buy Now' and 'Free Trial'. A 'REVIEWS' section shows a review titled 'Fun and Easy to Setup' with a 5-star rating. The bottom of the page displays a table of ratings for various aspects of the product.

Overall rating (Highest to Lowest)
5 4 3 2 1
Getting Started / Installation (optional)
5 4 3 2 1
Ease of Use (optional)
5 4 3 2 1
Features (optional)
5 4 3 2 1
Benefits (optional)
5 4 3 2 1

18402

18404

18406

18408

18410

18412

18414

18416

173/178

Figure 184

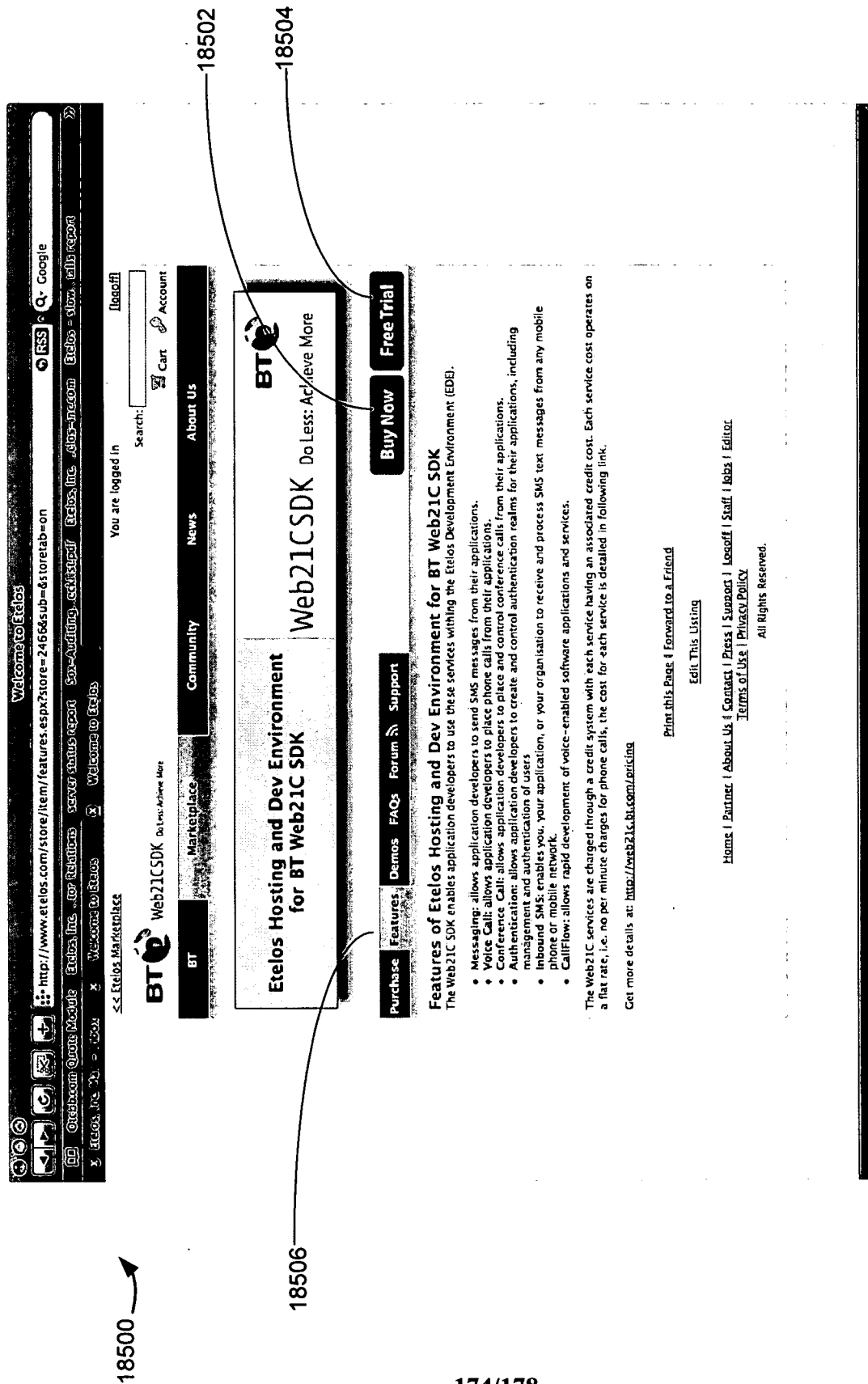


Figure 185

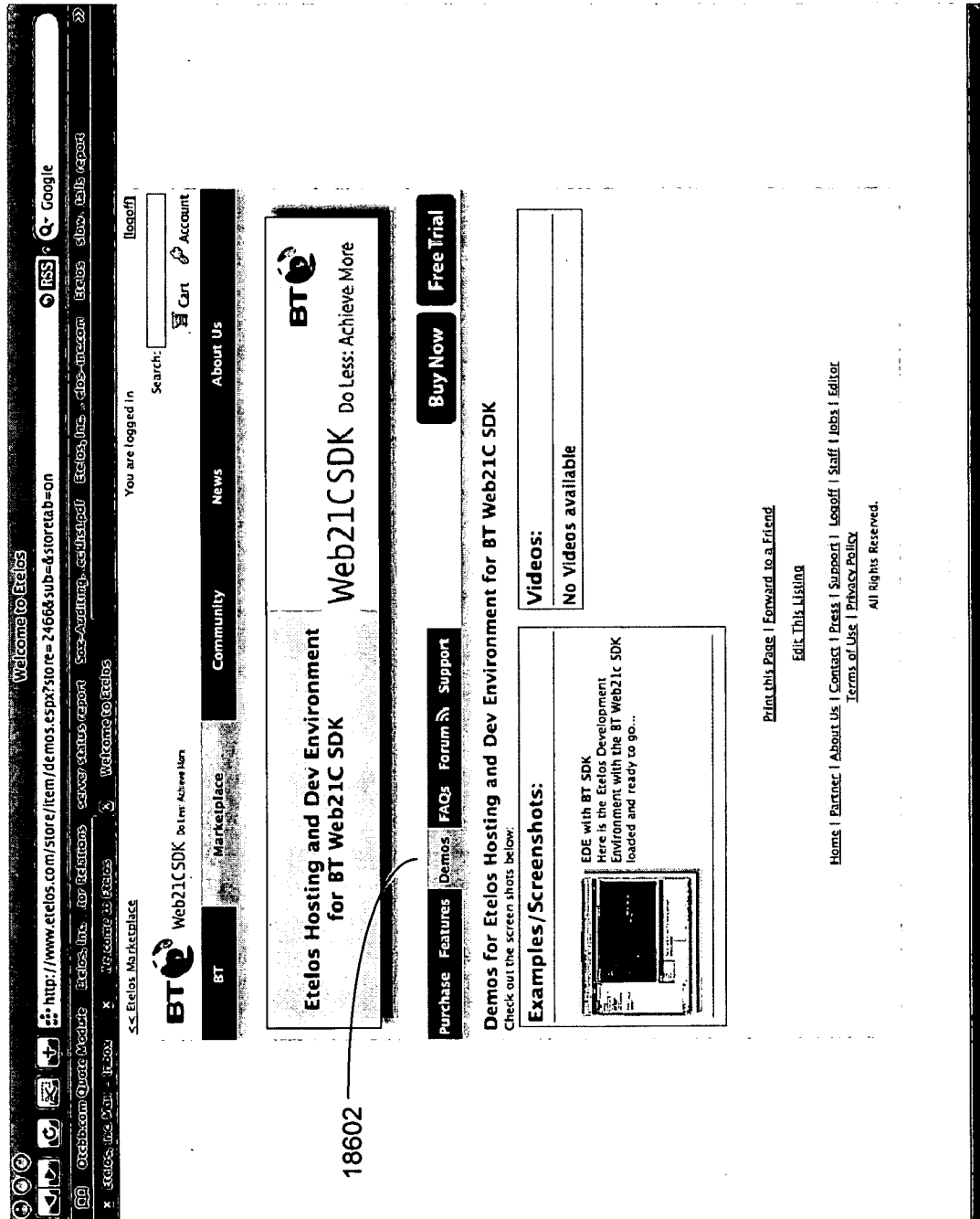


Figure 186

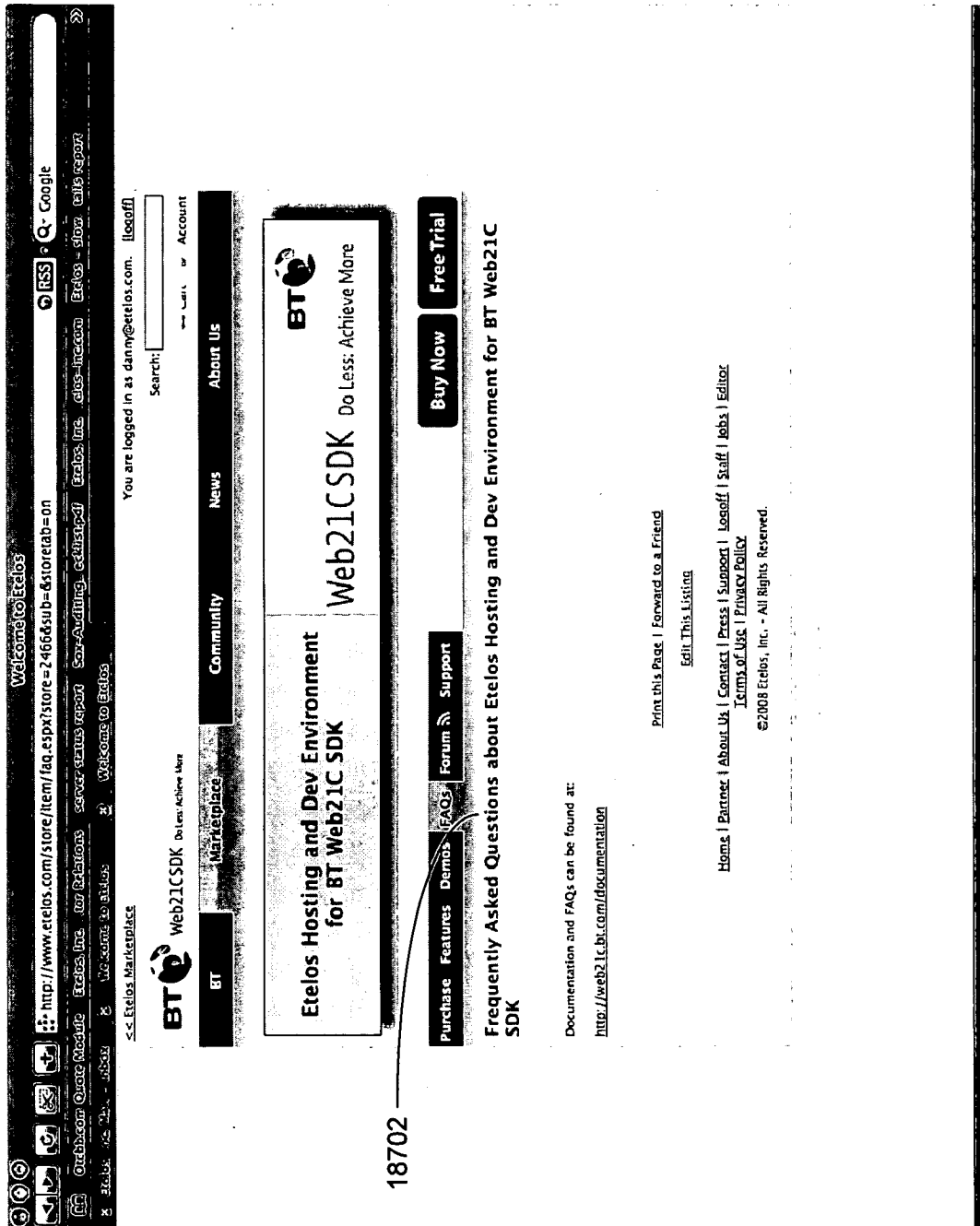


Figure 187

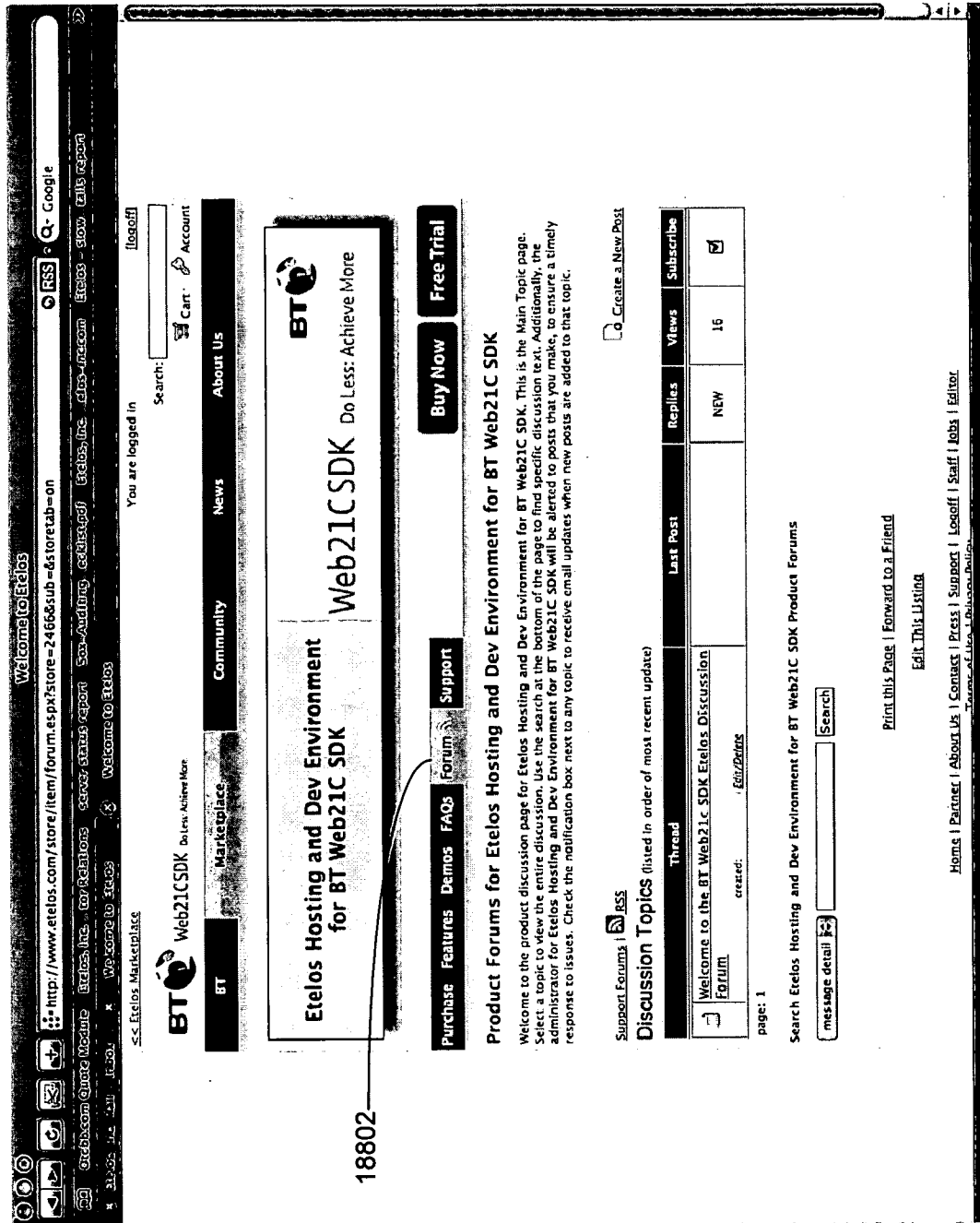


Figure 188

18900

Figure 189