US 20070130541A1

(19) United States

(12) Patent Application Publication

(10) Pub. No.: US 2007/0130541 A1

(43) Pub. Date: Jun. 7, 2007

Louch et al.

(54) SYNCHRONIZATION OF WIDGETS AND DASHBOARDS

(76) Inventors: **John O. Louch**, San Luis Obispo, CA (US); **Eric Steven Peyton**, Lisle, IL (US); **Christoper Hynes**, Santa Cruz, CA (US); **Scott Forstall**, Mountain View, CA (US); **Gregory N. Christie**, San Jose, CA (US)

Correspondence Address:
FISH & RICHARDSON P.C.
PO BOX 1022
MINNEAPOLIS, MN 55440-1022 (US)

(21) Appl. No.: 11/499,887

(22) Filed: Aug. 4, 2006

Related U.S. Application Data

(63) Continuation-in-part of application No. 10/877,968, filed on Jun. 25, 2004.

(60) Provisional application No. 60/583,125, filed on Jun. 25, 2004. Provisional application No. 60/734,016, filed on Nov. 4, 2005. Provisional application No. 60/730,956, filed on Oct. 27, 2005. Provisional application No. 60/737,899, filed on Nov. 18, 2005.

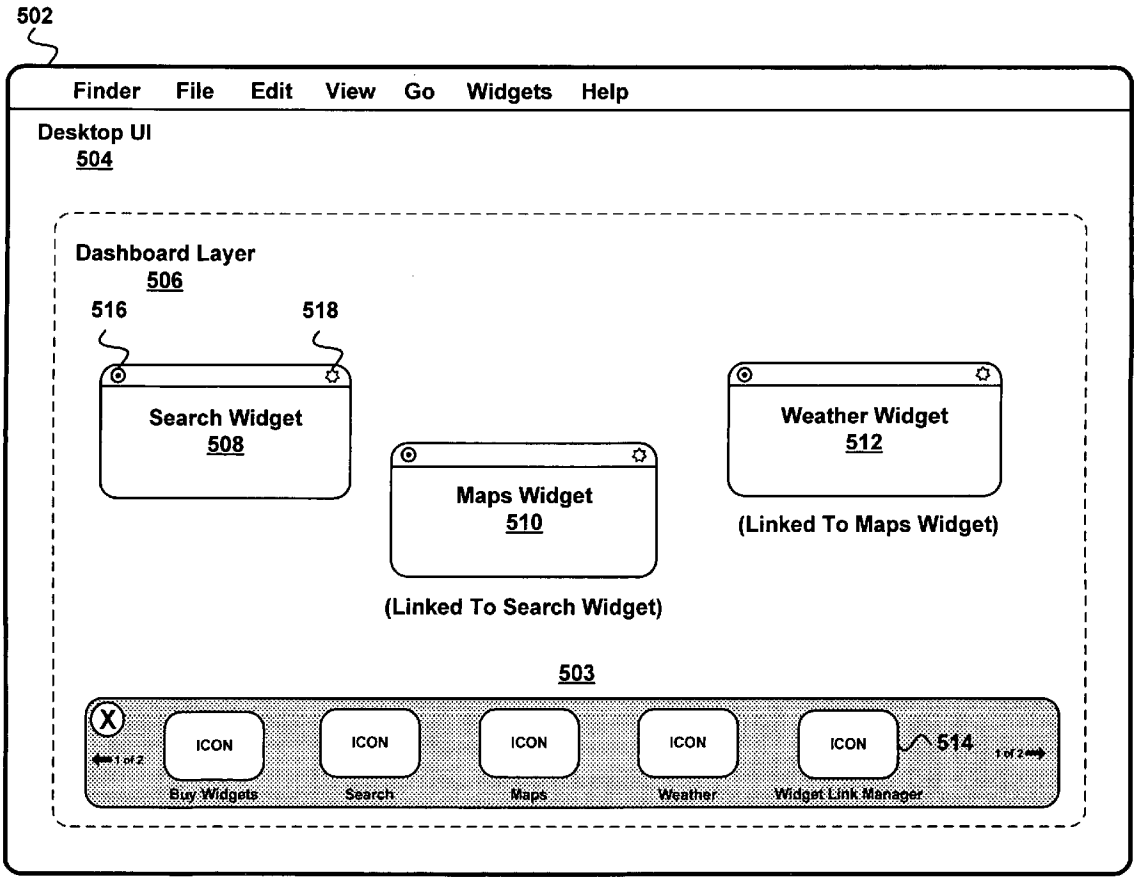
Publication Classification

(51) Int. Cl.
G06F 17/00 (2006.01)

(52) U.S. Cl. 715/804

(57) **ABSTRACT**

Systems, methods, computer-readable mediums, user interfaces and other implementations are disclosed for synchronizing widgets and dashboards.



SAP AMERICA, INC.
EXHIBIT 1039

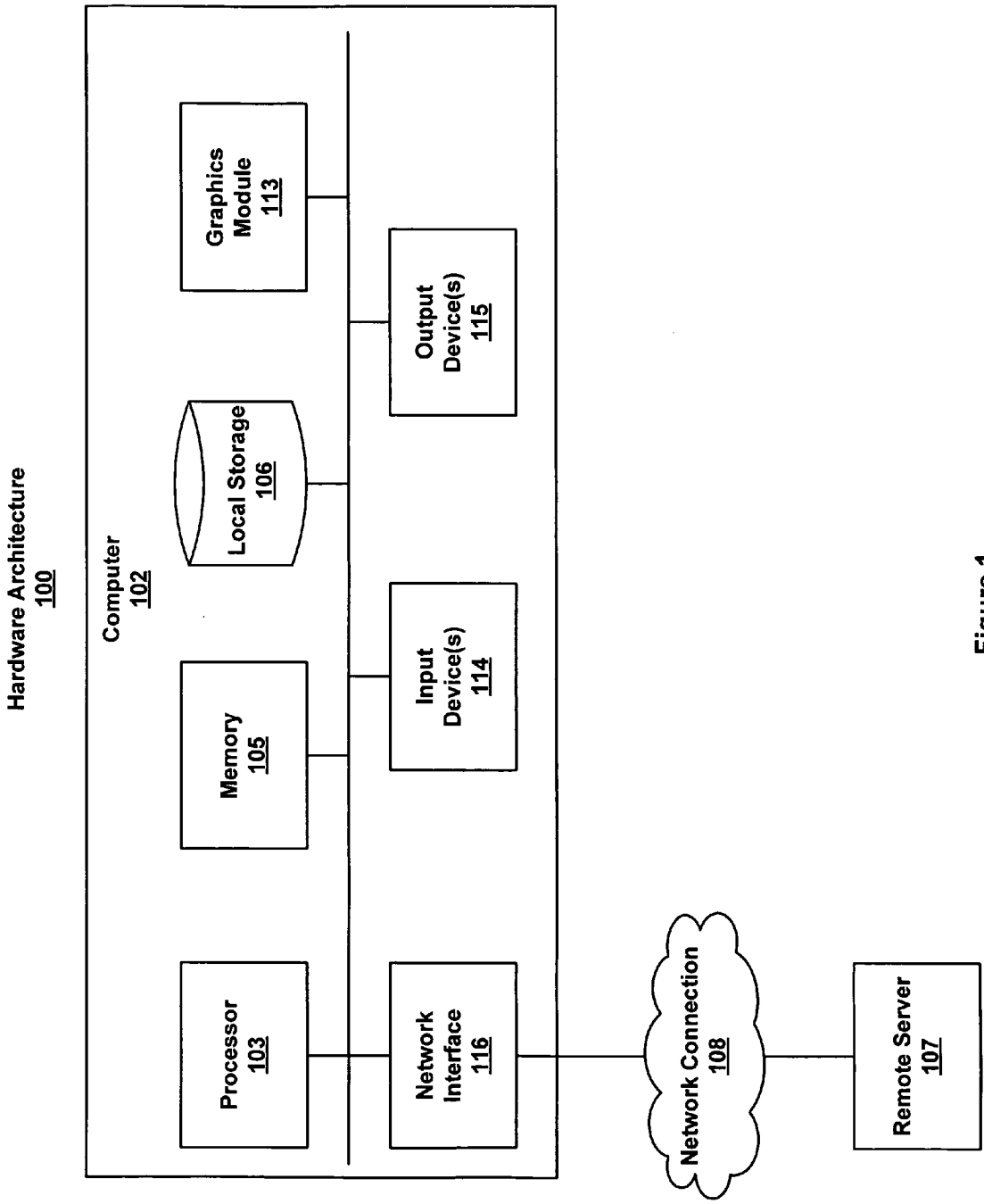

Hardware Architecture
100
Computer
102
Processor
103
Memory
105
Local Storage
106
Graphics
Module
113
Network
Interface
116
Input
Device(s)
114
Output
Device(s)
115
Network Connection
108
Remote Server
107

Figure 1

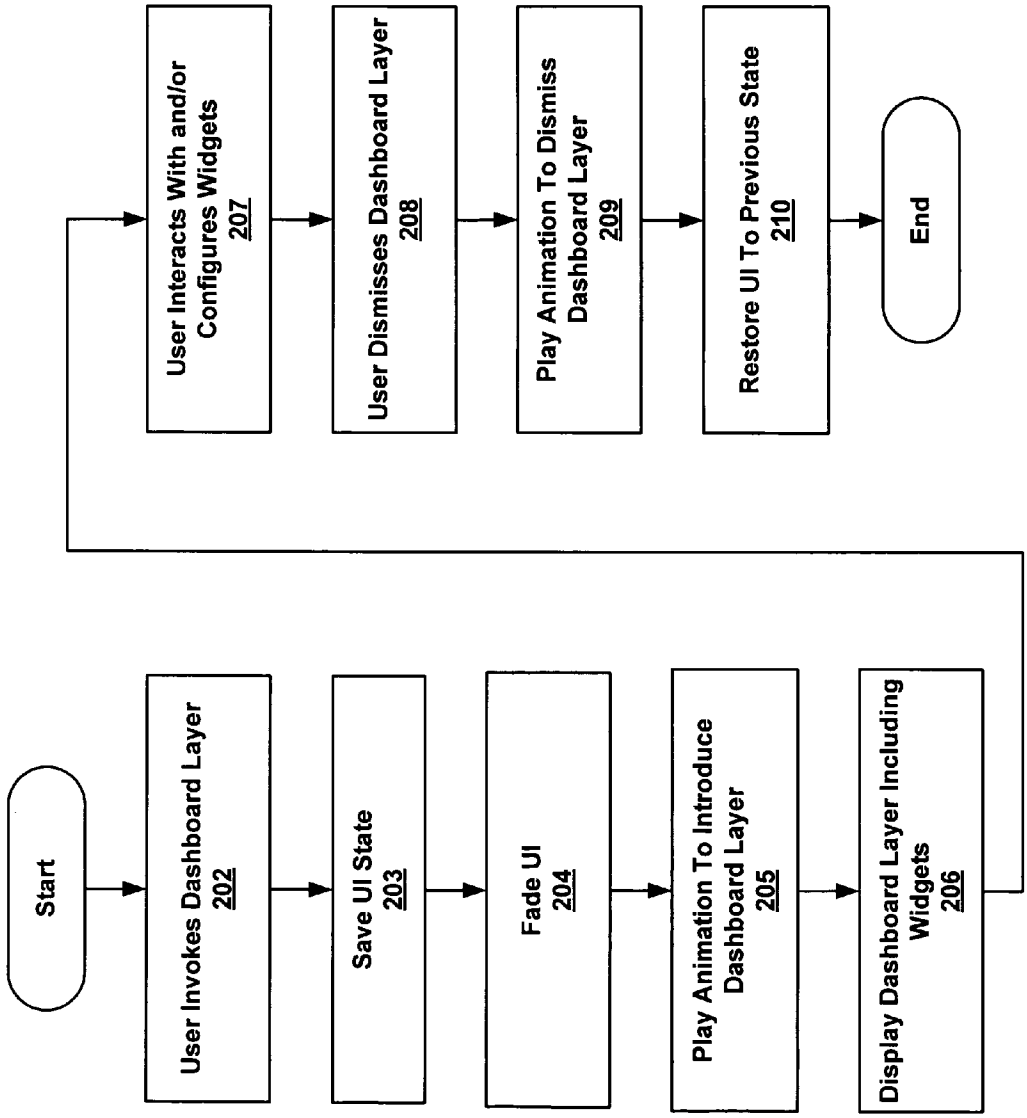

Start
User Invokes Dashboard Layer 202
Save UI State 203
Fade UI 204
Play Animation To Introduce Dashboard Layer 205
Display Dashboard Layer Including Widgets 206
User Interacts With and/or Configures Widgets 207
User Dismisses Dashboard Layer 208
Play Animation To Dismiss Dashboard Layer 209
Restore UI To Previous State 210
End

Figure 2

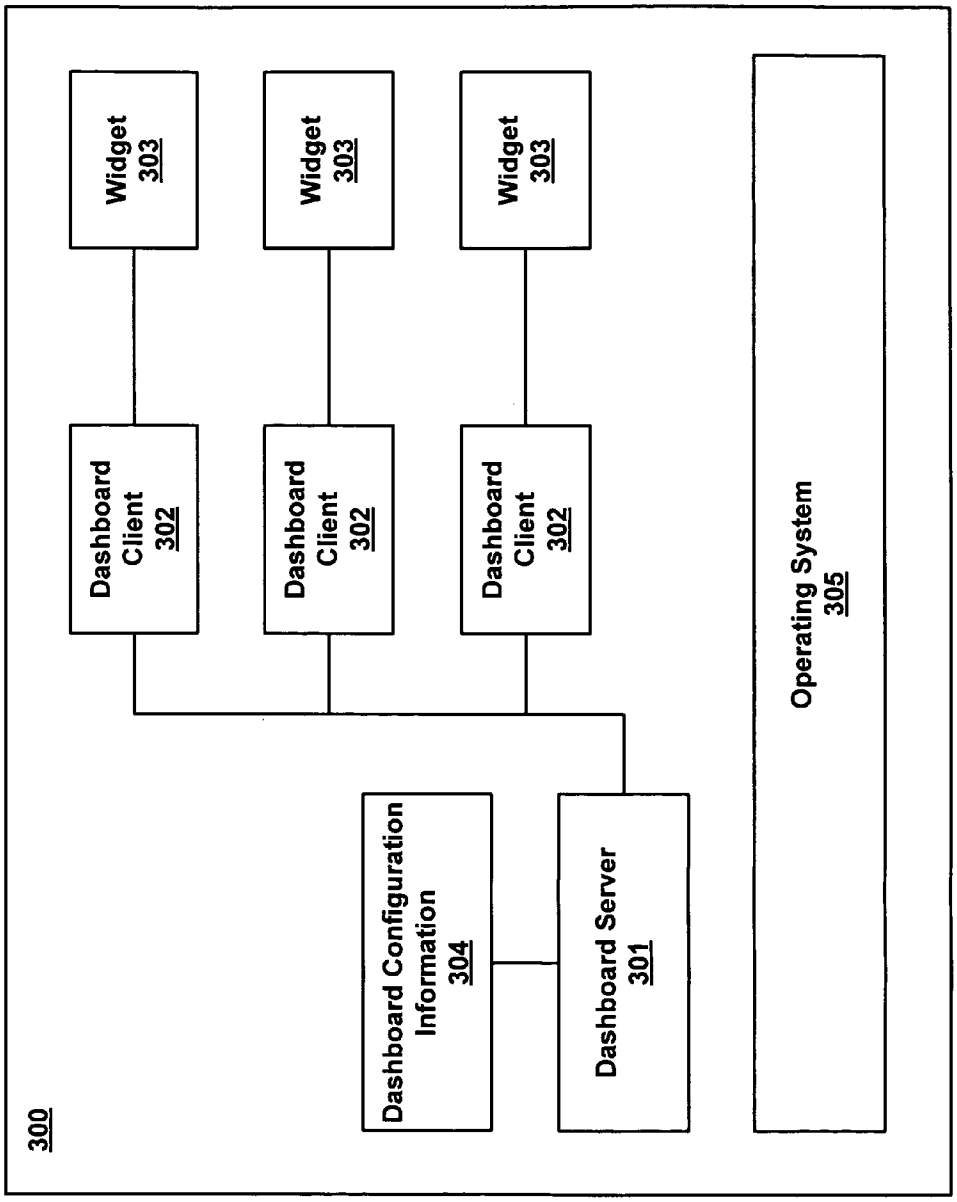

300
Widget 303
Widget 303
Widget 303
Dashboard Client 302
Dashboard Client 302
Dashboard Client 302
Dashboard Configuration Information 304
Dashboard Server 301
Operating System 305

Figure 3

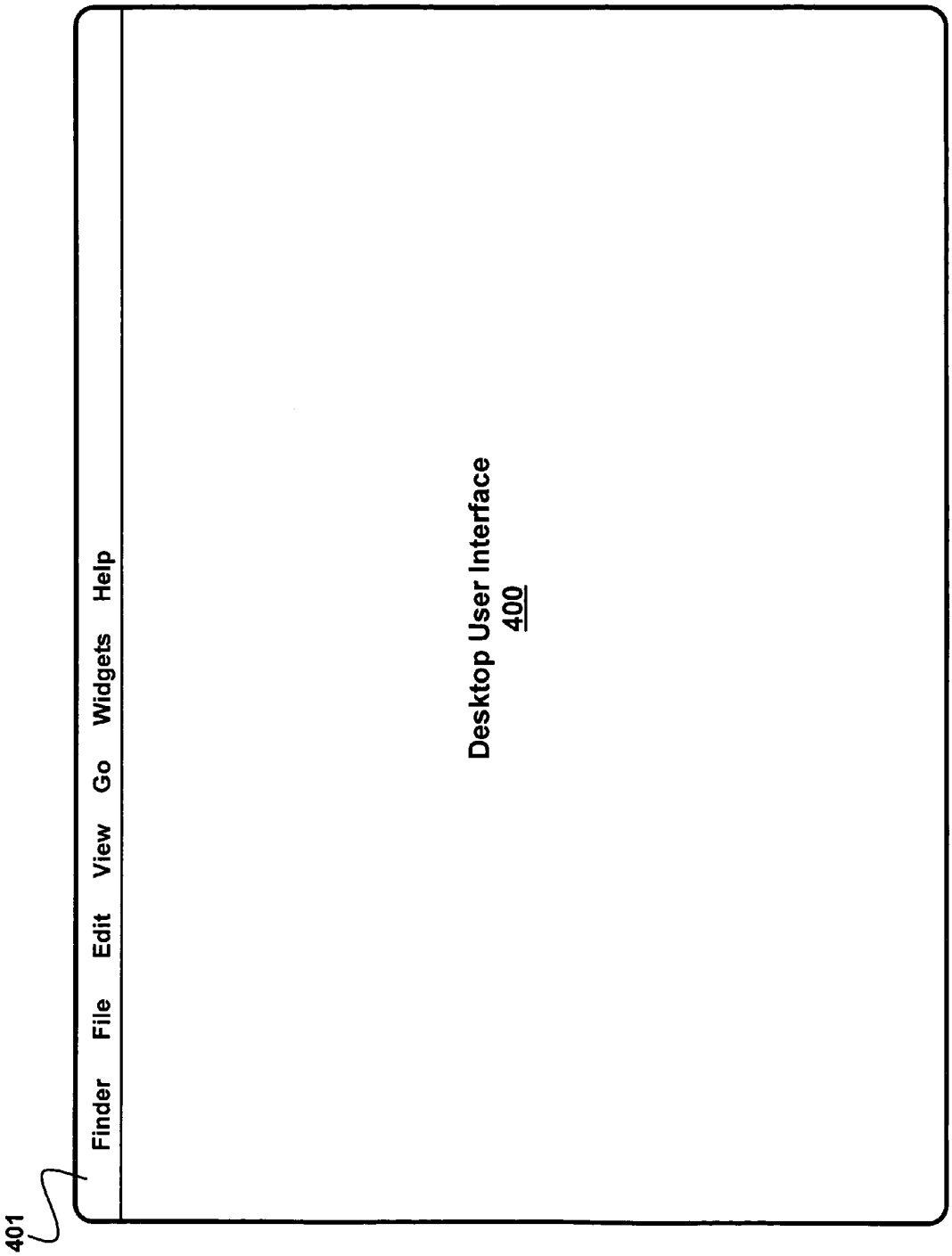

401
Finder
File
Edit
View
Go
Widgets
Help
Desktop User Interface
400

Figure 4A

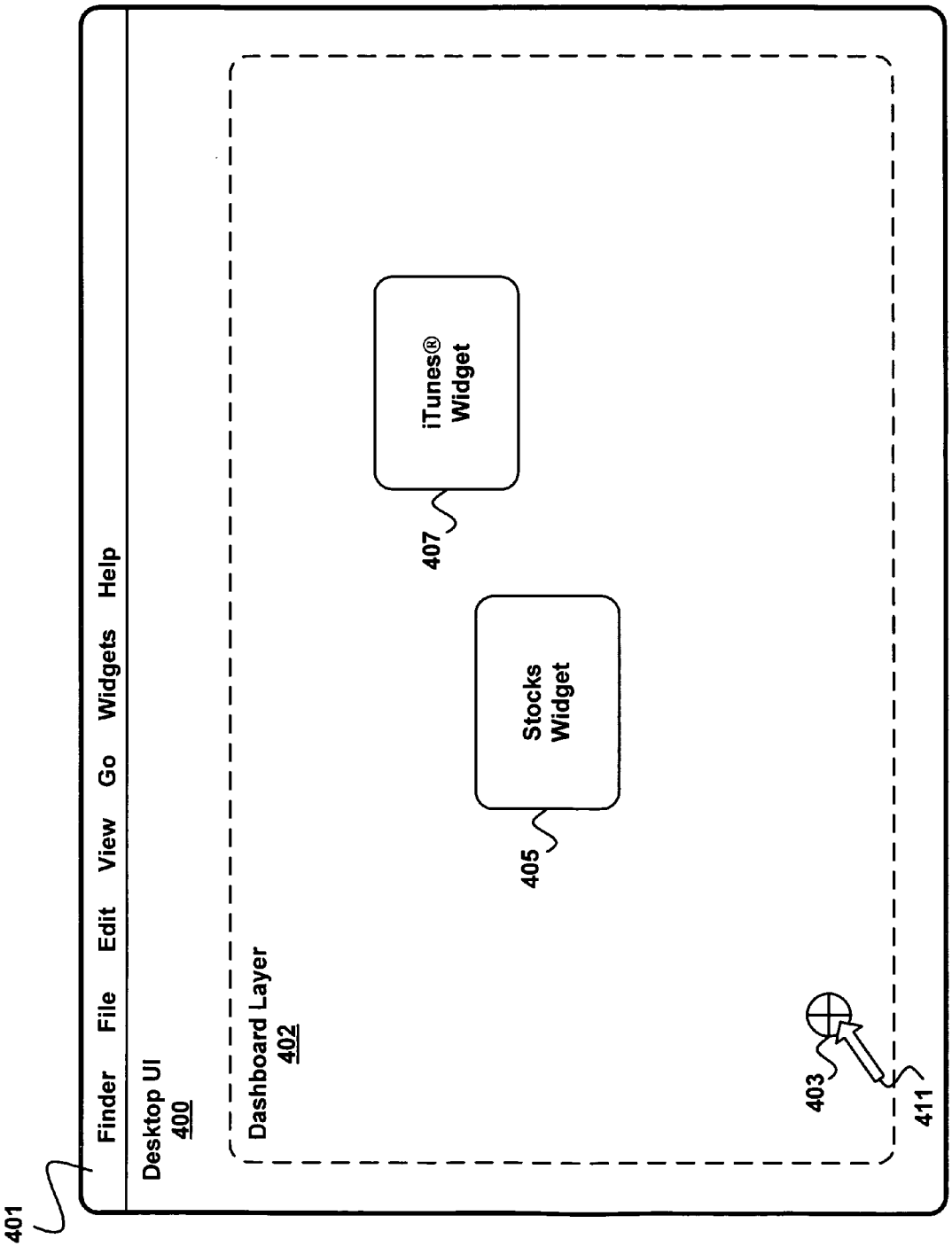

401
Finder File Edit View Go Widgets Help
Desktop UI
400
Dashboard Layer
402
iTunes® Widget
407
Stocks Widget
405
403
411

Figure 4B

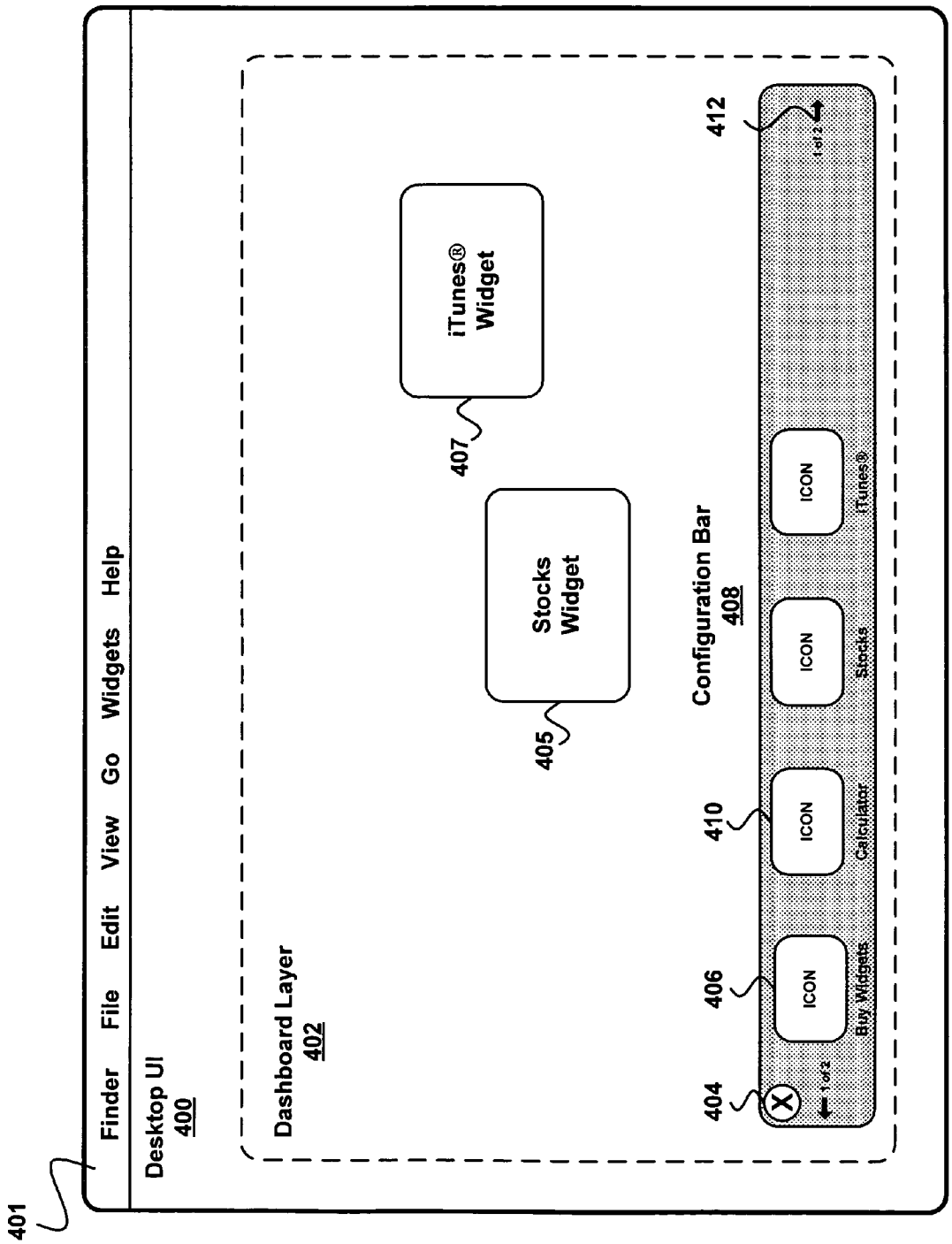

401
Finder File Edit View Go Widgets Help
Desktop UI
400
Dashboard Layer
402
iTunes® Widget
407
Stocks Widget
405
Configuration Bar
408
412
1 of 2
ICON
iTunes®
ICON
Stocks
ICON
Calculator
410
ICON
Buy Widgets
406
404
1 of 2

Figure 4C

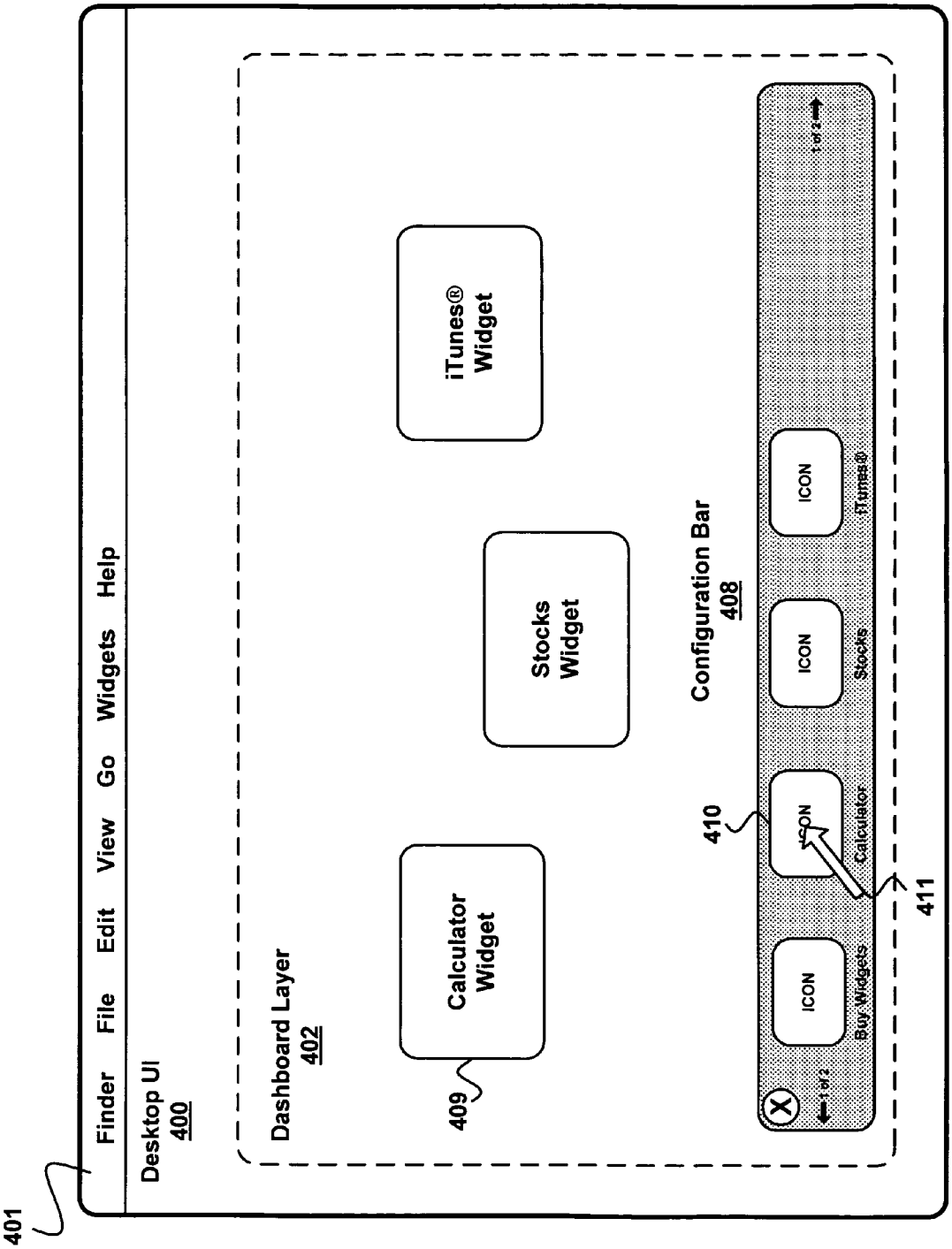

401
Finder File Edit View Go Widgets Help
Desktop UI
400
Dashboard Layer
402
Calculator Widget
409
Stocks Widget
iTunes® Widget
Configuration Bar
408
410
411
ICON
ICON
ICON
ICON
Buy Widgets
Calculator
Stocks
iTunes®
1 of 2
1 of 2

Figure 4D

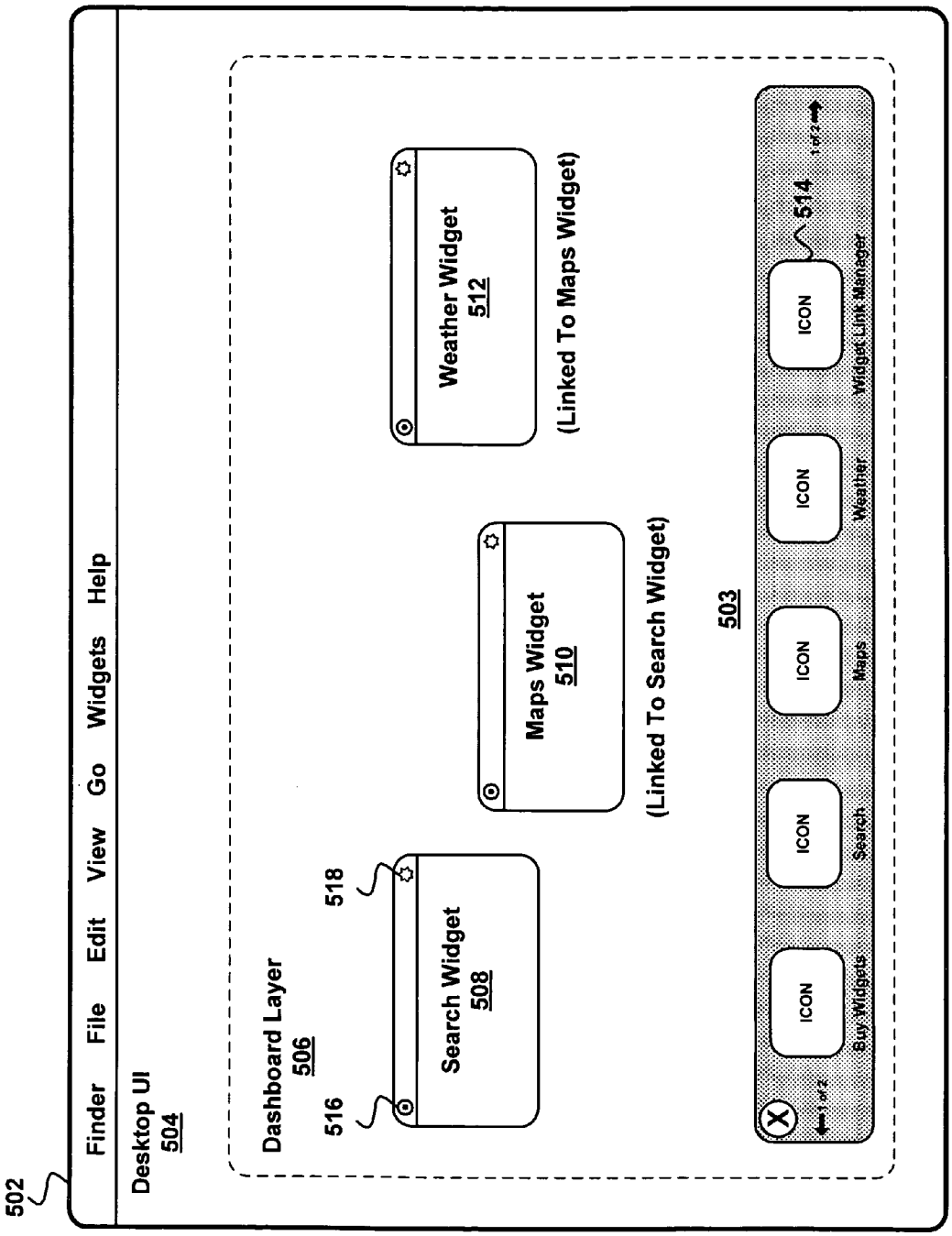

502
Finder File Edit View Go Widgets Help
Desktop UI
504
Dashboard Layer
506
516
518
Search Widget
508
Maps Widget
510
(Linked To Search Widget)
Weather Widget
512
(Linked To Maps Widget)
503
514
ICON
ICON
ICON
ICON
ICON
Buy Widgets
Search
Maps
Weather
Widget Link Manager

Figure 5A

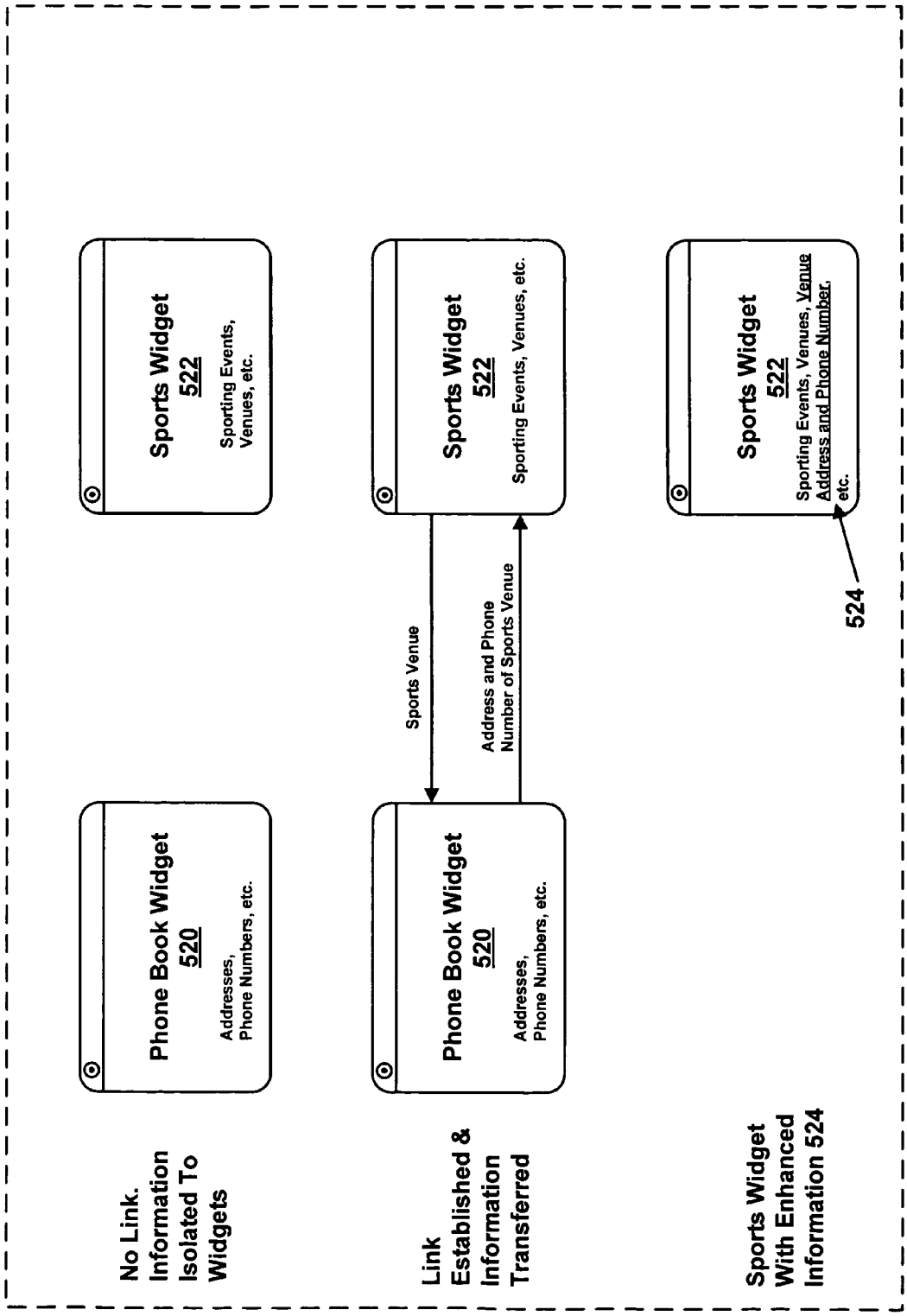

No Link. Information Isolated To Widgets
Phone Book Widget
520
Addresses, Phone Numbers, etc.
Sports Widget
522
Sporting Events, Venues, etc.
Link Established & Information Transferred
Phone Book Widget
520
Addresses, Phone Numbers, etc.
Sports Venue
Address and Phone Number of Sports Venue
Sports Widget
522
Sporting Events, Venues, etc.
Sports Widget With Enhanced Information 524
Sports Widget
522
Sporting Events, Venues, Venue Address and Phone Number, etc.
524

Figure 5B

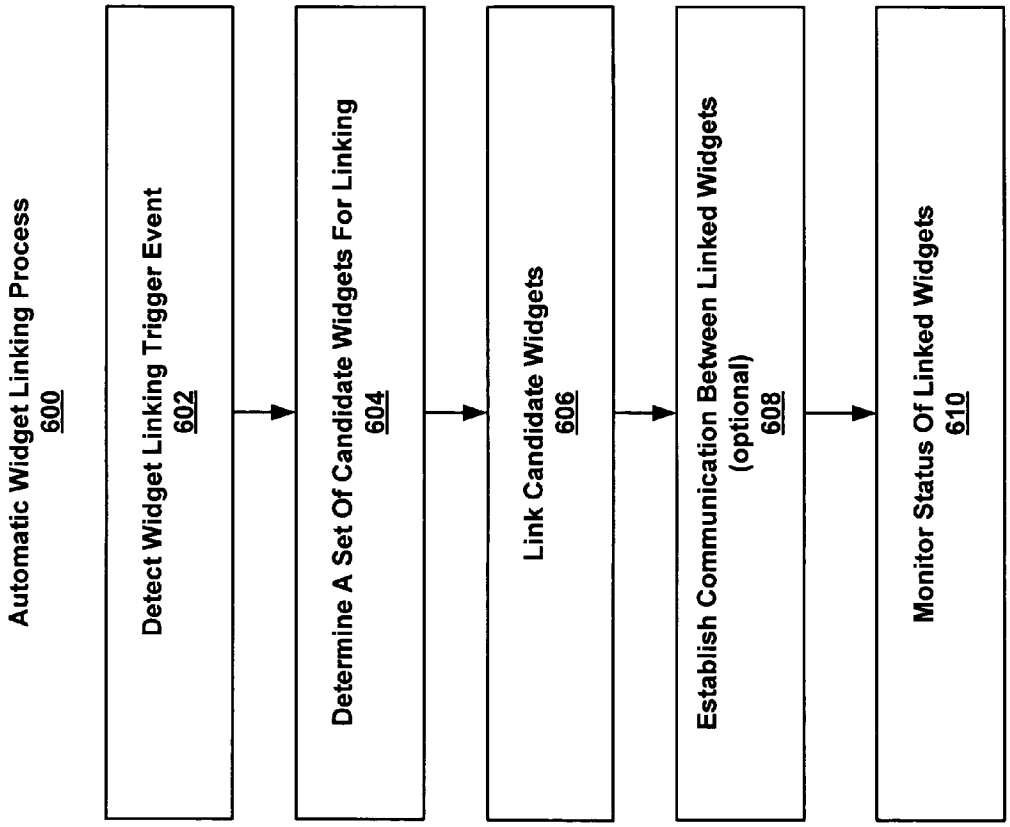

Automatic Widget Linking Process
600
Detect Widget Linking Trigger Event
602
Determine A Set Of Candidate Widgets For Linking
604
Link Candidate Widgets
606
Establish Communication Between Linked Widgets
(optional)
608
Monitor Status Of Linked Widgets
610

Figure 6

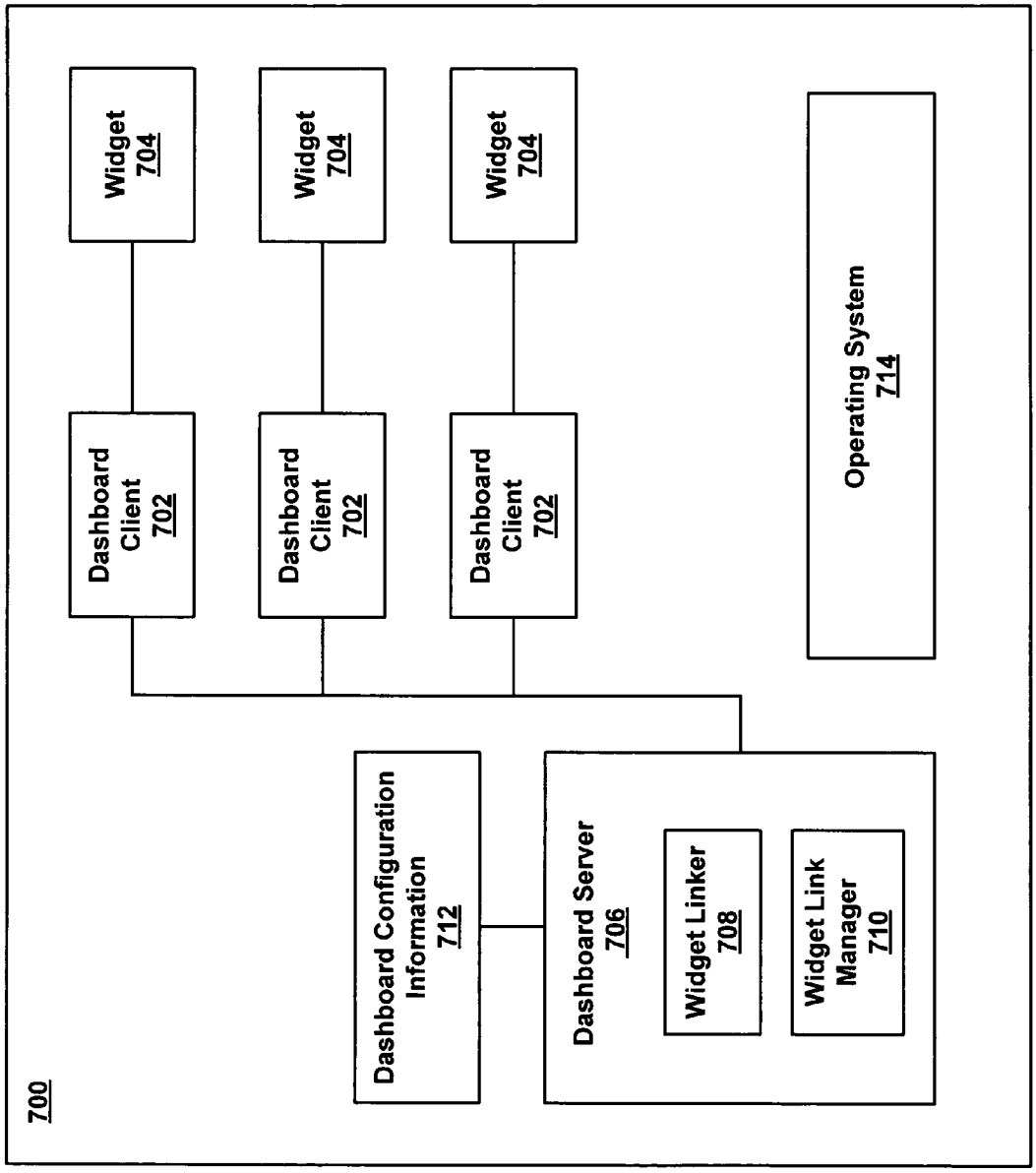

700
Widget 704
Widget 704
Widget 704
Dashboard Client 702
Dashboard Client 702
Dashboard Client 702
Operating System 714
Dashboard Configuration Information 712
Dashboard Server 706
Widget Linker 708
Widget Link Manager 710

Figure 7

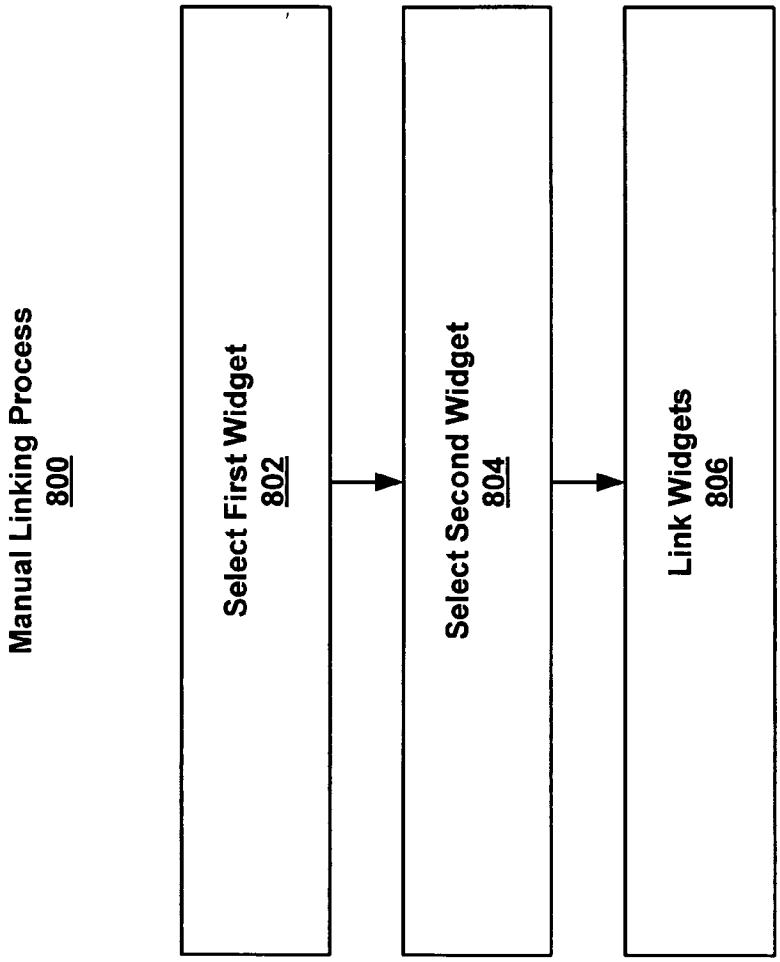

Manual Linking Process
800
Select First Widget
802
Select Second Widget
804
Link Widgets
806

Figure 8

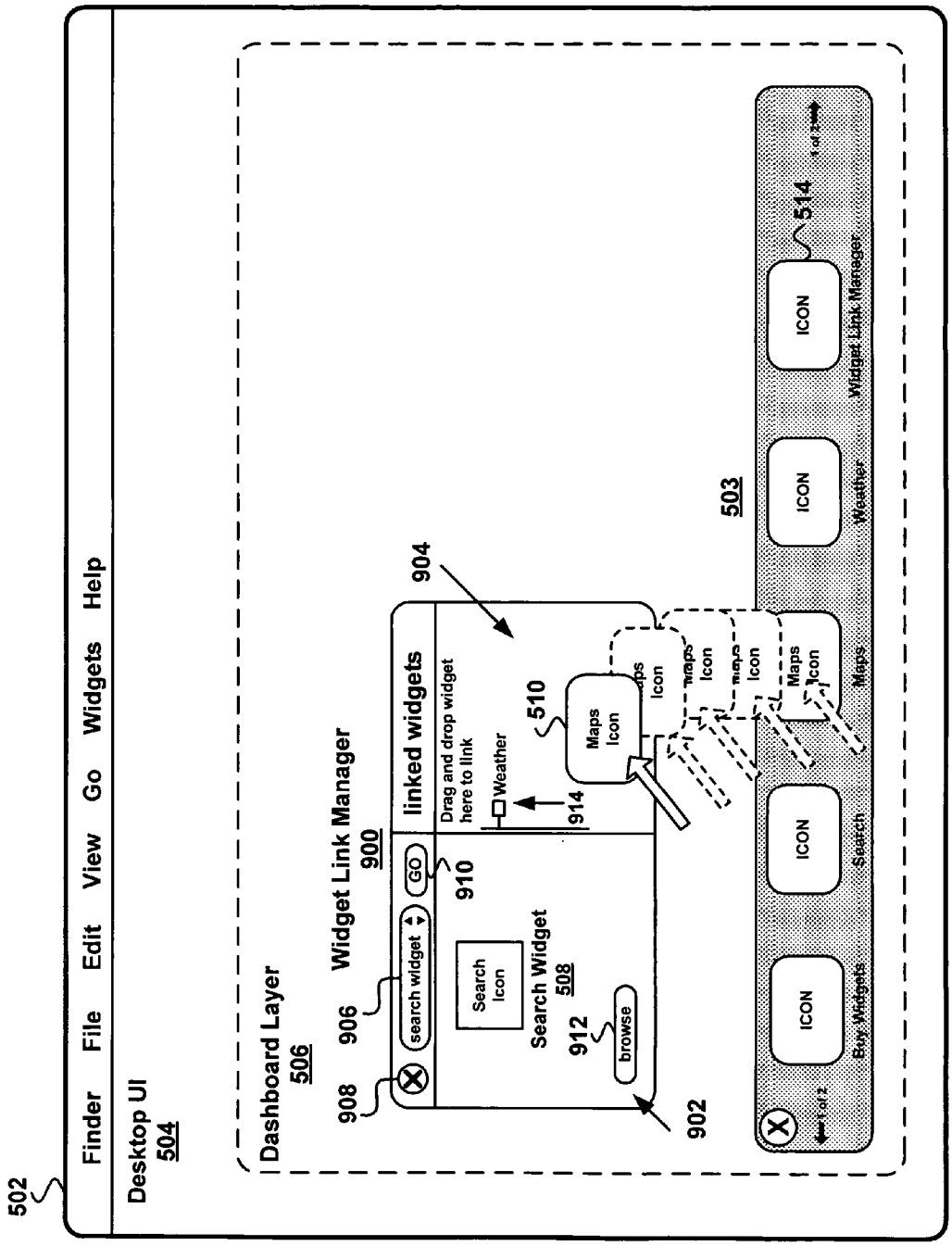

502
Finder File Edit View Go Widgets Help
Desktop UI
504
Dashboard Layer
506
Widget Link Manager
900
908
906
search widget
GO
910
linked widgets
Drag and drop widget here to link
Weather
914
904
Search Icon
Search Widget
508
912
browse
510
Maps Icon
902
503
ICON
514
Widget Link Manager
Weather
Maps
Search
Buy Widgets

Figure 9A

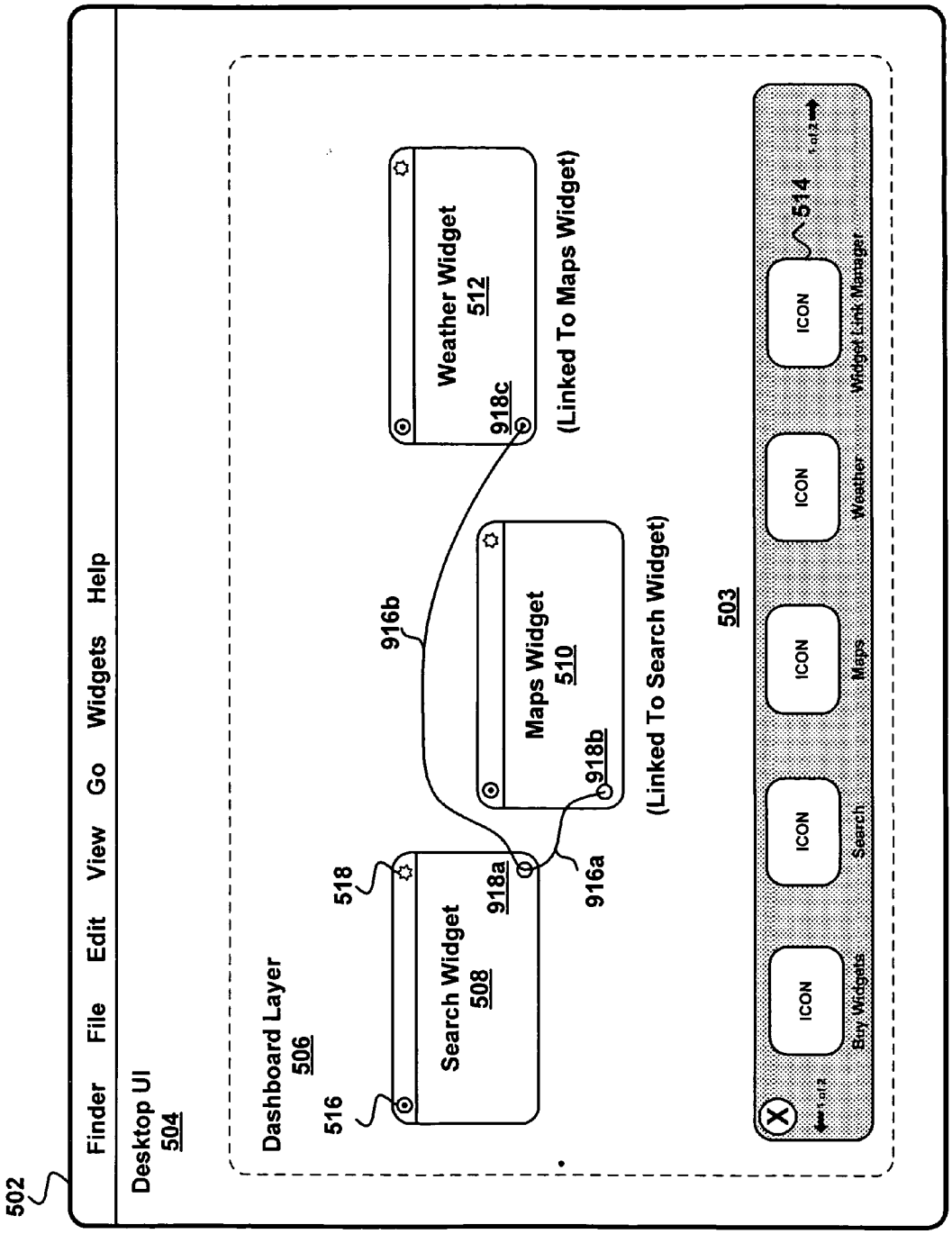

502
Finder File Edit View Go Widgets Help
Desktop UI
504
Dashboard Layer
506
516
518
Search Widget
508
918a
916a
916b
Maps Widget
510
918b
(Linked To Search Widget)
Weather Widget
512
918c
(Linked To Maps Widget)
503
ICON
ICON
ICON
ICON
514
Buy Widgets
Search
Maps
Weather
Widget Link Manager

Figure 9B

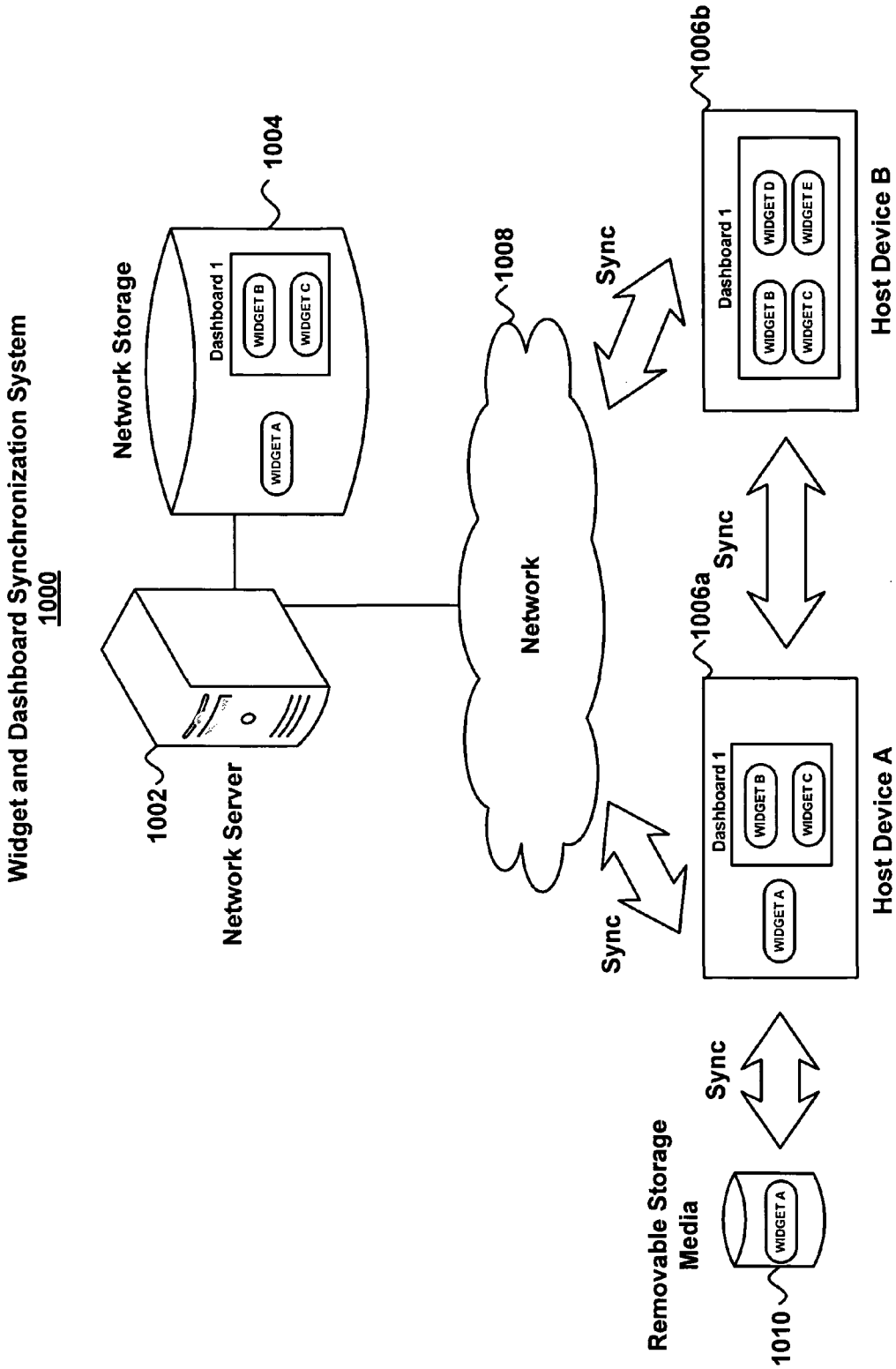

Widget and Dashboard Synchronization System
1000
1002
Network Server
Network Storage
1004
Dashboard 1
WIDGET A
WIDGET B
WIDGET C
Network
1008
Sync
Sync
Sync
Sync
1006a
Dashboard 1
WIDGET A
WIDGET B
WIDGET C
Host Device A
1006b
Dashboard 1
WIDGET B
WIDGET C
WIDGET D
WIDGET E
Host Device B
Removable Storage Media
1010
WIDGET A

Figure 10

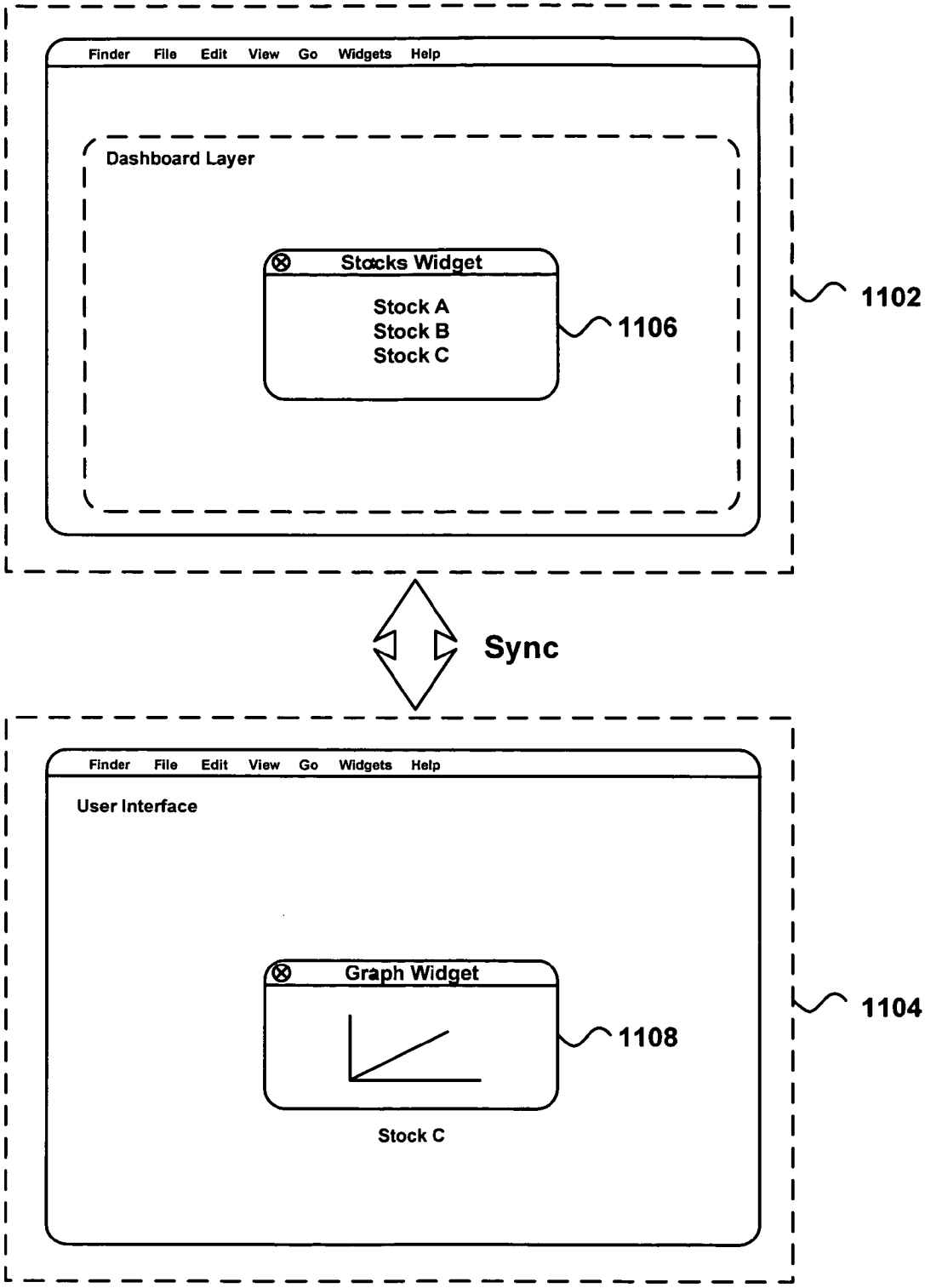

Finder File Edit View Go Widgets Help
Dashboard Layer
Stocks Widget
Stock A
Stock B
Stock C
1106
1102
Sync
Finder File Edit View Go Widgets Help
User Interface
Graph Widget
1108
Stock C
1104

Figure 11

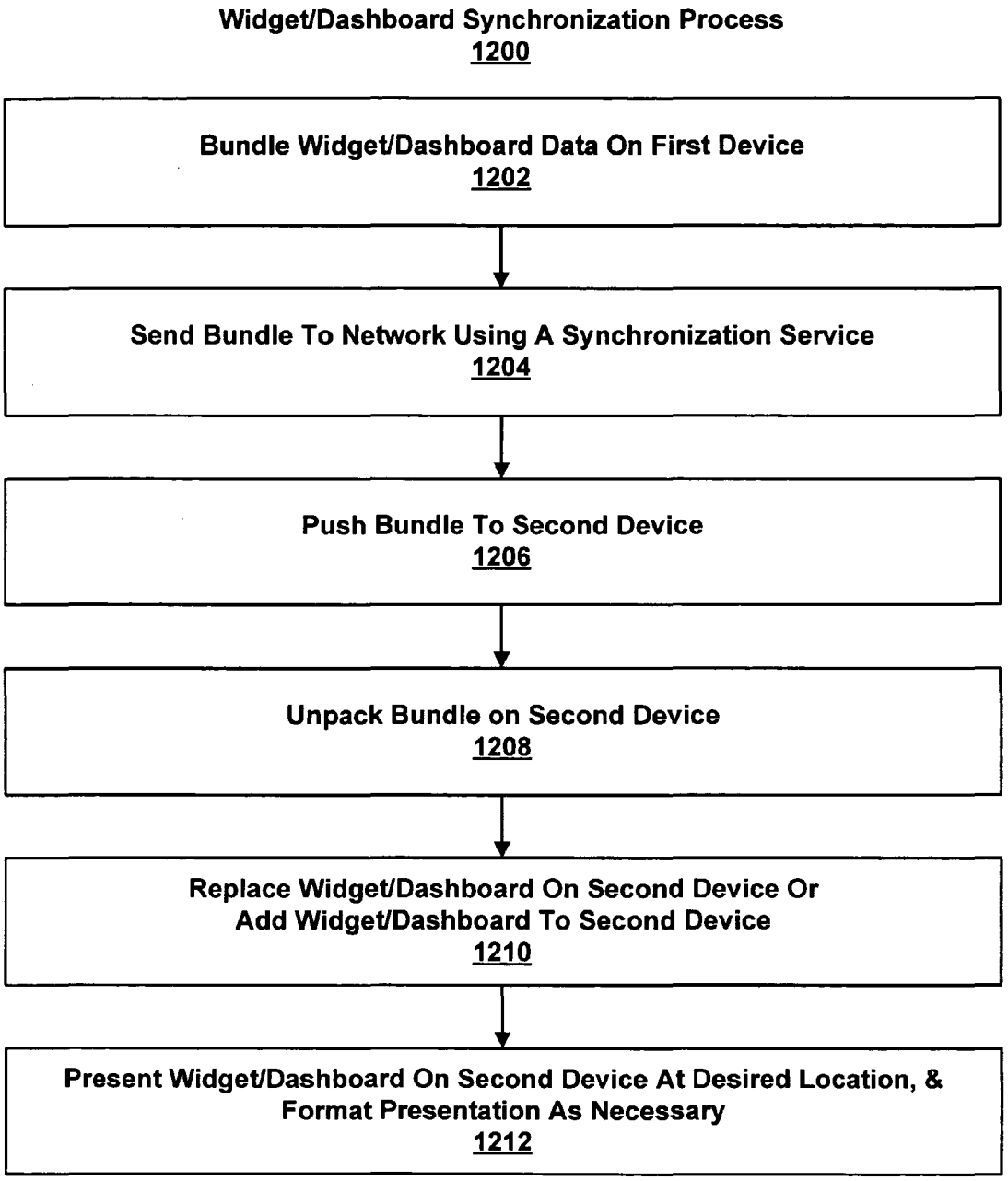

Widget/Dashboard Synchronization Process
1200
Bundle Widget/Dashboard Data On First Device
1202
Send Bundle To Network Using A Synchronization Service
1204
Push Bundle To Second Device
1206
Unpack Bundle on Second Device
1208
Replace Widget/Dashboard On Second Device Or
Add Widget/Dashboard To Second Device
1210
Present Widget/Dashboard On Second Device At Desired Location, &
Format Presentation As Necessary
1212

Figure 12

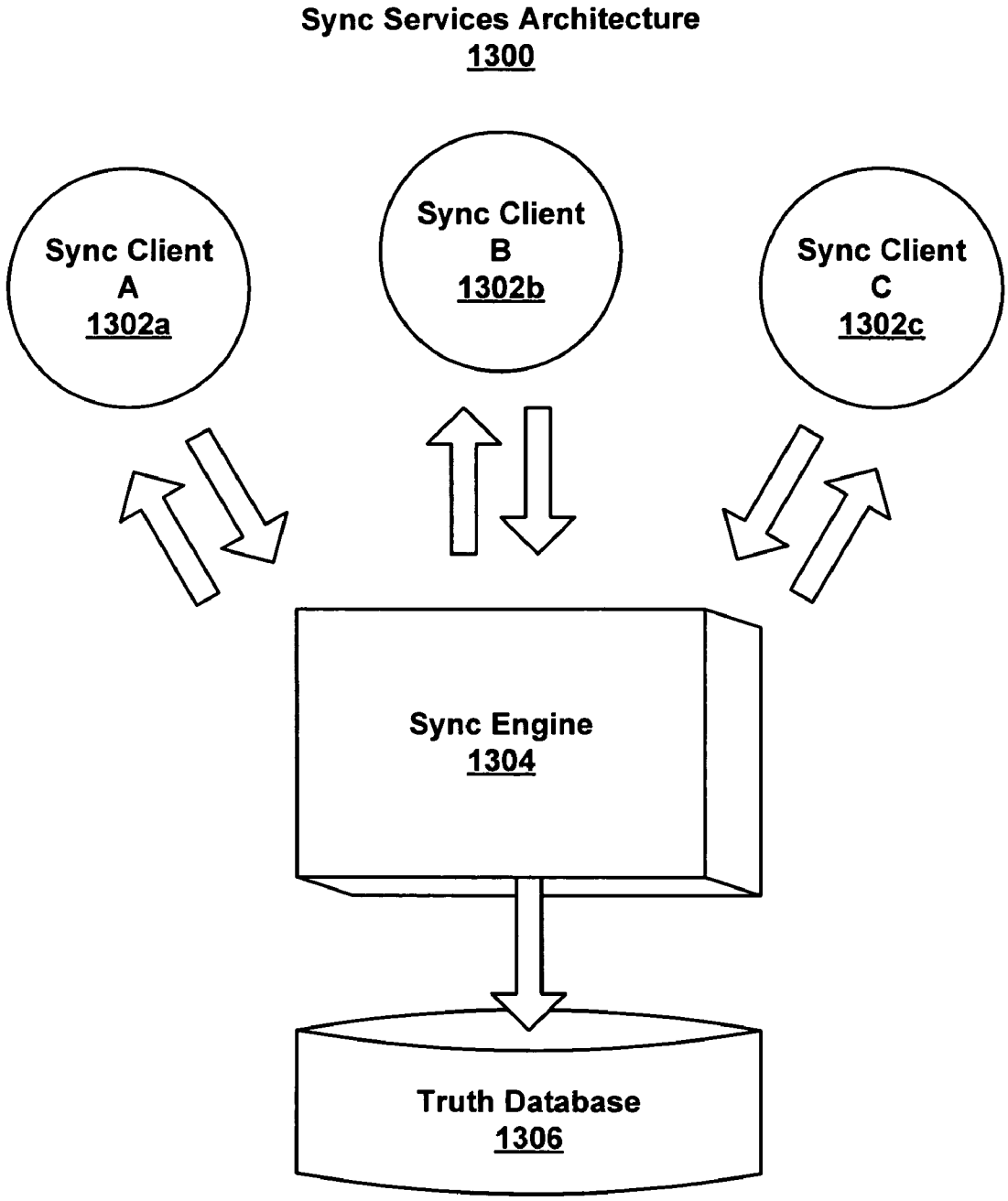

Sync Services Architecture
1300
Sync Client
A
1302a
Sync Client
B
1302b
Sync Client
C
1302c
Sync Engine
1304
Truth Database
1306

Figure 13

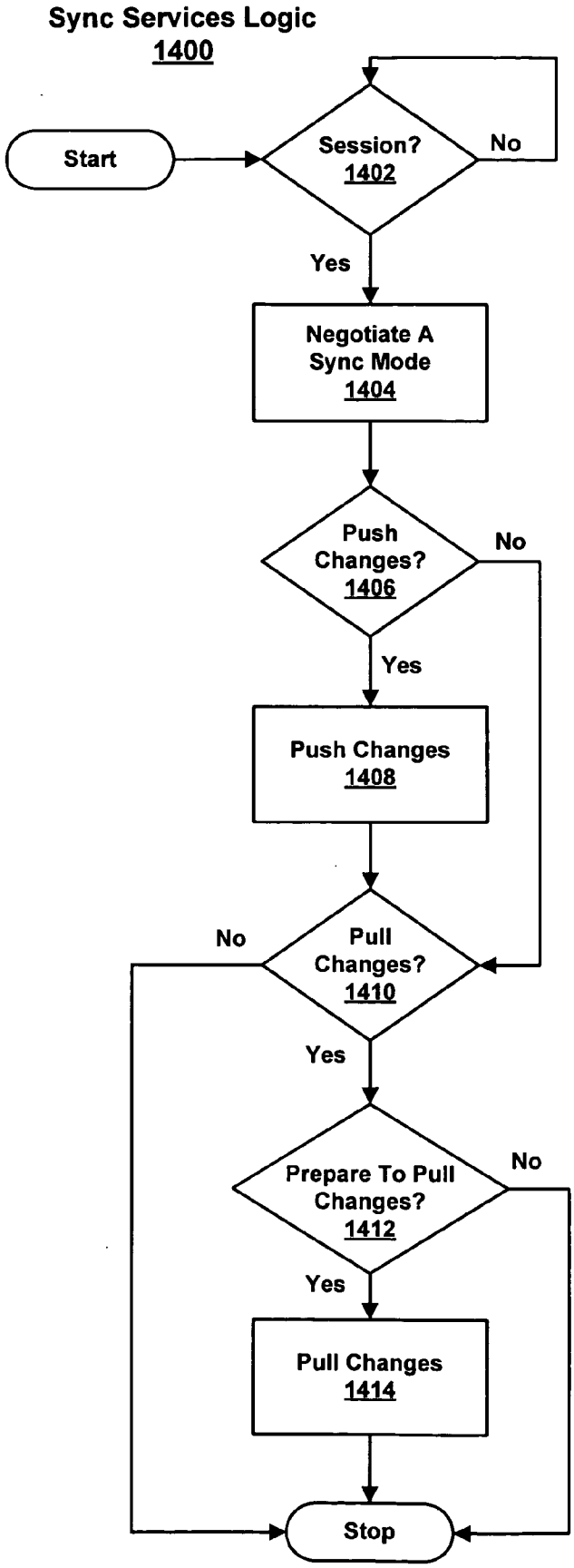

Sync Services Logic
1400
Start
Session?
1402
No
Yes
Negotiate A
Sync Mode
1404
Push
Changes?
1406
No
Yes
Push Changes
1408
No
Pull
Changes?
1410
Yes
Prepare To Pull
Changes?
1412
No
Yes
Pull Changes
1414
Stop

Figure 14

SYNCHRONIZATION OF WIDGETS AND DASHBOARDS

RELATED APPLICATIONS

[0001] This application is a continuation-in-part of U.S. patent application Ser. No. 10/877,968, for "Unified Interest Layer For User Interface," filed Jun. 25, 2004, which patent application is incorporated by reference herein in its entirety.

[0002] This application is related to the following jointly owned and co-pending patent applications, each incorporated herein by reference in its entirety:

[0003] U.S. Provisional Patent Application No. 60/583,125, for "Procedurally Expressing Graphic Objects for Web Pages," filed Jun. 25, 2004;

[0004] U.S. patent application Ser. No. 10/874,829, for "User Interface for Assisting in the Installation of an Asset," filed Jun. 22, 2004;

[0005] U.S. patent application Ser. No. 10/877,968, for "Unified Interest Layer For User Interface," filed Jun. 25, 2004;

[0006] U.S. patent application Ser. No. 11/145,561, for "Application Clipper," filed Jun. 3, 2005;

[0007] U.S. patent application Ser. No. 11/145,560, for "Web View Applications," filed Jun. 3, 2005;

[0008] U.S. patent application Ser. No. 11/145,023, for "Clip View Applications," filed Jun. 3, 2005;

[0009] U.S. patent application Ser. No. 11/148,010, for "Preview and Installation of User Interface Elements in a Display Environment," filed Jun. 7, 2005;

[0010] U.S. Provisional Patent Application No. 60/734,016, for "Preview Including Theme Based Installation of User Interface Elements In A Display Environment," filed Nov. 4, 2005, which provisional patent application is incorporated herein by reference in its entirety;

[0011] U.S. Provisional Patent Application No. 60/730,956, for "Widget Security," filed Oct. 27, 2005, which provisional application is incorporated herein by reference in its entirety;

[0012] U.S. patent application Ser. No. 11/282,110, for "Preview Including Theme Based Installation of User Interface Elements In A Display Environment," filed Nov. 16, 2005;

[0013] U.S. Provisional Patent Application No. 60/737,899, for "Management of User Interface Elements In A Display Environment," filed Nov. 18, 2005; and

[0014] U.S. patent application Ser. No. 11/346,603, for "Multiple Dashboards," filed Feb. 1, 2006;

[0015] U.S. patent application Ser. No. 11/403,644, for "Linked Widgets," filed Apr. 12, 2006; and

[0016] U.S. patent application Ser. No. _____, for "Management and Generation of Dashboards," filed Aug. 4, 2006, Attorney Docket No. P4210US1/18962-022001.

TECHNICAL FIELD

[0017] The disclosed implementations relate generally to graphical user interfaces.

BACKGROUND

[0018] A hallmark of modern graphical user interfaces is that they allow a large number of graphical objects or items to be displayed on a display screen at the same time. Leading personal computer operating systems, such as Apple Mac OS®, provide user interfaces in which a number of windows can be displayed, overlapped, resized, moved, configured, and reformatted according to the needs of the user or application. Taskbars, menus, virtual buttons and other user interface elements provide mechanisms for accessing and activating windows even when they are hidden behind other windows.

[0019] Although users appreciate interfaces that can present information on a screen via multiple windows, the result can be overwhelming. For example, users may find it difficult to navigate to a particular user interface element or to locate a desired element among a large number of onscreen elements. The problem is further compounded when user interfaces allow users to position elements in a desired arrangement, including overlapping, minimizing, maximizing, and the like. Although such flexibility may be useful to the user, it can result in a cluttered display screen. Having too many elements displayed on the screen can lead to "information overload," thus inhibiting the user to efficiently use the computer equipment.

[0020] Many of the deficiencies of conventional user interfaces can be reduced using "widgets." Generally, widgets are user interface elements that include information and one or more tools (e.g., applications) that let the user perform common tasks and provide fast access to information. Widgets can perform a variety of tasks, including without limitation, communicating with a remote server to provide information to the user (e.g., weather report), providing commonly needed functionality (e.g., a calculator), or acting as an information repository (e.g., a notebook). Widgets can be displayed and accessed through a user interface, such as a "dashboard layer," which is also referred to as a "dashboard." Widgets and dashboards are described in co-pending U.S. patent application Ser. No. 10/877,968, entitled "Unified Interest Layer For User Interface."

[0021] Due to the large number of widgets available to a user, a virtual desktop or dashboard can become cluttered and disorganized, making it difficult for the user to quickly locate and access a widget. Moreover, the user may have to invoke multiple widgets to perform a single task, which can lead to an inefficient use of widget resources. In some cases, the user may not readily recognize the relationship between two widgets, which leads to additional inefficiencies when using widgets.

SUMMARY

[0022] Systems, methods, computer-readable mediums, user interfaces and other implementations are disclosed for synchronizing widgets and dashboards.

[0023] In some implementations, a method comprises: receiving a first set of widget information associated with a first device; and synchronizing the first set of widget information with a data source.

[0024] In some implementations, a method, comprises: receiving a first set of dashboard information associated with

a first device; and synchronizing the first set of dashboard information with a data source.

[0025] In some implementations, a method comprises: identifying configuration information associated with one or more widgets; and updating the configuration information.

[0026] In some implementations, a system includes a first device, a second device and a sync engine. The first device includes a first dashboard including a first set of widgets. The second device includes a second dashboard including a second set of widgets. The sync engine is operatively coupled to the first and second devices and configurable to synchronize the first and second dashboards.

[0027] In some implementations, an apparatus includes a computer-readable medium adapted for storing a first set of dashboard information, and a sync engine operatively coupled to the computer-readable medium and configurable for synchronizing the first set of dashboard information with a data source.

[0028] Other implementations are disclosed which are directed to systems, methods, computer-readable mediums and user interfaces.

BRIEF DESCRIPTION OF THE DRAWINGS

[0029] FIG. **1** is a block diagram of a hardware architecture for implementing dashboards.

[0030] FIG. **2** is a flow diagram of a process for activating and using a dashboard.

[0031] FIG. **3** is a block diagram of a software architecture for implementing dashboards.

[0032] FIG. **4**A is a screen shot depicting a desktop user interface prior to activation of a dashboard.

[0033] FIG. **4**B is a screen shot depicting an initial state for a dashboard.

[0034] FIG. **4**C is a screen shot depicting a configuration bar for a dashboard.

[0035] FIG. **4**D is a screen shot depicting user selection of a widget from the configuration bar.

[0036] FIG. **5**A is a screen shot depicting an implementation of linked widgets.

[0037] FIG. **5**B illustrates the exchange of information between linked widgets.

[0038] FIG. **6** is a flow diagram of an exemplary automatic widget linking process.

[0039] FIG. **7** is a block diagram of an implementation of a software architecture for linked widgets.

[0040] FIG. **8** is a flow diagram of an implementation of a manual widget linking process.

[0041] FIG. **9**A is a screen shot depicting the manual linking of widgets using an exemplary widget link manager.

[0042] FIG. **9**B is a screen shot depicting the manual linking of widgets using bridging elements.

[0043] FIG. **10** is a block diagram of an exemplary synchronization system for widgets and dashboards.

[0044] FIG. **11** illustrates syncing of data between disparate widgets and dashboards.

[0045] FIG. **12** is a flow diagram of a exemplary widget/dashboard synchronization process.

[0046] FIG. **13** is an exemplary sync services architecture for syncing dashboards and widgets.

[0047] FIG. **14** is exemplary sync services logic for syncing dashboards and widgets.

DETAILED DESCRIPTION

Hardware Architecture

[0048] FIG. **1** is a block diagram of a hardware architecture **100** for synchronizing widgets and dashboards. The architecture **100** includes a personal computer **102** coupled to a remote server **107** via a network interface **116** and a network connection **108** (e.g., local area network, wireless network, Internet, intranet, etc.). The computer **102** generally includes a processor **103**, memory **105**, one or more input devices **114** (e.g., keyboard, mouse, etc.) and one or more output devices **115** (e.g., a display device). A user interacts with the architecture **100** via the input and output devices **114**, **115**.

[0049] The computer **102** also includes a local storage device **106** and a graphics module **113** (e.g., graphics card) for storing information and generating graphical objects, respectively. The local storage device **106** can be a computer-readable medium. The term "computer-readable medium" refers to any medium that participates in providing instructions to a processor for execution, including without limitation, non-volatile media (e.g., optical or magnetic disks), volatile media (e.g., memory) and transmission media. Transmission media includes, without limitation, coaxial cables, copper wire, fiber optics, and computer buses. Transmission media can also take the form of acoustic, light or radio frequency waves.

[0050] While widgets are described herein with respect to a personal computer **102**, it should be apparent that the disclosed implementations can be incorporated in, or integrated with, any electronic device that is capable of using widgets, including without limitation, portable and desktop computers, servers, electronics, media players, game devices, mobile phones, email devices, personal digital assistants (PDAs), televisions, etc.

[0051] A dashboard system and method for managing and displaying dashboards and widgets can be implemented as one or more plug-ins that are installed and run on the personal computer **102**. The plug-ins can be configured to interact with an operating system (e.g., MAC OS® X, WINDOWS XP, LINUX, etc.) and to perform the various dashboard and widget functions, as described with respect of FIGS. **2**-**8**. The dashboard system and method can also be implemented as one or more software applications running on a computer system (e.g., computer **102**). In some implementations, a dashboard system can be another widget that is configurable to communicate with other widgets, applications and/or operating systems. The dashboard system and method can also be characterized as a framework or model that can be implemented on various platforms and/or networks (e.g., client/server networks, stand-alone computers, portable electronic devices, mobile phones, etc.), and/or

embedded or bundled with one or more software applications (e.g., email, media player, browser, etc.).

[0052] For illustrative purposes, widgets (including linked widgets) are described as a feature of an operating system. Widgets, however, can be implemented in other contexts as well, including e-mail environments, desktop environments, application environments, hand-held displays, and any other display devices.

Dashboard Overview

[0053] FIG. **2** is a flow diagram of an implementation of a process for activating and using one or more dashboard layers. A dashboard layer (also referred to herein as a "unified interest layer" or "dashboard") is used to manage and display widgets (including linked widgets). A user can invoke a dashboard (**202**) by hitting a designated function key or key combination, or by clicking on an icon, or by selecting a command from an onscreen menu, or by moving an onscreen cursor to a designated corner of the screen. Alternatively, a dashboard layer can be invoked programmatically by another system, such as an application or an operating system, etc.

[0054] In response to such invocation, the current state of the user interface is saved (**203**), the user interface is temporarily inactivated (**204**), an animation or effect is played or presented to introduce the dashboard (**205**) and the dashboard is displayed with one or more widgets (**206**). If applicable, a previous state of the dashboard is retrieved, so that the dashboard can be displayed in its previous configuration.

[0055] In some implementations, the dashboard is overlaid on an existing user interface (UI) (e.g., a desktop UI). When the dashboard is activated, the existing UI may be faded, darkened, brightened, blurred, distorted, or otherwise altered to emphasize that it is temporarily inactivated. The existing UI may or may not be visible behind the dashboard. The UI can also be shrunk to a small portion of the display screen while the dashboard is active, and can be re-activated by clicking on it. In some implementations, the UI is shrunk and presented as a widget. The UI can be re-activated by clicking on the widget. In some implementations the UI remains active when the dashboard is active.

[0056] The user interacts with and/or configures widgets as desired (**207**). In some implementations, the user can move widgets around the screen, and can resize widgets if applicable. Some widgets are resizable and some have a fixed size. A widget author can specify whether a widget can be resized. Some widgets automatically resize themselves based on the amount or nature of the data being displayed. Widgets can overlap and or repel one another. For example, if the user attempts to move one widget to a screen position occupied by another widget, one of the widgets is automatically moved out of the way or repelled by the other widget. In some implementations, a user can link widgets together using a widget link manager, as described with respect to FIG. **8**.

[0057] In some implementations, the user dismisses the dashboard (**208**) by invoking a dismissal command, which causes the UI layer to return or re-present itself to the display screen. In some implementations, the dashboard is dismissed when the user presses a function key or key combination (which may be the same or different than the key or combination used to activate the dashboard), or clicks on a close box or other icon, or clicks on negative space within the dashboard (e.g., a space between widgets), or moves an onscreen cursor to a predefined corner of the screen. Other dismissal methods are possible.

[0058] In some implementations, the dashboard is automatically dismissed (i.e., without user input) after some predetermined period of time or in response to a trigger event. An animation or other effect can be played or presented to provide a transition as the dashboard is dismissed (**209**). When the dashboard is dismissed, the current configuration or state of the widgets (e.g., position, size, etc.) is stored, so that it can be retrieved the next time the dashboard is activated. In some implementations, an animation or effect is played or presented when re-introducing the UI. The UI is restored to its previous state (**210**) so that the user can resume interaction with software applications and/or the operating system.

[0059] In some implementations, the dashboard is configurable. The user can select a number of widgets to be displayed, for example, by dragging the widgets from a configuration bar (or other user interface element) onto the dashboard. The configuration bar can include different types of widgets, and can be categorized and/or hierarchically organized. In some implementations, in response to the user dragging a widget onto the configuration bar, the widget is downloaded from a server and automatically installed (if not previously installed). In some implementations, certain widgets can be purchased, so the user is requested to provide a credit card number or some other form of payment before the widget is installed on the user's device. In some implementations, widgets are already installed on the user's device, but are only made visible when they have been dragged from the configuration bar onto the dashboard. The configuration bar is merely an example of one type of UI element for configuring the dashboard. Other configuration mechanisms can be used, such as an icon tray or menu system.

[0060] It should be apparent that there are many ways in which dashboards and widgets can be displayed other than those implementations described herein. For example, widgets can be displayed on any surface, user interface or user interface element, including but not limited to desktops, browser or application windows, menu systems, trays, multi-touch sensitive displays and other widgets.

Software Architecture

[0061] FIG. **3** is a block diagram of a software architecture **300** for implementing dashboards for installing, displaying and launching linked widgets. The software architecture **300** generally includes a dashboard server **301**, one or more dashboard clients **302**, and one or more widgets **303**. The server **301** and/or clients **302** use dashboard configuration information **304** to specify configuration options for displaying the widgets **303**, including access levels, linking information and the like (if applicable). Such configuration information can include information for two or more dashboards configured by the same user or by different users.

[0062] In some implementations, the widgets **303** are displayed using HTML and related web technology. The dashboard server **301** manages and launches the dashboard client **302** processes. Each dashboard client **302** loads a widget **303** (e.g., an HTML webpage) and related resources needed to display the page. In some implementations, the dashboard clients **302** display the widgets **303** without a conventional window frame, menu bar, or other components typically associated with on-screen windows. This technique provides a clean, straightforward display of the overall dashboard to reduce confusion and clutter. The dashboard clients **302** display their respective widgets **303** by rendering web pages into a "WebView," as described in U.S. patent application Ser. No. 11/148,010, entitled "Preview and Installation of User Interface Elements in a Display Environment." The size of each WebView can be defined as metadata associated with the corresponding widget **303**. The server **301** provides data for rendering the dashboard layer that can be overlaid on a desktop user interface. In some implementations, the widgets **303** are rendered into the dashboard layer, which is drawn on top of the desktop user interface, so as to partially or completely obscure the desktop user interface while the dashboard layer is active.

Dashboard Server

[0063] The dashboard server **301** (also referred to as "server") can be a stand-alone process or embedded in another process. The server **301** can be located at the computer **102** or at the remote server **107**. In some implementations, the server **301** provides functionality for one or more processes, including but not limited to: non-widget UI management, window management, fast login, event management, loading widgets, widget arbitration, Core Image integration and widget preference management, as described in U.S. patent application Ser. No. 11/148,010, entitled "Preview and Installation of User Interface Elements in a Display Environment."

Dashboard Client

[0064] In some implementations, a dashboard client **302** is a process that uses, for example, objects that are defined as part of a development environment, such as Apple Computer's Cocoa Application Framework (also referred to as the Application Kit, or AppKit) for the Mac OS® operating system. In some implementations, the dashboard clients **302** can be implemented as simplified browser screens that omit conventional interface features such as a menu bar, window frame, and the like.

Widget Format

[0065] In one implementation, each widget **303** is implemented as an HTML file. The HTML file can reference other local and remote resources such as style sheets (e.g., Cascading Style Sheets), other HTML files, JavaScript files, images, and the like. Widgets **303** can be implemented using, for example, a flat bundle file format or a packaged HTML file format. In some implementations, the Flat Bundle format includes an info.plist file.

[0066] The Info.plist files describes a widget **303** and provides an identifier for a widget **303**. Table I provides an example of Info.plist file contents.

TABLE I

Example of Info.plist File Contents

Key	Type	Description/Value
CFBundleIdentifier	CFString	com.apple.widget <widget name>
CFBundleName	CFString	Name of the widget.
MainHTML	CFString	Name of main HTML resource.
Width	CFNumber	Default width of the widget.
Height	CFNumber	Default height of the widget.
DefaultImage	CFString	Resource name of default PNG file.
Plugin (optional)	CFString	Resource name of native plug-in.
AllowFileAccessOutsideofWidget	Boolean	Access to files across the file system; limited by the users permissions.
AllowFullAccess	Boolean	Access to the file system, Web Kit and standard browser plug-ins, Java applets, network resources, and command-line utilities.
AllowInternetPlugins	Boolean	Access to Web Kit and standard browser plug-ins.
AllowJava	Boolean	Access to Java applets.
AllowNetworkAccess	Boolean	Access to any resources that are not file based.
AllowSystem	Boolean	Access to command-line utilities using widget script object.
WidgetLinkInfo	CFString1 . . . CFStringN	Names of widgets that can be linked

[0067] The keys AllowFileAccessOutsideofWidget, AllowFullAccess AllowInternetPlugins, AllowJava, AllowNetworkAccess, and AllowSystem are Boolean types that can be set by a widget author to enable certain levels of resource access.

[0068] In some implementations, the Info.plist file includes N WidgetLinkInfo strings, for storing the names of widgets that can be linked to the widget associated with the Info.plist file. This information can be used to automatically link widgets, as described with respect to FIGS. **5-8**. Note that the additional widget link information can be included in the Info.plist file and/or one or more other files bundled with the widget, depending upon the widget design.

Dashboard Invocation

[0069] FIG. **4**A depicts a desktop user interface **400** prior to activation of a dashboard. The desktop user interface **400** (also referred to herein as "desktop") is a conventional user interface as may be provided by an operating system, such as Mac OS®. The desktop **400** has a background image, menu bar **401**, and other standard features. As is known in the art, the desktop **400** may also include windows, icons, and other elements (not shown). The user activates the dashboard by selecting an item from a menu, or by clicking

on an icon, or by pressing a function key or key combination, or by some other means for invoking activation. A dashboard does not have to be activated on a desktop; rather the dashboard can be activated and displayed on any display screen with or without a desktop.

[0070] FIG. 4B depicts an initial state for a dashboard layer **402**. In some implementations, a configuration bar icon **403** is initially displayed. Alternatively, upon activation the dashboard layer **402** can display one or more default widgets **405**, **407**. If the dashboard layer **402** has previously been activated and configured, the widgets **405**, **407**, can be displayed as previously configured. The dashboard layer **402** is not necessarily visible as a distinct layer. However, its various components (such as widgets, icons, and other features) are visible. In some implementations, these components are displayed in a transparent layer, thus maintaining the visibility of the desktop **400** to the user. In some implementations, the desktop **400** and its components are darkened (or blurred, or otherwise visually modified) while the dashboard layer **402** is active, so as to emphasize that the desktop **400** is temporarily inactive. In other implementations, the desktop **400** is not visible while the dashboard layer **402** is active. The user can reactivate the desktop **400** and dismiss the dashboard layer **402** by, for example, selecting on an area of the screen where no dashboard element is displayed (i.e., "negative space"). In some implementations, other commands, key combinations, icons, or other user input can be used to dismiss the dashboard layer **402**.

[0071] In some implementations, the user can drag the icon **403** to any location on the screen, and the position of the icon **403** will remain persistent from one invocation of the dashboard layer **402** to the next. The user can click on the icon **403** to activate the configuration bar **408**, as shown in FIG. 4C. The configuration bar **408** provides access to various widgets that can be placed on the dashboard. In some implementations, a text label is shown for each available widget (e.g., calculator, stocks, iTunes®, etc.). In some implementations, an icon is shown for each available widget (e.g., calculator icon **410**). If many widgets are available, the widgets may be arranged hierarchically by type (e.g., game widgets, utility widgets, etc.), or alphabetically, or by any other categorization methodology. For example, a number of categories may be displayed, and clicking on one of the categories causes a pull-down menu to be displayed, listing a number of widgets in that category. In some implementations, a buy widget **406** is also available, allowing the user to select widgets from an online store or website.

[0072] Note that the particular configuration and appearance of configuration bar **408** in FIG. 4C is merely exemplary, and that many other arrangements are possible. For example, widgets can be installed from other locations, other applications or other environments, without requiring that they first be part of the configuration bar **408**. The user can dismiss the configuration bar **408** by clicking on dismissal button or icon **404**.

Installation of Elements

[0073] Elements, including user interface elements such as widgets can be displayed as discussed below. One display, a dashboard layer, will be used for illustrative purposes. Installation can include a preview operation as is discussed below. Installation can include selection of the element, such as by a drag and drop action. Other selection means can be used. In one example, a user can drag widgets from configuration bar **408** onto the surface of the dashboard (in other words, anywhere on the screen), using standard drag-and-drop functionality for moving objects on a screen.

[0074] FIG. 4D depicts the selection of the calculator widget icon **410** from the configuration bar **408**. The calculator icon **410** which is associated with a calculator widget **409** is highlighted, or otherwise augmented or embellished, to indicate that it has been selected by a user with cursor **411**.

[0075] In some implementations, widgets in the configuration bar **408** are smaller than their actual size when installed. When the user clicks on a widget and begins to drag it into a dashboard or other display, the widget can be animated to its actual or installed size to assist the user in the real-time layout of the dashboard. By animating the widget to its actual size, the user will know the actual size of the widget prior to its installation.

[0076] In some implementations, an animation, such as a ripple animation, is shown when the user "drops" a widget by releasing a mouse button (or equivalent input device) to place a widget at the desired location. In one implementation, the dragging of the widget to the dashboard layer **402** invokes an installation process for installing the widget including previewing. After installation, the user can move a widget, to any other desired location, or can remove the widget from the screen, for example by dragging it off the screen, or dragging it back onto the configuration bar **408**, by invoking a remove command, disabling a widget in a menu associated with a widget manager or canceling the installation during the preview. In some implementations, the position, state, and configuration of a widget are preserved when the dashboard layer **402** is dismissed, so that these characteristics are restored the next time the dashboard layer **402** is activated.

[0077] In some implementations, widgets and/or dashboard layers (including widgets) can be installed from within a running application. For example, a widget and/or dashboard (including widgets) can be an attachment to an email. When the user clicks the attachment, an installation process is invoked for the widget and/or dashboard which can also include a preview.

[0078] Widgets can be created or instantiated using an installer process. The installer process can include a separate user interface or an integrated user interface (e.g., integrated in the display area or separate from the display area, for example, in another display area associated with another application, such as an email application) for selecting and installing widgets in a display area. For example, a widget received as an email attachment can be launched by a user from directly within a user interface of the email application.

[0079] Widgets can be created or instantiated using an installer process. The installer process can include a separate user interface or an integrated user interface (e.g., integrated in the display area or separate from the display area for example in another display area associated with another application, such as an email application) for selecting and installing widgets in a display area. Thus, the installation area for the widget can be embedded within an application display area or window. For example, if a user receives a

widget as an attachment to an email, the user can invoke and install the widget from within the email message window without the need for a separate installation window.

[0080] In general, an installer process is used to provide additional functionality to the creation/instantiation process, beyond the simple drag and drop operation describe above. Additional functionality can include preview, linking, security and deletion functionality in a singular interface. The installer process can be a separate process or combined in another process. The installer process can itself be a separate application that is executable to install widgets (or other elements) in a display area. As used herein, the term "process" refers to a combination of functions that can be implemented in hardware, software, firmware or the like.

[0081] The use of nested display areas associated with different dashboard environments enables users to organize dashboards into hierarchies. For example, a dashboard environment including widgets for use with operating system utilities can include nested display areas for displaying widgets associated with particular types of utilities (e.g., date and time, memory management, network resource management, etc.)

Linked Widgets

[0082] FIG. **5**A is a screen shot depicting an implementation of linked widgets. Linked widgets can be displayed in any display area, such as a dashboard layer **506** overlaid on a desktop user interface **504**. In some implementations, the user interface **504** can be a conventional UI provided by an operating system, such as the Mac OS® or Windows® operating systems.

[0083] In some implementations, the user interface **504** includes a background image, a menu bar **502**, and other UI common features (e.g., windows, icons, etc.). The user can activate the dashboard layer **506** by selecting an item from the menu bar **502**, or by clicking an icon, or by pressing a function key or key combination, or by some other means for invoking activation. In some implementations, a configuration bar **503** is displayed, which includes one or more widget icons and a widget link manager icon **514**. The configuration bar **503** can be scrolled from left to right to reveal more widget icons. A user can install a widget by dragging its associated widget icon from the configuration bar **503** and dropping it in the dashboard layer **506**, or by using other installation techniques, as described with respect to FIG. **4**D.

Automatic Widget Linking

[0084] In some implementations, two or more widgets are automatically linked in response to a trigger event. A trigger event can be generated by downloading, installing, previewing, launching, manipulating, updating, operating or otherwise interacting with a widget. A trigger event can also be generated by exercising the functionality of a widget.

[0085] Widgets can be automatically or manually linked together based on any suitable criteria or no criteria. For example, widgets can be linked based on the type of data or information the widgets use or share (e.g., time, date, place, etc.), the origin of the widgets (e.g., received from friends, downloaded from a common website, etc.) and the time when the widgets were downloaded (e.g., widgets downloaded at the same time of day). Widgets can be linked based on their membership in a widget class (e.g., financial widgets, stock market widgets, etc.). Widgets can also be linked based on their participation in a process or workflow (e.g., a stock market widget providing data to a 3D graph widget).

[0086] The concept of linked widgets can be illustrated by the following example involving a search widget **508**, a map widget **510** and a weather widget **512**. When the user installs the search widget **508** in the dashboard layer **506**, a trigger event is generated which causes the map widget **510** and the weather widget **512** to be automatically linked to the search widget **508** and displayed in the dashboard layer **506**. A user enters a search query in a search box of the search widget **508** to determine the location of a favorite restaurant. The search widget **508** returns location information (e.g., address, coordinates, etc.) for the restaurant, which is shared with the map widget **510** and the weather widget **512**. The map widget **510** uses the address to generate a map and driving directions. The weather widget **512** uses the address to determine the current weather for the location of the restaurant. If the map and weather widgets **510**, **512**, are already installed and launched in the dashboard layer **506** when the search widget **508** is installed, then the links are established directly without re-installing and launching the widgets **510**, **512**. If the widgets **510**, **512**, are not installed, then the widgets **510**, **512** are installed and launched before or after being linked. If the widgets **510**, **512** are not available on the computer system **102**, a message is displayed instructing the user on how to obtain the widgets **510**, **512** from another source. For example, the message can include a link to a website where the widgets **510**, **512** can be downloaded to the computer system **102**.

[0087] Generally, any two widgets can be linked and share information. The amount and type of information shared is dependant on the widgets that are linked. For example, the widgets **508**, **510** and **512** could share location information (e.g., address, latitude, longitude, etc.).

[0088] The widgets **508**, **510** and **512** are examples of widgets that can be linked based on the type of information the widgets use and share (i.e., location information). Other examples of widgets that can be linked include but are not limited to: a calendar widget that can be linked to a scheduling widget; a dictionary widget that can be linked to a word processing widget; a telephone directory widget that can be linked to dial-up widget; a stock widget that can be linked to a graph widget for presenting stock information in different types of graphs (e.g., pie graph, bar graph, etc.); an image processing/editing widget that can be linked to a picture frame widget for viewing a digital image; and a media player widget that can be linked to a ticket vending widget (e.g., a Ticketmaster™ widget) for providing a touring schedule and a mechanism for purchasing concert tickets to see an artist whose song is currently playing in the media player widget.

[0089] In some implementations, widgets can be linked to one or more non-widget applications and can interact with and receive data from those applications. A non-widget application can also provide a bridge between two or more widgets. For example, when an application is invoked (e.g., a word processor) can search for widgets that can support the application or specific feature of the application (e.g., dic-

tionary or thesaurus widgets). The widgets can communicate directly with each other and/or indirectly through the application.

[0090] In some implementations, links are established between widgets only if one or more conditions, events or triggers are satisfied. For example, a link may only be established upon completion of one or more tasks, or at a certain time of day, or only between widgets that are currently running, or only between widgets in the same dashboard layer, etc. The conditions for establishing links can be a set of rules that should be satisfied before a link is established. The rules can be generated manually by the user or programmatically at run time. Rules can also be dynamically generated by a running widget or non-widget application that is associated with widgets. The rules can be stored in a widget file or other data structure. The rules for linking widgets can be different based on the type of device where the widgets reside (e.g., a portable device, mobile phone, etc.).

[0091] Linked widgets can be located in the same dashboard or different dashboards in a multiple dashboard environment. In some implementations, linked widgets can communicate even when installed in different display areas. Linked widgets can reside on a single device or on multiple devices and communicate over a network connection established between the devices (e.g., Internet, Ethernet, wireless, etc.).

[0092] In some implementations, linked widgets include a link element **516** (e.g., a button), which if selected disables links to other widgets. For example, clicking on the link element **516** of the search widget **508** causes the map and weather widgets **510**, **512**, to be unlinked to the search widget **508**. In some implementations, when widgets are unlinked they are altered or obfuscated in the dashboard layer **506** (e.g., grayed out, dimmed, made semi-translucent, etc.). Alternatively, unlinked widgets can remain visible but a link indicator **518** (e.g., a virtual lamp, etc.) is altered to indicate a widget's link status. For example, the link indicator **518** can change color (e.g., from green to red) to assist users to visually identify the link status of a widget. In other implementations, the icon associated with a widget is modified to indicate the widgets link status (e.g., the icon change colors or displays informative text).

[0093] In some implementations, widgets that are linked (or that are capable of being linked) have a gravitational or magnetic attraction or repulsion to each other. For example, when two widgets are linked together, the widgets positions in a dashboard layer or other user interface can be automatically adjusted so the linked widgets are adjacent or proximate to each other. Under such a simulated gravitational attraction or repulsion, widgets can cluster together in the dashboard layer or user interface to indicate their linked status. The clustering visually indicates to the user that the widgets are linked (or not linked) or that the widgets can be linked. A visual indication of the strength of a link (or the potential to link) can be displayed by changing one or more properties of the widgets, such as the color of the widgets or the distance between widgets. For example, red widgets could indicate a strong link (or potential to link) and green widgets could indicate a weak link (or potential to link). Also, a shorter distance between linked widgets in a dashboard layer or user interface could indicate a stronger link (or potential to link) than widgets that are separated by a greater distance. If the user moves a widget around a dashboard layer or a user interface, other widgets in the dashboard layer or user interface can be attracted to or repulsed by the moving widget to indicate their link status or link potential.

[0094] Widgets can be linked in a variety of topologies. For example, a single widget can be linked to multiple widgets and configured to provide those widgets with common or personalized information (e.g., a broadcasting widget). In some implementations, a widget can behave like a server ("server widget") and interact and exchange information with one or more "client" widgets.

[0095] In some implementations, widgets can be linked at several levels and conceptually organized into a widget hierarchy, for example, forming a "tree" structure where the widget at the top of the tree is a "root" widget and the other widgets are "leafs" or "node" widgets.

[0096] In some implementations, the linkage between two or more widgets can be bi-directional, so that each widget in a pair of linked widgets can be invoked (e.g., launched, installed, updated, etc.) by the other widget in the pair. Also, each widget in a widget pair can transmit and receive information from the other widget in the pair.

[0097] In some implementations, the user interface **504** can be obfuscated to reveal a dashboard layer **506** containing only linked widgets. For example, the user can press a predetermined key combination or other input mechanism, which causes the appearance of unlinked widgets to be altered or otherwise obfuscated so that only linked widgets are visible on the display screen. A key combination can be specified by the user in a preference pane or other user input mechanism.

[0098] In some implementations, a widget link manager enables a user to manually establish and edit widget links, as described with respect to FIG. **8**. A widget link manager icon **514** for invoking the widget link manager can reside anywhere in the desktop user interface **504** and/or in the configuration bar **503**, as shown in FIG. **5**A. The icon **514** can be used to toggle between a dashboard layer or desktop and a user interface for the widget manager.

[0099] FIG. **5**B illustrates the exchange of information between linked widgets. In this example, a phone book widget **520** and a sports widget **522** are residing in a dashboard layer or user interface. The phone book widget **520** includes typical phonebook information, such as addresses and telephone numbers. The sports widget **522** includes information related to sporting events, including information related to sports venues. In this example, the sports widget **522** information does not include addresses and telephone numbers of sports venues.

[0100] When a link is manually or automatically established between the phone book widget **520** and the sports widget **522**, information can be exchanged between the phone book widget **520** and the sports widget **522**. For example, the sports widget **522** can send the phone book widget **520** the name of a sports venue. The phone book widget **520** can then use the name of the sport venue to look up the address and telephone number of the sports venue. Once the address and number are retrieved, the phone book widget **520** sends the address and telephone number to the

sports widget **522**, where the information can be used to augment or enhance the sporting event information **524**.

Automatic Widget Linking Process

[**0101**] FIG. **6** is a flow diagram of an exemplary automatic widget linking process **600**. The steps of process **600** do not necessarily have to occur in a specific order, and at least some steps can be executed in a multi-threading and/or multi-processing environment.

[**0102**] The process **600** begins when a widget linking trigger event is detected (**602**). A widget linking trigger event can generated (e.g., by a dashboard server) in response to the downloading, installation, previewing, launching, updating, manipulation, operation and/or interaction with a widget. A widget linking trigger event can also be the exercise of a feature or functionality of one or more widgets. A widget linking trigger event can be generated by user input or programmatically by software (e.g., operating system, application, etc.) or hardware (e.g., mouse click, hardware plug-in, etc.).

[**0103**] In response to a trigger event, the process **600** determines a set of candidate widgets that can be linked (**604**). Candidate widgets can be determined from predetermined or dynamically generated link information. In some implementations, predetermined link information can be stored in the dashboard configuration information **304**, as shown in FIG. **3**. The link information can be included by a widget's author in files associated with each widget (e.g., info.plist), or created by the user with the widget link manager, as described with respect to FIG. **8**. Link information can also be dynamically created while a widget is running or not. For example, widgets that are running can be forming new links, terminating or reviving existing links, generating or receiving new data sources and the like. This dynamically generated link information can be stored during runtime in memory or other computer-readable medium (e.g., hard disk).

[**0104**] In some implementations, a dashboard can scan for installed widgets to create a collection of linkages, or possible linkages which are stored during runtime. In some implementations, the user can control the linkages which can be stored by the dashboard server. For example, the user can manually establish linkages using a widget link manager or bridging elements, as described with respect to FIGS. **9**A and **9**B.

[**0105**] When a set of candidate widgets is determined, the process **600** automatically links (and installs and launches, if necessary) the candidate widgets (**606**). In some implementations a manual step can be used in combination with the process **600** by automatically presenting a user interface that includes a list of candidate widgets that can be linked to a particular widget (e.g., the widget that generated the trigger event). The candidate widgets can be organized into a file system that can be navigated. For example, launched widgets that are running on the host device can be listed separately from widgets that are residing on the host device but have not been launched.

[**0106**] The user can manually select one or more widgets from the candidate widget list for linking. In other embodiments, a link or other mechanism can be provided in the dashboard layer and/or in a configuration bar that the user can select to invoke the user interface having candidate widgets and/or to direct the user to another source of candidate widgets (e.g., a link to a website).

[**0107**] In some implementations, a communication channel is established between the linked widgets using known object oriented programming (OOP) techniques and languages (e.g., Java, C++, Smalltalk, etc.) for transmitting and receiving messages (**608**). In some implementations, each widget in a linked pair of widgets can "pull" information from the other widget, "push" information to the other widget, or both (i.e., bidirectional communication). In other implementations, each widget writes information to a shared memory or storage location (e.g., local storage **106**) where it can be read by other widgets. The type and amount of information shared is dependent on the needs of the widgets that are linked. An examples of shared link information would be the coordinate or location data between the widgets **508**, **510** and **512**, as described with respect to FIG. **5**A.

[**0108**] In some implementations, the widgets share security information (e.g., keys, secret data, etc.) for secured communications. When widgets share information, there is a danger that malicious widgets will gain access to restricted information. If confidential information is to be shared between widgets, then the widgets can be signed and undergo an authentication procedure during linking using one or more known authentication techniques (e.g., Digital Signature Algorithm (DSA)).

[**0109**] When communication between linked widgets is established, the process **600** monitors the link status (**610**) for changes. Changes could be failed links, temporarily disable links, new links, etc. The process **600** detects any changes and modifies the link information as appropriate. For example, if a new link is established the process **600** will add the link to the link information associated with the widgets forming the link, as described with respect to Table I.

Software Architecture for Linking Widgets

[**0110**] FIG. **7** is a block diagram of an implementation of a software architecture for linking widgets. The software architecture **700** is similar to the architecture described with respect to FIG. **3**. However, the dashboard server **701** includes a widget linker **702** and a widget link manager **704**. The widget link manager **704** is described with respect to FIG. **8**. The software architecture **700** is exemplary and other architectures can be realized having more or fewer components and/or processes. For example, the widget linker **702** and widget link manager **704** can be independent or standalone applications, processes, components, or services that can operate independent of the dashboard server **701**, including as an operating system component or plug-in.

[**0111**] The widget linker **702** is responsible for implementing the process **600**, as described with respect to FIG. **6**. In some implementations, the widget linker **702** monitors the downloading, installation, previewing and launching of widgets and detects trigger events for linking. In response to a trigger event, the widget linker **702** can match a unique widget identifier (e.g., a hash) for the trigger widget with a list of candidate widgets that can be linked to the triggering widget ("candidate link widgets"). In some implementations, the widget linker **702** can store and maintain a link flag or link key for each candidate link widget, together with a

memory address for accessing shared information. Setting the flag will cause the widget to read information from the address provided by the widget linker **702**. The widget linker **702** will also install and/or launch the candidate linked widgets (widgets with set link flags), if the candidate link widgets have not been installed and/or launched. An example data structure for linked widgets is show in Table I below.

TABLE I

Data Structure For Link Information

Widget Name	Widget ID	Linked To:	Sharing:	Link Flag
Search	ID1	ID2, ID3, . . .	location, . . .	On
Map	ID2	ID1, ID3 . . .	None.	On
Weather	ID3	ID1, ID2 . . .	None.	On
Other Widgets	. . .	. . .	. . .	. . .

[**0112**] Referring to Table I, an exemplary data structure includes a Name field, a Widget ID field, a Linked To field and a Sharing field. More or fewer fields can be used, as desired. The Name field stores the name of the widget, the Widget ID field stores a unique ID for the widget (e.g., a fingerprint or hash), the Linked To field includes the Widget IDs for all the widgets that are linked to the widget identified in the Name and Widget ID fields. The Sharing field includes a description of information to be shared by the widget having the Widget ID. The data structure can be stored as a text file in a directory where it can be edited by a user through, for example, a text editor.

[**0113**] In some implementations, the widget linker **702** keeps track of the physical and/or logical address locations of shared information, the types of data that can be shared and the widgets that are allowed to share data. In some implementations, the user or a system administrator can prevent a widget from sharing its data with any other widget (e.g., as a security precaution) by setting the "sharing" field to None. Content feeds and other external information sources can be similarly protected. In some implementations, some widget data can be shared and other widget data can remain private. Such an implementation can be realized by adding one or more additional fields in the data structure shown in Table I. For example, one or more fields can be added that list the source of a trigger event or that stores instantaneous IDs for multiple instances of widgets.

Manual Widget Linking Process

[**0114**] FIG. **8** is a flow diagram of an implementation of a manual widget linking process **800**. The steps of process **800** do not necessarily have to occur in a specific order, and at least some steps can be executed in a multi-threading and/or multi-processing environment.

[**0115**] The process **800** begins when the user selects a first widget for linking (**802**). In some implementations, the first widget includes a linking mechanism (e.g., menu item, button) which, when selected (e.g., mouse clicked), configures the widget for linking with other widgets, and invokes a widget link manager for manually establishing links with other widgets. In other implementations, the widget flips when the linking mechanism is selected, and a widget link manager user interface is presented on the backside of the widget. An example of a user interface for a widget link manager is described with respect to FIG. **9**A.

[**0116**] After the first widget is selected, the user can select information belonging to the first widget which can be shared with other widgets. In some cases, the user may desire to keep certain widget information private but allow other widget information to be made public (i.e., shared with other widgets). In some cases, there may be restrictions on the number and types of widgets that can be linked to the first widget. For example, widgets that have access to certain local or network resources (e.g., file systems, private information, etc.) may be restricted by the user (or the user's privileges) from linking with other widgets for security reasons. For example, a user can turn off automatic widget linking for all or some widgets, or restrict certain widgets from linking with certain other widgets. Such restrictions or any other security methods can be specified or set by a user or administrator through a security preference pane or other input mechanism. These restrictions can be placed in a widget file that is distributed with the widget (e.g., info.plist) or added at a later time by a system administrator or user through a widget manager, as described with respect to FIG. **9**A.

[**0117**] The user selects a second widget for linking with the first widget (**804**). The user can also select information belonging to the second widget which can be shared with the first widget. Once the first and second widgets are specified, including a specification of shared data, the widgets can be linked (**806**).

Manually Linking Widgets Using a Widget Link Manager

[**0118**] FIG. **9**A is a screen shot depicting the manual linking of widgets using an exemplary widget link manager **900**. The widget link manager **900** can be a stand-alone application, an operating system component or plug-in or a widget. The functionality of the widget link manager **900** will now be described in reference to the exemplary search, map and weather widgets **508**, **510** and **512**. It should be appreciated that the widget link manager **900** can operate on any number or type of widgets and is not limited to the widgets disclosed.

[**0119**] In some implementations, the widget link manager **900** is invoked by clicking on the widget link manager icon **514** or other input mechanism (e.g., key combination, menu option, etc.). The widget link manager **900** can be closed by clicking the button **908** or other input mechanism. When invoked the widget link manager provides a display area including a search panel **802** and a link panel **904**. The search panel **902** includes a search box **906** for searching for widgets. For example, a user can put name of a widget in the search box **906** and click a "Go" button **910** to run a search for the search widget **508**. Alternatively, the user can browse a directory structure for widgets, using techniques commonly employed by file systems to search for files (e.g., Mac OS® Finder or Spotlight).

[**0120**] When the search widget **508** is selected its icon or other identifier can be displayed in the search area **902**. To link the search widget **508** with the map widget **510**, the user can, for example, drag the icon for the map widget **510** from the configuration bar **503** and drops the icon in the link area **904**. In some implementations, the widget link manager **900** determines whether the map widget **510** can be linked to the search widget **508** by, for example, examining link infor-

mation for the widgets. If the map widget **510** can be linked, the map widget **510** can appear in a list **914** of linked widgets in the link area **904**. If the map widget **510** cannot be linked to the search widget **508**, the map widget **510** is not displayed in the list **914** and the user is notified (e.g., by an alert message). A link failure could occur if the map and search widgets **508**, **510**, are restricted from being linked to each other, or if there is insufficient link information available for one or both widgets, or for any other reason (e.g., security restrictions). Note that the link information shared is dependant on the widgets that are linked. For example, the search, map and weather widgets would share location information (e.g., address, latitude, longitude, etc.).

[**0121**] It should be apparent that other implementations of the widget link manager **900** are possible. For example, all or part of the functionality of the widget link manager **900** can be accessible on the flip-side of a widget. If a user wants to link widgets, the user can flip the widget to display the search area **902** and link area **904**.

Manually Linking Widgets Using Bridging Elements

[**0122**] FIG. **9**B is a screen shot depicting the manual linking of widgets **508**, **510** and **512** using bridging elements **916**. Bridging elements are objects that can be used to visually connect two or more objects in a user interface. For example, a user can select or click on a dock **918***a* located on widget **508** to grab hold of a first bridging element **916***a*. The user can then click and drag the first bridging element **916***a* to a dock **918***b* located on widget **510**. In this example, the first bridging element **916***a* is shown as a solid line, but other bridging elements are possible (e.g., a dashed line). In some implementations, the user can grab the first bridging element **916***a* by clicking a mouse button and holding the button down while dragging the end of the first bridging element **916***a* to the dock **918***b*. In some implementations, the user clicks on the docks **918***a* and **918***b* and a bridging element **916***a* is automatically displayed between the two docks **918***a* and **918***b*.

[**0123**] When the end of the first bridging element **916***a* is over or in the proximity of the dock **918***b*, the user releases the mouse button and the bridge between the widgets **508** and **510** is completed, resulting in the widgets **508** and **510** being linked. The user can repeat the same procedure using a second bridging element **916***b* and docks **918***a* and **918***c*. In some implementations, the bridging element is displayed until the widgets are linked at which time is removed or otherwise obfuscated. For example, once the widgets have been bridged, the bridging element can disappear or be obfuscated and the widget's properties or characteristics can be altered to indicate link status (e.g., the widgets change color).

[**0124**] In some implementations, the widgets can be unlinked by clicking on each widget of a link, which causes the bridging element to be displayed again. The user can then manually "snip" the bridging element by clicking on it with a mouse or other pointing device. Other techniques for removing links using bridging elements are possible.

Synchronizing Widgets and Dashboards

[**0125**] FIG. **10** is a block diagram of an exemplary synchronization system **1000** for widgets (including linked widgets) and dashboards. In some implementations, the system **1000** includes a network server **1002** operatively coupled to one or more host devices **1006** through a network **1008**. In the example shown, there are two host devices **1006***a* and **1006***b*. The network server **1002** is shown operatively coupled to a network storage **1004** (e.g., optical disk, hard disk, a storage area network (SAN)). The host device **1006** is shown operatively coupled to an external storage media **1010**. The system **1000** can be implemented using a variety of configurations and topologies, and can include more or fewer host devices, servers, storage devices and other devices typically used in a network (e.g., hubs, routers). Examples of a network **1008** include but are not limited to: the Internet, an intranet, a local or wide area network, a wireless network, optical network, etc. Examples of host devices include any device that is capable of using widgets, including without limitation, portable and desktop computers, servers, electronics, media players, game devices, mobile phones, email devices, personal digital assistants (PDAs), televisions, etc.

[**0126**] Users may have several devices that utilize widgets and dashboards. For example, a user may have a desktop computer with certain dashboards and widgets installed and a portable computer with the same or different dashboards and widgets installed. The system **1000** allows a user to synchronize dashboards and widgets installed on a host device, network or storage media to one or more data sources. Data sources can be any source that provides data for creating, installing, operating or managing widgets and/or dashboards, or any data used or presented by widgets and/or dashboards. In the example shown, the host device **1006***a* ("host device A") can synchronize directly with the host device **1006***b* ("host device B") using a known or standard bus technology (e.g., USB, FireWire™) or indirectly through the network server **1002** and network **1080**.

[**0127**] In some implementations, a synchronization service can be used to non-destructively synchronize widgets and/or dashboards between two devices. In the example shown, the host device **1006***a* has installed widgets A, B, and C. Widget A is installed on a user interface provided by, for example, an operating system or application running on the host device **1006***a*. The widgets B and C are installed in a dashboard layer, referred to as "dashboard **1**." The host device **1006***a* can be connected directly to the host device **1006***b* and a synchronization can be initiated by the host device **1006***a*. Synchronization can be initiated manually by user or automatically on a scheduled based or in response to a trigger event. Manual synchronization can be initiated by selecting an option from a menu or other user interface element (e.g., virtual button) presented on a display device of the host device **1006***a* and/or by a hardware mechanism (e.g., a mechanical button, switch, key). When synchronization is initiated, the user can be presented with several options for synchronization. For example, the user can be presented with a list of items (e.g., dashboards, widgets, files, database records, etc.) that can be synchronized with corresponding checkboxes that can be checked by the user to allow the item to be included or excluded from the synchronization process. Once the synchronization process has begun, the user can be presented with a dialog reporting the progress of the synchronization and a summary of the synchronization results. In some implementations, the synchronization results provides a list of potential conflicts and allows the user to manually resolve the conflicts. For

example, a conflict may arise between two different versions of the same widget and/or dashboard. The user can be prompted in real time to resolve the conflict by selecting one version over another version. In some implementations, such conflict resolutions and other specifications for synchronization can be pre-selected by the user through a preference pane or other dialog.

[**0128**] The synchronization process can be performed using known synchronization technologies and/or services. An example of a suitable synchronization service is "Sync Services" provided by Apple Computer, Inc. Sync Services is a framework containing components needed by a developer to sync an application and devices. Data can be synced with other applications and devices on the same device, or other devices over a network using, for example, .Mac (a web-based service provided by Apple). Sync Services is publicly available as an Objective-C Application Programming Interface (API) for Mac OS® X version 10.4 and later. The architecture and logic of Sync Services is described with reference to FIGS. **13** and **14**. An example of a widget/dashboard synchronization process using a network server is described in reference to FIG. **12**.

[**0129**] In some implementations, the system **1000** can use peer-to-peer or network-less syncing. In such an implementation, the network device that is used for syncing may or may not be independent of the target device which is receiving the synced data. For example, Apple Computer Inc.'s "Sync Services" can sync information to a device or to Mac and then to another device. In the former case, there is no "network" other than the network formed by the two devices being synced. In the latter case, there is a network server that holds the information before it is synced to the target device. In some implementations, two peer-to-peer devices can sync in an ad hoc network where no network server is available.

[**0130**] A desirable feature of the system **1000** is the ability to synchronize non-destructively. For example, assume that Widgets A, B and C are installed on host device **1006***a* and Widgets B, C, D and E are installed on host device **1600***b*, but not Widget A. If host device **1006***a* initiates a synchronization with host device **1006***b*, then Widget A will be added to host device **1600***b* and Widgets B and C on host device **1006***b* will be replaced with Widgets B and C on host device **1006***a* (assuming the user or an application has specified such a replacement). After synchronization has completed, the host device **1006***b* will have Widgets A, B, C, D and E. Now if the host device **1006***b* initiates a synchronization with the host device **1006***a*, then Widgets D and E will be added to the host device **1006***a*, Widgets B and C will be replaced and Widget A will remain unaffected. Since Widget A is not removed from the host device **1006***a*, the synchronization is referred to as "non-destructive."

[**0131**] In some implementations, local configuration information (e.g., parameter data) can be identified as related to dashboards and/or widgets on a host device and evaluated for consistency with master configuration information stored locally on the host device or remotely on, for example, a network device (e.g., network server **1002**). If the information is different, then local configuration information can be updated with master configuration information or vice versa.

[**0132**] In some implementations, different versions of the same widget/dashboard and/or different widget/dashboard can be synced based on sync preferences, which can be specified by a user. If a device does not have a widget/dashboard installed, then the settings for the widget/dashboard can be synced, so that when the widget/dashboard is later installed, the widget/dashboard is invoked with the correct settings. If a widget cannot be synced (e.g. it is a purchased widget), then a "dummy" widget can be synced and the user can be provided instructions on how to obtain the widget.

[**0133**] In some implementations, a removable storage media can be synchronized with a host device. In the example shown, the removable storage media **1010**, which includes Widget A, can be synchronized with the host device **1006***a*, which also includes a version of Widget A. Examples of removable storage media **1010** include but are not limited to: external hard drives, USB flash drives, Firewire™ drives, floppy disks, compact discs, an any other storage media that can be plugged into or otherwise connected with a host device. In some implementations, the system **1000** can be used with Portable Home Directories (PHDs) as provided by Apple Computer's Mac OS® X (Tiger) operating system or similar technologies.

[**0134**] In some implementations, the host device **1006***a* scans for removable media **1010**. If detected, the host device **1006***a* mounts the removable media **1010** and searches for widgets/dashboards to be synchronized. If found, then the configurations for the widgets/dashboards can be compared against master information stored locally on the host device **1006***a* or remotely on a network device, and a synchronization process can be initialized manually (e.g., by the user) or automatically, such as the synchronization process described in reference to FIG. **12**.

[**0135**] The use of removable media allows users to maintain widgets and dashboards on multiple devices and to transfer changes made to widgets and dashboards on one device to other devices. For example, a user could carry their customized dashboards and widgets on a USB flash drive. The user can plug the drive into any device capable of supporting dashboards and widgets, and install or cause to be installed, the customized dashboards and widgets. The user can then make changes to widgets and dashboards, which can be stored on the USB flash drive. When the user plugs the drive into another device the changes are detected and synchronized to the new device or other user devices through a network synchronization process, such as is described with reference to FIG. **12**.

[**0136**] In some implementations, the network server **1002** can act as a centralized repository for information relating to a user's widgets and dashboards, which can be downloaded to a device through a network connection. The user can manage the information using the device.

[**0137**] In some implementations, one or more uses can subscribe to a synchronization service. The service can provide a web site where a user can login and specify certain synchronization services. The service can be part of a service that aggregates and distributes widgets and dashboards. For example, a user device could receive updated versions of widgets and/or dashboards by syncing with a network device. The syncing can be initiated by the user device ("sync client") or by the sync service and changes to widgets and/or dashboards can be either pushed by or pulled from a sync engine running on the network device, as

described in reference to FIG. **14**. In some implementations, the synchronization services can be part of a broader service offering, such as described in U.S. patent application Ser. No. _____, for "Management and Generation of Dashboards," filed Aug. 4, 2006, Attorney Docket No. P4210US1/18962-022001.

[**0138**] FIG. **11** illustrates syncing of data between disparate widgets and dashboards. In the example shown, a first device **1102** includes a dashboard layer A for displaying a "Stock Widget"**1106**. A second device **1104** includes a "Graph Widget"**1108**, which is displayed on a operating system desktop. One or both of the widgets **1106**, **1108**, can reside in a dashboard layer or any user interface or display area or surface. When the devices **1102**, **1104**, are synced, data associated with the Stock Widget **1106** is synced with data associated with widget **1108**. For example, if the user adds a "Stock C" to the widget **1106**, the "Stock C" data is transferred to the Graph Widget **1108**, where it can be plotted.

[**0139**] It should be apparent that the widgets **1106** and **1108** are different or disparate and that it is the data associated with the widgets **1106** and **1108** that is synced. This process can be compared to the example synchronization process described in reference to FIG. **10** where additional widgets/dashboards were added to a device or existing widgets/dashboards were replaced with updated versions of the same widget/dashboard.

Widget/Dashboard Synchronization Process

[**0140**] FIG. **12** is a flow diagram of a exemplary widget/dashboard synchronization process **1200**. The process **1200** can be performed on a scheduled basis or initiated by a trigger event. Examples of trigger events include, but are not limited to: downloading, installing or invoking a widget; adding or deleting a widget or dashboard; changing parameters, themes or other information associated with a widget or dashboard; and context (e.g., place, time or subject matter) in which a widget or dashboard is being used. In some implementations, the process **1200** begins by bundling widget/dashboard data on a first device (step **1202**). The bundle is then sent to a network device using, for example, a synchronization service (step **1204**). The bundle can be compared against information accessible by the network device, such as user synchronization specifications. The bundle is then pushed to a second device (step **1206**). In some implementations, the bundle can be pulled from the network by the second device. When the bundle is received at the second device, it is unpacked (step **1208**) and new widgets/dashboards are added to the second device and/or existing widgets/dashboards are replaced (step **1210**). In some implementations, the widgets are presented on the second device (e.g., presented on a display screen) and formatted, if necessary (step **1212**). Formatting would be necessary if, for example, the second device (e.g., a mobile phone) has smaller display screen then the first device (e.g., a desktop computer monitor). In such a case, the widgets can be scaled to fit within the display screen. In some implementations, the widgets/dashboards can be automatically positioned in a predetermined location on the display screen or other location specified by the user.

Sync Services Architecture and Logic

[**0141**] FIG. **13** is an exemplary sync services architecture **1300** for syncing dashboards and widgets. In some implementations, the architecture **1300** includes one or more sync clients **1302**, a sync engine **1304** and a truth database **1306**. The sync engine **1304** and truth database **1306** can be located on a network device (e.g., network server **1002**) or on a sync client **1302**. In the latter case, the sync engine **1304** can be part of an operating system or a separate application. The sync services architecture **1300** is implemented by Apple Computer Inc.'s "Sync Services." A detailed description of the architecture **1300** can be found in the "Sync Services Programming Guide," published by Apple Computer, Inc. (Mar. 8, 2006) and is available to the public from Apple's developer website (http://developer.apple.com). The "Sync Services Programming Guide" is incorporated by reference herein in its entirety.

[**0142**] In some implementations, the sync engine **1304** merges changes to be pulled by different sync clients **1302**. The sync engine **1304** can be invoked on a scheduled basis or triggered by an event. A network-based sync engine **1304** can coordinate the requests of multiple sync devices **1302** simultaneously and can notify a dependent sync device **1302** that an observed sync device **1302** is syncing, and allow the sync device **1302** to join a sync session.

[**0143**] In some implementations, the sync engine **1304** selects an appropriate sync mode for each client **1302**. In a "slow syncing mode," the first time a sync client **1302** syncs, it pushes all its widget/dashboard information in a "bundle" to the sync engine **1304** and pull changes computed by the sync engine **1304**. In a "fast syncing mode," while a client **1302** is pushing and pulling information, the sync engine **1304** keeps track of the client's state using, for example, a snapshot so that subsequent syncs can be more efficient. The next time a client **1302** syncs, only changes are pushed and pulled. In some implementations, the sync engine **1304** assumes the client **1302** is fast syncing unless the client negotiates another sync mode or some state has changed that requires a different mode. Intelligence can be built into the sync engine **1304** to resolve conflicts and duplicates without requiring user input. In some implementations, the sync engine **1304** is a differencing engine that processes changes to individual parameters associated with a widget or dashboard. If two clients **1302** modify the same parameter for a widget or dashboard, the sync engine **1304** can generate a conflict.

[**0144**] In some implementations, the truth database **1306** contains an aggregate of all the client's widget and dashboard information. The truth database **1306** can use a canonical scheme that is an aggregate of all the widget and dashboard schemas used by all the clients **1302**.

[**0145**] FIG. **14** is exemplary sync services logic for syncing dashboards and widgets. In some implementation, the logic **1400** begins when a sync session is started by, for example, the sync engine **1304**. If the session is started (step **1402**), the client negotiates a sync mode with the sync engine **1304** (step **1404**). This can be a slow syncing mode or a fast syncing mode. If the client has changes to push to the sync engine **1304** (step **1406**), the changes are pushed by the client (step **1408**). If the client needs to pull changes from the sync engine **1304** (step **1410**), then the client prepares to pull changes (step **1412**) and pulls changes from the sync engine **1304** (step **1414**). If there are no push or pull changes after negotiating a sync mode with a client, then the

sync services session is terminated. The sync services logic **1400** is implemented by Apple Computer Inc.'s "Sync Services."

[**0146**] It will be understood by those skilled in the relevant art that the above-described implementations are merely exemplary, and many changes can be made without departing from the true spirit and scope of the present invention. Therefore, it is intended by the appended claims to cover all such changes and modifications that come within the true spirit and scope of this invention.

What is claimed is:

1. A method, comprising:

receiving a first set of widget information associated with a first device; and

synchronizing the first set of widget information with a data source.

2. The method of claim 1, further comprising:

providing a synchronized set of widget information to a second device; and

replacing widget information associated with the second device with the synchronized set of widget information.

3. The method of claim 1, where synchronizing the first set of widget information with a data source, further comprises:

synchronizing the first set of widget information with a second set of widget information associated with a second device.

4. The method of claim 1, where synchronizing the first set of widget information with a data source, further comprises:

synchronizing the first set of widget information with a second set of widget information stored on a network device.

5. A method, comprising:

receiving a first set of dashboard information associated with a first device; and

synchronizing the first set of dashboard information with a data source.

6. The method of claim 5, further comprising:

providing a synchronized set of dashboard information to a second device; and

replacing dashboard information associated with the second device with the synchronized set of dashboard information.

7. The method of claim 5, where synchronizing the first set of dashboard information with a data source, further comprises:

synchronizing the first set of dashboard information with a second set of dashboard information associated with a second device.

8. The method of claim 5, where synchronizing the first set of dashboard information with a data source, further comprises:

synchronizing the first set of dashboard information with a second set of dashboard information stored on a network device.

9. A method, comprising:

identifying configuration information associated with one or more widgets; and

updating the configuration information.

10. The method of claim 9, further comprising:

evaluating the configuration information for consistency with master configuration information; and

updating the configuration information based on the results of the evaluation.

11. A system, comprising:

a first device including a first dashboard, the first dashboard including a first set of widgets;

a second device including a second dashboard, the second dashboard including a second set of widgets; and

a sync engine operatively coupled to the first and second devices and configurable to synchronize the first and second dashboards.

12. The system of claim 11, where the sync engine synchronizes widgets in the first and second dashboards.

13. The system of claim 11, where the widgets perform different functions.

14. The system of claim 11, where the widgets are different versions of the same widget.

15. The system of claim 11, where the widgets are synchronized based on synchronization preferences.

16. The system of claim 11, where the sync engine synchronizes widget settings.

17. The system of claim 11, where the synchronization is initiated by the first device.

18. The system of claim 11, where the synchronization is initiated by the second device.

19. The system of claim 11, where the synchronization is non-destructive.

20. An apparatus, comprising:

a computer-readable medium adapted for storing a first set of dashboard information; and

a sync engine operatively coupled to the computer-readable medium and configurable for synchronizing the first set of dashboard information with a data source.

21. The system of claim 20, where the data source is a second set of dashboard information stored on a network.

22. The system of claim 20, where the data source is a second set of dashboard information stored on another apparatus.

23. The system of claim 20, where the data source is a second set of dashboard information stored on removable storage media.

24. A computer-readable medium having instructions stored thereon, which, when executed by a processor, causes the processor to perform the operations of:

receiving a first set of widget information associated with a first device; and

synchronizing the first set of widget information with a data source.

25. The computer-readable medium of claim 24, further comprising:

providing a synchronized set of widget information to a second device; and

replacing widget information associated with the second device with the synchronized set of widget information.

26. The computer-readable medium of claim 24, where synchronizing the first set of widget information with a data source, further comprises:

synchronizing the first set of widget information with a second set of widget information associated with a second device.

27. The computer-readable medium of claim 24, where synchronizing the first set of widget information with a data source, further comprises:

synchronizing the first set of widget information with a second set of widget information stored on a network device.

28. A computer-readable medium having instructions stored thereon, which, when executed by a processor, causes the processor to perform the operations of:

receiving a first set of dashboard information associated with a first device; and

synchronizing the first set of dashboard information with a data source.

29. The computer-readable medium of claim 28, further comprising:

providing a synchronized set of dashboard information to a second device; and

replacing dashboard information associated with the second device with the synchronized set of dashboard information.

30. The computer-readable medium of claim 28, where synchronizing the first set of dashboard information with a data source, further comprises:

synchronizing the first set of dashboard information with a second set of dashboard information associated with a second device.

31. The computer-readable medium of claim 28, where synchronizing the first set of dashboard information with a data source, further comprises:

synchronizing the first set of dashboard information with a second set of dashboard information stored on a network device.

* * * * *