IN THE UNITED STATES DISTRICT COURT
FOR THE WESTERN DISTRICT OF TEXAS
WACO DIVISION

DODOTS LICENSING SOLUTIONS LLC, Plaintiff, vs. SAMSUNG ELECTRONICS CO., LTD., SAMSUNG ELECTRONICS AMERICA, INC., BEST BUY STORES, L.P., BESTBUY.COM, LLC, and BEST BUY TEXAS.COM, LLC, Defendants.	Case No. 6:22-cv-00535 Jury Trial Demanded

COMPLAINT FOR PATENT INFRINGEMENT
AND DEMAND FOR JURY TRIAL

This is an action for infringement of U.S. Patent Nos. 9,369,545; 8,020,083; and 8,510,407 (the "patents-in-suit"), in which Plaintiff DoDots Licensing Solutions LLC ("DoDots"), makes the following allegations against Defendants Samsung Electronics Co., Ltd. and Samsung Electronics America, Inc. (collectively, "Samsung"), and Best Buy Stores, L.P., Bestbuy.com, LLC and Best Buy Texas.com, LLC (collectively, "Best Buy," or "BBY) (collectively with Samsung, "Defendants"):

THE PARTIES

1. DoDots is a Texas limited liability company with a place of business at 32932 Pacific Coast Highway #14-164, Dana Point, CA 92629.

2. Upon information and belief, Samsung Electronics Co., Ltd. is company organized and existing under the laws of the Republic of Korea, with a principal place of business at 129 Samseong-ro, Yeongtong-gu Gyeonggi-do 16677 Suwon-Shi, Republic of Korea (South).

3. Samsung Electronics Co., Ltd. operates a wholly owned subsidiary, Samsung Electronics America, Inc. ("SEA"), that has been registered to do business in the State of Texas and has been since at least June 10, 1996.

4. SEA is a New York corporation with its principal place of business at 85 Challenger Rd., Ridgefield Park, New Jersey 97660. SEA is a wholly-owned subsidiary of Samsung Electronics Co., Ltd. SEA may be served through its registered agent CT Corporation System, 1999 Bryan Street, Suite 900, Dallas, Texas 75201.

5. Samsung Electronics Co., Ltd. exercises direction and control over the performance of SEA. Alternatively, Defendants form a joint business enterprise such that the performance by one Defendant is each attributable to the other Defendant.

6. Samsung has maintained regular and established places of business or offices and/or other facilities in Texas at least at 12100 Samsung Blvd., Austin, Texas 78754; 2800 Wells Branch Pkwy, Austin, TX 78728; 1301 East Lookout Drive, Richardson, Texas 75082; and 6635 Declaration Drive, Plano, TX 75023.

7. Samsung's products are offered for sale through numerous mobile carriers in this judicial District, including, but not limited to Verizon stores at 2812 W Loop 340 Suite# H-12, Waco, TX 76711; 1820 S Valley Mills Dr, Waco, TX 7671; and 3590 Greenlawn Blvd Suite 103, Round Rock, TX 78664; T-Mobile Stores at 2448 W Loop 340

Suite 24a, Waco, TX 76711 and 208 Hewitt Dr Suite #200, Waco, TX 76712; and AT&T Stores at 4330 W Waco Dr, Waco, TX 76710; 2320 W Loop 340 #100A, Waco, TX 76711; and 1515 Hewitt Dr Ste A, Waco, TX 76712 (collectively, "Waco and Austin Carrier Stores"). On information and belief, Samsung products relevant to the allegations in this Complaint have been sold and used at the Waco and Austin Carrier Stores, and are offered for sale at the Waco and Austin Carrier Stores.

8. Samsung has authorized sellers and sales representatives that offer and sell accused Samsung products relevant to this Complaint throughout the State of Texas, including in this District, and to consumers throughout this District, such as: Best Buy, 4627 S Jack Kultgen Expy, Waco, TX 76706 and 11066 Pecan Park Blvd Ste 300, Cedar Park, TX 78613.

9. Defendant Best Buy Stores, L.P. is a corporation organized and existing under the laws of Virginia with its principal place of business at 7601 Penn Ave South, Richfield, MN 55423.

10. Defendant BestBuy.com, LLC is a corporation organized and existing under the laws of Virginia with its principal place of business at 7601 Penn Ave South, Richfield, MN 55423.

11. Defendant Best Buy Texas.com, LLC is a corporation organized and existing under the laws of Virginia with its principal place of business at 7601 Penn Ave South, Richfield, MN 55423.

JURISDICTION AND VENUE

12. This is an action for infringement of U.S. patent nos. 9,369,545; 8,020,083; and 8,510,407 arising under the patent laws of the United States, Title 35 of the United States Code.

13. This Court has subject matter jurisdiction pursuant to 28 U.S.C. §§ 1331 and 1338(a).

14. This Court has personal jurisdiction over Samsung in this action pursuant to due process, Federal Rule of Civil Procedure 4(k)(2)(B), and/or the Texas Long Arm Statute, by virtue of at least the substantial business Samsung conducts in this forum, directly and/or through intermediaries, including but not limited to: (1) having committed acts within the Western District of Texas giving rise to this action and having established minimum contacts with this forum such that the exercise of jurisdiction over Samsung would not offend traditional notions of fair play and substantial justice; (2) having directed its activities to customers in the State of Texas and this District, solicited business in the State of Texas and this District, transacted business within the State of Texas and this District and attempted to derive financial benefit from residents of the State of Texas and this District, including benefits directly related to the instant patent infringement causes of action set forth herein; (3) having placed its products and services into the stream of commerce throughout the United States and having been actively engaged in transacting business in Texas and in this District; and (4) either individually, as members of a common business enterprise, and/or in conjunction with third parties, having committed acts of infringement within Texas and in this District.

15. Samsung has committed and continues to commit acts of infringement in this District directly and through third parties by, among other things, making, selling, advertising (including through websites), offering to sell, distributing, and/or importing products and/or services that infringe the Asserted Patents as defined below.

16. Samsung has, directly or through its distribution network, purposefully and voluntarily placed infringing products in the stream of commerce knowing and expecting them to be purchased and used by consumers in Texas.

17. Samsung has committed direct infringement in Texas.

18. Samsung has committed indirect infringement based on acts of direct infringement in Texas.

19. Samsung has transacted, and as of the time of filing of the Complaint, continues to transact business within this District.

20. Samsung derives substantial revenues from its infringing acts in this District, including from its manufacture and sale of infringing products in the United States.

21. Venue is proper against Samsung Electronics Co., Ltd. in this District pursuant to 28 U.S.C. § 1391(c)(3) because Samsung Electronics Co., Ltd. is a foreign corporation not resident in the United States and venue is proper in any district against a foreign corporation. Venue is proper for Samsung Electronics America, Inc. under 28 U.S.C. § 1400 because SEA (1) has a regular and established place of business in this Judicial District, and (2) has committed and continues to commit acts of patent

infringement in this Judicial District by, *inter alia*, directly and/or indirectly using, selling, or offering for sale, the accused Samsung products discussed below.

22. Samsung has answered multiple complaints in this District, without contesting Venue and Personal Jurisdiction. For example, Samsung has filed answers in: *Scramoge, Ltd. v. Samsung Electronics Co., Ltd. et al,* 6-21-cv-00454 (W.D. Tex); *Wepay Global Payments LLC v. Samsung Electronics Co., Ltd. et al,* 6-21-cv-01095 (W.D. Tex); and *VOIP-PAL.com.Inc. v. Samsung Electronics Co., Ltd et al,* 6-21-cv-01246 (W.D. Tex.).

23. BBY has committed acts of infringement in this judicial district.

24. BBY has a regular established place of business in this judicial district at 4627 S. Jack Kultgen Expy, Waco, TX 76706.

25. On information and belief, the Court has personal jurisdiction over BBY because BBY has committed, and continues to commit, acts of infringement in the state of Texas, has conducted business in the State of Texas, and/or has engaged in continuous and systematic activities in the State of Texas.

26. On information and belief, BBY's instrumentalities that are alleged herein to infringe were and continue to be used, imported, offered for sale, and/or sold in the Western District of Texas.

27. BBY has agreed, on multiple occasions, that Best Buy Stores, L.P., BestBuy.com, LLC, and Best Buy Texas.com LLC, all subsidiaries of Defendant Best Buy Co., Inc., were the proper defendants in this district and have agreed to not challenge venue for those defendants. *See, e.g., MV3 Partners, LLC v. Best Buy Co.*, Case No. 18-cv-374 (W.D. Tex.), ECF No. 29 and *NXP USA Inc., v. Mediatek Inc. et al.,* Case No. 21-cv-318, (W.D. Tex.), ECF No. 40 ("Substitute Best Buy Defendants are the proper parties to defend against allegations made in this patent infringement lawsuit.").

Background

28. This case arises from groundbreaking technology that the named inventors of the patents-in-suit developed at the turn of the 21st century. At that time, accessing content on the internet generally involved the use of web browsers such as Microsoft's Internet Explorer or Netscape Navigator running on a personal computer or primitive mobile device. Viewing internet content on many devices was hindered by the fact that existing web content and web applications were designed to fit an entire web page displayed on a traditional computer monitor. Many web pages were also slow and difficult to navigate. Various attempts to enhance the traditional web pages, such as the addition of "plug-ins," were equally unsuccessful because they only added to the "mess" of the web page. *See* https://www.forbes.com/forbes/2000/0515/6511334a.html

29. John Kembel and George Kembel, twin brothers, recognized that there was dissatisfaction with the traditional web browser and that there was a "growing desire for individual users to fully control the aggregation and presentation of content and web applications that appears on a client computer." *See, e.g.,* U.S. patent no. 9,369,545, col. 1, ll. 48-51.

30. The Kembel brothers are Stanford engineering, business, and design school alumnae. They are the original founders of DoDots, Inc. Since that time, they have created other start-ups that were acquired by leading companies like Oracle Corporation.

31. In view of this need in the marketplace, the Kembels sought to develop a unique and novel technical solution to a computer-specific process of retrieving and viewing content. The Kembels wanted to eliminate the need for a web browser all together. *See* https://www.forbes.com/forbes/2000/0515/6511334a.html

32. So, in 1999, the Kembels, along with fellow Stanford graduate student, Tony Medrano, founded DoDots, Inc. in Silicon Valley. They developed a novel approach to delivering content from the internet in the form of connected widgets or applications, called "Dots" rather than via a web browser. Those "Dots," also referred to as "Network Information Monitors," were "fully configurable frame[s] with one or more controls; the frame through which content is optionally presented." *See, e.g.,* U.S. patent no. 9,369,545, col. 4, ll. 56-60.

33. The Dots used one-tenth of the data that a traditional web page would use, thus allowing for faster loading and display of internet content. *See* Exh. 1 (Business 2.0: "Windows on the World," August 22, 2000).

34. DoDots, Inc. raised over $20M in funding from leading Silicon Valley venture capital companies such as Softbank, Chase HQ and Merrill Lynch due to strength of their "Dot" technology.

35. To commercialize this technology, DoDots, Inc. created a system and platform for its businesses and other third-parties to develop such widgets or apps and make them available to desktop and mobile devices. The technology was groundbreaking and revolutionary.

36. As noted in an article by CNN in April 2000, the DoDots, Inc. technology was the "Web without a browser," and "DoDots is an application made up of small windows called dots. Through these windows, you can take advantage of the features and services offered by certain Web sites without actually visiting them through a browser. Because the dots are small and operate outside the browser, they provide a faster, more direct link to content providers, according to representatives of DoDots, the new Internet company that makes the application. Each dot handles a specific task. 'Essentially, it's a little Web application on your desktop,' says John Kembel, the company's chief technology officer." https://www.cnn.com/2000/TECH/computing/04/07/dodots.idg/index.html

37. At its height, DoDots, Inc. employed more than 100 people that were designing, innovating, and selling the DoDots, Inc. technology. *See*

https://www.thefreelibrary.com/Back+to+the+launch+pad%3a+after+a+few+dormant+years%2c+tech+entrepreneurs...-a0169825785.

38. The success of DoDots, Inc. saw it valued at $275 million. The company listed dozens of customers that had used the technology to distribute their own Dots, including ABC, Bloomberg, Edmunds, CNET and Merriam-Webster. Seeking to capitalize on this marketplace adoption, the company evangelized the concept of Dots and demonstrated the technology to all who would listen, including at conferences attended by many leading technology companies of today. *See* Exh. 1 (Business 2.0: "Windows on the World," August 22, 2000).

39. Indeed, companies like ABC saw the value of the Dot technology and were extremely excited to partner with DoDots, Inc. As Alan Cohen, executive vice president of marketing and advertising of ABC stated "In our continuing effort to find new ways to connect with our audience, the ABC Dot truly stands out as a revolutionary new communication device…. the ABC Dot will give our viewers a chance to use their computer desktops in ways they never imagined." The ABC Dot was used with such popular shows like "Who Wants to be a Millionaire" and "NYPD Blue", among others. *See* Exh. 2, DoDots, Inc. Press Release, October 2, 2000.

40. DoDots, Inc. launched and scaled a developer program, cultivating a community of over 400 independent Dot developers who were deploying Dots and a base of over 250,000 end-users.

41. DoDots, Inc. also sought and entered into partnerships with leading wireless solutions providers such as 2Roam, to expand its reach to the wireless market.

The CEO of 2Roam, Bryan Wargo, stated "DoDots technology is a killer application for wireless devices as it supports the information needs of the on-the-go mobile professional and, like 2Roam, enables users to maintain a constant state with their wireless content or application." And Bob D'Acquisto, 2Roam's director of business development, recognized that "[the DoDots, Inc. technology] gives 2Roam a new and unique way to package and distribute content to [its] customers…it's a win-win for everyone." *See* Exh. 3, DoDots, Inc. Press Release, September 7, 2000.

42. DoDots, Inc. won back-to-back awards from DemoGod at the DEMO2000 and DEMOMobile 2001 conferences, the leading industry event for disruptive technologies at the time.

43. DoDots, Inc. was named as an "Investor's Choice" winner at the Technologic Partners' Internet Outlook Conference held in Silicon Valley in September 2000. *See* Exh. 4, DoDots, Inc. Press Release, September 20, 2000.

44. Unfortunately, when the industry-wide dot com bubble burst, investors withdrew support at a critical stage of its growth, leaving DoDots, Inc. with limited options. Notwithstanding the collapse of DoDots, Inc., the technology it pioneered has been co-opted by numerous companies selling mobile devices, computers, and web applications, including Defendants.

THE PATENTS-IN-SUIT

45. On June 14, 2016, the U.S. Patent and Trademark Office ("USPTO") duly and lawfully issued U.S. Patent No. 9,369,545 (the "'545 Patent"), entitled "Accessing and Displaying Network Content," naming John Albert Kembel, George Andrew

Kembel, Daniel S. Kim, John Russell, Jake Wobbrock, Geoffrey S. Kembel, Jeremy L. Kembel, and Lynn D. Gabbay as inventors.

46. DoDots is the lawful owner of all right, title and interest in the ’545 Patent and has the right to sue and to recover for past infringement of the ’545 Patent. A copy of the ’545 Patent is attached as Exh. 5.

47. On September 9, 2020, the USPTO’s Patent and Trial Appeal Board (“PTAB”) issued a final written decision finding that “Petitioner has not shown by a preponderance of the evidence that claims 1–10 and 12–15 of the ’545 patent are unpatentable.” Specifically, the PTAB rejected the assertion that any of the challenged claims were invalid as obvious under § 103. The Federal Circuit affirmed the PTAB’s decision on December 8, 2021. *See Lenovo Holding Co. v. DoDots Licensing Sols. LLC*, Nos. 2021-1247, 2021-1521, 2021-1580, 2021 U.S. App. LEXIS 36126, at *2 (Fed. Cir. Dec. 8, 2021).

48. On September 13, 2011, the USPTO duly and lawfully issued U.S. Patent No. 8,020,083 (the “’083 Patent”), entitled “System and Methods for Creating and Authoring Internet Content Using Application Media Packages,” naming John Kembel et al. as the inventors.

49. DoDots is the lawful owner of all right, title and interest in the ’083 Patent and has the right to sue and to recover for past infringement of the ’083 Patent. A copy of the ’083 Patent is attached as Exh. 6.

50. On January 19, 2021, the PTAB issued a final written decision finding that “claims 1–16 of the ’083 patent have not been shown to be unpatentable.” Specifically,

the PTAB rejected the assertion that any of the challenged claims were invalid as obvious under § 103. The Federal Circuit affirmed the PTAB's decision on December 8, 2021. *See Lenovo Holding Co. v. DoDots Licensing Sols. LLC*, Nos. 2021-1247, 2021-1521, 2021-1580, 2021 U.S. App. LEXIS 36126, at *2 (Fed. Cir. Dec. 8, 2021).

51. On August 13, 2013, the USPTO duly and lawfully issued U.S. Patent No. 8,510,407 (the "'407 Patent", collectively with the '545 and '083 patent, the "patents-in-suit"), entitled "Displaying Time-Varying Internet Based Data Using Application Media," naming John Kembel et al. as the inventors.

52. DoDots is the lawful owner of all right, title and interest in the '407 Patent and has the right to sue and to recover for past infringement of the '407 Patent. A copy of the '407 Patent is attached as Exh. 7.

53. On January 5, 2021, the PTAB issued a final written decision finding that "Petitioner has not demonstrated by a preponderance of the evidence that any of claims 1, 8–13, and 20–24 are unpatentable." Specifically, the PTAB rejected the assertion that any of the challenged claims were invalid as obvious under § 103. The Federal Circuit affirmed the PTAB's decision on December 8, 2021. *See Lenovo Holding Co. v. DoDots Licensing Sols. LLC*, Nos. 2021-1247, 2021-1521, 2021-1580, 2021 U.S. App. LEXIS 36126, at *2 (Fed. Cir. Dec. 8, 2021).

Samsung's Infringing Devices and Activities

54. Defendants make, have made, use, have used, sell, have sold, offer for sale, and/or import into the United States devices including Samsung Galaxy Z Series Mobile Phones, Galaxy S Series Mobile Phones, Galaxy Note Series Mobile Phones,

Galaxy A Series Mobile Phones, Galaxy M Series Mobile phones, and Galaxy Tab Series Tablets (collectively, "Accused Samsung Devices").

55. Additionally, with each device, Samsung launched and continues to operate, use, and sell an operating system customized from the Android OS (*e.g.* Android OS12, OS 11, QOS 10, Pie (9.0),Oreo (8.0), Nougat (7.0), Marshmallow (6.0), Lollipop (5.0), KitKat (4.4), Jellybean (4.3, 4.2 and 4.1), Ice Cream Sandwich (4.0), Honeycomb (3.0), Gingerbread (2.3), Froyo (2.2), Éclair (2.1), Donut (1.6) (collectively, "the Samsung OS") along with other software (*e.g.*, installers, the Play Store app, and the Galaxy App Store app) that are pre-installed or updated on each Accused Samsung Device (the "Accused Samsung Software"). Samsung programmed, customized, preinstalled, and developed the Accused Samsung Software specifically for its Accused Samsung Devices and is directly responsible for, and has direct control over the use of the Samsung OS along with other software.

56. Each and every iteration of the Accused Samsung Software is specifically designed by Samsung to cause the Accused Samsung Devices to download applications from an App Store ("Samsung-Supported Apps") in a specific manner. Through these stores various apps can be downloaded for usage on the Accused Samsung Devices. More particularly, Samsung is directly responsible for, and has direct control over, because of the way it programmed and developed the Accused Samsung Software, each and every Accused Samsung Device that is configured to execute the Accused Samsung Software code to obtain Samsung-Supported Apps by transmitting a request to the App Store and receiving the Samsung-Supported App in response to that request.

57. Moreover, each Samsung-Supported App, which runs on the Accused Samsung Software contains specific information that allows the user experience (including the graphical user interface) of the Samsung-Supported App to be presented on the display of the Accused Samsung Devices.

58. By making, selling, offering for sale, and importing the Accused Samsung Devices that require the Accused Samsung Software which executes specific code to obtain, install, and use Samsung-Supported Apps, Samsung directly infringes the patents-in-suit. Further, by making, selling, offering for sale, importing, operating, and using the Accused Samsung Software installed and running on the Accused Samsung Devices that require the Accused Samsung Software, which executes specific code to obtain and utilize Samsung-Supported Apps, Samsung directly infringes the patents-in-suit.

The Accused Samsung Devices Infringe the '545 Patent

59. Samsung directly infringes all of the claims of the '545 patent.

60. For example, Claim 1 of the '545 patent reads as follows:

> (Claim 1 Preamble) A computer-implemented method of obtaining content over a network and displaying the content to a user, the method being implemented in a client computing device in operative communication with a server over a network, the client computing device including electronic storage, a display, and one or more processors configured to execute one or more computer program modules, the method comprising:
>
> (Claim 1 limitation (a)) transmitting a request to the server over the network, the request requesting networked information monitor template;
>
> (Claim 1 limitation (b)) receiving the requested networked information monitor template from the server over the internet, the requested networked information monitor template having been transmitted from the

server over the network responsive to the transmitted request, the networked information monitor template comprising:

> a definition of a viewer graphical user interface within which content in a web browser-readable language may be presented on the display of the client computing device; and
>
> a definition of a first content element for the networked information monitor template, the definition of the first content element referencing a first network location from which the first content element for the networked information monitor template is served over the network;

(Claim 1 limitation (c)) responsive to instructions included in the requested networked information monitor template, presenting the viewer graphical user interface defined by the networked information monitor on the display of the client computing device separate from and outside of any other graphical user interface that includes user controls for specifying the first network location from which the first content element for the networked information monitor is served over the network;

(Claim 1 limitation (d)) responsive to instructions included in the requested networked information monitor template, transmitting over the network a first content request to the first network location referenced by the definition of the first content element for the networked information monitor template;

(Claim 1 limitation (e)) receiving, over the network, the first content element transmitted responsive to the first content request;

(Claim 1 limitation(f)) presenting the received the first content element in the viewer graphical user interface defined by the networked information monitor template, wherein the definition of the viewer graphical user interface and/or the first content element define all controls for enabling a user to interact with the first content element through the viewer graphical user interface.

61. Samsung infringes each step of the computer-implemented method recited in Claim 1 of the '545 patent because it implements, operates and uses its Accused Samsung Software, which executes specific code to obtain, display and use Samsung-Supported Apps, on its Accused Samsung Devices, which are in operative

communication with a server over a network and include electronic storage, a display, and one or more processors configured to execute one or more computer program modules.

62. First, the preamble of Claim 1 is met because Samsung executes, operates uses, and has direct control over a computer-implemented method of obtaining content over a network (such as the internet) and displaying the content to a user that is implemented on each and every Accused Samsung Device, which are in operative communication with a server over a network and include electronic storage, a display, and one or more processors configured to execute one or more computer program modules:

Source: CNET: "Here's every Galaxy S phone since 2010" accessed at (https://www.cnet.com/pictures/evolution-history-samsung-galaxy-phones/)

63. On each of the Accused Samsung Devices, the Accused Samsung Software, because Samsung directly and specifically programmed it to do so, practices the claimed method by implementing code on a client computing device (*i.e.*, each Accused Samsung Device) in operative communication with a server over a network

(such as the internet), the client computing device (*i.e.,* each Accused Samsung Device) including electronic storage, a display (such as each Accused Samsung Device's screen), and one or more processors (such as each Accused Samsung Device's processor(s)) configured to execute one or more computer program modules.

64. Specifically, the Accused Samsung Devices that execute the Accused Samsung Software have electronic storage, display, and processor that are used to communication over a wireless network to access the internet, as seen in the product specifications shown below:

Source: https://news.samsung.com/global/specifications-of-the-galaxy-s-series

65. Samsung infringes limitation (a) of Claim 1 because the Accused Samsung Software in each and every Accused Samsung Device transmits a request to a server

over the network, the request requesting a networked information monitor template. In particular, Samsung programs, executes and uses, and has direct control over the Accused Samsung Software in each and every Accused Samsung Device in a specific and particular manner so that the Accused Samsung Software sends a request to the App Store for an application package (the application package herein is an APK file) to the server over the network and that request requests a networked information monitor template (*e.g.,* APK file, which is a data structure including data structures that constitute the NIM template).

66. Samsung infringes limitation (b) of Claim 1 because Samsung programmed and executes the Accused Samsung Software in its Accused Samsung Devices to receive the requested networked information monitor ("NIM") template (such as, for example, a Stock app/widget) from a server over the internet, the requested networked information monitor template having been transmitted from a server over the network responsive to the Accused Samsung Software's transmitted request.

67. Moreover, the Accused Samsung Software requires the APK file for any Samsung-Supported App, which includes a NIM template, to include:

> a definition of a viewer graphical user interface within which content (*e.g.,* how and where the graphical user interface presents a stock price)) in a web browser-readable language (such as a xml. or a .json files) may be presented on the display (monitor) of the client computing device (*i.e.,* each Accused Samsung Device); and
>
> a definition of a first content element (incorporating the present price of a stock) for the networked information monitor template, the definition of the first content element referencing a first network location (such as using

uniform resource locators) from which the first content element for the networked information monitor template is served over the network.

68. Specifically, the data structures in APK file for any Samsung-Supported App comprises a definition of a viewer graphical user interface within which content in a web browser-readable language may be presented on the display of the client computing device.

69. For example, the data structures in APK files for any Samsung-Supported App is used to define a viewer graphical user interface (*e.g.,* a user interface presented on the screen) that may include menus, buttons, and other features. The app resource contains the files related to the visual presentation of the application, as suggested by the excerpts below.

App resources

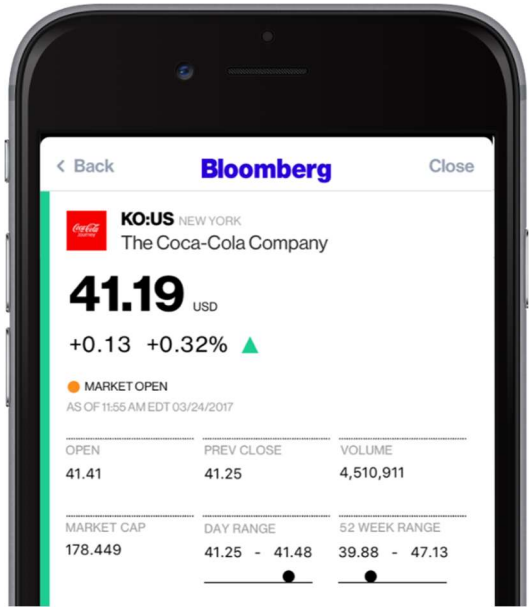
An Android app is composed of more than just code—it requires resources that are separate from the source code, such as images, audio files, and anything relating to the visual presentation of the app. For example, you can define animations, menus, styles, colors, and the layout of activity user interfaces with XML files. Using app resources makes it easy to update various characteristics of your app without modifying code. Providing sets of alternative resources enables you to optimize your app for a variety of device configurations, such as different languages and screen sizes.

Source: https://developer.android.com/guide/components/fundamentals

70. Additionally, the data structures in APK file for any Samsung-Supported App (*i.e.,* the NIM template) comprises a definition of a first content element for the networked information monitor template, the definition of the first content element referencing a first network location from which the first content element for the networked information monitor template is served over the network.

71. For example, the data structures in APK file for any Samsung-Supported App comprises a definition of a first content element (for the example of a Stock app, a

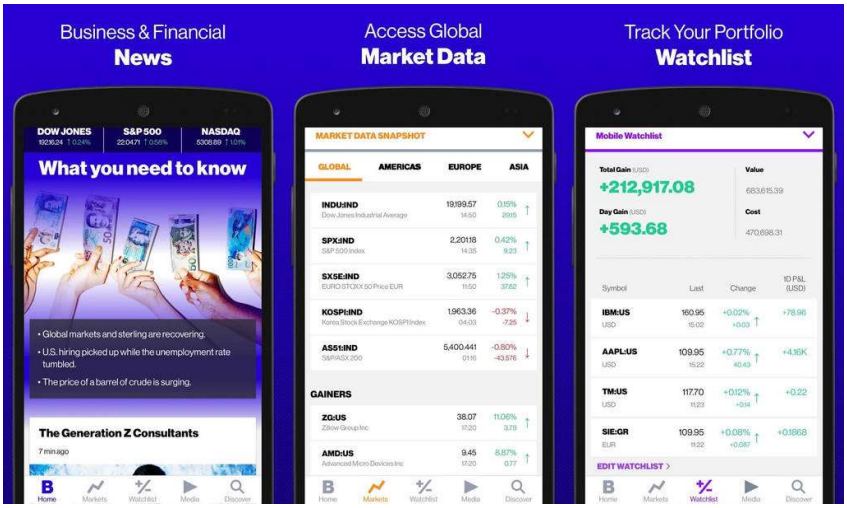
definition of Stock data (*e.g.*, stock price, daily change, percentage change, other information, etc.) that is displayed on the user interface) referencing a first network location (*e.g.*, a location from which the Stock data may be acquired from the internet) from which the first content is served, as seen in the image below:



72. Samsung infringes Claim 1, limitation (c) because its Accused Samsung Software in each of its Accused Samsung Devices is responsive to instructions included in the requested networked information monitor template (such as the Stock app/widget), presenting the viewer graphical user interface defined by the networked information monitor on the display (monitor) of the client computing device (*i.e.*, each Accused Samsung Devices) separate from and outside of any other graphical user interface that includes user controls for specifying the first network location from which the first content element (such as stock price) for the networked information monitor is served over the network.

73. For example, responsive to (*e.g.,* pursuant to, or as defined by) instructions included in the requested NIM template (*e.g.,* the data structures in APK files associated with the Samsung-Supported Apps), the Accused Samsung Software implemented in each of the Accused Samsung Devices presents the viewer with a graphical user interface defined by the NIM template on the display of the client computing device (*e.g.,* the user interface presented on the screen of the Accused Samsung Device) as seen in the paragraph above. The user interface includes user controls (*e.g.,* buttons and menus) for specifying the first network location (*e.g.,* a record of a stock price file at a particular web address) from which the first content element for the networked information monitor (*e.g.,* the Stock data displayed in the user interface frame of the app) is served over the network (*e.g.,* over the Internet).

74. Samsung further infringes Claim 1, limitation (d) because the Accused Samsung Software in each Accused Samsung Device is responsive to instructions included in the requested networked information monitor template, transmitting over the network a first content request to the first network location referenced by the definition of the first content element for the networked information monitor template. Specifically, the Accused Samsung Software implemented in each of the Accused Samsung Devices transmits a request to the first network location (source of the stock market data), as seen in the image below, which shows the results of the Samsung-Supported App pulling information based on a specific watch list.

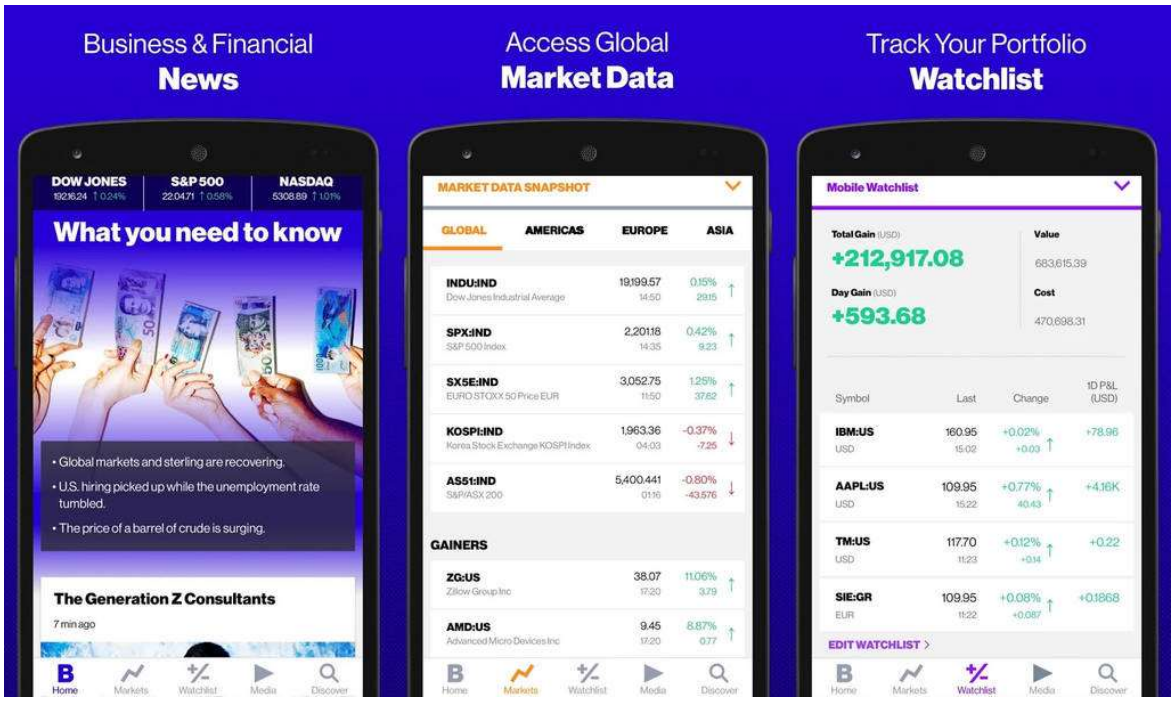


75. Samsung further infringes Claim 1, limitation (e) because the Accused Samsung Software in each Accused Samsung Device receives, over the network, the first content element (such as stock price) transmitted responsive to the first content request. For example, responsive to instructions included in the requested networked information monitor template (*e.g.*, responsive to instructions included in the data structures in APK files associated with the Samsung-Supported Apps requested from the Google Play Store or Galaxy App Store), the Accused Samsung Software in each Accused Samsung Device transmits over a network a first content request (*e.g.*, a request for Stock data) to the first network location referenced by the definition of the first content element for the networked information monitor template (*e.g.*, stock data, such as pricing, daily change, percentage change, other information, is served over the Internet).

76. Finally, Samsung infringes Claim 1, limitation (f) because the Accused Samsung Software in each Accused Samsung Device presents, though the Samsung-Supported Apps, the received first content element (such as stock price) in the viewer

graphical user interface defined by the networked information monitor template (such as the data structures in APK file associated with the Stock app/widget), wherein the definition of the viewer graphical user interface and/or the first content element (such as stock price) define all controls for enabling a user to interact with the first content element (such as stock price) through the viewer graphical user interface (such as allowing the user to select specific stocks to track).

77. For example, the mobile snapshot below discloses different selection features (see details etc.), search menu etc.



<u>The Accused Samsung Devices Infringe the '083 Patent</u>

78. The Accused Samsung Devices infringe all of the claims of the '083 patent.

79. For example, each Accused Samsung Device infringes Claim 1 of the '083 patent, which recites the following limitations:

(preamble) A client device, the client device comprising:

(limitation (a)) electronic storage having stored thereon a plurality of networked information monitor templates defining a plurality of networked information monitors, the plurality of networked information monitor templates comprising a first networked information monitor template defining a first networked information monitor, wherein the first networked information monitor template comprises:

(limitation (b)) a content reference that comprises a network location at which content for the first networked information is accessible via a TCP/IP protocol;

(limitation (c)) a definition of a graphical user interface of the first networked information monitor that lacks controls for manually navigating a network, and that includes a frame within which content received from the network location can be displayed, and frame characteristics defining one or more color, a size, or a position on the electronic display of the frame; and

(limitation (d)) instructions configured (i) to cause the first networked information monitor to request content from the network location in the content reference via the TCP/IP protocol, and (ii) to cause the first networked information monitor to generate the graphical user interface of the first networked information monitor with the content received from the network location via the TCP/IP protocol within the frame;

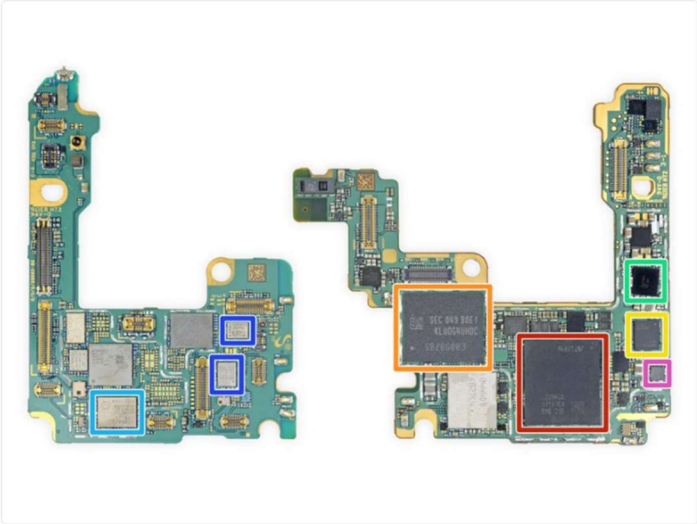
(limitation (e)) an electronic display; and

(limitation (f)) one or more processors configured to access the first networked information monitor template such that the graphical user interface of the first networked information monitor is presented to a user on the electronic display having content received from the content reference therein.

80. Specifically, each Accused Samsung Device, such as the ones seen below, meets the preamble of Claim 1 because each is a client device:

Source: CNET: "Here's every Galaxy S phone since 2010" accessed at (https://www.cnet.com/pictures/evolution-history-samsung-galaxy-phones/)

81. And each of the Accused Samsung Devices meets limitation (a) of Claim 1 because each Accused Samsung Device has an electronic storage having stored thereon a plurality of networked information monitor templates defining a plurality of networked information monitors, the plurality of networked information monitor templates comprising a first networked information monitor template defining a first networked information monitor. Such electronic storage is shown, for example, by the following breakdown of the Samsung Galaxy S21 that shows flash storage:

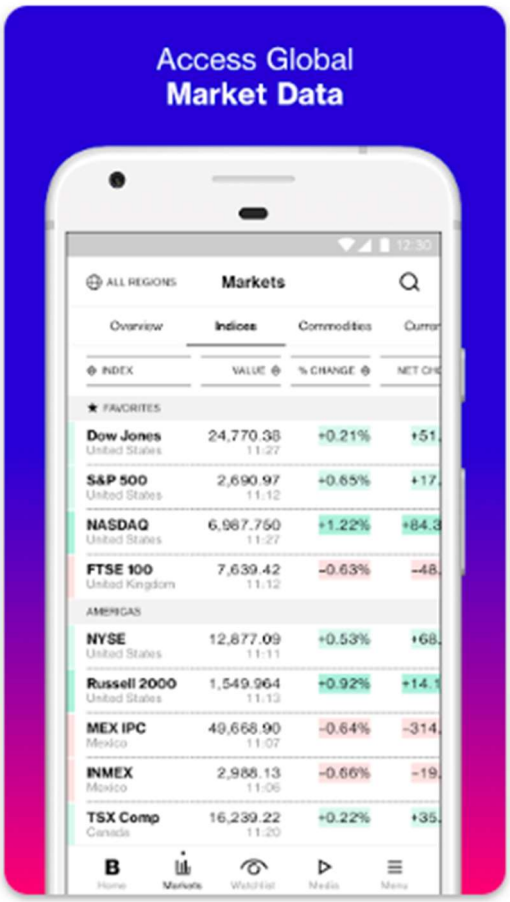


- Do these boards look like howling dogs? Anyways, let's check out what makes this phone tick:
  - Qualcomm Snapdragon 888 layered beneath Samsung K3LK4K40CM-BGCP 12 GB LPDDR5 RAM
  - Samsung flash storage KLUDG4UHDC-B0E1 128 GB
  - Qualcomm SMR526 5G modulator
  - Maxim MAX77705C power management IC
  - Qualcomm QPM5825 power management IC
  - Qualcomm QDM5872 and QDM4820 Front-End Module
  - Cirrus Logic CS35L40 audio amplifier IC

Source: https://www.ifixit.com/Teardown/Samsung+Galaxy+S21+Ultra+Teardown/141188

82. Moreover, the electronic storage of the Accused Samsung Devices is shown to have stored various networked information monitor templates as seen in the image below, which shows an Accused Samsung Devices storing Samsung-Supported Apps, which, as discussed above, are constructed from the data structures in APK files (NIM templates) that are downloaded to Accused Samsung Devices by the Accused Samsung Software and placed in electronic storage:

83. Additionally, each Accused Samsung Device meets limitation (b) of Claim 1 because the electronic storage in each Accused Samsung Device contains the first networked information monitor template that has a content reference (such as a uniform resource locator ("URL")) that comprises a network location at which content for the first networked information (such as stock price data) is accessible via the TCP/IP protocol employed by the internet. For example, the stock app is able to pull stock prices from a network location as seen in the image below:



84. Moreover, each Accused Samsung Device infringes limitation (c) of Claim 1 because a definition of a graphical user interface of the first networked information monitor (such as the graphical user interface of the Stock app) that lacks controls for manually navigating a network, and that includes a frame within which content (such as stock price data) received from the network location can be displayed, and frame characteristics defining one or more color, a size, or a position on the electronic display of the frame.

85. Specifically, the data structures in APK files associated with Samsung-Supported Apps in each Accused Samsung Device are used to define a viewer graphical user interface (*e.g.*, a user interface presented on the screen) that may include menus,

buttons, and other features. The data structures in APK files associated with Samsung-Supported Apps in each Accused Samsung Device contain the files related to the visual presentation of the application as suggested by Samsung developer guides and seen in the excerpt below:

App resources

An Android app is composed of more than just code—it requires resources that are separate from the source code, such as images, audio files, and anything relating to the visual presentation of the app. For example, you can define animations, menus, styles, colors, and the layout of activity user interfaces with XML files. Using app resources makes it easy to update various characteristics of your app without modifying code. Providing sets of alternative resources enables you to optimize your app for a variety of device configurations, such as different languages and screen sizes.

Source: https://developer.android.com/guide/components/fundamentals

86. Additionally, the Accused Samsung Devices infringe limitation (d) of Claim 1 because Samsung-Supported Apps in each Accused Samsung Device include instructions configured (i) to cause the first networked information monitor to request content (such as stock price data) from the network location in the content reference via the TCP/IP protocol, and (ii) to cause the first networked information monitor to generate the graphical user interface of the first networked information monitor with the content (such as stock price data) received from the network location via the TCP/IP protocol within the frame.

87. Specifically, the NIM template (*e.g.*, the data structures in APK file associated with a Stock app) includes instructions configured to request content from the network location (*e.g.*, address of the server containing the Stock data) via a TCP/IP protocol, as shown by the excerpt below:

Web-based content

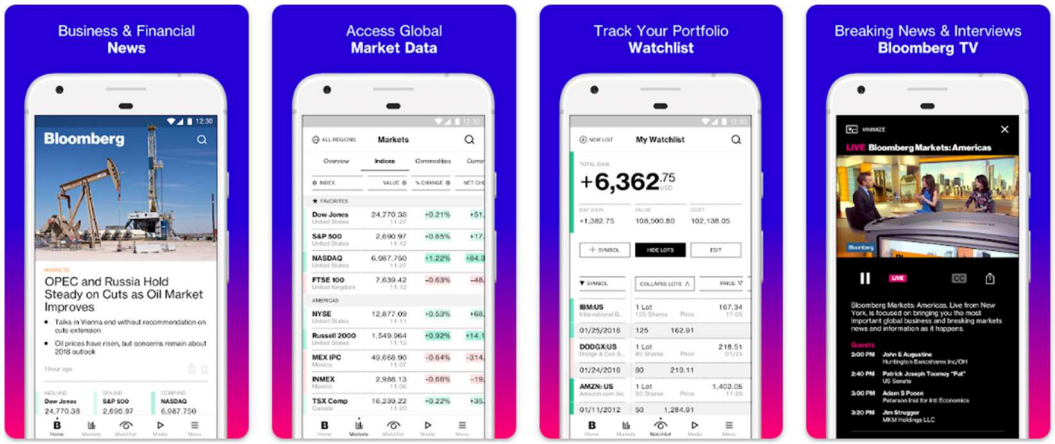
Android offers a variety of ways to present content to a user. To provide a user experience that's consistent with the rest of the platform, it's usually best to build a native app that incorporates framework-provided experiences, such as Android App Links or Search. Additionally, you can use Google Play-based experiences, such as App Actions and Slices, where Google Play services is available. Some apps, however, may need increased control over the UI. In this case, a `WebView` is a good option for displaying trusted first-party content.

Figure 1 illustrates how you can provide access to your web pages from either a browser or your own Android app. The `WebView` framework allows you to specify viewport and style properties that make your web pages appear at the proper size and scale on all screen configurations for all major web browsers. You can even define an interface between your Android app and your web pages that allows JavaScript in the web pages to call upon APIs in your app —providing Android APIs to your web-based application.

Source: https://developer.android.com/guide/webapps.

Notably, this excerpt suggests that Samsung-Supported Apps are able to access web-based content, and on information and belief, such access is through a TCP/IP protocol.

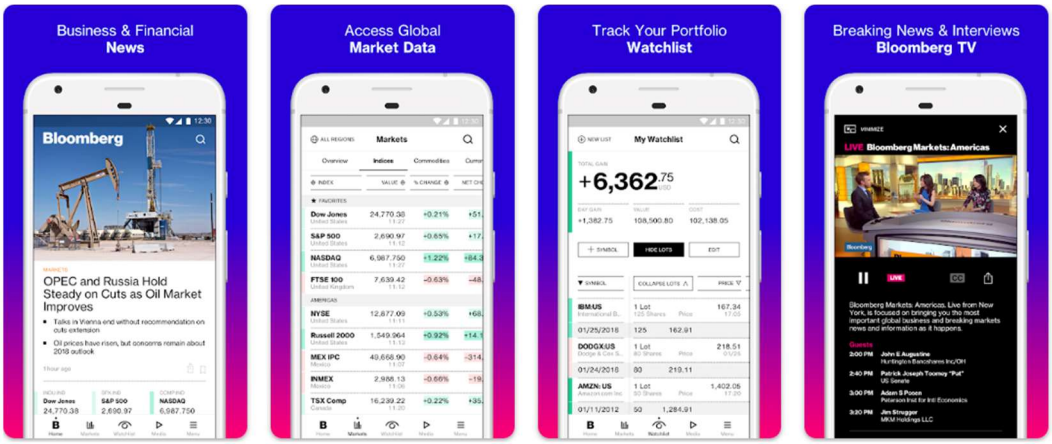
88. Further, the data structures in APK files associated with Samsung-Supported Apps are used to define a viewer graphical user interface (*e.g.*, a user interface presented on the screen) that may include menus, buttons, and other features. The data structures in APK files associated with Samsung-Supported Apps contain the files defining the visual presentation of the application, as shown by the display of the app below:



89. Finally, each Accused Samsung Device has an electronic display (such as the Accused Samsung Devices' display, noted in the image below); and one or more processors (highlighted below):

90. And that processor and display is configured to access the first networked information monitor template such that the graphical user interface of the first networked information monitor is presented to a user on the electronic display having

content (such as stock price data) received from the content reference therein, as exemplified by the images below:



<u>The Accused Samsung Devices Infringe the '407 Patent</u>

91. The Accused Samsung Devices infringe all of the claims of the '407 patent.

92. For example, each Accused Samsung Device, satisfies each limitation of Claim 1 of the '407 patent, which recites the following limitations:

> (preamble) A client computing device configured to access content over a network, the client computing device comprising:
>
> (limitation a) electronic storage configured to store networked information monitor template associated with a networked information monitor, the networked information monitor template having therein a definition of a viewer graphical user interface having a frame within which time-varying content in a web browser-readable language may be presented on a display associated with the client computing device, wherein the frame of the viewer graphical user interface lacks controls for enabling a user to specify a network location at which content for the networked information monitor is available; and
>
> (limitation b) one or more processors configured to execute one or more computer program modules, the one or more computer program modules being configured to access the networked information monitor defined by the networked information monitor template, wherein accessing the

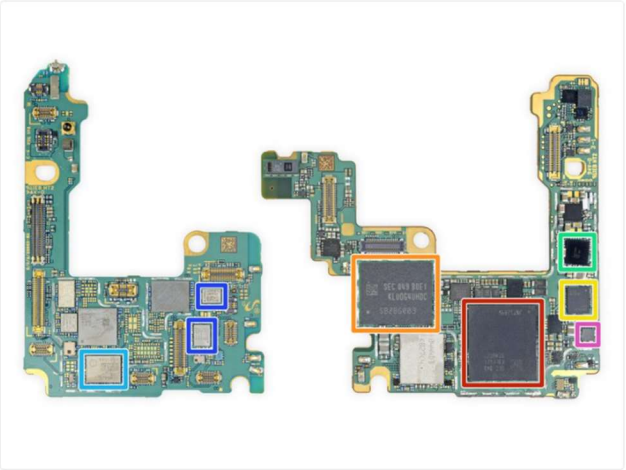
networked information monitor defined by the networked information monitor template results in:

> (limitation b1) transmission, over a network to a web server at a network location, of a content request for content to be displayed within the frame of the viewer graphical user interface defined by the networked information monitor template;
>
> (limitation b2) reception, over the network from the web server at the network location, of content transmitted from the web server in response to the content request, the content being time-varying;
>
> (limitation b3) presentation, on the display, of the viewer graphical user interface defined by the networked information monitor template outside of and separate from any graphical user interface of any other application; and
>
> (limitation b4) presentation, on the display within the frame of the viewer graphical user interface defined by the networked information monitor, of the time-varying content received from the web server.

93. Specifically, each Accused Samsung Device meets the preamble because they are client computing devices configured to access content over a network (such as the internet).

94. And each Accused Samsung Device meets limitation (a) of Claim 1 because it has electronic storage (such as flash memory) configured to store a networked information monitor template associated with a networked information monitor, the networked information monitor template having therein a definition of a viewer graphical interface having a frame within which time-varying content (such as stock price, which varies over time) in a web browser-readable language may be presented on a display (monitor) associated with the client computing device (*i.e.*, each Accused Samsung Device), wherein the frame of the viewer graphical user interface

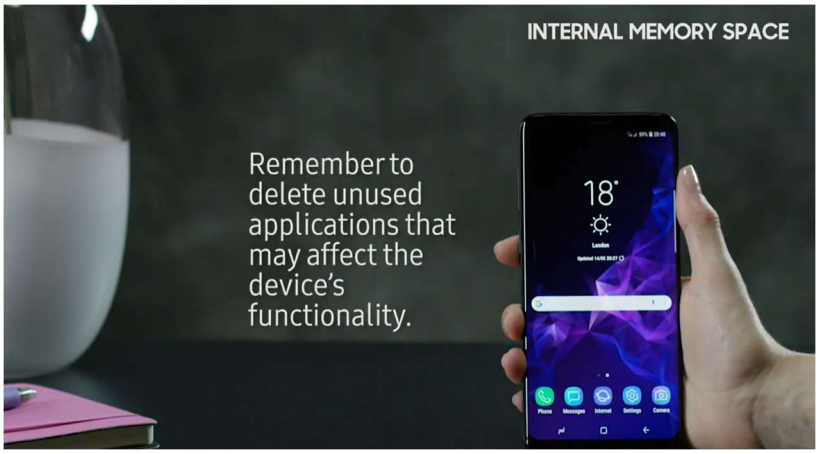
lacks controls for enabling a user to specify a network location (such as a URL) at which content (such as stock price data) for the networked information monitor is available. Such electronic storage is shown, for example, by the following breakdown of the Samsung Galaxy S21 that shows flash storage:



- Do these boards look like howling dogs? Anyways, let's check out what makes this phone tick:
  - Qualcomm Snapdragon 888 layered beneath Samsung K3LK4K40CM-BGCP 12 GB LPDDR5 RAM
  - Samsung flash storage KLUDG4UHDC-B0E1 128 GB
  - Qualcomm SMR526 5G modulator
  - Maxim MAX77705C power management IC
  - Qualcomm QPM5825 power management IC
  - Qualcomm QDM5872 and QDM4820 Front-End Module
  - Cirrus Logic CS35L40 audio amplifier IC

Source: https://www.ifixit.com/Teardown/Samsung+Galaxy+S21+Ultra+Teardown/141188

95. And each Accused Samsung Device stores the data structures in APK files associated with the Samsung-Supported Apps from the App store in its internal storage. Specifically, the Accused Samsung Devices store the data structures in APK files associated with Samsung-Supported Apps and their associated data from the App store in its internal storage, as suggested by the image below:



Source: https://www.samsung.com/sg/support/mobile-devices/how-to-free-up-space-on-samsung-mobile-phone-when-it-is-full/

96. The data structures in APK files for the Samsung-Supported Apps are used to define a viewer graphical user interface (*e.g.*, a user interface presented on the screen) that may include menus, buttons, and other features. The data structures in APK files for each Samsung-Supported App contains the files related to the visual presentation of the application, as suggested by Android developer guides, and seen in the excerpt below:

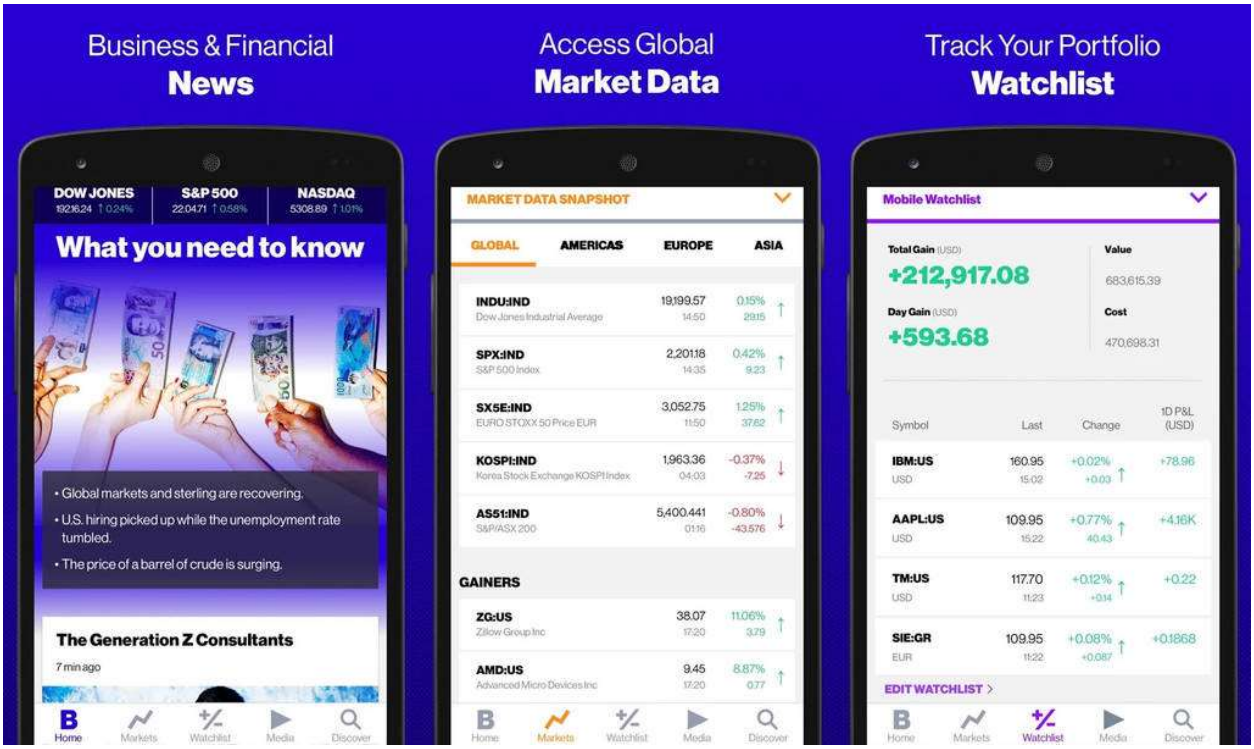
App resources

An Android app is composed of more than just code—it requires resources that are separate from the source code, such as images, audio files, and anything relating to the visual presentation of the app. For example, you can define animations, menus, styles, colors, and the layout of activity user interfaces with XML files. Using app resources makes it easy to update various characteristics of your app without modifying code. Providing sets of alternative resources enables you to optimize your app for a variety of device configurations, such as different languages and screen sizes.

Source: https://developer.android.com/guide/components/fundamentals

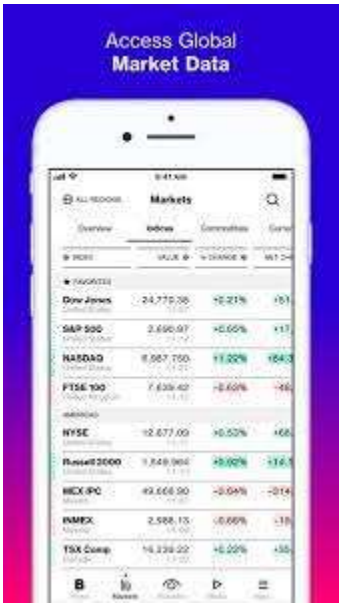
97. Moreover, Samsung-Supported Apps, like the Stock app, provide time varying content which is presented on a display associated with the client computing device (*e.g.*, the Accused Samsung Devices), wherein the frame of the viewer's graphical user interface lacks controls for enabling a user to specify a network location (*e.g.*, user

cannot specify the network location) at which content for the networked information monitor is available, as seen in the description from Stock app below that notes a person can track their watchlist but does not permit a person to specify which server from which to obtain stock information.

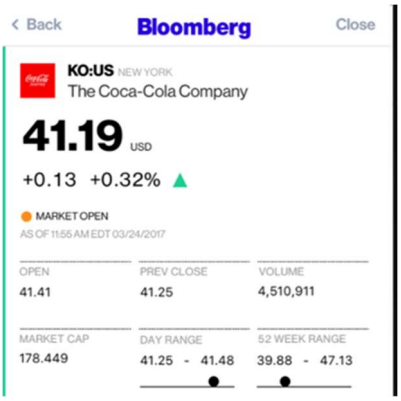


98. Additionally, the Accused Samsung Devices meet limitation (b) of Claim 1 because they include one or more processors configured to execute one or more computer program modules, the one or more computer program modules being configured to access the networked information monitor defined by the networked information monitor template, as exemplified by the image below:

99. The Accused Samsung Devices also meet limitations (b1-b4) of Claim 1 because each Accused Samsung Device includes, implements and uses Accused Samsung Software resulting in transmission, over a network (such as the internet) to a web server at a network location, of a content request (such as a request for stock price data) for content (such as the stock price data) to be displayed within the frame of the viewer graphical user interface defined by the networked information monitor template.



100. Specifically, limitation (b1) of Claim 1 is met because each of the Accused Samsung Devices transmits a request for that content, as suggested by the screen shot below, which shows a response to a request for a specific stock price:



101. And limitation (b2) of Claim 1 is met because in response to the request, there is a reception, over the network (such as the internet) from the web server at the network location, of content (such as stock price data) transmitted from the web server in response to the content request (such as the request for stock price data), the content being time-varying (such as stock price, which varies over time). For example, each

Accused Samsung Device receives Stock data (*e.g.*, current price) over the network from the web server at the network location that is time–varying because stock prices are time dependent, as evidenced by the ability to track a watchlist of stocks:

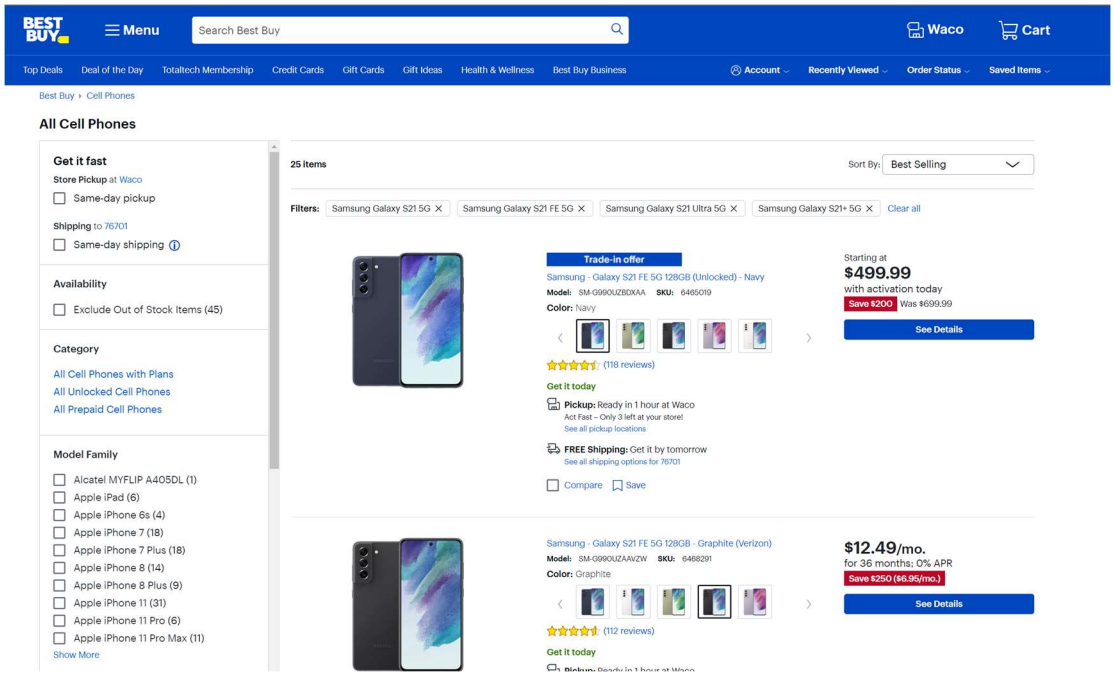
102. Moreover, limitation (b3) of Claim 1 is met as each Accused Samsung Device presents, on the display monitor/screen), a viewer graphical user interface defined by the networked information monitor template outside of and separate from any graphical user interface of any other application (*i.e.*, the viewer graphical user interface of the Stock app), as seen in the screen shot below:

103. Finally, limitation (b4) of Claim 1 is met because on each Accused Samsung Device, the presentation, on the display (monitor/screen) within the frame of the viewer graphical user interface, is defined by the networked information monitor (*i.e.*, the viewer graphical user interface of the stock app), of the time-varying content (such as stock price) received from the web server.

<u>Best Buy's Infringing activities</u>

104. Upon information and belief, Defendant BBY has sold, sells, and offers for sale in its stores in this district certain Accused Samsung Devices.

105. BBY also sold and offered for sale each of the Accused Samsung Devices in this district through its online store, as seen in the image below:



106. Persons with direct knowledge of BBY's infringement in this District include Alejandro Torrez, a specialty sales manager at Best Buy and Todd Melikan, General Manager of Best Buy Stores in this District.

107. DoDots will identify additional Accused Samsung Devices pursuant to the Court's scheduling order.

COUNT I – SAMSUNG'S INFRINGEMENT OF THE '545 PATENT

108. DoDots realleges and incorporates by reference here each of the allegations set forth in the preceding paragraphs.

109. DoDots owns the entire right, title, and interest in the '545 patent and is entitled to sue for past, current and future infringement.

110. Samsung directly infringes one or more claims of the '545 patent in violation of 35 U.S.C. § 271 by, at least, implementing, operating, executing and using in the United States, the Accused Samsung Software in each Accused Samsung Device such that the implementation, operation, execution and use of an Accused Samsung Software infringes one or more claims of the '545 patent. For example, Samsung directly infringes, literally or under the doctrine of equivalents, at least claims 1-2, 9-10, 12 and 13 of the '545 patent.

111. Samsung directly infringes one or more claims of the '545 patent in violation of 35 U.S.C. § 271 by, at least, directly controlling and managing, and having direct responsibility for the implementation, operation, execution and use in the United States of the Accused Samsung Software in each Accused Samsung Device such that the implementation, operation, execution and use of an Accused Samsung Software under Samsung's direct control infringes one or more claims of the '545 patent. For example, Samsung directly infringes, literally or under the doctrine of equivalents, at least claims 1-2, 9-10, 12 and 13 of the '545 patent.

112. Samsung directly infringes one or more claims of the ’545 patent in violation of 35 U.S.C. § 271 by, at least, selling and offering to sell Accused Samsung Devices together with an Accused Samsung Software in the United States, wherein Samsung specifically and intentionally designed and configured each of its Accused Samsung Devices to implement, execute and use an Accused Samsung Software such that the Accused Samsung Software infringes one or more claims of the ’545 patent. For example, Samsung directly infringes, literally or under the doctrine of equivalents, at least claims 1-2, 9-10, 12 and 13 of the ’545 patent.

113. DoDots has not authorized, licensed, or otherwise permitted Samsung to infringe the claims of the ‘545 patent.

114. Samsung is on notice of the ‘545 patent and, despite that notice, continues to infringe the ‘545 patent.

115. As a result of Samsung’s infringement of the ’545 patent, DoDots has suffered damages and will continue to suffer damages.

116. Samsung is therefore liable to DoDots in an amount that adequately compensates DoDots for Samsung’s infringement, which, by law, cannot be less than a reasonable royalty, together with interest and costs as fixed by this Court under 35 U.S.C. §§ 284-285.

COUNT II - SAMSUNG’S INFRINGEMENT OF THE ’083 PATENT

117. DoDots realleges and incorporates by reference here each of the allegations set forth in the preceding paragraphs.

118. DoDots owns the entire right, title, and interest in the ’083 patent and is entitled to sue for past, current and future infringement.

119. Samsung directly infringes one or more claims of the ’083 patent in violation of 35 U.S.C. § 271 by, at least, making, using, supplying, distributing, importing, exporting, selling and/or offering for sale in the United States the Accused Samsung Devices that read on one or more claims of the ’083 patent. For example, Samsung’s Accused Devices, either directly infringe literally or under the doctrine of equivalents, at least claim 1 of the ’083 patent.

120. DoDots has not authorized, licensed or otherwise permitted Samsung to infringe the claims of the ‘083 patent.

121. Samsung is on notice of the ‘083 patent and continues to infringe the ‘083 patent.

122. As a result of Samsung’s infringement of the ’083 patent, DoDots has suffered damages and will continue to suffer damages.

123. Samsung is therefore liable to DoDots in an amount that adequately compensates DoDots for Samsung’s infringement, which, by law, cannot be less than a reasonable royalty, together with interest and costs as fixed by this Court under 35 U.S.C. §§ 284-285.

COUNT III - SAMSUNG’S INFRINGEMENT OF THE ’407 PATENT

124. DoDots realleges and incorporates by reference here each of the allegations set forth in the preceding paragraphs.

125. DoDots owns the entire right, title, and interest in the ’407 patent and is entitled to sue for past, current and future infringement.

126. Samsung is directly and indirectly infringing one or more claims of the ’407 patent in violation of 35 U.S.C. § 271 by, at least, making, using, supplying, distributing, importing, exporting, selling and/or offering for sale in the United States the Accused Samsung Devices that read on one or more claims of the ’407 patent. For example, Samsung’s Accused Devices directly infringe, literally or under the doctrine of equivalents, at least claim 1 of the ’407 patent.

127. DoDots has not authorized, licensed, or otherwise permitted Samsung to infringe the claims of the ‘407 patent.

128. Samsung is on notice of the ‘407 patent and continues to infringe the ‘407 patent.

129. As a result of Samsung’s infringement of the ’407 patent, DoDots has suffered damages and will continue to suffer damages.

130. Samsung is therefore liable to DoDots in an amount that adequately compensates DoDots for Samsung’s infringement, which, by law, cannot be less than a reasonable royalty, together with interest and costs as fixed by this Court under 35 U.S.C. §§ 284-285.

COUNT IV – BBY’S INFRINGEMENT OF THE ’083 PATENT

131. DoDots realleges and incorporates by reference here each of the allegations set forth in the preceding paragraphs.

132. DoDots owns the entire right, title, and interest in the ’083 patent and is entitled to sue for past, current and future infringement.

133. BBY directly infringes one or more claims of the ’083 patent in violation of 35 U.S.C. § 271 by supplying, distributing, importing, exporting, selling and/or offering for sale in the United States the Accused Samsung Devices that read on one or more claims of the ’083 patent. For example, the Samsung Accused Devices, directly infringe literally or under the doctrine of equivalents, at least claims 1-8 of the ’083 patent.

134. DoDots has not authorized, licensed or otherwise permitted BBY to infringe the claims of the ‘083 patent.

135. As a result of BBY’s infringement of the ’083 patent, DoDots has suffered damages and will continue to suffer damages.

136. BBY is therefore liable to DoDots in an amount that adequately compensates DoDots for BBY’s infringement, which, by law, cannot be less than a reasonable royalty, together with interest and costs as fixed by this Court under 35 U.S.C. §§ 284-285.

COUNT V - BBY’S INFRINGEMENT OF THE ’407 PATENT

137. DoDots realleges and incorporates by reference here each of the allegations set forth in the preceding paragraphs.

138. DoDots owns the entire right, title, and interest in the ’407 patent and is entitled to sue for past, current and future infringement.

139. BBY is directly infringing one or more claims of the ’407 patent in violation of 35 U.S.C. § 271 by, at least supplying, distributing, importing, exporting,

selling and/or offering for sale in the United States the Accused Samsung Devices that read on one or more claims of the ’407 patent. For example, the Accused Samsung Devices either directly or indirectly infringe literally or under the doctrine of equivalents, at least claims 1-12 of the ’407 patent.

140. DoDots has not authorized, licensed or otherwise permitted BBY to infringe the claims of the ‘407 patent.

141. As a result of BBY’s infringement of the ’407 patent, DoDots has suffered damages and will continue to suffer damages.

142. BBY is therefore liable to DoDots in an amount that adequately compensates DoDots for BBY’s infringement, which, by law, cannot be less than a reasonable royalty, together with interest and costs as fixed by this Court under 35 U.S.C. §§ 284-285.

JURY DEMAND

143. DoDots hereby respectfully requests a trial by jury of all issues so triable, pursuant to Rule 38, Fed. R. Civ. P.

REQUEST FOR RELIEF

WHEREFORE, DoDots respectfully requests the following relief:

A. A judgment that Samsung has infringed the ’545 Patent, the ‘083 Patent, and the ‘407 Patent;

B. A judgment that BBY has infringed the ‘083 Patent and the ‘407 Patent;

C. An accounting and an award of damages pursuant to 35 U.S.C. § 284 adequate to compensate for Defendants’ infringement of DoDots’ patents-in-suit, and in

no event less than a reasonable royalty for Defendants' acts of infringement, including all pre-judgment and post-judgment interest at the maximum rate permitted by law;

D. A finding that this is an exceptional case under 35 U.S.C. § 285 entitling DoDots to enhanced damages up to a trebling and an award of attorneys' fees;

E. That Defendants be ordered to pay all of DoDots' costs associated with this action; and

F. Any other remedy to which DoDots may be entitled.

Dated: May 24, 2022

Respectfully submitted,

/s/ Raymond W. Mort, III
Raymond W. Mort, III
Texas State Bar No. 00791308
raymort@austinlaw.com
THE MORT LAW FIRM, PLLC
100 Congress Ave, Suite 2000
Austin, Texas 78701
Tel/Fax: (512) 865-7950

Of Counsel:
Ronald M. Daignault (*pro hac vice to be requested*)*
Chandran B. Iyer (*pro hac vice to be requested*)
Shalu Maheshwari (*pro hac vice to be requested*)*
Richard Juang (*pro hac vice to be requested*)*
Oded Burger (*pro hac vice to be requested*)*
Zachary H. Ellis (State Bar No. 24122606)*
rdaignault@daignaultiyer.com
cbiyer@daignaultiyer.com
smaheshwari@daignaultiyer.com
rjuang@daignaultiyer.com
oburger@daignaultiyer.com

zellis@daignaultiyer.com
DAIGNAULT IYER LLP
8618 Westwood Center Drive
Suite 150
Vienna, VA 22102
**Not admitted in Virginia*

Brian S. Seal (pro hac vice to be requested)
TAFT STETTINIUS & HOLLISTER LLP
200 Massachusetts Ave., Suite 400
Washington, D.C. 20001
Tele: (202) 664-1537
bseal@taftlaw.com

ATTORNEYS FOR PLAINTIFF

JS 44 (Rev. 06/17)

CIVIL COVER SHEET

The JS 44 civil cover sheet and the information contained herein neither replace nor supplement the filing and service of pleadings or other papers as required by law, except as provided by local rules of court. This form, approved by the Judicial Conference of the United States in September 1974, is required for the use of the Clerk of Court for the purpose of initiating the civil docket sheet. *(SEE INSTRUCTIONS ON NEXT PAGE OF THIS FORM.)*

I. (a) PLAINTIFFS

DoDots Licensing Solutions LLC

(b) County of Residence of First Listed Plaintiff _____
(EXCEPT IN U.S. PLAINTIFF CASES)

(c) Attorneys *(Firm Name, Address, and Telephone Number)*
The Mort Law Firm, PLLC
100 Congress Ave, Suite 2000, Austin, TX 78701
(512) 865-7950

DEFENDANTS

Samsung Electronics Co., LTD., Samsung Electronics America, Inc., Best Buy Stores, L.P., BestBuy.com, LLC, and Best Buy Texas.com, LLC

County of Residence of First Listed Defendant _____
(IN U.S. PLAINTIFF CASES ONLY)

NOTE: IN LAND CONDEMNATION CASES, USE THE LOCATION OF THE TRACT OF LAND INVOLVED.

Attorneys *(If Known)*

II. BASIS OF JURISDICTION *(Place an "X" in One Box Only)*

- ☐ 1 U.S. Government Plaintiff
- ☒ 3 Federal Question *(U.S. Government Not a Party)*
- ☐ 2 U.S. Government Defendant
- ☐ 4 Diversity *(Indicate Citizenship of Parties in Item III)*

III. CITIZENSHIP OF PRINCIPAL PARTIES *(Place an "X" in One Box for Plaintiff and One Box for Defendant)*
(For Diversity Cases Only)

	PTF	DEF		PTF	DEF
Citizen of This State	☐ 1	☐ 1	Incorporated *or* Principal Place of Business In This State	☐ 4	☐ 4
Citizen of Another State	☐ 2	☐ 2	Incorporated *and* Principal Place of Business In Another State	☐ 5	☐ 5
Citizen or Subject of a Foreign Country	☐ 3	☐ 3	Foreign Nation	☐ 6	☐ 6

IV. NATURE OF SUIT *(Place an "X" in One Box Only)*

Click here for: Nature of Suit Code Descriptions.

CONTRACT
- ☐ 110 Insurance
- ☐ 120 Marine
- ☐ 130 Miller Act
- ☐ 140 Negotiable Instrument
- ☐ 150 Recovery of Overpayment & Enforcement of Judgment
- ☐ 151 Medicare Act
- ☐ 152 Recovery of Defaulted Student Loans (Excludes Veterans)
- ☐ 153 Recovery of Overpayment of Veteran's Benefits
- ☐ 160 Stockholders' Suits
- ☐ 190 Other Contract
- ☐ 195 Contract Product Liability
- ☐ 196 Franchise

REAL PROPERTY
- ☐ 210 Land Condemnation
- ☐ 220 Foreclosure
- ☐ 230 Rent Lease & Ejectment
- ☐ 240 Torts to Land
- ☐ 245 Tort Product Liability
- ☐ 290 All Other Real Property

TORTS

PERSONAL INJURY
- ☐ 310 Airplane
- ☐ 315 Airplane Product Liability
- ☐ 320 Assault, Libel & Slander
- ☐ 330 Federal Employers' Liability
- ☐ 340 Marine
- ☐ 345 Marine Product Liability
- ☐ 350 Motor Vehicle
- ☐ 355 Motor Vehicle Product Liability
- ☐ 360 Other Personal Injury
- ☐ 362 Personal Injury - Medical Malpractice

PERSONAL INJURY
- ☐ 365 Personal Injury - Product Liability
- ☐ 367 Health Care/ Pharmaceutical Personal Injury Product Liability
- ☐ 368 Asbestos Personal Injury Product Liability

PERSONAL PROPERTY
- ☐ 370 Other Fraud
- ☐ 371 Truth in Lending
- ☐ 380 Other Personal Property Damage
- ☐ 385 Property Damage Product Liability

CIVIL RIGHTS
- ☐ 440 Other Civil Rights
- ☐ 441 Voting
- ☐ 442 Employment
- ☐ 443 Housing/ Accommodations
- ☐ 445 Amer. w/Disabilities - Employment
- ☐ 446 Amer. w/Disabilities - Other
- ☐ 448 Education

PRISONER PETITIONS

Habeas Corpus:
- ☐ 463 Alien Detainee
- ☐ 510 Motions to Vacate Sentence
- ☐ 530 General
- ☐ 535 Death Penalty

Other:
- ☐ 540 Mandamus & Other
- ☐ 550 Civil Rights
- ☐ 555 Prison Condition
- ☐ 560 Civil Detainee - Conditions of Confinement

FORFEITURE/PENALTY
- ☐ 625 Drug Related Seizure of Property 21 USC 881
- ☐ 690 Other

LABOR
- ☐ 710 Fair Labor Standards Act
- ☐ 720 Labor/Management Relations
- ☐ 740 Railway Labor Act
- ☐ 751 Family and Medical Leave Act
- ☐ 790 Other Labor Litigation
- ☐ 791 Employee Retirement Income Security Act

IMMIGRATION
- ☐ 462 Naturalization Application
- ☐ 465 Other Immigration Actions

BANKRUPTCY
- ☐ 422 Appeal 28 USC 158
- ☐ 423 Withdrawal 28 USC 157

PROPERTY RIGHTS
- ☐ 820 Copyrights
- ☒ 830 Patent
- ☐ 835 Patent - Abbreviated New Drug Application
- ☐ 840 Trademark

SOCIAL SECURITY
- ☐ 861 HIA (1395fi)
- ☐ 862 Black Lung (923)
- ☐ 863 DIWC/DIWW (405(g))
- ☐ 864 SSID Title XVI
- ☐ 865 RSI (405(g))

FEDERAL TAX SUITS
- ☐ 870 Taxes (U.S. Plaintiff or Defendant)
- ☐ 871 IRS—Third Party 26 USC 7609

OTHER STATUTES
- ☐ 375 False Claims Act
- ☐ 376 Qui Tam (31 USC 3729(a))
- ☐ 400 State Reapportionment
- ☐ 410 Antitrust
- ☐ 430 Banks and Banking
- ☐ 450 Commerce
- ☐ 460 Deportation
- ☐ 470 Racketeer Influenced and Corrupt Organizations
- ☐ 480 Consumer Credit
- ☐ 490 Cable/Sat TV
- ☐ 850 Securities/Commodities/ Exchange
- ☐ 890 Other Statutory Actions
- ☐ 891 Agricultural Acts
- ☐ 893 Environmental Matters
- ☐ 895 Freedom of Information Act
- ☐ 896 Arbitration
- ☐ 899 Administrative Procedure Act/Review or Appeal of Agency Decision
- ☐ 950 Constitutionality of State Statutes

V. ORIGIN *(Place an "X" in One Box Only)*

- ☒ 1 Original Proceeding
- ☐ 2 Removed from State Court
- ☐ 3 Remanded from Appellate Court
- ☐ 4 Reinstated or Reopened
- ☐ 5 Transferred from Another District *(specify)*
- ☐ 6 Multidistrict Litigation - Transfer
- ☐ 8 Multidistrict Litigation - Direct File

VI. CAUSE OF ACTION

Cite the U.S. Civil Statute under which you are filing ***(Do not cite jurisdictional statutes unless diversity)***:
35 U.S.C. § 1 et seq.

Brief description of cause:
Patent Infringement

VII. REQUESTED IN COMPLAINT:

☐ CHECK IF THIS IS A **CLASS ACTION** UNDER RULE 23, F.R.Cv.P.

DEMAND $

CHECK YES only if demanded in complaint:
JURY DEMAND: ☒ Yes ☐ No

VIII. RELATED CASE(S) IF ANY

(See instructions): JUDGE _____ DOCKET NUMBER _____

DATE: May 24, 2022

SIGNATURE OF ATTORNEY OF RECORD: /s/ Raymond W. Mort, III

FOR OFFICE USE ONLY

RECEIPT # _____ AMOUNT _____ APPLYING IFP _____ JUDGE _____ MAG. JUDGE _____

BUSINESS 2.0

AUGUST 22, 2000

FILTER

Windows on the World

It wasn't long ago that push technology was going to make order of the overflow of data vying for desktop space. You were supposed to be able to tell a service such as PointCast which information most interested you and it would "push" it to your monitor. You remember what happened to push, right? Down the tubes.

As Webpages grow denser and take longer to download, there should be a way to request only the information you need without the extraneous GIFs, JPEGs, Flash animations, and other digital detritus. DoDots wants to provide customized Web applications. The dots in DoDots are little windows of content extracted from company partners—including ZDNet, ABCNews.com, mySimon, and Merriam-Webster. These dots sit on your desktop as individual applications, separate from whatever pages you're viewing in your browser.

www.dodots.com

GRAPHIC: XPLANE, WWW.XPLANE.COM

DoDots offers users access to specific tasks that are accomplished via the Internet while occupying a minimum amount of space on their desktops. Users sign up for their free DoDots account and download the Home Dot software that allows them to manage the applications they decide to use. Several partners have signed up, including AnyDay.com, PCWorld, Epinions, and GreatRestaurants.com. Other services include:

Assuming you're already connected to the Net, you can turn on the Merriam-Webster dot while you're in an Excel spreadsheet and use it to look up definitions without even opening your browser. The Web-based datebook and contacts organizer dot from AnyDay.com keep a daily agenda open while surfing other Webpages. Or you can scan news headlines from the CNET dot while waiting for that important email. Users download a menu to the desktop that dynamically generates a customizable list of dots. The dot windows link directly to the content provider. Because the dots use one-tenth of the data of a full Webpage, they load fast, placing content control back in your hands, not in the mitts of a proprietary technology.

Exhibit No.
1

Reprinted with permission from BUSINESS 2.0
#30495 Reprinted by Reprint Management Services, (717) 399-1900, www.rmsreprints.com — sales@rmsreprints.com

DoDots™

DoDots, Inc.
830 Stewart Drive, Suite A
Sunnyvale, CA 94085

tel 408 331 7200
fax 408 331 7410
www.DoDots.com

FOR IMMEDIATE RELEASE

Press Contacts:

Julie O'Grady, DoDots, Inc.
408-331-7294; julie@dodots.com

Virginia Jamieson, Eastwick Communications
650-480-4068; virginia@eastwick.com

DoDots Announces Partnership With Wireless Solution Provider 2Roam at DEMOmobile 2000 Wireless Conference in Pasadena, California

PASADENA, Calif. – September 7, 2000 – DoDots, Inc., a digital infrastructure company, today previewed mobile Dot technology at DEMOmobile 2000, a conference highlighting emerging technologies that will shape the future of transportable computing. DoDots announced its new mobile partnership initiative by teaming with 2Roam, a wireless Web Application Service Provider (ASP) provider that enables Web site businesses to quickly and profitably extend their business models to the wireless world. Under the partnership, 2Roam will empower consumers and business professionals to receive DoDots' innovative Dot technology on their wireless PDAs and cell phones.

"Our goal is to extend the most innovative Internet content, applications, and services to all wireless devices," said Bryan Wargo, president and CEO of 2Roam. "DoDots technology is a killer application for wireless devices as it supports the information needs of the on-the-go mobile professional and, like 2Roam, enables users to maintain a constant state with their wireless content or application, insuring the optimal user experience. We look forward to working closely with DoDots to help reach millions of wireless users."

"The wireless space is a natural extension of the DoDots offering," said Bob D'Acquisto, director of business development at 2Roam. It gives 2Roam a new and unique way to package and distribute content to our customers, making for a more complete content delivery system and a new revenue source. It's a win-win for everyone."

DoDots has been developing its technology for the mobile space, and partnering with 2Roam will allow DoDots to take advantage of this space as an extension of its existing capabilities. "2Roam's technology will offer our customers a robust solution for deploying their existing content on mobile devices," said George Kembel, CEO of DoDots. "We will continue to extend our infrastructure through partnerships to meet our customers' needs as the Internet moves off the desktop."

-more-

Exhibit No.
2

DoDots™

DoDots, Inc.
830 Stewart Drive, Suite A
Sunnyvale, CA 94085

tel 408 331 7200
fax 408 331 7410
www.DoDots.com

FOR IMMEDIATE RELEASE

Press Contacts:
Julie O'Grady - DoDots, Inc., (408) 331-7294
Virginia Jamieson - Eastwick Communications, (650) 480-4068
Kevin Brockman - ABC, (310) 557-6676
Michelle Bergman - ABC.com, (818) 623-3944

DoDots Technology Embraced by ABC Television Network and ABC.com to Deliver Innovative ABC Dots on the Internet

ABC Dots provide viewers with an enriched online experience on their desktops

SUNNYVALE, Calif. – October 2, 2000 - DoDots, Inc., a digital infrastructure company, today announced that the ABC Television Network and ABC.com are using DoDots technology to launch ABC Dots, an exciting and innovative way to allow viewers to interact with their favorite ABC shows and characters via the Internet. In addition to using ABC Dots to provide its viewers with a direct online connection to its fall line-up and stars, ABC Dots will give ABC a persistent branded presence on their viewers' desktops.

"In our continuing effort to find new ways to connect with our audience, the ABC Dot truly stands out as a revolutionary new communication device," said Alan Cohen, executive vice president, marketing, advertising & promotion, ABC Entertainment Television Group. "The ABC Dots will allow people to experience the stars, the shows, and everything ABC has to offer in a whole new way." Michael Benson, senior vice president, advertising & promotion, ABC Entertainment added, "In addition to being an innovative marketing and branding tool, the ABC Dot will give our viewers a chance to use their computer desktops in ways they never imagined.

The ABC Dot provides immediate access to ABC's programming schedule and talent information, fun and informative games and puzzles, and other applications. Little Dot, the classic Harvey comic book character, will greet users when the ABC Dot is accessed and will guide them through the Dot features. Initially, ABC Dots will provide users with direct access to ABC's complete fall line up, including **"Who Wants to Be a Millionaire," "Dharma & Greg," "Norm," "The Drew Carey Show," "NYPD Blue," "Once and Again," "The Practice,"** and more. Additional ABC Dots and features are being developed and will be added as they become available.

-more-

Exhibit No.
3

DoDots™

DoDots, Inc.
830 Stewart Drive, Suite A
Sunnyvale, CA 94085

tel 408 331 7200
fax 408 331 7410
www.DoDots.com

FOR IMMEDIATE RELEASE

Press Contacts:

Julie O'Grady, DoDots, Inc.
408-331-7294; julie@dodots.com

Virginia Jamieson, Eastwick Communications
650-480-4068; virginia@eastwick.com

DoDots Named "Investors' Choice" Winner at Technologic Partners' Internet Outlook Conference in Burlingame, California

SUNNYVALE, California – September 20, 2000 – DoDots, Inc., a digital infrastructure company, has been named one of those "most likely to succeed" at Internet Outlook 2000, a conference where key industry executives and investors rate company prospects for success. Among those attending were venture capitalists, money managers, investment bankers, and industry executives from leading e-commerce companies like MP3.com, Screaming Media, Lycos, Travelocity, Modem Media, and Vignette.

"As a recipient of the Investors' Choice award, DoDots was recognized above and beyond other presenting companies to receive one of 10 Investors' Choice awards. We believe DoDots exemplifies the key characteristics needed to succeed in this competitive market environment and are looking forward to seeing its positive growth in the coming year," said Richard A. Shaffer, founder of Technologic Partners. Votes by the audience were supplemented by recommendations from a panel of distinguished technology investors and observers including Stewart Alsop, general partner at New Enterprise Associates; Alex Baluta, principal at Robertson Stephens; and Steve Jurvetson, managing director at Draper Fisher Jurvetson.

"We are honored to have been selected by the Internet Outlook audience and the esteemed technology investor panel to receive the Investors' Choice Award," said George Kembel, CEO of DoDots. "This validation of our business is strongly supported by the results our technology is delivering to customers who are extending their businesses online."

Issues discussed at the conference included what it will take to get the next wave of consumers online, whether privacy issues could stymie effective marketing tactics, and whether technology can make a difference for the customer experience.

Exhibit No.
4

-more-

US009369545B2

(12) **United States Patent**
Kembel et al.

(10) **Patent No.: US 9,369,545 B2**
(45) **Date of Patent: *Jun. 14, 2016**

(54) **ACCESSING AND DISPLAYING NETWORK CONTENT**

(71) Applicant: **MAINSTREAM SCIENTIFIC, LLC**, Mountain View, CA (US)

(72) Inventors: **John Albert Kembel**, Palo Alto, CA (US); **George Andrew Kembel**, Menlo Park, CA (US); **Daniel S. Kim**, Palo Alto, CA (US); **John Russell**, Palo Alto, CA (US); **Jake Wobbrock**, Palo Alto, CA (US); **Geoffrey S. Kembel**, Menlo Park, CA (US); **Jeremy L. Kembel**, Palo Alto, CA (US); **Lynn D. Gabbay**, Sunnyvale, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 320 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **13/975,227**

(22) Filed: **Aug. 23, 2013**

(65) **Prior Publication Data**

US 2013/0346485 A1 Dec. 26, 2013

Related U.S. Application Data

(63) Continuation of application No. 11/932,392, filed on Oct. 31, 2007, now Pat. No. 8,521,833, which is a continuation of application No. 09/558,925, filed on Apr. 26, 2000, now Pat. No. 7,660,868.

(Continued)

(51) **Int. Cl.**
G06F 15/16 (2006.01)
H04L 29/06 (2006.01)

(Continued)

(52) **U.S. Cl.**
CPC ***H04L 67/42*** (2013.01); ***G06F 3/0484*** (2013.01); ***G06F 8/65*** (2013.01);

(Continued)

(58) **Field of Classification Search**
None
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,297,250 A 3/1994 Leroy et al. 715/763
5,375,199 A 12/1994 Harrow et al. 715/771

(Continued)

FOREIGN PATENT DOCUMENTS

EP 798634 A1 10/1997
WO WO 01/80086 10/2001

OTHER PUBLICATIONS

"Customizing the Active Desktop", Windows 98 Unleashed, published May 12, 1998, retrieved from the Internet from <http://my.safaribooksonline.com/print?xmlid=0-672-31235-2/ch05lev1sec3>, 9 pages.

(Continued)

Primary Examiner — Kenny Lin
(74) *Attorney, Agent, or Firm* — Pillsbury Winthrop Shaw Pittman, LLP

(57) **ABSTRACT**

A method for accessing and displaying network content employs an informational component which includes a definition for rendering a graphical user interface within which content may be presented outside of and without utilization of another application. The informational component also includes one or more definitions of content locations from which content elements may be transmitted over a network and presented within the graphical user interface. A provider of an informational component or informational components may thus create customized displays within which intended content may be rendered. An application component may be used in association with an informational component or informational components to manage the collection, organization, sharing, and rendering of plurality of the informational component(s).

21 Claims, 37 Drawing Sheets

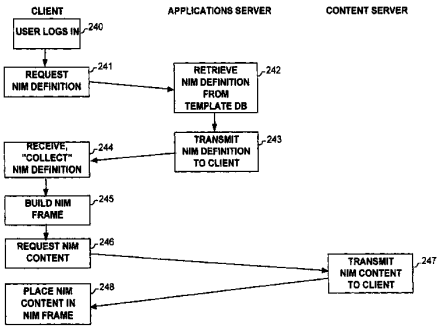


Exhibit No.
5

Related U.S. Application Data

(60) Provisional application No. 60/131,083, filed on Apr. 26, 1999, provisional application No. 60/131,114, filed on Apr. 26, 1999, provisional application No. 60/131,115, filed on Apr. 26, 1999, provisional application No. 60/176,687, filed on Jan. 18, 2000, provisional application No. 60/176,699, filed on Jan. 18, 2000.

(51) **Int. Cl.**

G06F 17/30	(2006.01)
G06F 3/0484	(2013.01)
H04L 12/24	(2006.01)
G06F 9/445	(2006.01)
H04L 29/08	(2006.01)

(52) **U.S. Cl.**
CPC ***G06F 17/30899*** (2013.01); ***H04L 41/22*** (2013.01); ***H04L 65/60*** (2013.01); ***H04L 67/02*** (2013.01); ***H04L 67/10*** (2013.01)

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,599,756	A	2/1997	Matsuo	501/127
5,625,781	A	4/1997	Cline et al.	715/854
5,649,186	A	7/1997	Ferguson	
5,682,511	A	10/1997	Sposato et al.	715/716
5,727,175	A	3/1998	Malone et al.	
5,740,549	A	4/1998	Reilly et al.	705/14
5,745,718	A	4/1998	Cline et al.	715/777
5,761,071	A	6/1998	Bernstein et al.	364/479.07
5,761,662	A	6/1998	Dasan	
5,774,670	A	6/1998	Montulli	395/200.57
5,778,187	A	7/1998	Monteiro et al.	395/200.61
5,793,964	A	8/1998	Rogers et al.	395/200.32
5,794,230	A	8/1998	Horadan et al.	705/35
5,796,393	A	8/1998	MacNaughton et al.	715/733
5,796,952	A	8/1998	Davis et al.	709/224
5,801,702	A	9/1998	Dolan et al.	715/854
5,802,530	A	9/1998	Van Hoff	715/513
5,805,829	A	9/1998	Cohen et al.	395/200.32
5,809,248	A	9/1998	Vidovic	709/219
5,818,446	A	10/1998	Bertram et al.	715/746
5,835,088	A	11/1998	Jaaskelainen, Jr.	715/803
5,838,906	A	11/1998	Doyle et al.	709/219
5,860,068	A	1/1999	Cook	705/26.81
5,864,676	A	1/1999	Beer et al.	709/229
5,864,868	A	1/1999	Contois	707/104
5,890,172	A	3/1999	Borman et al.	715/205
5,892,905	A	4/1999	Brandt et al.	395/187.01
5,893,091	A	4/1999	Hunt et al.	707/3
5,896,533	A	4/1999	Ramos et al.	395/680
5,915,001	A	6/1999	Uppaluru	379/88.22
5,918,237	A	6/1999	Montalbano	715/206
5,919,247	A	7/1999	Van Hoff et al.	709/217
5,922,044	A	7/1999	Banthia	709/203
5,923,845	A	7/1999	Kamiya et al.	709/206
5,923,885	A	7/1999	Johnson et al.	717/176
5,948,061	A	9/1999	Merriman et al.	709/219
5,948,070	A	9/1999	Fujita	
5,959,621	A	9/1999	Nawaz et al.	345/329
5,963,950	A	10/1999	Nielsen et al.	
5,966,715	A	10/1999	Sweeney et al.	707/203
5,973,692	A	10/1999	Knowlton et al.	345/348
5,974,446	A	10/1999	Sonnenreich et al.	709/204
5,974,546	A	10/1999	Anderson	713/2
5,977,964	A	11/1999	Williams et al.	345/721
5,983,227	A	11/1999	Nazem et al.	707/10
5,987,454	A	11/1999	Hobbs	
5,987,513	A	11/1999	Prithviraj et al.	709/223
5,995,756	A	11/1999	Herrmann	717/178
6,006,252	A	12/1999	Wolfe	709/203
6,012,090	A	1/2000	Chung et al.	709/219
6,012,098	A	1/2000	Bayeh et al.	709/246
6,018,344	A	1/2000	Harada et al.	715/818
6,023,698	A	2/2000	Lavey, Jr. et al.	707/10
6,026,433	A	2/2000	D'Arlach et al.	709/217
6,031,904	A	2/2000	An et al.	379/201
6,034,652	A	3/2000	Freiberger et al.	715/730
6,037,935	A	3/2000	Bates et al.	
6,044,403	A	3/2000	Gerszberg et al.	709/225
6,061,695	A	5/2000	Slivka et al.	707/513
6,061,696	A	5/2000	Lee et al.	707/513
6,065,044	A	5/2000	Ogasawara	709/207
6,088,717	A	7/2000	Reed et al.	709/201
6,091,411	A	7/2000	Straub et al.	715/747
6,091,412	A	7/2000	Simonoff et al.	345/335
6,101,510	A	8/2000	Stone et al.	707/513
6,104,391	A	8/2000	Johnston, Jr. et al.	715/745
6,105,063	A	8/2000	Hayes, Jr.	709/223
6,115,040	A	9/2000	Bladow et al.	345/335
6,128,655	A	10/2000	Fields et al.	709/219
6,133,916	A	10/2000	Bukszar et al.	715/744
6,141,693	A	10/2000	Perlman et al.	
6,144,990	A	11/2000	Brandt et al.	709/203
6,161,112	A	12/2000	Cragun et al.	715/501.1
6,177,936	B1	1/2001	Cragun	715/760
6,184,878	B1	2/2001	Alonso et al.	
6,192,407	B1	2/2001	Smith et al.	709/229
6,199,082	B1	3/2001	Ferrel et al.	715/522
6,208,335	B1	3/2001	Gordon et al.	
6,215,490	B1	4/2001	Kaply	715/788
6,216,141	B1	4/2001	Straub et al.	707/513
6,230,173	B1	5/2001	Ferrel et al.	715/513
6,237,030	B1	5/2001	Adams et al.	709/218
6,240,555	B1	5/2001	Shoff et al.	725/110
6,268,856	B1	7/2001	Bruck et al.	715/733
6,275,854	B1	8/2001	Himmel et al.	709/224
6,278,448	B1	8/2001	Brown et al.	345/333
6,278,449	B1	8/2001	Sugiarto et al.	715/826
6,286,034	B1	9/2001	Sato et al.	709/204
6,289,362	B1	9/2001	Van Der Meer	715/273
6,292,185	B1	9/2001	Ko et al.	715/763
6,292,186	B1	9/2001	Lehman et al.	345/335
6,297,819	B1	10/2001	Furst	715/733
6,307,574	B1	10/2001	Ashe et al.	715/765
6,314,451	B1	11/2001	Landsman et al.	709/203
6,317,759	B1	11/2001	Osmond	715/513
6,324,578	B1	11/2001	Cox et al.	709/223
6,339,826	B2	1/2002	Hayes, Jr. et al.	713/166
6,341,305	B2	1/2002	Wolfe	709/203
6,342,907	B1	1/2002	Petty et al.	345/762
6,343,377	B1	1/2002	Gessner et al.	717/10
6,356,905	B1	3/2002	Gershman et al.	707/10
6,369,840	B1	4/2002	Barnett et al.	345/853
6,370,552	B1	4/2002	Bloomfield	715/513
6,374,273	B1	4/2002	Webster	707/513
6,385,596	B1	5/2002	Wiser et al.	705/51
6,393,407	B1	5/2002	Middleton, III et al.	705/14
6,401,134	B1	6/2002	Razavi et al.	709/310
6,411,992	B1	6/2002	Srinivasan et al.	709/218
6,414,677	B1	7/2002	Robertson et al.	345/419
6,418,440	B1	7/2002	Kuo et al.	707/10
6,421,694	B1	7/2002	Nawaz et al.	707/526
6,434,563	B1	8/2002	Pasquali et al.	707/10
6,434,598	B1	8/2002	Gish	709/203
6,452,609	B1	9/2002	Katinsky et al.	345/716
6,453,348	B1	9/2002	Barnier et al.	709/225
6,460,029	B1	10/2002	Fries et al.	
6,476,833	B1	11/2002	Moshfeghi	345/854
6,484,149	B1	11/2002	Jammes et al.	705/26
6,487,566	B1	11/2002	Sundaresan	707/513
6,487,663	B1	11/2002	Jaisimha et al.	713/193
6,496,203	B1	12/2002	Beaumont et al.	345/762
6,510,466	B1	1/2003	Cox et al.	709/229
6,537,324	B1	3/2003	Tabata et al.	715/513
6,538,673	B1	3/2003	Maslov	345/853
6,549,612	B2	4/2003	Gifford et al.	379/67.1

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,560,639	B1	5/2003	Dan et al.	709/218
6,571,245	B2	5/2003	Chun et al.	
6,584,507	B1	6/2003	Bradley et al.	
6,594,682	B2	7/2003	Peterson et al.	709/219
6,606,657	B1	8/2003	Zilberstein et al.	709/224
6,618,754	B1	9/2003	Gosling	
6,629,143	B1	9/2003	Pang	709/226
6,662,341	B1	12/2003	Cooper et al.	715/513
6,681,368	B1	1/2004	Kawabata	715/501.1
6,687,745	B1	2/2004	Franco et al.	709/219
6,691,130	B2	2/2004	Kawasaki et al.	707/102
6,694,484	B1	2/2004	Mueller	715/513
6,718,015	B1	4/2004	Berstis	379/88.17
6,724,403	B1	4/2004	Santoro et al.	345/765
6,751,606	B1	6/2004	Fries et al.	707/3
6,757,716	B1	6/2004	Blegen et al.	709/217
6,766,454	B1	7/2004	Riggins	713/185
6,784,900	B1	8/2004	Dobronsky et al.	345/744
6,816,880	B1	11/2004	Strandberg et al.	709/217
6,819,343	B1	11/2004	Sobeski et al.	715/848
6,819,345	B1	11/2004	Jones et al.	345/856
6,834,302	B1	12/2004	Harvell	709/224
6,842,779	B1	1/2005	Nishizawa	715/757
6,879,994	B1	4/2005	Matsliach et al.	709/204
6,938,041	B1	8/2005	Brandow et al.	707/10
7,039,857	B2	5/2006	Beck et al.	715/500.1
7,039,859	B1	5/2006	Sundaresan	715/513
7,076,737	B2	7/2006	Abbott et al.	715/744
7,107,548	B2	9/2006	Shafron	715/826
7,216,300	B2	5/2007	Dang	715/783
7,222,303	B2	5/2007	Oren et al.	715/744
7,257,638	B2	8/2007	Celik et al.	709/231
7,290,217	B2	10/2007	Jones et al.	715/764
7,356,569	B1	4/2008	Kembel et al.	709/217
7,478,142	B1	1/2009	Veditz	709/218
7,574,649	B1	8/2009	Safars et al.	715/200
7,660,868	B1	2/2010	Kembel et al.	709/217
7,756,967	B1	7/2010	Kembel et al.	709/224
7,792,947	B1	9/2010	Kembel et al.	709/224
8,020,083	B1	9/2011	Kembel et al.	715/201
8,346,887	B1	1/2013	Kembel et al.	709/217
8,510,406	B2	8/2013	Kembel et al.	709/217
8,510,407	B1	8/2013	Kembel et al.	709/217
8,521,833	B1	8/2013	Kembel et al.	709/217
8,621,034	B1	12/2013	Kembel et al.	709/217
2001/0011341	A1	8/2001	Hayes, Jr. et al.	712/11
2001/0032220	A1	10/2001	Ven Hoff	
2001/0042107	A1	11/2001	Palm	709/218
2002/0010768	A1	1/2002	Marks et al.	
2002/0032709	A1	3/2002	Gessner	
2002/0054085	A1	5/2002	Taylor et al.	345/744
2002/0065896	A1	5/2002	Burakoff et al.	709/206
2002/0078136	A1	6/2002	Brodsky et al.	709/203
2002/0089526	A1	7/2002	Buxton et al.	345/700
2002/0089536	A1	7/2002	Dang	345/749
2002/0091697	A1	7/2002	Huang et al.	707/10
2002/0130900	A1	9/2002	Davis	345/744
2002/0136167	A1	9/2002	Steele et al.	370/260
2002/0161879	A1	10/2002	Richard	709/223
2003/0051027	A1	3/2003	Aupperle et al.	709/224
2003/0069944	A1	4/2003	Barlock et al.	
2004/0041836	A1	3/2004	Zaner et al.	345/751
2004/0165007	A1	8/2004	Shafron	715/781
2005/0273718	A1	12/2005	Naas	715/745
2008/0040681	A1	2/2008	Synstelien et al.	715/765
2008/0134018	A1	6/2008	Kembel et al.	715/234
2008/0163202	A1	7/2008	Kembel et al.	717/178
2008/0229217	A1	9/2008	Kembel et al.	715/760
2010/0235757	A1	9/2010	Kembel et al.	715/745
2010/0257442	A1	10/2010	Kembel et al.	715/234
2012/0117479	A1	5/2012	Kembel et al.	715/736
2013/0339866	A1	12/2013	Kembel et al.	715/738
2013/0339867	A1	12/2013	Kembel et al.	715/738
2014/0101559	A1	4/2014	Kembel et al.	715/736

OTHER PUBLICATIONS

Rose, Charlton, "Windows: Pop-Up Dialog Boxes", Nov. 28, 1996, retrieved from the Internet <URL: http://sharkysoft.com/archive/1997/jsa/content/008.html>, 2 pages.

No Stated Author, "Window Spawning and Remotes: Window Attributes—Doc JavaScript", created Nov. 18, 1997, retrieved from the Internet <URL: http://webreference.com/js/column7/attributes.html>, 4 pages.

Merrick et al., "Web Interface Definition Language (WIDL)", Sep. 22, 1997, retrieved from the Internet: <URL: http://www.w3.org/TR/NOTE-widl>, 14 pages.

No Stated Author, "CoffeeCup: Uploading Files to Your Server", Dec. 31, 1996, retrieved from the Internet <URL: http://www.coffeecup.com/help/articles/uploading-files-to-your-server/>, 2 pages.

Ellerman, Castedo, "Channel Definition Format (CDF)", World Wide Web Consortium (W3C), available at http://www.w3.org/TR/NOTE-CDFsubmit.html, Mar. 9, 1997, 10 pages.

"HTML 4.0 Specification", World Wide Web Consortium (W3C), available at http://www.w3.org/TR/1998/REC-html40-19980424.html, revised Apr. 24, 1998, 30 pages.

U.S. Appl. No. 60/153,917, filed Sep. 14, 1999, Franco et al.

Patent Application entitled "Parallel Web Sites", U.S. Appl. No. 09/192,633, filed Nov. 16, 1998.

Alexa 1.4.1 Support Pages, 9 pages, downloaded from www.alexa.com/support/index1.html, Jan. 1999.

Alexa General FAQs, 4 pages, downloaded from www.alexa.com/whatisalexa.faq.html#general, Jan. 1999.

"Custom Explorer Bars Give Sites an Edge", 2 pages, downloaded from www.microsoft.com/Windows/Ie/IE5/custom.asp, Jan. 1999.

"Flexibility Across the Web", 2 pages, downloaded from www.microsoft.com/Windows/Ie/IE5/choice.asp, Jan. 1999.

"Web Accessories Overview", 2 pages, downloaded from www.microsoft.com/workshop/...er/accessory/overview/overview.asp, Jan. 1999.

"Browser Extensions Overview", 2 pages, downloaded from www.microsoft.com/workshop/browser/ext/overview/overview.asp, Jan. 1999.

Alexa Technology, 4 pages, downloaded from www.alexa.com/support/technology.html, Jan. 1999.

"Creating Custom Explorer Bars and Desk Bands", 13 pages, downloaded from www.microsoft.com/workshop/browser/ext/overview/Bands.asp, Jan. 1999, 13 pages.

Alexa Internet Tour, 1 page, downloaded from www.alexa.com.whatisalexa.index.html, Jan. 1999.

"Revolutionary Ad Model", Advertise on Alexa, 1 page, downloaded from www.alexa.com/company/advertise/html, Jan. 1999.

"The Alexa Service Appears on Your Desktop in Its Own Window", 1 page, downloaded from www.alexa.com/tour/overview.html, Jan. 1999.

"Know More About the Sites You Visit", 1 page, downloaded from www.alexa.com/tour/site_stats.html, Jan. 1999.

"Find Related Web Sites", 1 page, downloaded from www.alexa.com/tour/related_links.html, Jan. 1999.

"500,000 Sites and Growing", 1 page, downloaded from www.alexa.com/tour/archive/html, Jan. 1999.

"Research Tools at Your Fingertips", 1 page, downloaded from www.alexa.com/tour/eb.html, Jan. 1999.

"Reporting", 1 page, downloaded from www.alexa.com/company/reporting.html, Jan. 1999.

"Alexa Internet's Related Links Integrated into Netscape Browsers", 1 page, downloaded from www.alexa.com/company/netscape.html, Jan. 1999.

"Demographics", 1 page, downloaded from www.alexa.com/company/demographics.html, Jan. 1999.

"Ads Appear in the Pop-up and on the Bar", 1 page, downloaded from www.alexa.com/company/adspecs.html, Jan. 1999.

"Alexa Why Crawl", 1 page, downloaded from www.alexa.com/support/why_crawl.html, Jan. 1999.

GIF Image 590x329 pixels, Alexa, 1 page, downloaded from www.alexa.com/tour/images/alexa_overview.gif, Jan. 1999.

"Its X-treme!", Alexa, PC Magazine: The Best of 1998, 1 page, downloaded from www.zdnet.com/pcmag/special/bestof98/internet5.html, Jan. 1999.

(56) **References Cited**

OTHER PUBLICATIONS

"Search While You Surf", *PC Magazine: Search the Web*, by Sarah Roberts-Witt, 1 page, downloaded from www.zdnet.com/pcmag/features/websearch98/surf.html, Jan. 1999.
Online document; MindSpring, My Yahoo!; © MindSpring Enterprises, Inc.; http://www.mindspring.com/myyahoo/contents.htm; Dec. 1997, pp. 1-16.
Morrison, Michael, XML Unleashed, Sams Publishing, Dec. 21, 1999, 5 pages.
Flanagan, David, "JavaScript: The Definitive Guide", 3rd Edition, O'Reilly, Jun. 1998, 6 pages.
Microsoft Computer Dictionary, Fifth Edition, 2002, Definition of "Web Browser".
Williams, Margot, "Cyberspace Calendars: The Web's Growing Date Base", Nov. 30, 1998, The Washington Post, p. 1.
Bott, Ed., et al., "Special Edition Using Windows 95 with Internet Explorer 4.0", Publisher: Que, Feb. 17, 1998, pp. 585 and 435.
McFedries, Paul, "Windows 98 Unleashed", Publisher: Sams, May 12, 1998, pp. 594-596.
McFedries, Paul, "Windows 98 Unleashed", Publisher: Sams Publishing, May 12, 1998, pp. xix, xx, xxi, xxiv, 2-4, 44-46, 69, 70, 79, 80, 97-116 ,148, 158-163, 251, 551, 787-792, 799-807, 885, 899, 900, and 904-906.
McCrickard, D. Scott, et al., "Supporting Information Awareness Using Animated Widgets", *Proceedings of the 7th USENIX Tel/Tk Conference*, Feb. 14-18, 2000, 12 pages.
"Streaming Internet Technologies", *PR Newswire*, May 18, 1999, 2 pages.
"NewsEdge Delivers the Power of Real-Time News in a Browser", *Business Wire*, Nov. 9, 1998, 3 pages.
McCartney, Terrance Paul, "End-User Construction and Configuration of Distributed Multimedia Applications", *ProQuest Dissertations and Theses*, 1996, 197 pages.
"Microsoft Eyes Marimba's Castanet", by CNET_News.com_ Staff, Dec. 24, 1996, printed from <http://news.cnet.com/Microsoft-eyes-Marimbas-Castanet/2100-1001_3-257491.html>, 2 pages.
Williams, Dennis, "Application Delivery on a Grand Scale", *Network World Fusion*, Mar. 22, 1999, printed from <http://www.networkworld.com/reviews/0322revmarimba.html>, 4 pages.
Whitehead et al., "WEBDAV: IETF Standard for Collaborative Authoring on the Web", Sep. and Oct. 1998, Retrieved from the Internet <URL:ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=722228>, pp. 1-7 as printed.
Spencer, Steven, "JAVA Tip 66: Control browsers from your JAVA application," Jan. 1, 1999, retrieved from http://javaworld.com/article/2077567/core-java/java-tip-66--control-browsers-from-your-java-application.html.
Finai Office Action mailed Jan. 15, 2016 in corresponding U.S. Appl. No. 12/702,156 (37 pages).
Fina Office Action mailed Jan. 20, 2016 in corresponding U.S. Appl. No. 13/965,093 (26 pages).
Final Office Action mailed Jan. 25, 2016 in corresponding U.S. Appl. No. 13/965,067 (35 pages).
Final Office Action mailed Feb. 3, 2016 in corresponding U.S. Appl. No. 14/104,370 (11 pages).
Notice of Allowance mailed Mar. 28, 2016 in corresponding U.S. Appl. No. 12/820,442 (6 pages).
Nielsen, Jakob, "When to Open Web-Based Applications in a New Window", NN/g Nielsen Norman Group, Oct. 15, 1997, downloaded from https://www.nngroup.com/articles/when-to-open-web-based-applications-in-a-new-window/ (5 pages).
Notice of Allowance mailed Apr. 21, 2016 in corresponding U.S. Appl. No. 13/215,874 (6 pages).
Martin-Flatin, Jean-Philippe, "Push vs. Pull in Web-Based Network Management", Proceedings of the Sixth IFIP/IEEE international Symposium on Integrated Network Management, May 1999, (16 pages).

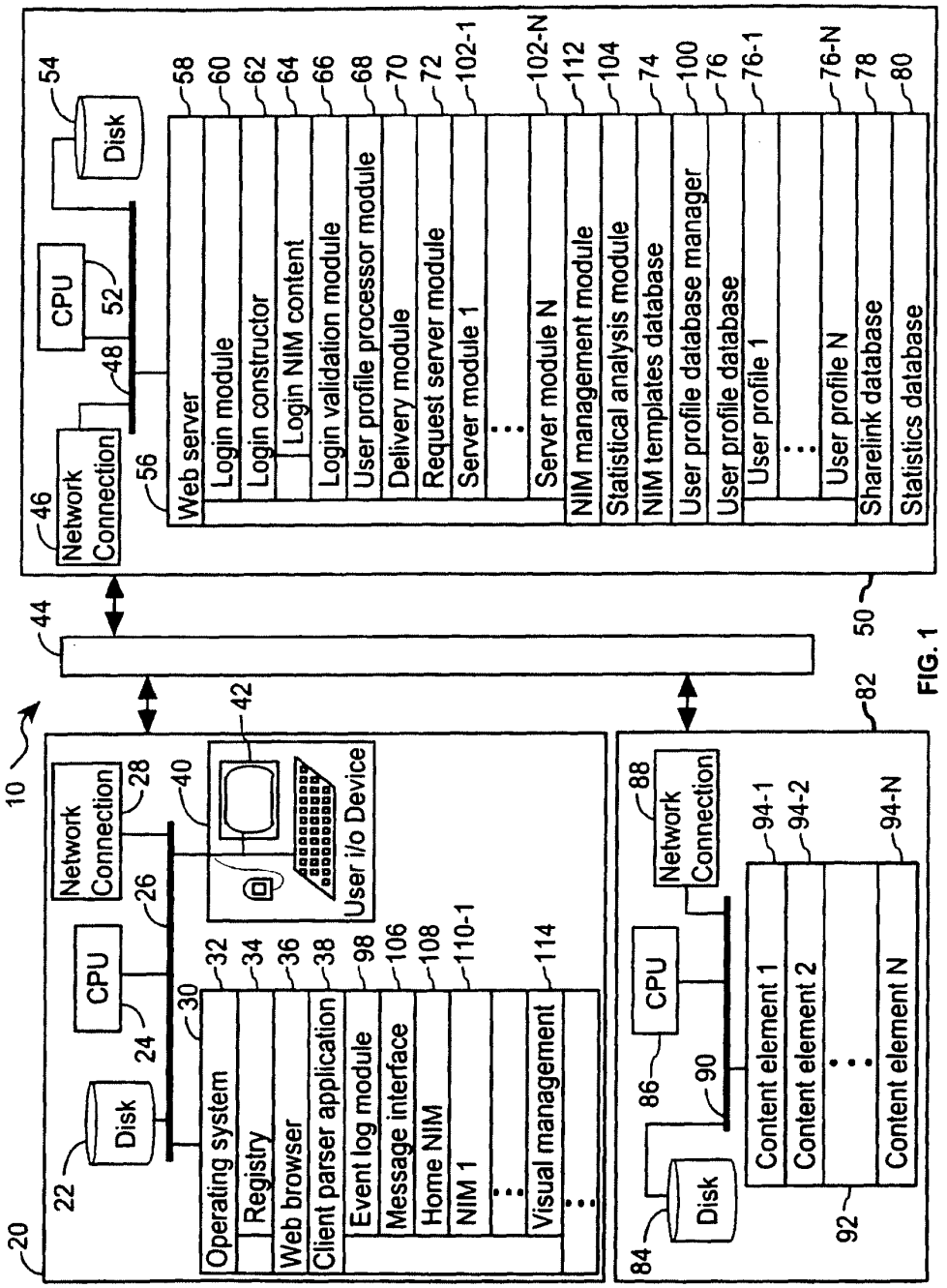

10
20
22
Disk
24
CPU
26
Network Connection
28
30
Operating system
32
Registry
34
Web browser
36
Client parser application
38
Event log module
98
Message interface
106
Home NIM
108
NIM 1
110-1
Visual management
114
40
42
User i/o Device
44
46
Network Connection
48
CPU
52
Disk
54
56
Web server
58
Login module
60
Login constructor
62
Login NIM content
64
Login validation module
66
User profile processor module
68
Delivery module
70
Request server module
72
Server module 1
102-1
Server module N
102-N
NIM management module
112
Statistical analysis module
104
NIM templates database
74
User profile database manager
100
User profile database
76
User profile 1
76-1
User profile N
76-N
Sharelink database
78
Statistics database
80
50
82
84
Disk
86
CPU
88
Network Connection
90
92
Content element 1
94-1
Content element 2
94-2
Content element N
94-N

FIG. 1

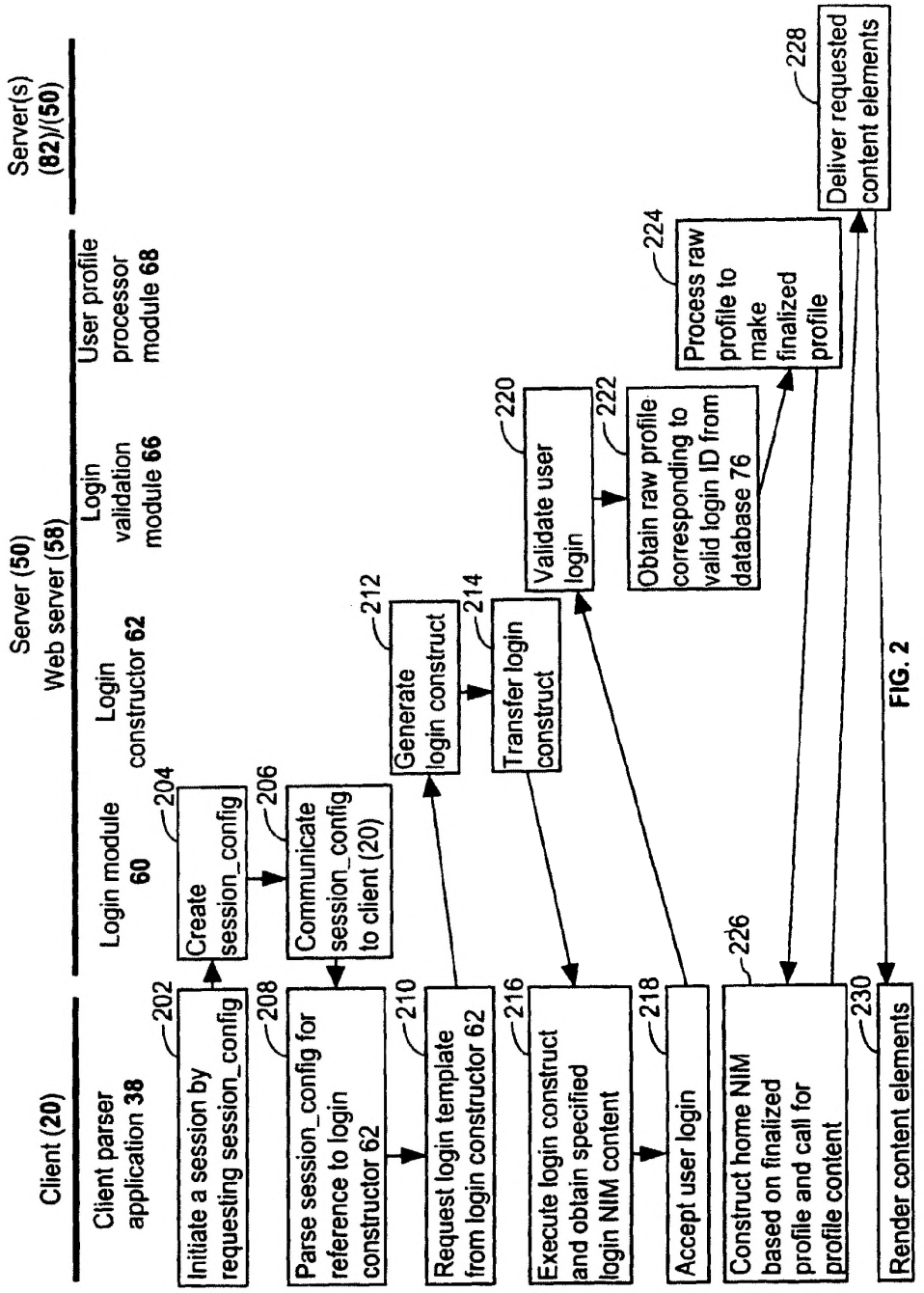

Client (20)
Client parser application 38
Server (50)
Web server (58)
Login module 60
Login constructor 62
Login validation module 66
User profile processor module 68
Server(s) (82)/(50)
202
Initiate a session by requesting session_config
204
Create session_config
206
Communicate session_config to client (20)
208
Parse session_config for reference to login constructor 62
210
Request login template from login constructor 62
212
Generate login construct
214
Transfer login construct
216
Execute login construct and obtain specified login NIM content
218
Accept user login
220
Validate user login
222
Obtain raw profile corresponding to valid login ID from database 76
224
Process raw profile to make finalized profile
226
Construct home NIM based on finalized profile and call for profile content
228
Deliver requested content elements
230
Render content elements

FIG. 2

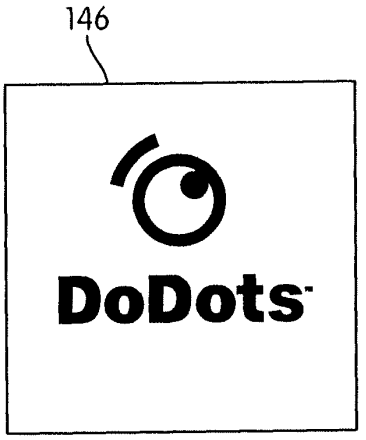

146
DoDots

FIG. 3A

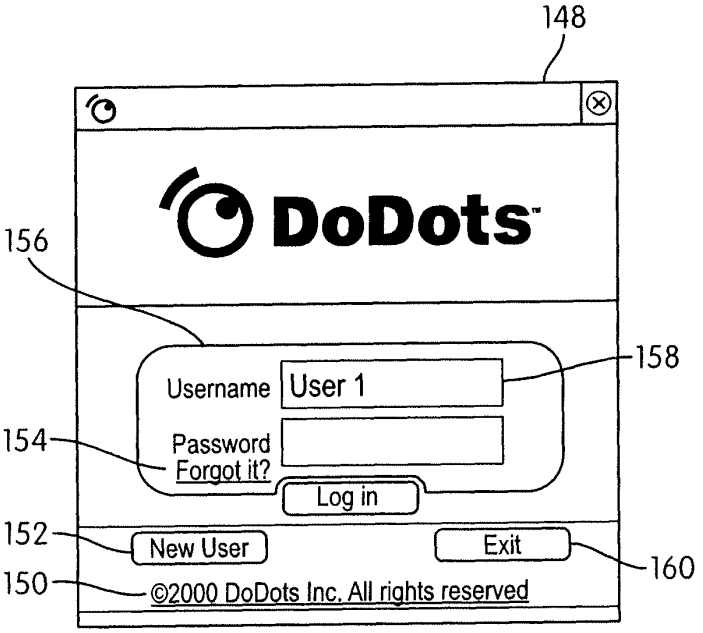

148
DoDots
156
158
Username
User 1
154
Password
Forgot it?
Log in
152
New User
Exit
160
150
©2000 DoDots Inc. All rights reserved

FIG. 3B

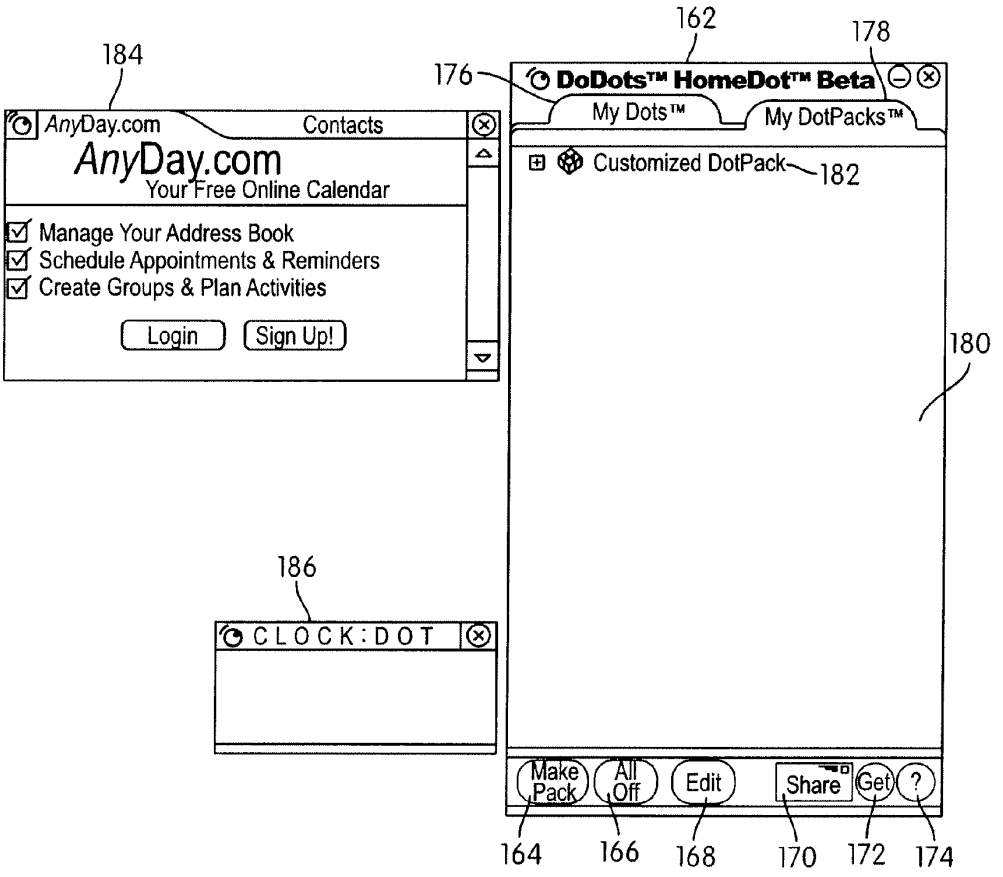

184
AnyDay.com
Contacts
AnyDay.com
Your Free Online Calendar
Manage Your Address Book
Schedule Appointments & Reminders
Create Groups & Plan Activities
Login
Sign Up!
176
162
178
DoDots™ HomeDot™ Beta
My Dots™
My DotPacks™
Customized DotPack
182
180
186
CLOCK:DOT
Make Pack
All Off
Edit
Share
Get
?
164
166
168
170
172
174

FIG. 4

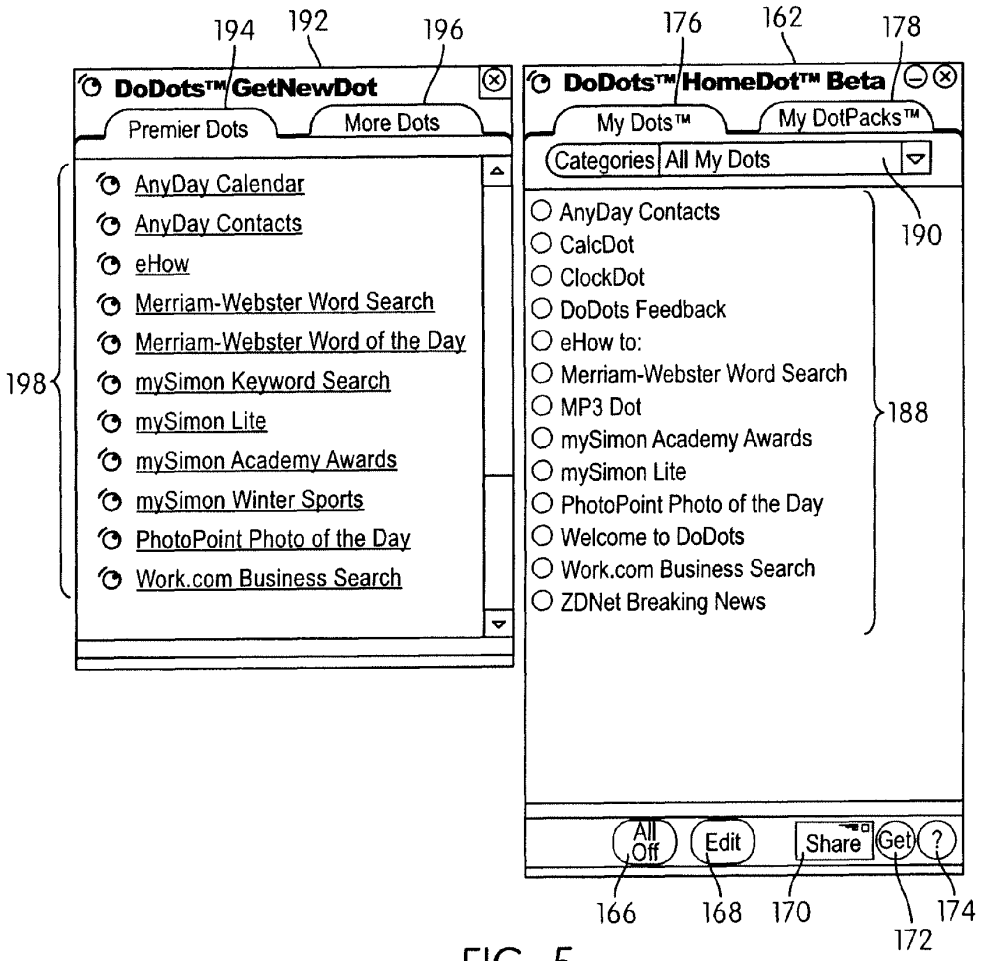

194
192
196
DoDots™ GetNewDot
Premier Dots
More Dots
AnyDay Calendar
AnyDay Contacts
eHow
Merriam-Webster Word Search
Merriam-Webster Word of the Day
mySimon Keyword Search
mySimon Lite
mySimon Academy Awards
mySimon Winter Sports
PhotoPoint Photo of the Day
Work.com Business Search
198
176
162
178
DoDots™ HomeDot™ Beta
My Dots™
My DotPacks™
Categories
All My Dots
190
AnyDay Contacts
CalcDot
ClockDot
DoDots Feedback
eHow to:
Merriam-Webster Word Search
MP3 Dot
mySimon Academy Awards
mySimon Lite
PhotoPoint Photo of the Day
Welcome to DoDots
Work.com Business Search
ZDNet Breaking News
188
All Off
Edit
Share
Get
?
166
168
170
172
174

FIG. 5

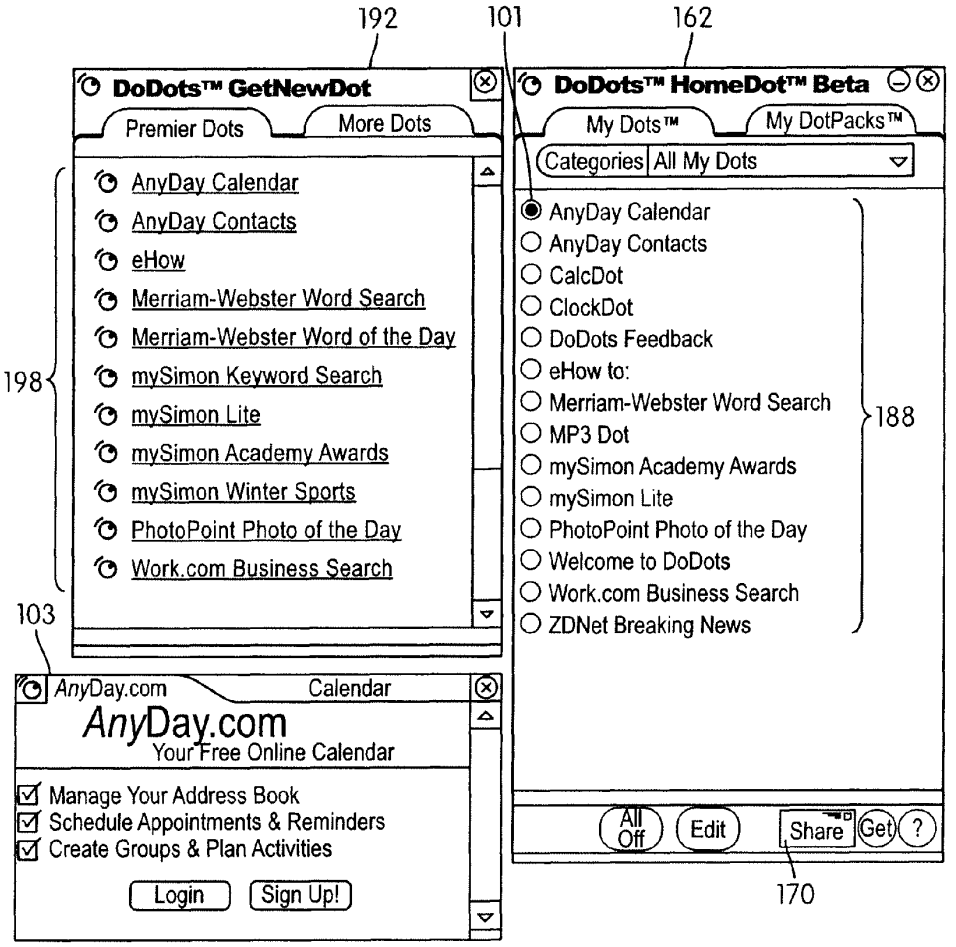

192
101
162
DoDots™ GetNewDot
Premier Dots
More Dots
AnyDay Calendar
AnyDay Contacts
eHow
Merriam-Webster Word Search
Merriam-Webster Word of the Day
mySimon Keyword Search
mySimon Lite
mySimon Academy Awards
mySimon Winter Sports
PhotoPoint Photo of the Day
Work.com Business Search
198
103
AnyDay.com
Calendar
AnyDay.com
Your Free Online Calendar
Manage Your Address Book
Schedule Appointments & Reminders
Create Groups & Plan Activities
Login
Sign Up!
DoDots™ HomeDot™ Beta
My Dots™
My DotPacks™
Categories
All My Dots
AnyDay Calendar
AnyDay Contacts
CalcDot
ClockDot
DoDots Feedback
eHow to:
Merriam-Webster Word Search
MP3 Dot
mySimon Academy Awards
mySimon Lite
PhotoPoint Photo of the Day
Welcome to DoDots
Work.com Business Search
ZDNet Breaking News
188
All Off
Edit
Share
Get
?
170

FIG. 6

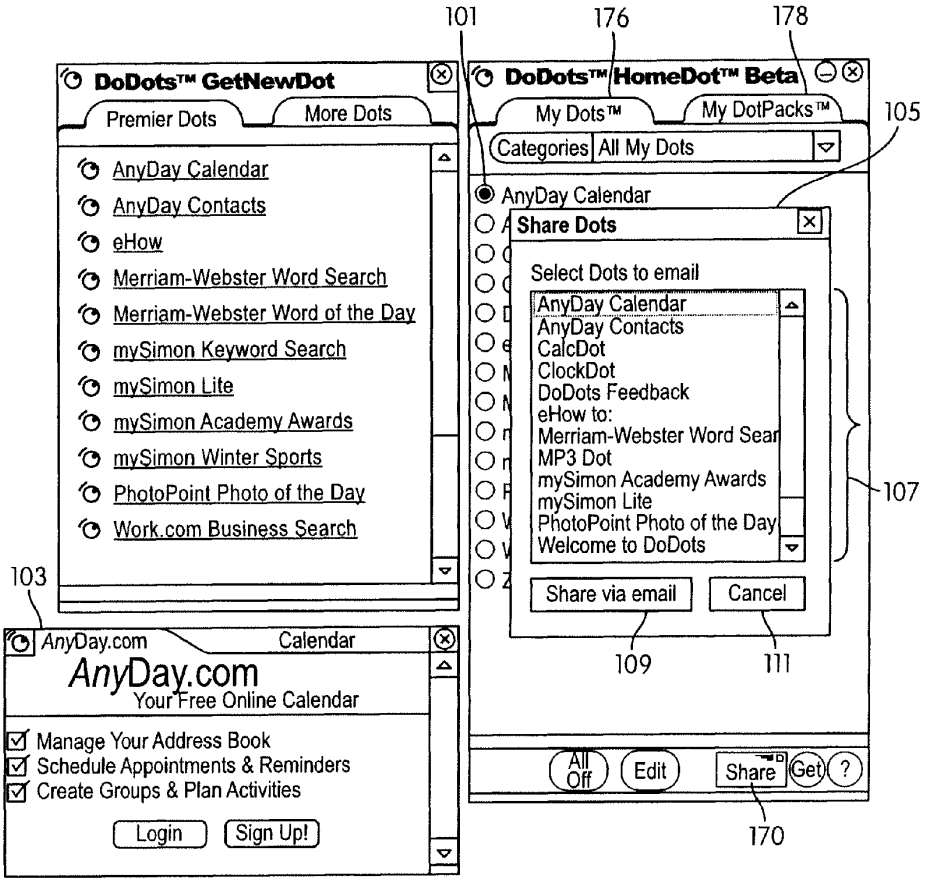

101
176
178
DoDots™ GetNewDot
Premier Dots
More Dots
AnyDay Calendar
AnyDay Contacts
eHow
Merriam-Webster Word Search
Merriam-Webster Word of the Day
mySimon Keyword Search
mySimon Lite
mySimon Academy Awards
mySimon Winter Sports
PhotoPoint Photo of the Day
Work.com Business Search
DoDots™ HomeDot™ Beta
My Dots™
My DotPacks™
105
Categories
All My Dots
AnyDay Calendar
Share Dots
Select Dots to email
AnyDay Calendar
AnyDay Contacts
CalcDot
ClockDot
DoDots Feedback
eHow to:
Merriam-Webster Word Sear
MP3 Dot
mySimon Academy Awards
mySimon Lite
PhotoPoint Photo of the Day
Welcome to DoDots
107
Share via email
Cancel
109
111
103
AnyDay.com
Calendar
AnyDay.com
Your Free Online Calendar
Manage Your Address Book
Schedule Appointments & Reminders
Create Groups & Plan Activities
Login
Sign Up!
All Off
Edit
Share
Get
?
170

FIG. 7

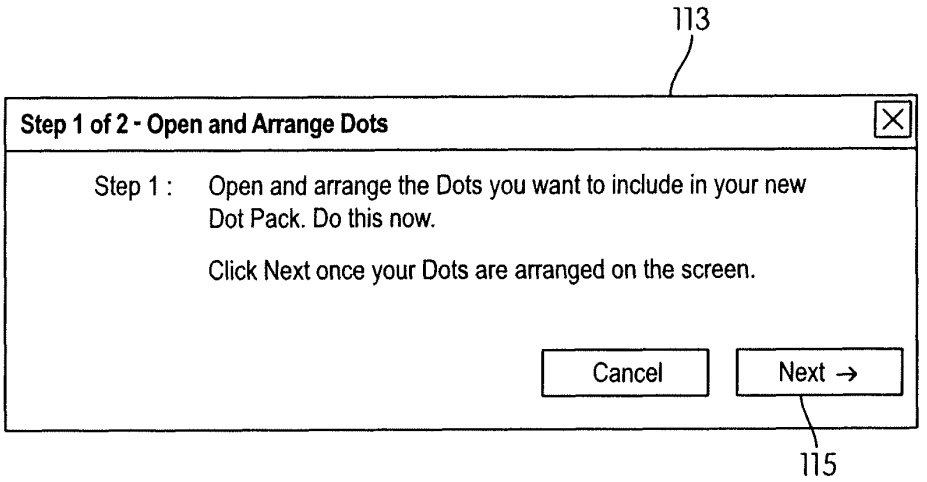

113
Step 1 of 2 - Open and Arrange Dots
Step 1 :
Open and arrange the Dots you want to include in your new Dot Pack. Do this now.
Click Next once your Dots are arranged on the screen.
Cancel
Next →
115

FIG. 8A

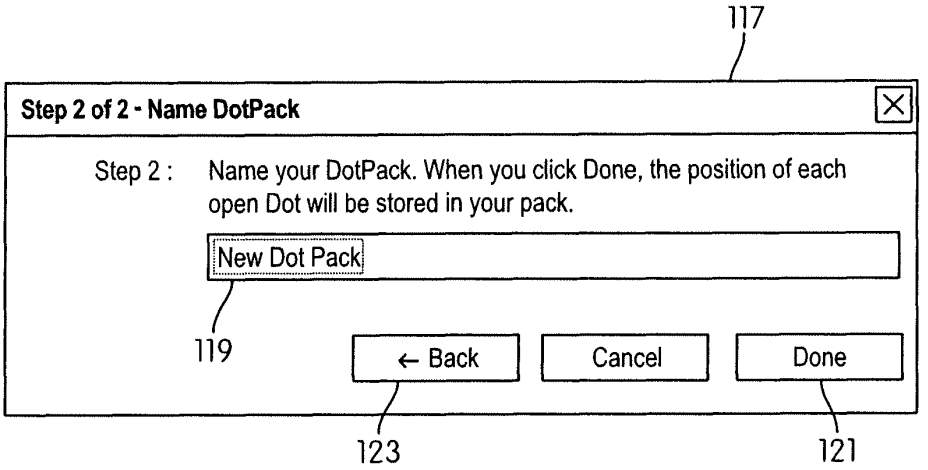

117
Step 2 of 2 - Name DotPack
Step 2 :
Name your DotPack. When you click Done, the position of each open Dot will be stored in your pack.
New Dot Pack
119
← Back
Cancel
Done
123
121

FIG. 8B

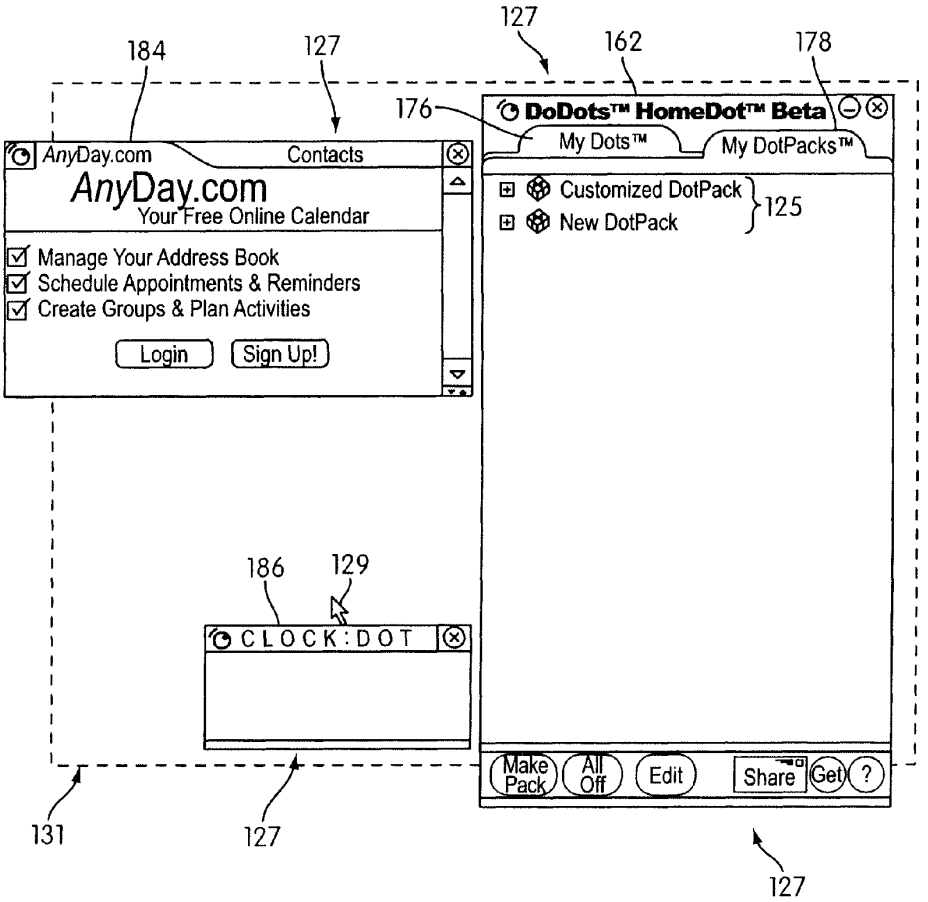

127
184
127
162
178
176
DoDots™ HomeDot™ Beta
My Dots™
My DotPacks™
AnyDay.com
Contacts
AnyDay.com
Your Free Online Calendar
Manage Your Address Book
Schedule Appointments & Reminders
Create Groups & Plan Activities
Login
Sign Up!
Customized DotPack
New DotPack
125
186
129
CLOCK:DOT
Make Pack
All Off
Edit
Share
Get
?
131
127
127

FIG. 9A

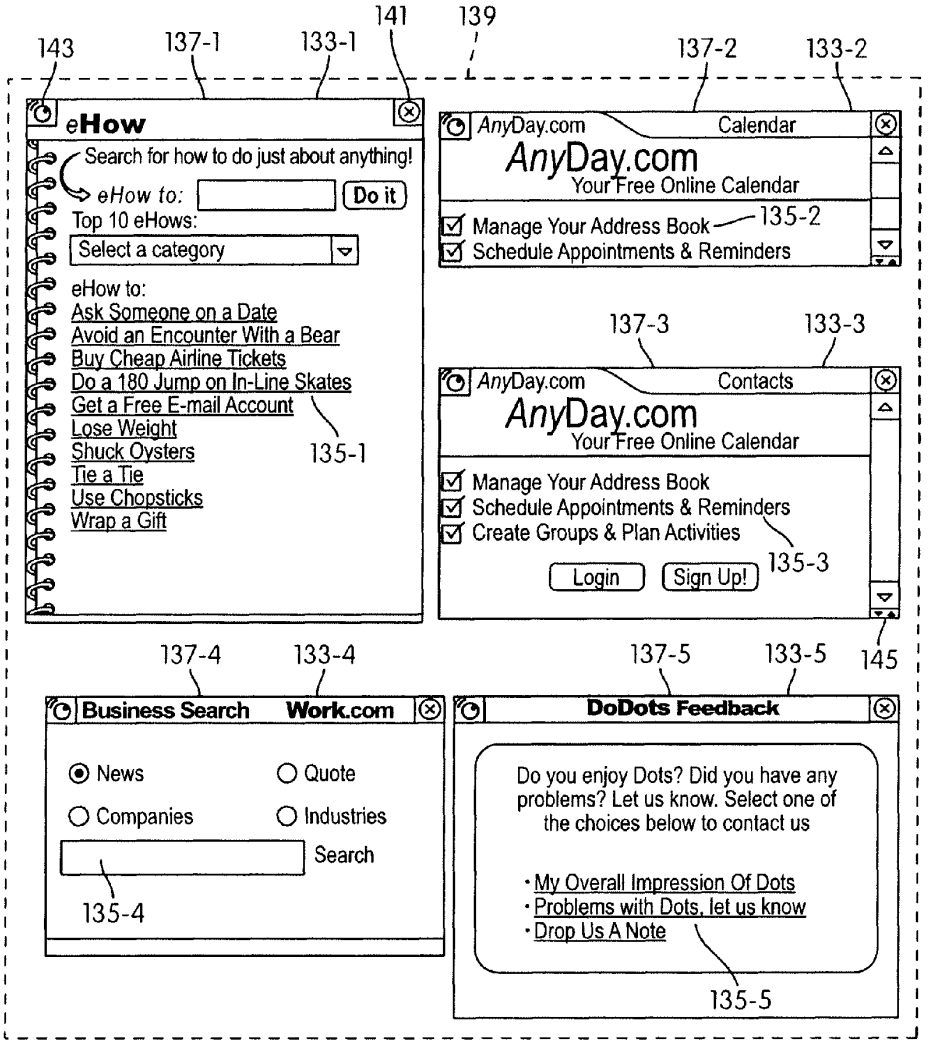

143
137-1
133-1
141
139
137-2
133-2
eHow
Search for how to do just about anything!
eHow to:
Do it
Top 10 eHows:
Select a category
eHow to:
Ask Someone on a Date
Avoid an Encounter With a Bear
Buy Cheap Airline Tickets
Do a 180 Jump on In-Line Skates
Get a Free E-mail Account
Lose Weight
Shuck Oysters
Tie a Tie
Use Chopsticks
Wrap a Gift
135-1
AnyDay.com
Calendar
AnyDay.com
Your Free Online Calendar
Manage Your Address Book
Schedule Appointments & Reminders
135-2
137-3
133-3
AnyDay.com
Contacts
AnyDay.com
Your Free Online Calendar
Manage Your Address Book
Schedule Appointments & Reminders
Create Groups & Plan Activities
Login
Sign Up!
135-3
145
137-4
133-4
Business Search
Work.com
News
Quote
Companies
Industries
Search
135-4
137-5
133-5
DoDots Feedback
Do you enjoy Dots? Did you have any problems? Let us know. Select one of the choices below to contact us
• My Overall Impression Of Dots
• Problems with Dots, let us know
• Drop Us A Note
135-5

FIG. 9B

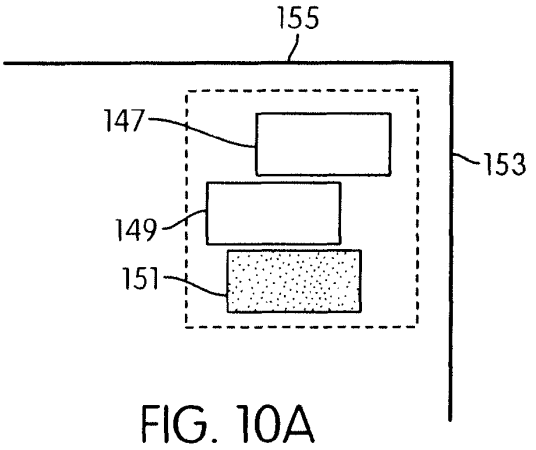


FIG. 10A

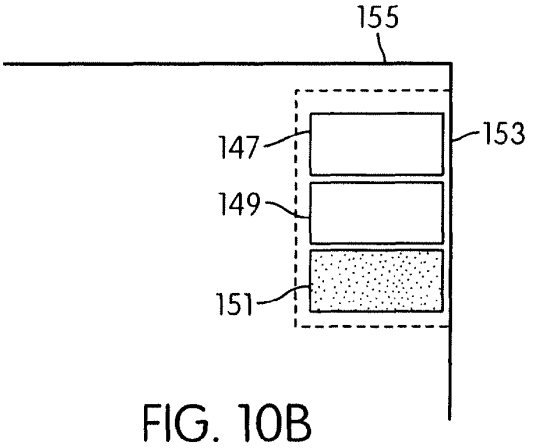


FIG. 10B

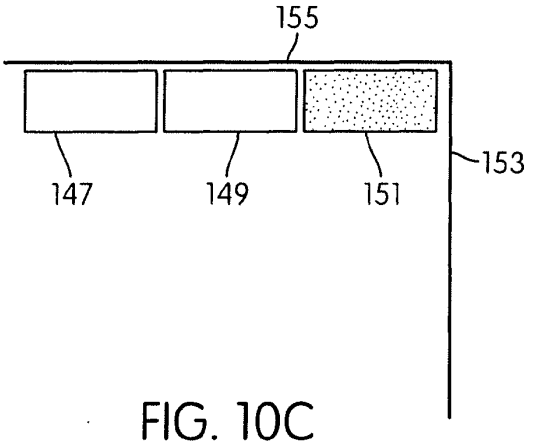


FIG. 10C

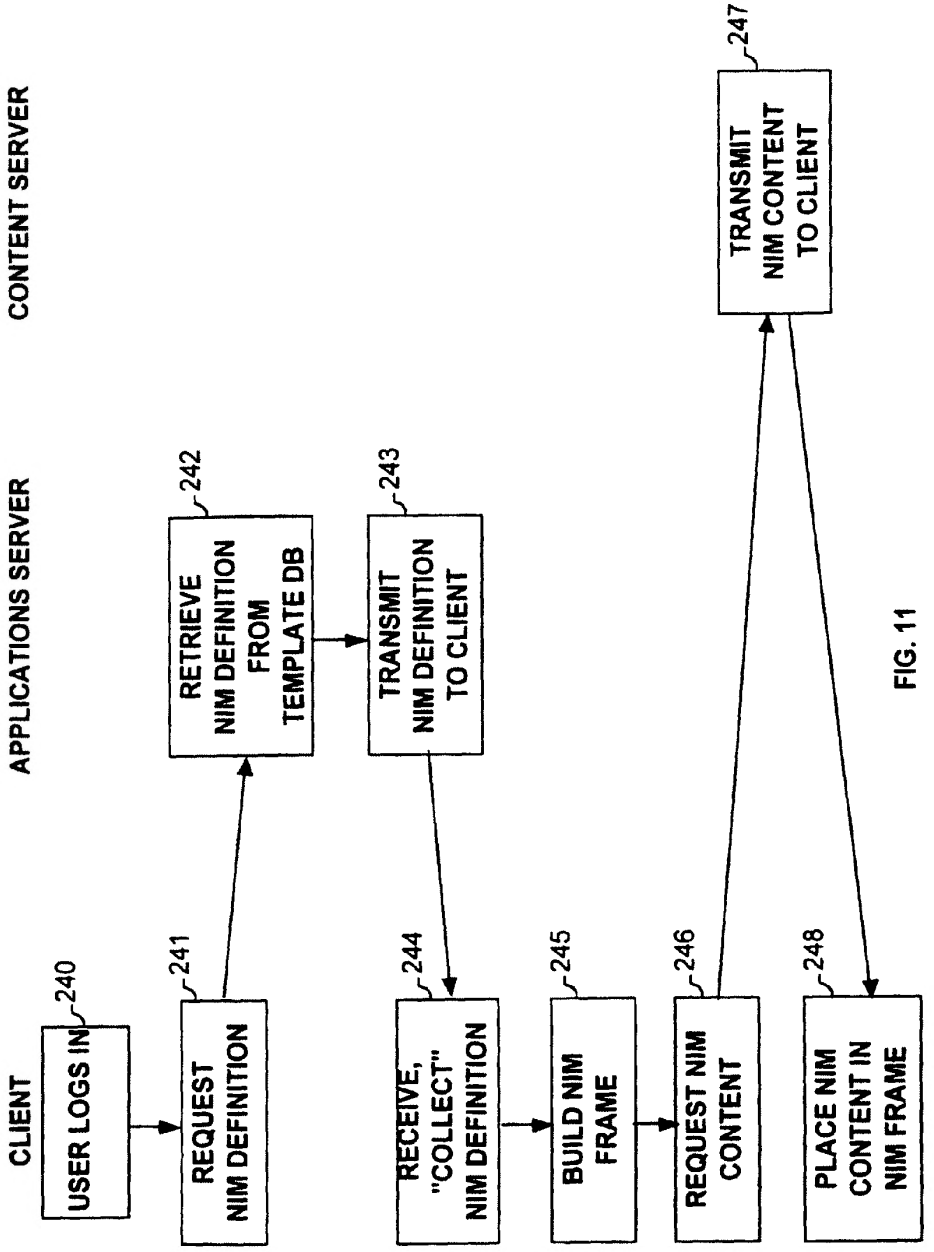

CLIENT
APPLICATIONS SERVER
CONTENT SERVER
240
USER LOGS IN
241
REQUEST NIM DEFINITION
242
RETRIEVE NIM DEFINITION FROM TEMPLATE DB
243
TRANSMIT NIM DEFINITION TO CLIENT
244
RECEIVE, "COLLECT" NIM DEFINITION
245
BUILD NIM FRAME
246
REQUEST NIM CONTENT
247
TRANSMIT NIM CONTENT TO CLIENT
248
PLACE NIM CONTENT IN NIM FRAME

FIG. 11

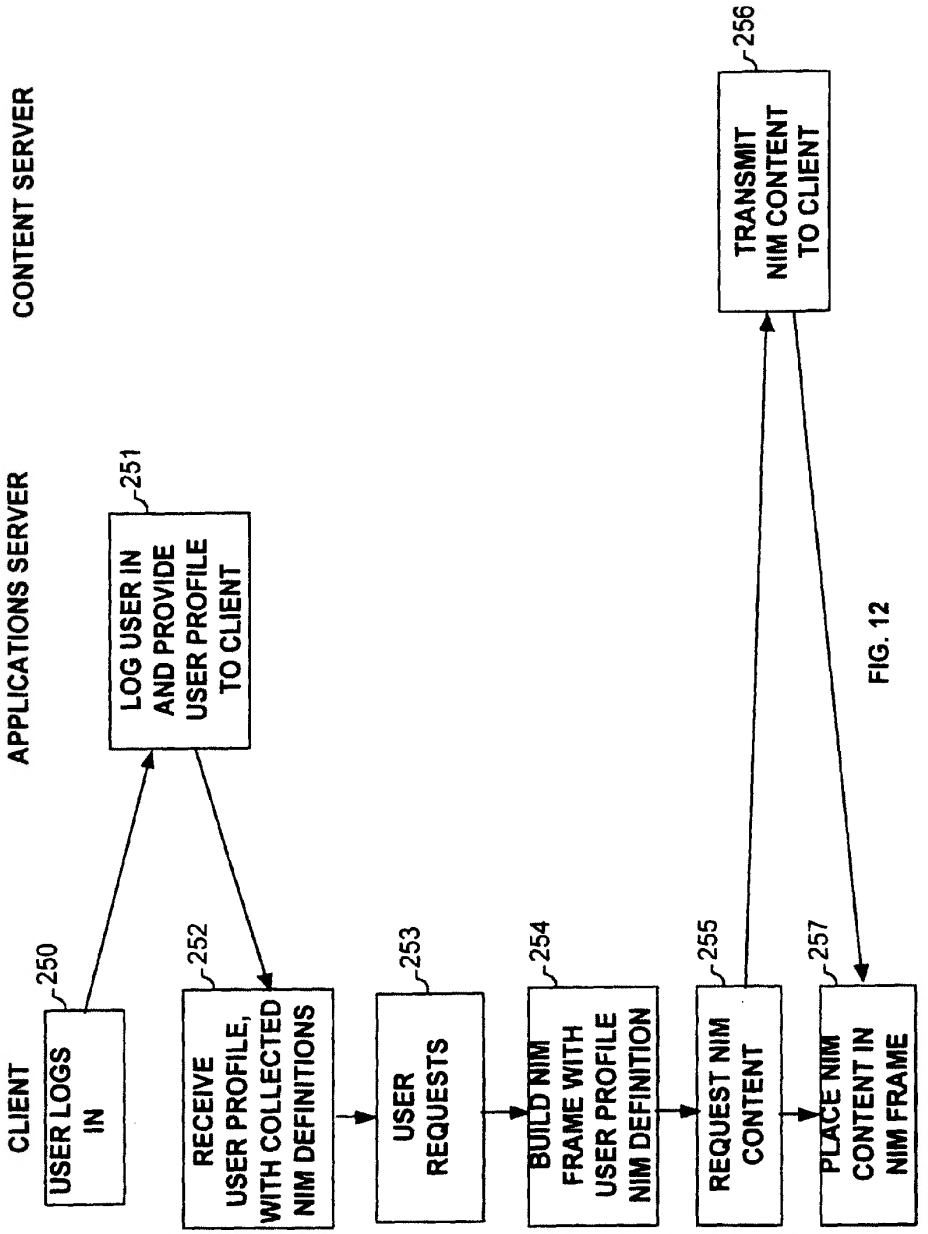

CLIENT
APPLICATIONS SERVER
CONTENT SERVER
250
USER LOGS IN
251
LOG USER IN AND PROVIDE USER PROFILE TO CLIENT
252
RECEIVE USER PROFILE, WITH COLLECTED NIM DEFINITIONS
253
USER REQUESTS
254
BUILD NIM FRAME WITH USER PROFILE NIM DEFINITION
255
REQUEST NIM CONTENT
256
TRANSMIT NIM CONTENT TO CLIENT
257
PLACE NIM CONTENT IN NIM FRAME

FIG. 12

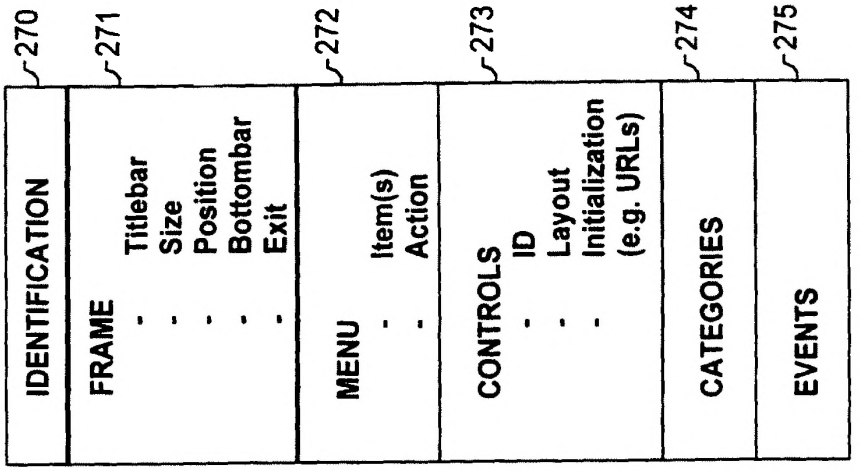

IDENTIFICATION
270
FRAME
- Titlebar
- Size
- Position
- Bottombar
- Exit
271
MENU
- Item(s)
- Action
272
CONTROLS
- ID
- Layout
- Initialization (e.g. URLs)
273
CATEGORIES
274
EVENTS
275

FIG. 13

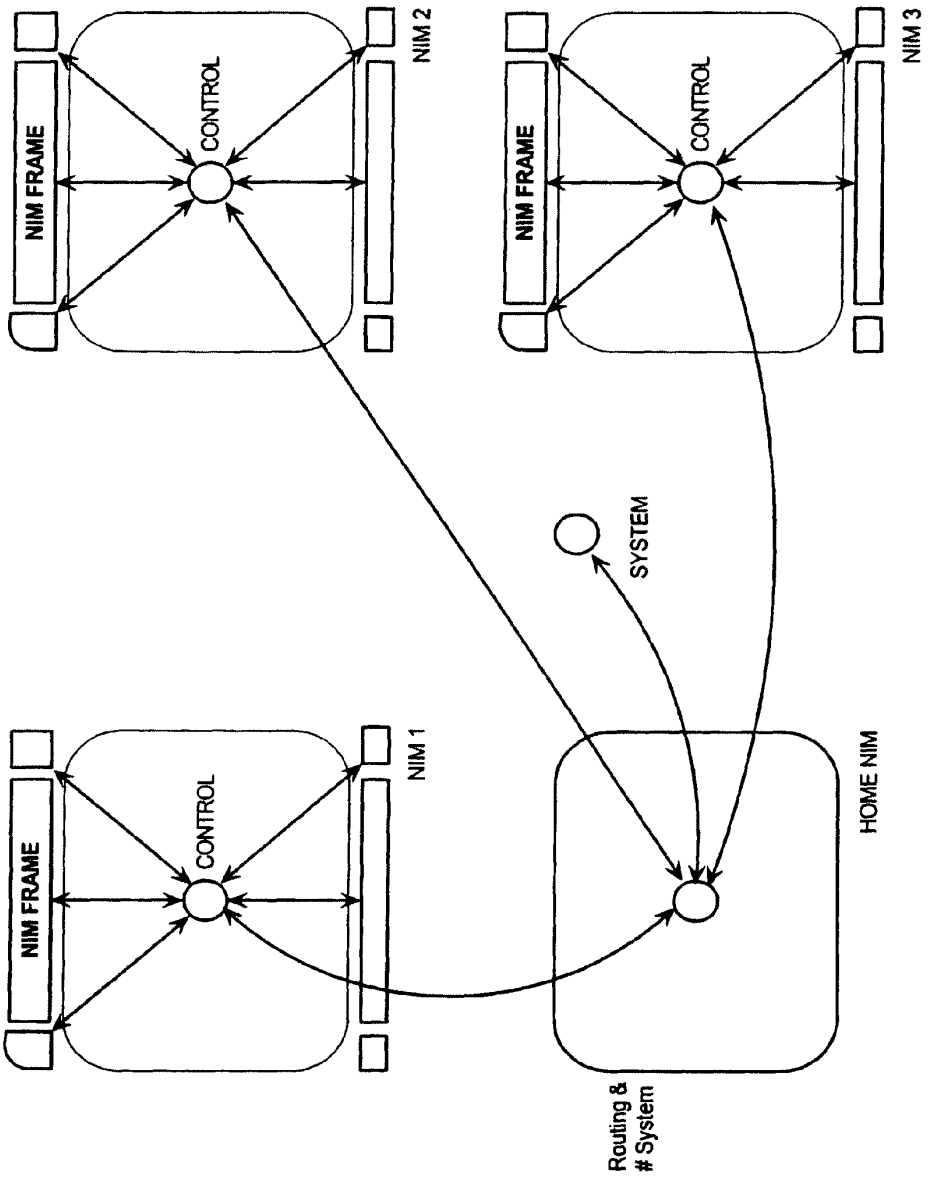

NIM FRAME
CONTROL
NIM 1
NIM FRAME
CONTROL
NIM 2
NIM FRAME
CONTROL
NIM 3
SYSTEM
HOME NIM
Routing &
System

FIG. 14

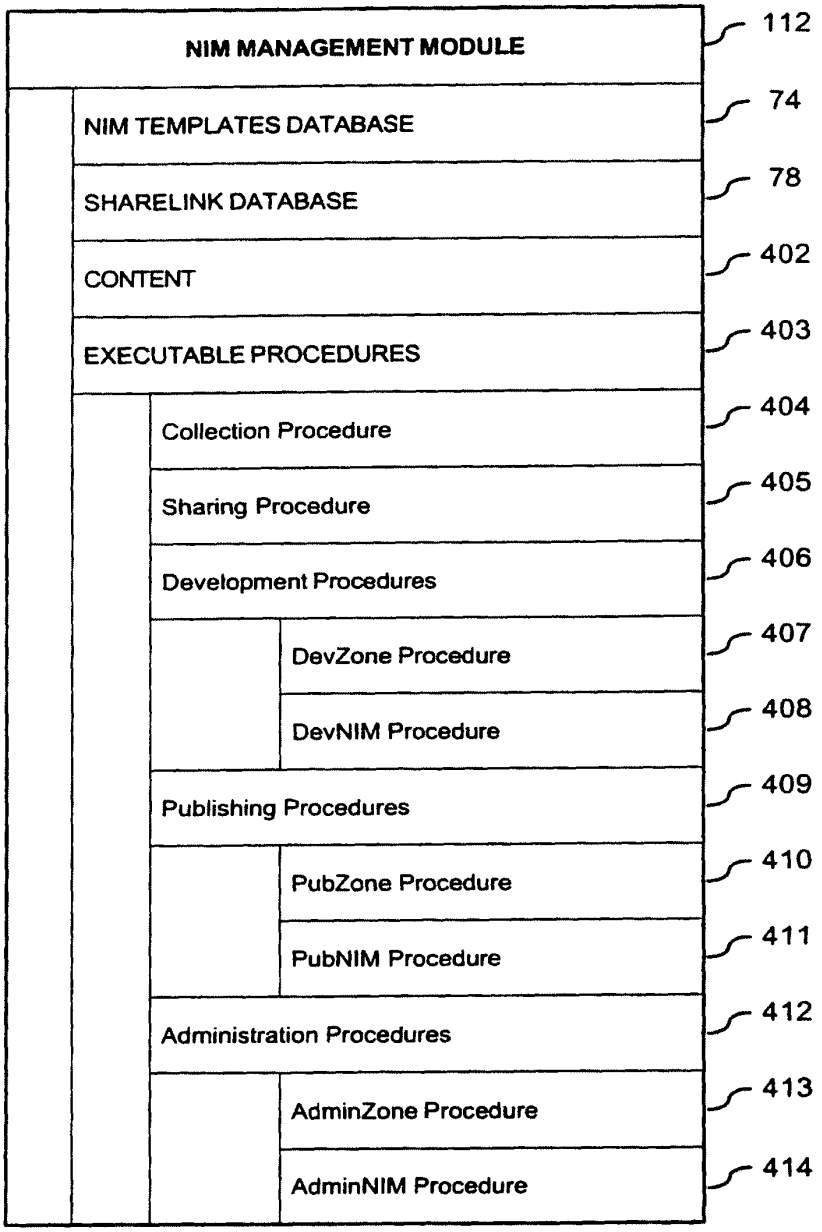

NIM MANAGEMENT MODULE
112
NIM TEMPLATES DATABASE
74
SHARELINK DATABASE
78
CONTENT
402
EXECUTABLE PROCEDURES
403
Collection Procedure
404
Sharing Procedure
405
Development Procedures
406
DevZone Procedure
407
DevNIM Procedure
408
Publishing Procedures
409
PubZone Procedure
410
PubNIM Procedure
411
Administration Procedures
412
AdminZone Procedure
413
AdminNIM Procedure
414

FIG. 15

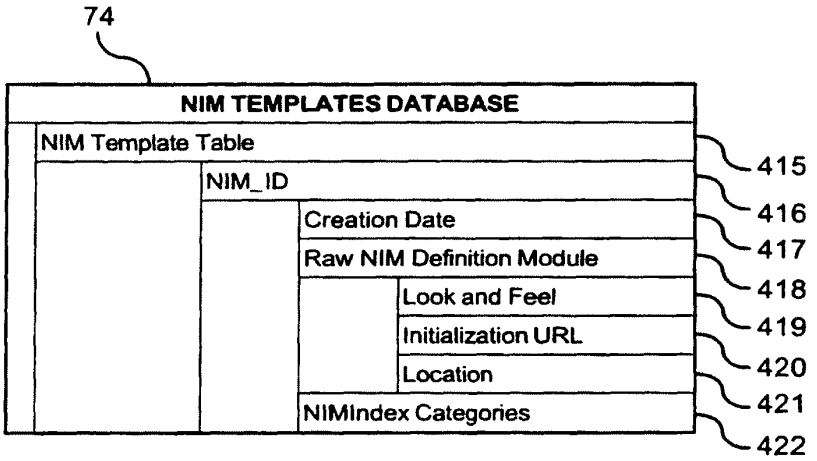

74
NIM TEMPLATES DATABASE
NIM Template Table
NIM_ID
Creation Date
Raw NIM Definition Module
Look and Feel
Initialization URL
Location
NIMIndex Categories
415
416
417
418
419
420
421
422

FIG. 16

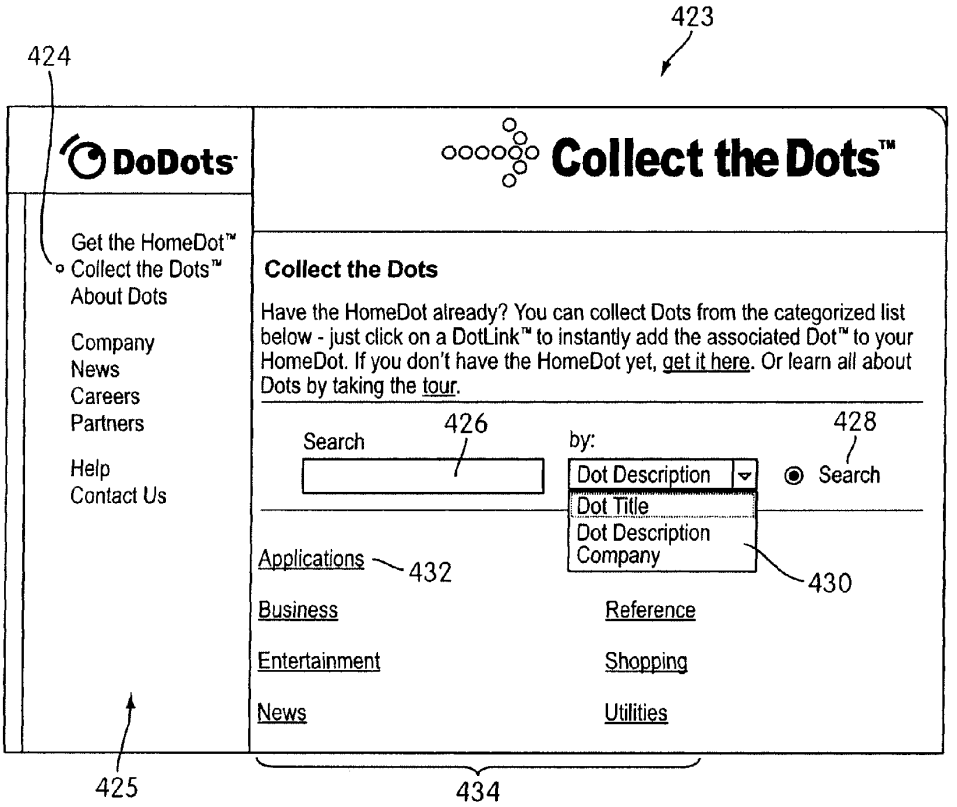

423
424
DoDots
Collect the Dots™
Get the HomeDot™
Collect the Dots™
About Dots
Company
News
Careers
Partners
Help
Contact Us
Collect the Dots
Have the HomeDot already? You can collect Dots from the categorized list below - just click on a DotLink™ to instantly add the associated Dot™ to your HomeDot. If you don't have the HomeDot yet, get it here. Or learn all about Dots by taking the tour.
Search
426
by:
428
Dot Description
Search
Dot Title
Dot Description
Company
430
Applications
432
Business
Reference
Entertainment
Shopping
News
Utilities
425
434

FIG. 17

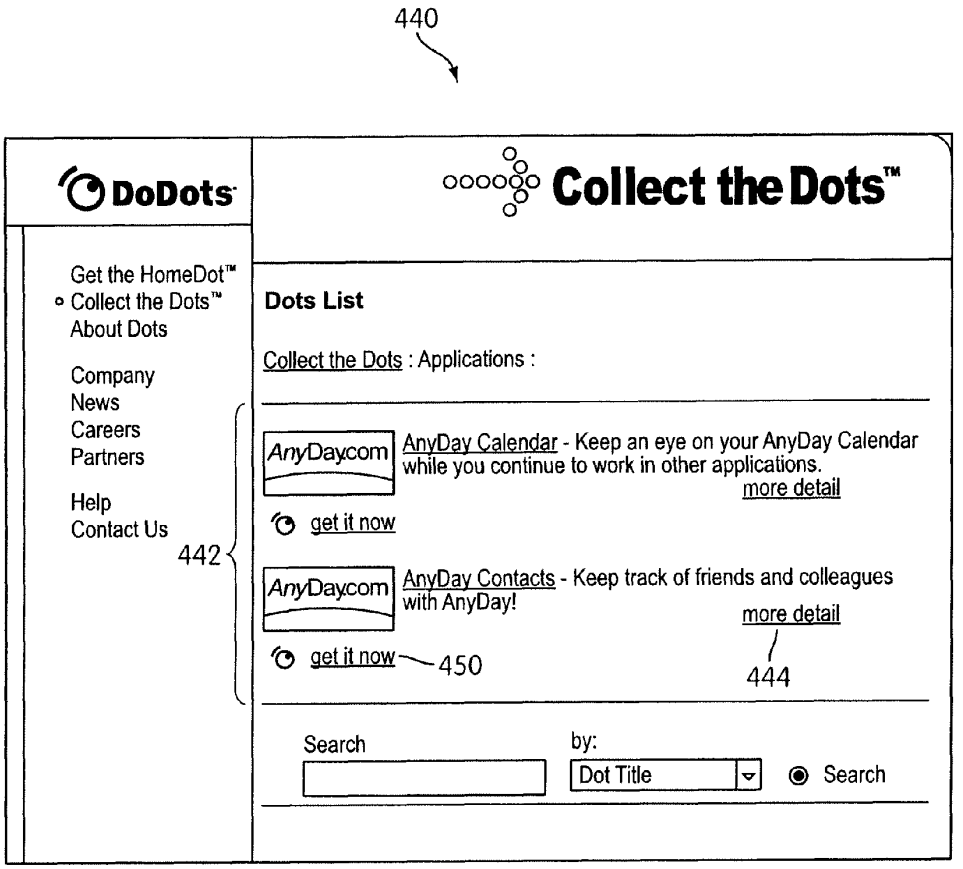

440
DoDots
Collect the Dots™
Get the HomeDot™
Collect the Dots™
About Dots
Company
News
Careers
Partners
Help
Contact Us
442
Dots List
Collect the Dots : Applications :
AnyDay.com
AnyDay Calendar - Keep an eye on your AnyDay Calendar while you continue to work in other applications.
more detail
get it now
AnyDay.com
AnyDay Contacts - Keep track of friends and colleagues with AnyDay!
more detail
get it now
450
444
Search
by:
Dot Title
Search

FIG. 18

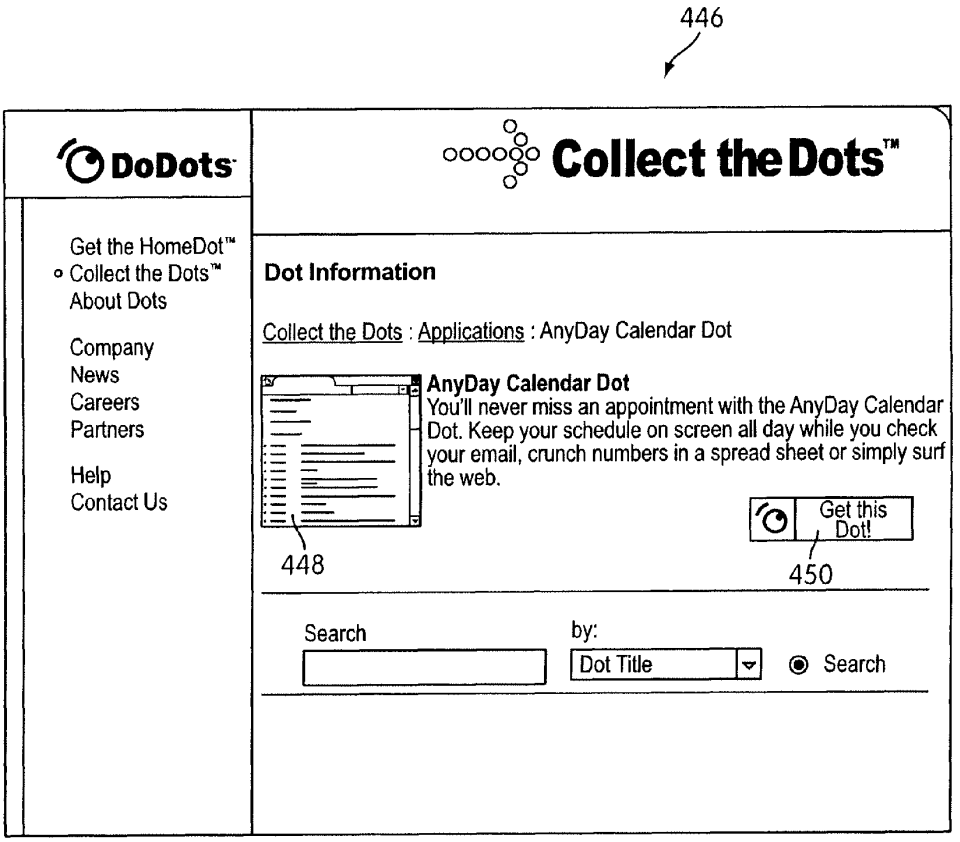

446
DoDots
Collect the Dots™
Get the HomeDot™
Collect the Dots™
About Dots
Company
News
Careers
Partners
Help
Contact Us
Dot Information
Collect the Dots : Applications : AnyDay Calendar Dot
AnyDay Calendar Dot
You'll never miss an appointment with the AnyDay Calendar Dot. Keep your schedule on screen all day while you check your email, crunch numbers in a spread sheet or simply surf the web.
Get this Dot!
448
450
Search
by:
Dot Title
Search

FIG. 19

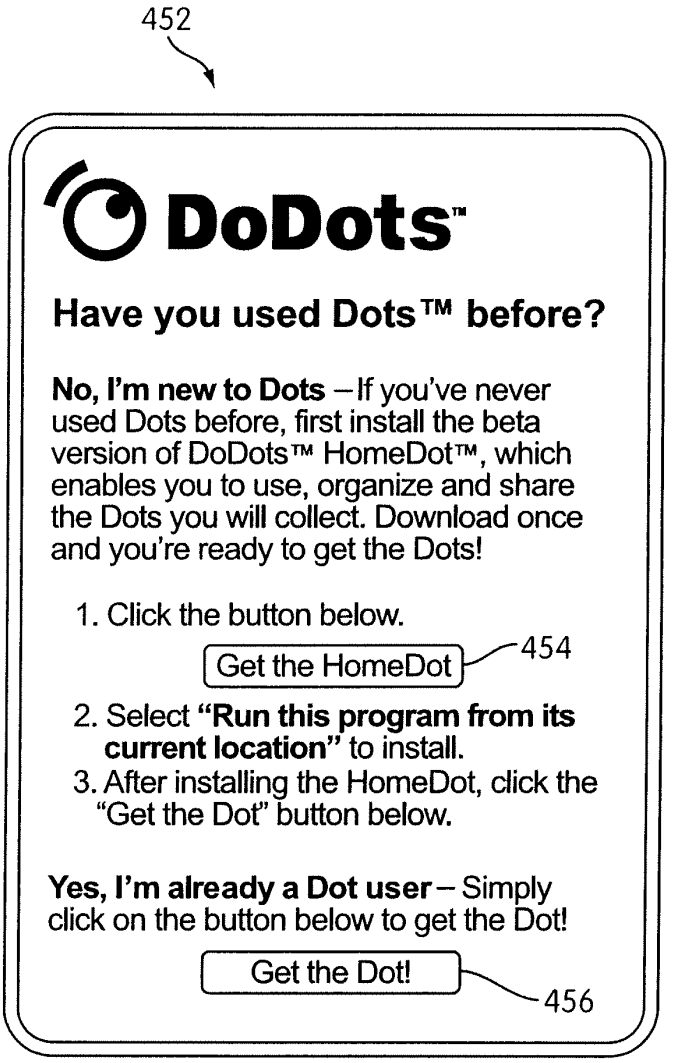

452
DoDots™
Have you used Dots™ before?
No, I'm new to Dots – If you've never used Dots before, first install the beta version of DoDots™ HomeDot™, which enables you to use, organize and share the Dots you will collect. Download once and you're ready to get the Dots!
1. Click the button below.
Get the HomeDot
454
2. Select "Run this program from its current location" to install.
3. After installing the HomeDot, click the "Get the Dot" button below.
Yes, I'm already a Dot user – Simply click on the button below to get the Dot!
Get the Dot!
456

FIG. 20

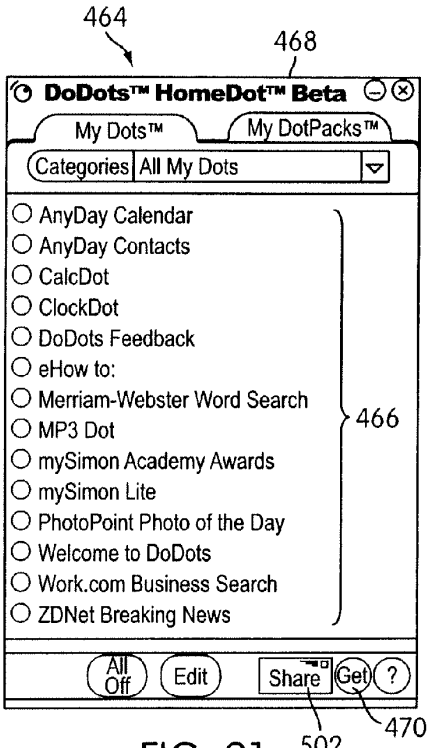


FIG. 21

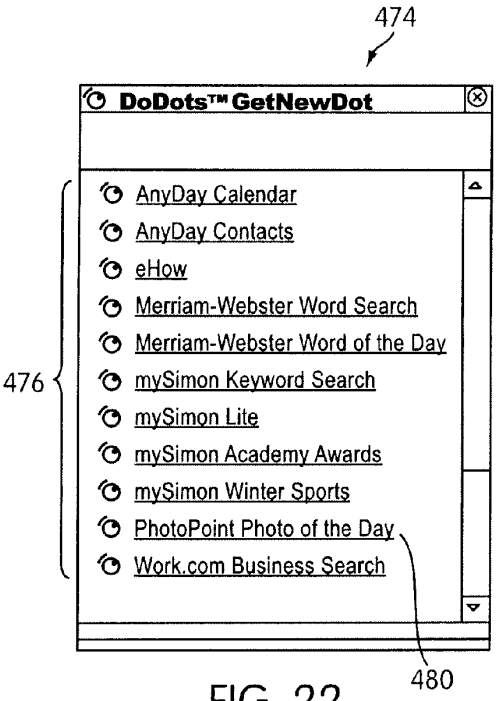


FIG. 22

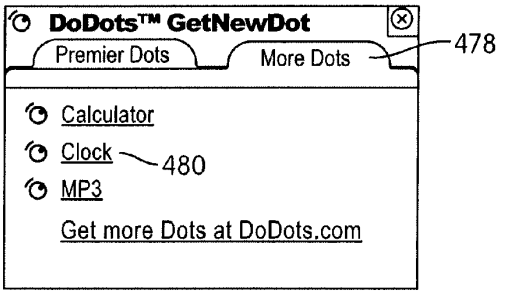


FIG. 23

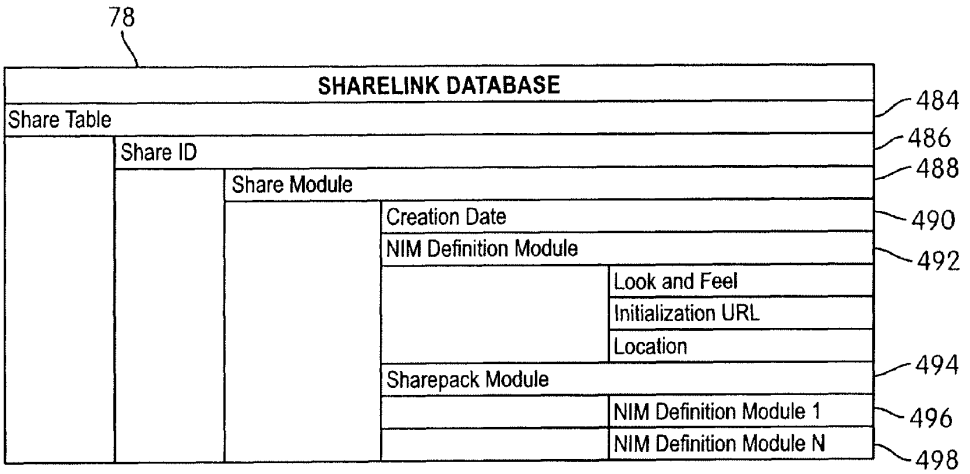

78
SHARELINK DATABASE
Share Table
Share ID
Share Module
Creation Date
NIM Definition Module
Look and Feel
Initialization URL
Location
Sharepack Module
NIM Definition Module 1
NIM Definition Module N
484
486
488
490
492
494
496
498

FIG. 24

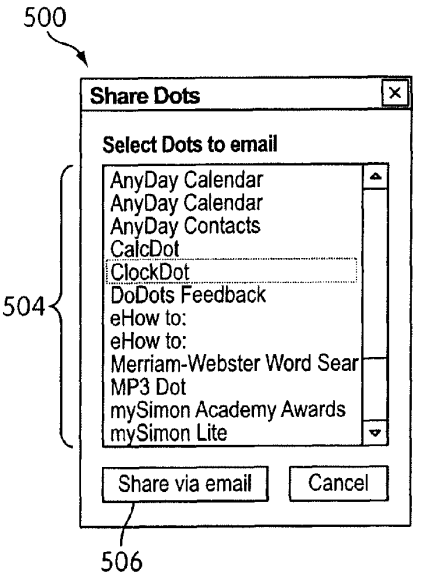

500
Share Dots
Select Dots to email
AnyDay Calendar
AnyDay Calendar
AnyDay Contacts
CalcDot
ClockDot
DoDots Feedback
eHow to:
eHow to:
Merriam-Webster Word Sear
MP3 Dot
mySimon Academy Awards
mySimon Lite
504
Share via email
Cancel
506

FIG. 25

510

DoDots

Get the HomeDot™
Collect the Dots™
About Dots

Company
News
Careers
Partners

Help
Contact Us
Log Out

Collect the Dots™

Your Dot List FAQ Dot Design Guide Development Support

Your Dot List

Welcome to your Dot Developer Zone, *User 2*. Your Dot List contains a list of all of your Dots. You can preview and modify your Dots from this list, as well as sort them using the sort arrows (⇕) above the heading you wish to sort by. You can also create new Dots.

522 ◉ Create a Dot 524

STAGE ⇕	DOT NAME ⇕	DATE CREATED ⇕	ACTION
	DoDots DevDot	03/11/2000	preview modify
	mySimon Tax Day	03/28/2000	preview modify
	Zupit Radio	04/04/2000	preview modify
•	MP3 Dot	03/04/2000	preview modify
•	CalcDot	03/04/2000	preview modify
•	ClockDot	03/04/2000	preview modify

◉ Create a Dot

516 518 520 522 514

The preview function will open the Dot on you desktop for preview and testing purposes.

development • public

FIG. 26

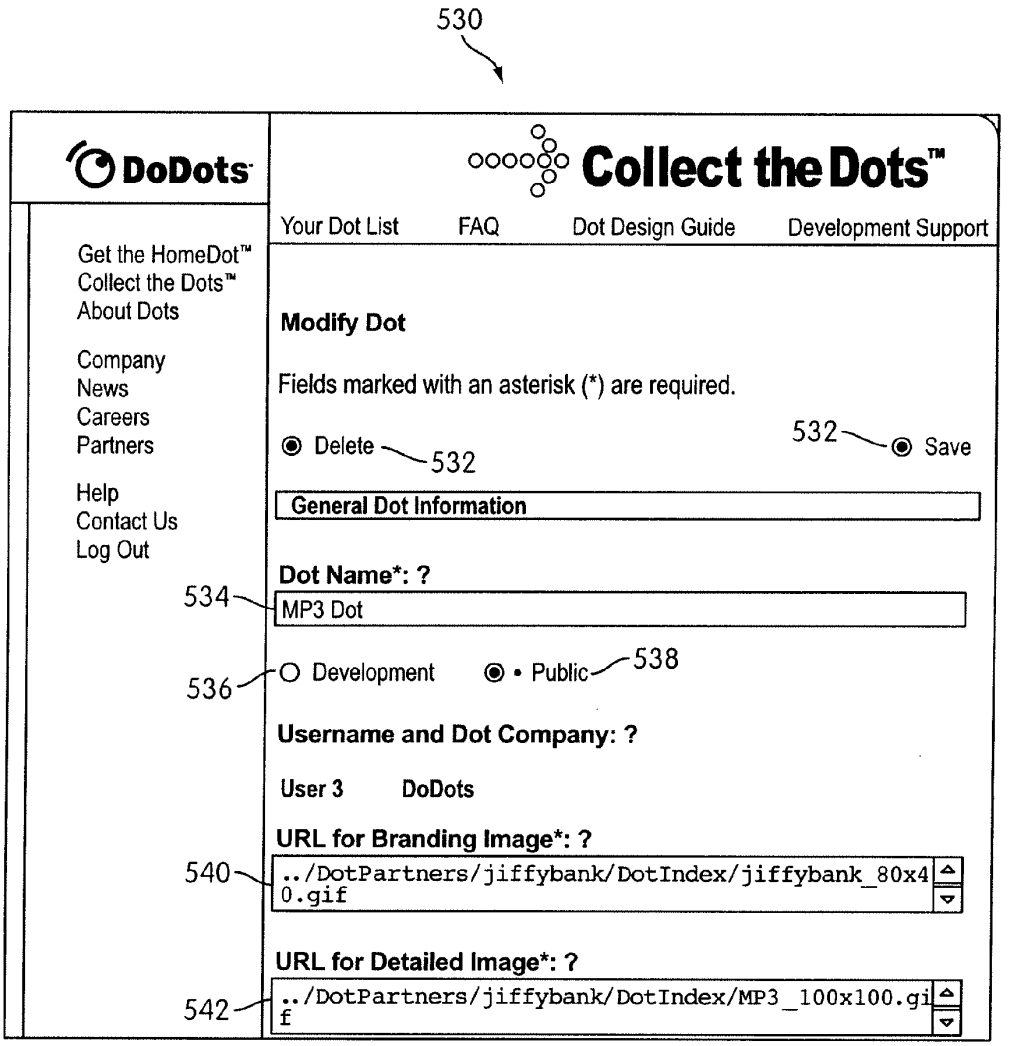

530
DoDots
Collect the Dots™
Your Dot List
FAQ
Dot Design Guide
Development Support
Get the HomeDot™
Collect the Dots™
About Dots
Company
News
Careers
Partners
Help
Contact Us
Log Out
Modify Dot
Fields marked with an asterisk (*) are required.
Delete
532
532
Save
General Dot Information
Dot Name*: ?
534
MP3 Dot
536
Development
Public
538
Username and Dot Company: ?
User 3
DoDots
URL for Branding Image*: ?
540
../DotPartners/jiffybank/DotIndex/jiffybank_80x40.gif
URL for Detailed Image*: ?
542
../DotPartners/jiffybank/DotIndex/MP3_100x100.gif

FIG. 27

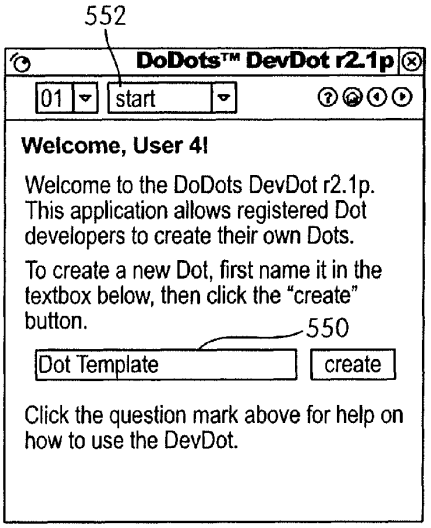


FIG. 28A

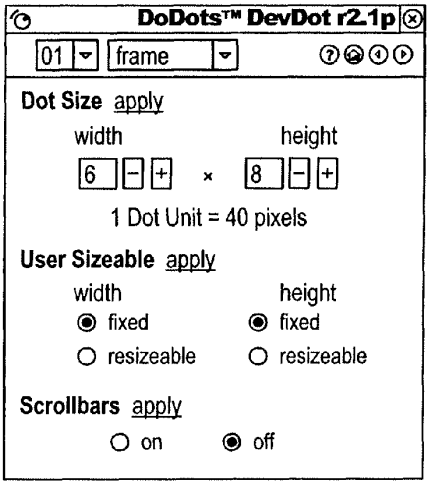


FIG. 28B

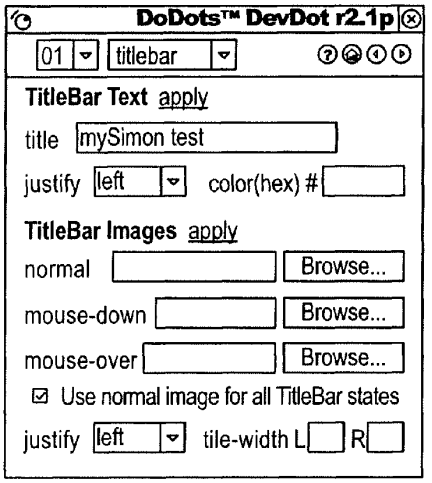


FIG. 28C

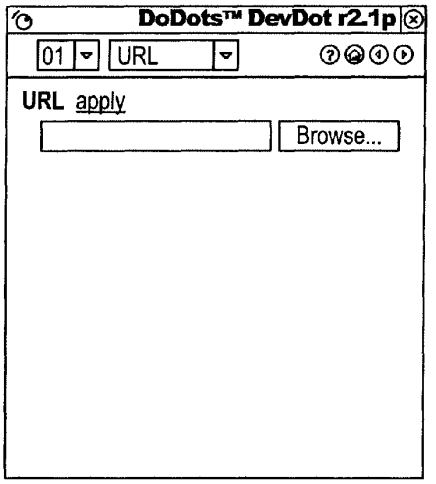


FIG. 28D

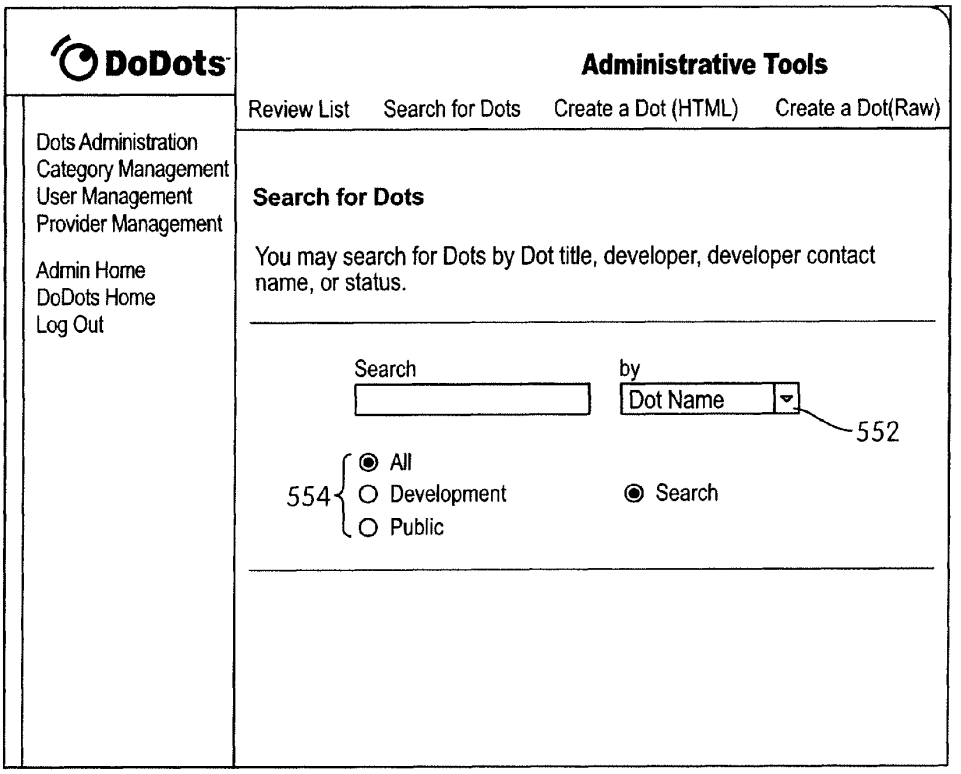

DoDots
Administrative Tools
Review List
Search for Dots
Create a Dot (HTML)
Create a Dot(Raw)
Dots Administration
Category Management
User Management
Provider Management
Admin Home
DoDots Home
Log Out
Search for Dots
You may search for Dots by Dot title, developer, developer contact name, or status.
Search
by
Dot Name
552
554
All
Development
Public
Search

FIG. 29A

DoDots

Administrative Tools

Review List Search for Dots Create a Dot

Dots Administration
Category Management
User Management
Provider Management

Admin Home
DoDots Home
Log Out

Modify Dot

Fields marked with an asterisk (*) are required.

Delete Submit

General Dot Information

Dot Name*: ?

AnyDay Contacts

Development • Public

Username and Dot Company: ?

anyday anyday

URL for Branding Image*: ?

../DotPartners/jiffybank/DotIndex/jiffybank_80x40.gif

URL for Detailed Image*: ?

../DotPartners/jiffybank/DotIndex/MP3_100x100.gif

Dot Categorization*: ? text text text

Brief Dot Description*: ?

Keep stock of friends and colleagues with AnyDay!

Full Dot Description*: ?

Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text

Text*: ?

Comments

Text

Text ## ? Text Yes No

Text ## ? Text Yes No

FIG. 29B

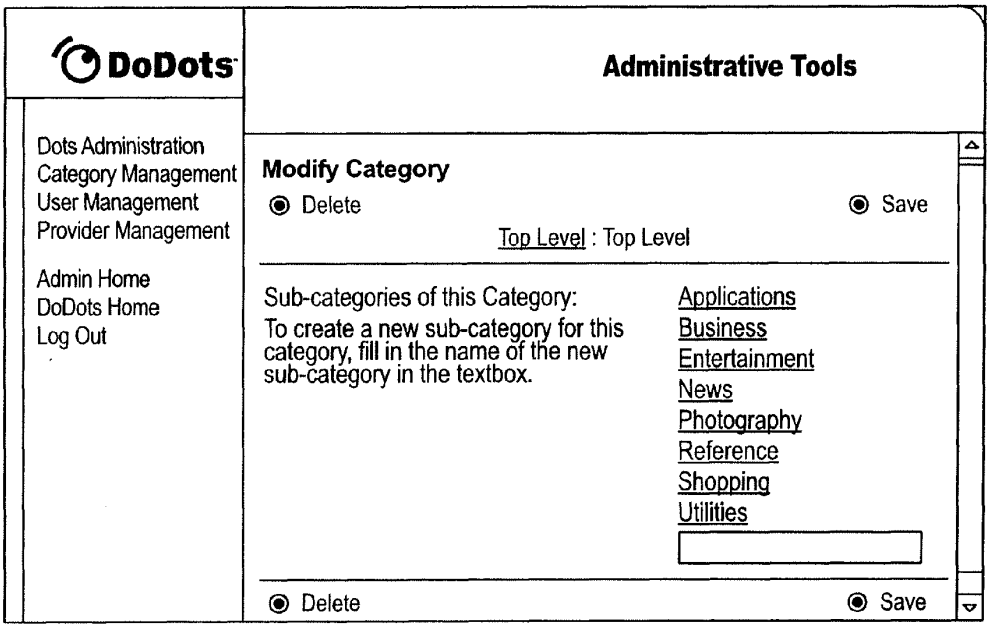


FIG. 30A

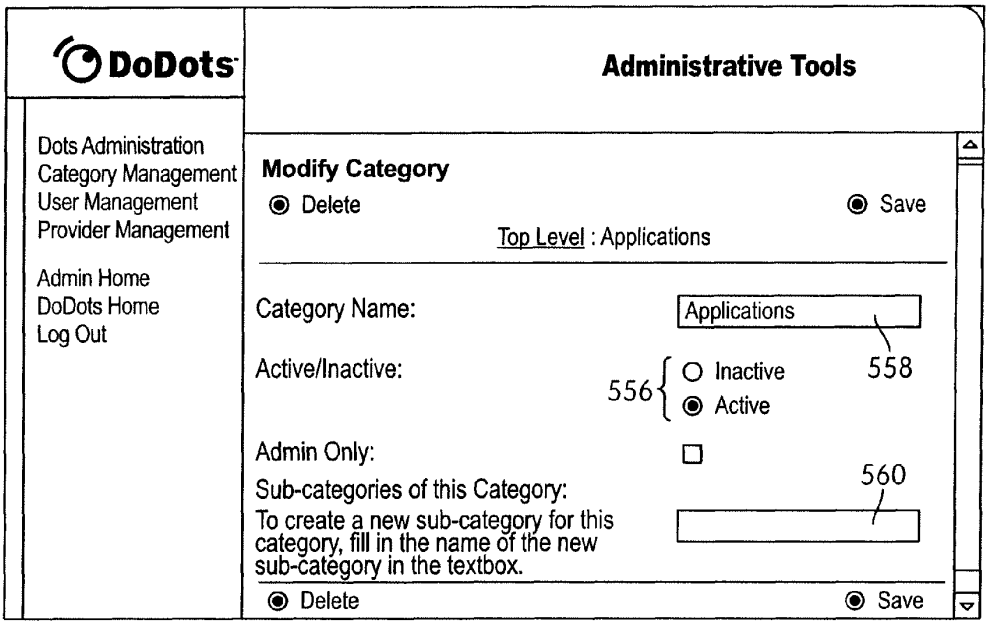


FIG. 30B

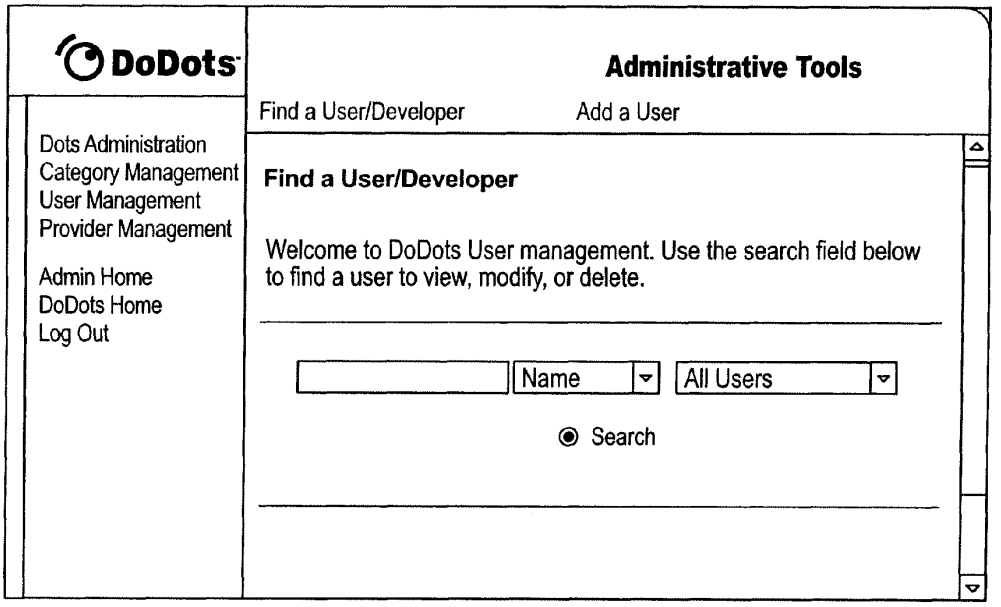

DoDots
Administrative Tools
Find a User/Developer
Add a User
Dots Administration
Category Management
User Management
Provider Management
Admin Home
DoDots Home
Log Out
Find a User/Developer
Welcome to DoDots User management. Use the search field below to find a user to view, modify, or delete.
Name
All Users
Search

FIG. 31A

DoDots

Administrative Tools

Find a User/Developer Add a User

Dots Administration
Category Management
User Management
Provider Management

Admin Home
DoDots Home
Log Out

Modify A User
text
text
text
text

◉ Delete ◉ Update this User

User Status:
○ HomeDot User ◉ Dot Developer ○ Dot Administrator
Partner

Active/Inactive:
◉ Active ○ Inactive

First Name: User 5
Last Name: User 5
Login*: ehow
Password*: ********
Retype Password*: ********
E-Mail*: User5@ehow.com
Address 1:
Address 2:
City:
State/Province: -Select-
Postal Code:
Country: -Select-
Age: -Select-
Gender: -Select-
Occupation: None

☐ Please contact me from time to time about specials and new products

FIG. 31B

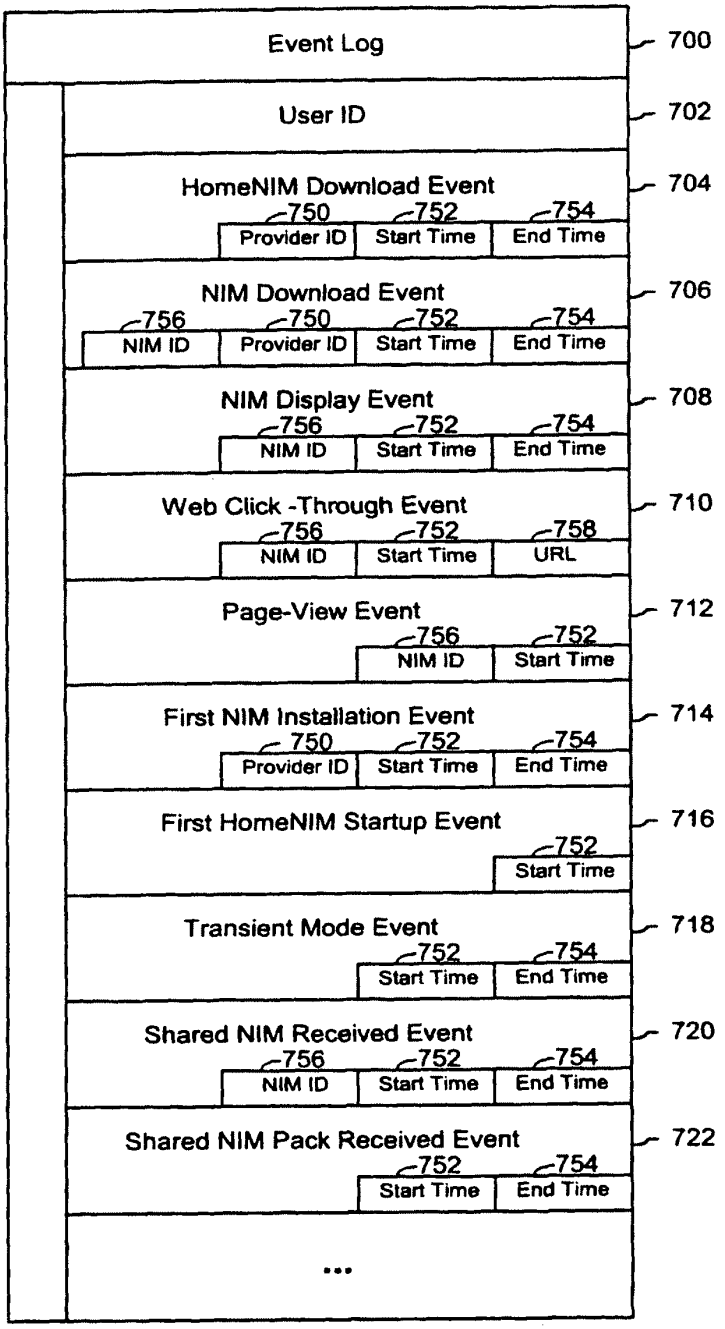

Event Log
700
User ID
702
HomeNIM Download Event
704
750
752
754
Provider ID
Start Time
End Time
NIM Download Event
706
756
750
752
754
NIM ID
Provider ID
Start Time
End Time
NIM Display Event
708
756
752
754
NIM ID
Start Time
End Time
Web Click -Through Event
710
756
752
758
NIM ID
Start Time
URL
Page-View Event
712
756
752
NIM ID
Start Time
First NIM Installation Event
714
750
752
754
Provider ID
Start Time
End Time
First HomeNIM Startup Event
716
752
Start Time
Transient Mode Event
718
752
754
Start Time
End Time
Shared NIM Received Event
720
756
752
754
NIM ID
Start Time
End Time
Shared NIM Pack Received Event
722
752
754
Start Time
End Time
...

FIG. 32

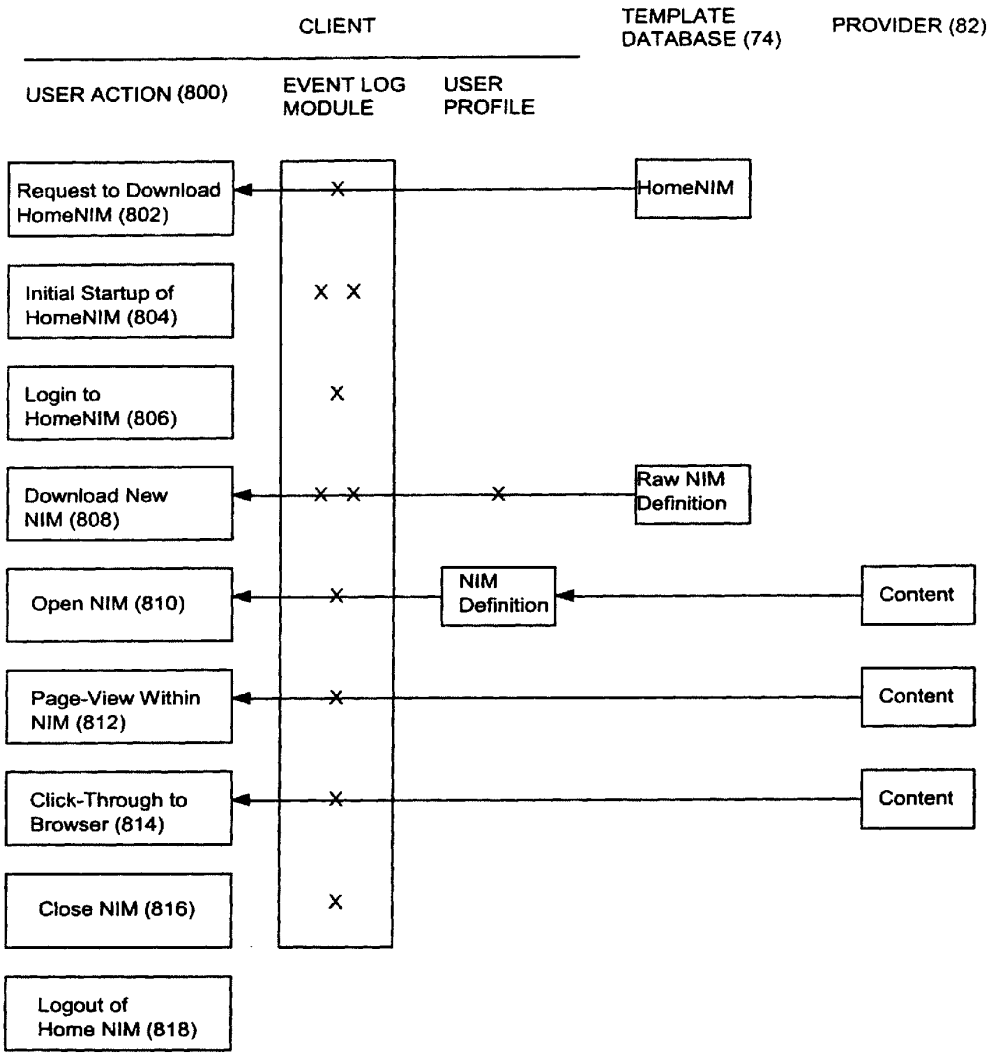

CLIENT
TEMPLATE DATABASE (74)
PROVIDER (82)
USER ACTION (800)
EVENT LOG MODULE
USER PROFILE
Request to Download HomeNIM (802)
HomeNIM
Initial Startup of HomeNIM (804)
Login to HomeNIM (806)
Download New NIM (808)
Raw NIM Definition
Open NIM (810)
NIM Definition
Content
Page-View Within NIM (812)
Content
Click-Through to Browser (814)
Content
Close NIM (816)
Logout of Home NIM (818)

FIG. 33

80

Statistics Database

Events (1002)	User ID (1004)	Start Time (1006)	End Time (1008)	NIM ID (1010)	Provider ID (1012)	URL (1014)	NIM Pack ID (1016)
Event 1 ⋮ Event n	⋮	⋮	⋮	⋮	⋮	⋮	⋮

FIG. 34

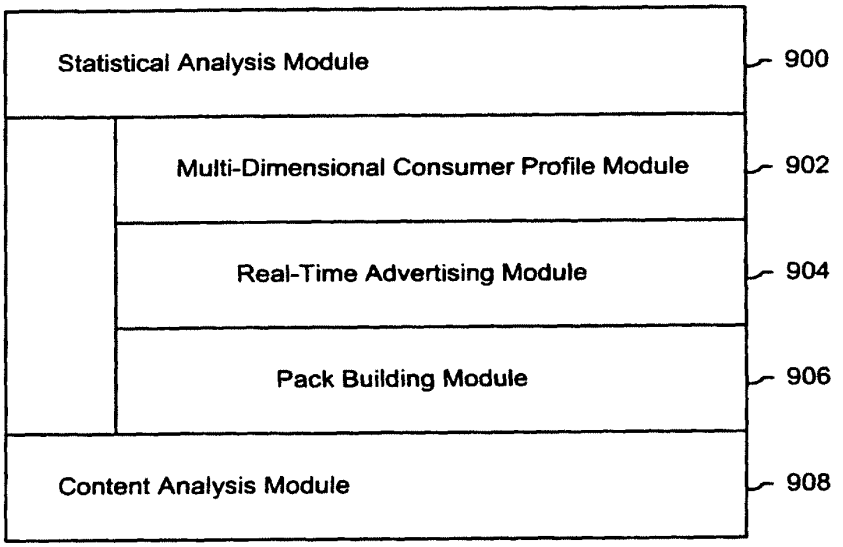

Statistical Analysis Module
900
Multi-Dimensional Consumer Profile Module
902
Real-Time Advertising Module
904
Pack Building Module
906
Content Analysis Module
908

FIG. 35

1050

Content Database

1052 Content Descriptor	1054 NIM ID	1056 Start Time	End Time 1058
Content Descriptor 1 ⋮ Content Descriptor n	⋮	⋮	⋮

FIG. 36

1100

User Account Database	
User ID (1102)	User Information Provided at Login (1104)
User ID 1 ⋮ User ID n	⋮

FIG. 37

ACCESSING AND DISPLAYING NETWORK CONTENT

CROSS-REFERENCE TO RELATED APPLICATIONS

The present application is a continuation of and incorporates by reference U.S. Non-Provisional patent application Ser. No. 11/932,392 filed Oct. 31, 2007, which is a continuation of and incorporates by reference U.S. Non-Provisional patent application Ser. No. 09/558,925, filed Apr. 26, 2000, now U.S. Pat. No. 7,660,868, which claims priority from and incorporates by reference U.S. Provisional Application Ser. Nos. 60/131,083, filed Apr. 26, 1999, 60/131,114, filed Apr. 26, 1999, 60/131,115, filed Apr. 26, 1999, 60/176,687, filed Jan. 18, 2000, and 60/176,699, filed Jan. 18, 2000. The present application claims priority to U.S. Non-Provisional patent application Ser. No. 11/932,392, filed Oct. 31, 2007 and each of the aforementioned applications to which it claims priority.

BACKGROUND OF THE INVENTION

A user operating a client computer typically accesses the Internet by using a viewer application, such as a browser to view web content provided at a destination address, typically a web page. In this context, web content and web applications are designed to fill the entire web page. It is known to divide the web content into different regions of a single web page. For example, personalized web pages can be specified, such that a user views a variety of content sources in a single page, such as stock information, weather information, and sports information, which is aggregated at the server that delivers the web page to the user, who then views the aggregated content in a single web page. Observe that even when disparate content is aggregated, in this manner, it is reassembled into a full web page and is served through a full-screen browser. Web content and application developers therefore have limited control over the user experience: content is typically trapped within the frame of the browser. A developer's only alternative to engaging a user page-by-page in a browser is to develop, distribute, and support custom client software. In the Web browser scenario, it is the content provider, not the user that aggregates the information that is viewed by the user. Thus, the user is not in a position to separately aggregate the content at a client computer, instead the user is constrained to view the content that has been delivered in the manner provided by the server computer hosting the web page. There is a growing desire for individual users to fully control the aggregation and presentation of content and web applications that appears on a client computer.

A user who wishes to view multiple web pages or applications can open multiple instances of a browser. However, the user will not be able to view each "full-screen" page at the same time. Instead, the user must adjust the windows corresponding to each browser instance and view only part of each page. The information appearing in each browser is not designed for viewing in this manner. Thus, the user cannot create an optimized display of content from multiple sources.

Currently, content providers and end users have limited tools to alter the browser in which content appears. That is, the controls associated with a browser are not fully configurable. Thus, the vendor of a browser is in a position to brand the browser and regulate the controls associated with the browser. There is a growing desire for content providers to not only fill a browser with their content, but to also fully brand and control the frame in which the content appears. Further, in

some instances, content providers desire to limit the controls associated with a browser or viewer, so that a user is more inclined to view a single set of content, for example, by having limited access to previously viewed content.

At the present time, it is relatively difficult to trace the content viewing activity of a client computer. In other words, it is difficult to identify the type of content that a particular user of a client computer favors. Consequently, there are limited tools **25** available to provide a user with tailored information that would be of particular interest to the user.

In view of the foregoing, there is a need in the art to provide a technique for accessing multiple instances of distributable computer readable media in their entirety simultaneously, where these instances are typically smaller than the full pages used in 30 current web pages and web applications. There is a further need for providing the user with flexibility in selecting, collecting, relating and viewing such computer readable media, and for giving the media provider flexibility in directing media to a specific user and controlling the framework in which media is presented. Finally, there is a need to gather more accurate information regarding the type of content that a user enjoys, so that the user can be automatically provided with this content.

SUMMARY OF THE INVENTION

The invention includes a method of presenting distributable computer readable media to a user in response to a user request. The method comprises the steps of identifying a definition of a Networked Information Monitor (NIM). A NIM frame is defined for the NIM using the definition. Content is then retrieved for the NIM. Then, the content is placed in a NIM viewer defined by the frame.

The invention also includes a method of altering a Networked Information Monitor (NIM). The method includes the step of receiving a message at a NIM. The message specifies a configurable feature of the NUM. The NIM is altered in accordance with the configurable feature of the message.

The apparatus of the invention includes a computer readable memory to direct a computer to function in a specific manner. The computer readable memory includes a first executable module to identify a definition of a Networked Information Monitor (NIM). A second executable module defines a NIM frame for the NIM using the definition. A third executable module retrieves content for the NIM. A fourth executable module places the content in a NUM viewer defined by the frame.

The apparatus of the invention further includes a computer readable memory with a first executable module to receive a Networked Information Monitor (NIM) message. The NIM message specifies a configurable feature of a NIM. A second executable module alters the NIM in accordance with the configurable feature of the NIM message.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** illustrates a system for processing distributable computer readable media in accordance with one embodiment of the present invention;

FIG. **2** shows exemplary processing steps used to initiate an instance of a home networked information monitor (NIM) using the apparatus of FIG. **1**;

FIG. **3**A illustrates a screen logo in accordance with one embodiment of the present invention;

FIG. **3**B illustrates a login construct in accordance with one embodiment of the present invention;

FIG. **4** illustrates a set of visual manifestations corresponding to a collection of NIMs, including a home NIM;

FIG. **5** illustrates a visual manifestation of a home NIM as well as a visual manifestation of a reference NIM that references additional NIMs;

FIG. **6** illustrates how a NIM, referenced by the reference NIM of FIG. **5**, is added to a processed user profile in response to a designated keyboard entry sequence or mouse click;

FIG. **7** illustrates how a NIM is shared with other users in accordance with an embodiment of the present invention;

FIGS. **8**A and **8**B illustrate screen panels that facilitate the collection of the description of a set of designated NIMs into a pack;

FIG. **9**A illustrates how the relative position of visual manifestations that correspond to NIMs remains fixed when the visual manifestations are within a predetermined distance of each other;

FIG. **9**B illustrates a representative pack in accordance with the present 20 invention;

FIGS. **10**A, **10**B and **10**C illustrate how a set of visual manifestations corresponding to a collection of NIMs is aligned against a boundary when a user selects the visual manifestations and pushes them against the boundary;

FIG. **11** is a flow chart of the steps taken by a client to give a user access to a NIM where the user "collects" the NIM, in accordance with one embodiment of the invention;

FIG. **12** is a flow chart of the steps taken by a client to present a NIM to a user, where the NIM has been "collected" previously by the user, in accordance with one embodiment of the invention;

FIG. **13** illustrates a data structure for a NIM definition, stored in the NIM application server's template database or user profile database;

FIG. **14** illustrates NMA message routing between NIMs and the message interface in the client parser application;

FIG. **15** is a diagrammatic illustration of an embodiment of a NIM Management Module utilized in accordance with an embodiment of the invention;

FIG. **16** is a diagrammatic illustration of an embodiment of the NIM Templates database utilized in accordance with an embodiment of the invention;

FIG. **17** is an illustration of a main NIMIndex Web page used in accordance with an embodiment of the invention;

FIG. **18** is an illustration of a single NIMIndex category used in accordance with an embodiment of the invention;

FIG. **19** is an illustration of a full description of NIM content provided in accordance with an embodiment of the invention;

FIG. **20** is an illustration of a Web page displayed to the user once the user has clicked to collect the NIM;

FIG. **21** is an illustration of the main home NIM graphical user interface used in accordance with an embodiment of the invention;

FIG. **22** is an illustration of a "Get New NIM" graphical user interface that may be used in accordance with an embodiment of the invention;

FIG. **23** is an illustration of a "More NIMs" graphical user interface representative of an embodiment of the invention;

FIG. **24** is a diagrammatic illustration of the ShareLink database used in accordance with an embodiment of the invention;

FIG. **25** is an illustration of a Share NIM's graphical user interface according to an embodiment of the invention;

FIG. **26** is an illustration of the main DevZone Web page utilized in accordance with an embodiment of the invention;

FIG. **27** is a partial view of a NIM modification web page utilized in accordance with an embodiment of the invention;

FIGS. **28**A to **28**D are graphical user interfaces of development NIMs (DevNIMs) utilized in accordance with an embodiment of the invention;

FIGS. **29**A and **29**B are illustrations of Administrative Zone (AdminZone) Web pages utilized in accordance with an embodiment of the invention;

FIGS. **30**A and **30**B are also illustrations of Administrative Zone (AdminZone) Web pages utilized in accordance with an embodiment of the invention;

FIGS. **31**A and **31**B are further illustrations of Administrative Zone (AdminZone) Web pages utilized in accordance with an embodiment of the invention;

FIG. **32** illustrates an embodiment of an event log that may be used in accordance with an embodiment of the invention;

FIG. **33** illustrates the tracking of events in an event log module in accordance with an embodiment of the invention;

FIG. **34** illustrates a statistics database that may be used in accordance with an embodiment of the invention;

FIG. **35** illustrates a statistical analysis module and a content analysis module that may be used in accordance with an embodiment of the invention;

FIG. **36** illustrates a content database that may be used in accordance with an embodiment of the invention; and

FIG. **37** illustrates a user account database that may be used in accordance with an embodiment of the invention.

DETAILED DESCRIPTION OF THE INVENTION

The present invention discloses a technology that is capable of processing distributable computer readable media. Distributable computer readable media includes, but is not limited to, standard web content, such as HTML, dHTML, images, imbedded ActiveX and Java applications, JavaScript, CSS, Pen scripts, Streaming Media, and/or Flash. The present invention is advantageous relative to prior art systems and methods because it provides improved mechanisms for simultaneously interacting with several independent sources of distributable computer readable media, collecting references to such media, and sharing such references with other users. The ac disclosed technology is further advantageous because it provides improved systems and methods for on screen management of distributable computer readable media.

In the system and method of the present invention, a user logs into a server by providing a login identifier to a login construct. The login identifier is used by the server to obtain an unprocessed user profile that corresponds to the user. The unprocessed user profile is processed by the server to generate a processed user profile. Advantageously, this processing step allows for up-to-date refinement of the user profile. Up-to-date refinements include, for example, the addition of advertisements directed to the user based on one or more characteristics in the user profile. The processed user profile is delivered to the client computer associated with the user. The processed user profile includes references to the networked information monitors (NIMs). As used herein, the term networked information monitor or NIM refers to a fully configurable frame with one or more controls; the frame through which content is optionally presented. The fully configurable frame utilized in accordance with the invention stands in contrast to present web browsers, which are branded by the browser vendor and which have limited means by which to alter the controls associated with the browser.

Attention is initially directed toward the home NIM of the invention, which coordinates the activities of all other NIMs that are accessed by a user. The home NIM facilitates much of the technology of the present invention, including the ability

to simultaneously review multiple sources of distributable computer readable media as well as to package and distribute such media.

FIG. **1** is a general illustration of a system in accordance with one embodiment of the present invention. In FIG. **1**, a network **10** is operated in accordance with the present invention. Network **10** includes at least one user or client computer **20**, at least one server computer of class **50**, and optionally one or more server computers of class **82**. User computer **20** as well as server computers of class **20 50** and **82** are each connected by transmission channel **44**, which is any wired or wireless transmission channel.

User computer **20** is any device that includes a Central Processing Unit (CPU) **24** connected to a random access memory **30**, a network connection **28**, and one or more user input/output ("i/o") devices **40** including output means **42**. Output means **42** is any device capable of communicating with a user and includes, for example, a video monitor, a liquid crystal display, voice user interfaces, and/or integrated graphic means such as mini-displays present in web-phones. Typically, user computer **20** includes a main non-volatile storage unit **22**, preferably a hard disk drive, for storing software and data. Further, user computer **20** includes one or more internal buses **26** for interconnecting the aforementioned elements. In a typical embodiment, memory **30** includes an operating system **32** for managing files and programs associated with user computer **20**. In some embodiments, operating system **32** includes a registry **34** that has one or more references to specified locations in system **10**. The exemplary memory **30** of FIG. **1** further includes a web browser **36** for viewing web content and a client parser application **38** for facilitating low level functionality, such as login and logout procedures, in accordance with the present invention. In some embodiments, client parser application **38** uses the one or more references in registry **34** to obtain a login construct from server **50**. In various embodiments, in accordance with the present invention, client parser application **38** runs in conjunction with one or more software modules, such as an event log module **98**, which tracks user activity, a message interface module **106**, which serves as a communication interface between the client parser application **38** and web server **58** and/or external web servers, a home NIM **108**, which references one or more NIMs **110**, and a visual management system **114** which regulates the characteristics of visual manifestations of NIMs **108** and **110** when displayed on output device **42**. Furthermore, in some embodiments, client parser application **38** runs in conjunction with instances of web browser **36** as well as web server **58** as detailed below.

Server computer **50** includes standard server components, including a network connection device **46**, a CPU **52**, a main non-volatile storage unit **54**, and a random access memory **56**. Further, server computer **50** includes one or more internal buses **48** for interconnecting the aforementioned elements. Memory **56** stores a set of computer programs, modules and data to implement the processing associated with the present invention.

The embodiment of memory **56** illustrated in FIG. **1** includes a web server **58** for processing requests received from client computer **20**. Web server **58** has many components, including a variety of modules and data structures to assist users that want to log into system **10**. Namely, login module **60** handles an entry request from a client computer **20** and accepts a login identifier that corresponds to a user from client computer **20**. Login constructor **62** generates a login construct in response to a call for a login construct and transfers the login construct to client **20**. Login constructor **62** dynamically generates login constructs using updated login NIM content **64**. Login validation module **66** works in conjunction with user profile database manager **100** to identify an unprocessed user profile, which is associated with a user provided login, in user profile database **76**. If an unprocessed user profile corresponding to the user provided login does not exist in user profile database **76**, login validation module **66** associates a new unprocessed user profile with the login identifier.

When an unprocessed user profile is identified by login validation module **66**, it is processed by user profile processor module **68** to produce a processed profile. It will be appreciated that the services of user profile processor module **68** are highly advantageous because they allow for last minute user profile revisions. Such revisions include, for example, the addition or customization of NIMs referenced by the user profile, and/or server redirect information that is a function of current server load in system **10**. Importantly, the processed user profile includes a reference to a home NIM. The home NIM is capable of accessing each of the NIMs that are represented in the processed user profile. Memory **56** further includes delivery module **70**, which coordinates the delivery of portions of the home NIM to the client based on parameters specified in the processed user profile.

Once a user has successfully logged into system **10**, request server module **72** handles requests for specified NIMs from client **20**. When such a request is received, request server module **72** routes the request to an address that corresponds to the specified NIM and transmits the specified NIM to client **20**. One class of specified networked information handled by request server module **72** is requests for NIMs. When such a request is received, request server module **72** searches NIM templates database **74** for the specified NIM. NIM templates database **74** includes a large number of NIM templates. Each NIM template defines the characteristics of a specific NIM, including fully configurable frame characteristics, viewer and control characteristics, and NIM content references.

The web server **58** illustrated in FIG. **1** further includes additional modules **102** to handle specialized features of the present invention. For example, one embodiment of the present invention provides a mechanism that allows users to distribute NIMs to each other. In such embodiments, a special server module **102** provides instructions for storing the NIMs, which are to be distributed, in sharelink database **78**. Advantageously, NIMs that are distributed to other users are customizable. A user can, for example, resize and position a particular NIM prior to sharing it with another user. Indeed, it is possible, in such embodiments, for a user to arrange a series of NIMs in a unique arrangement and then distribute the collection of NIMs in the designated NIMs in the designed arrangement. As an illustration, a user arranges a first NIM that represents a scrolling stock ticker at the bottom of an output means, such as a computer screen, a second NIM that tracks the NASDAQ top ten most heavily traded stocks in the upper left corner of the output means, and a third NIM that tracks headline news on the upper right hand corner of the output means. Then, the user distributes the three NIMs in this customized arrangement to other users. Observe that in this example a user of a client computer is aggregating separate sets of information in different NIMs. This stands in contrast to prior art approaches where a web server running on a server computer aggregates information in a single page.

System **10** is highly scalable and thus supports a large number of users. This scalability stems from the fact that the server **50** is delivering the definition associated with a NIM. The content displayed in the NIM may be located on a separate computer.

Memory **56** provides a statistical analysis module **104** for tracking key events associated with users. This information is stored in statistics database **80**. The information collected by statistical analysis module **104** is used for a wide variety of purposes, including server load optimization and directed advertising, as discussed below. As described below, the statistical information gathered in accordance with the invention includes fully traced events defining the type of content and the duration over which all content is viewed by a user. This type of comprehensive information is not available using present techniques.

Much of the distributable computer readable media that is available for processing is stored as content elements **94** on server **82**. Server **82** is a standard web server that includes components such as a network connection device **88**, a CPU **86**, a main non-volatile storage unit **84**, a random access memory (RAM) **92**, and one or more internal buses **90** for interconnecting the aforementioned elements. RAM **92** includes some of the content elements **94** stored by server **82**. Other content elements **94** are stored in storage unit **84**. In some embodiments, a single web server **58** is capable of directly accessing content elements **94** located on one or more servers **82**. In other embodiments, each server **82** has a resident web server module that works in conjunction with server **50** to identify, optionally dynamically generate, and serve content elements **94** upon demand.

Now that general architecture of a system in ac dance with the present invention has been disclosed, attention turns to FIG. **2**, which discloses a method for logging into system **10** (FIG. **1**). In the first processing step shown in FIG. **2** (**202**), a user initiates a session on system **10** by requesting the global login script "session_conFIG." It will be appreciated that the term "session_config" merely provides an illustrative name for the global login script and that the technology of the present invention is by no means limited to this name or the script described.

The request for "session_config" originates on client **20** and is sent to server **50** where it is processed by login module **60** of web server **58**. Upon receiving request **202**, login module **60** creates a "session_config" global login script (**204**). Processing step **204** is advantageous relative to systems that have static global login scripts because it allows for the incorporation of highly variable information. This highly variable information includes, for example, system settings such as up-to-date server redirect information, server content address changes, directed advertisements, and messages. An exemplary "session_config" is found in Example 1 below. Each line of data has an associated numeral. The remaining text in the example describes the purpose of select data.

EXAMPLE 1

```
   Version tag that identifies the latest home NIM version
(1) </SESSION__CONFIG VERSION="alpha:3"
   Upgrade event that is sourced when home NIM version is outdated
(2) <UPGRADE=http://www.NIM.com /QuickOpen.exe>
   LOCATION OF SERVER-SUPPORTED FUNCTIONALITY Default base address
for server supported functionality
(3) <METHODS BASEURL=http://neo.NIM.com /servlet/NIMServer/>
(4) <ADD__USER URL="addUser/>
(5) <GET__USER URL="getUser"/>
(6) <SET__USER URL="setUser/>
(7) <GET__SES__CONFIG URL=http://www.NIM.com /home__NIM/s__cnfg.xml"/>
   Server-based functionality for setting password
(8) <SET__PASSWORD URL="setPassword"/>
   List of all publically available NIMs
(9) <GET__MASTER__NIM__LIST URL="getMasterNIMList"/>
(10) <GET__ALL__CONFIG URL="getAllConfig"/>
(11) <SET__ALL__CONFIG URL="setAllConfig/>
(12) <SET__ALL__STATS URL="setAllStats"/>
(13) <GET__NIM__TEMPLATE URL="get__NIM__Template"/>
   Location of server-side NIM and pack sharing functionality
(14) <ADD__SHARE URL="addShare"/>
(15) <GET__SHARE URL="getShare"/>
(16) <AUTH__TEST URL="DOeCHO?AUTH=TRUE"/>
   Redirect information
(17) <DO__REDIR URL="doRedir"/>
(18) </METHODS>
   Flexible content layer that defines default NIM frame appearance, including the default
appearance of the frame of a home NIM
(19) <FRAMES>
   Default NIM frame appearance
(20) <NIM>
(21) <IMAGES BASEURL=
http://www.NIM.com /home__NIM/NIM__FrameImages/>
(22) </NIM>
   Default home NIM frame appearance
(23) <HOME__NIM>
(24) <IMAGES BASEURL=
http://www.NIM.com /home__NIMImages/>
(25) </HOME__NIM>
(26) </FRAMES>
   Location of system NIM templates
(27) <NIMs>
(28) <ADD TEMPLATE="http://www.NIM.com / . . . /add__NIM__XML.xml"/>
(29) <HELP TEMPLATE="http://www.NIM.com / . . . /help__NIM__XML.xml"/
(30) <LOGIN TEMPLATE="http://www.NIM.com / . . . /login2.xml"/
(31) </NIMs>
(32) </SESSION__CONFIG>
```

Line **1** of the exemplary "session_config" of Example 1 provides the version tag for the expected version of the home dot system that corresponds to the "session_config" script. In one embodiment, client parser application **38** determines whether it is up-to-date using the information in line **1**. If client parser application **38** determines that it is outdated, an upgrade request is made in accordance with the instructions provided by the UPGRADE flag of line **2**. In one embodiment, the UPGRADE flag in line **2** of Example 1 describes the location of an executable program, one of skill in the art will appreciate that this flag can in fact reference any form of instruction, including a flat file, a web page, a script, a symbol, or an address.

Lines **3** through **18** in Example 1 define the functionality that is provided by a server, such as server **50**. For example, line **8** of Example 1 provides the location of a set of instructions that are called when a user requests a password change. Furthermore, line **9** of Example 1 provides the location of master list of NIMs that are publically available. Lines **14** and **15** of Example 1 provide the location of specialized server-side functionality that allows users to share data such as NIM definitions.

Lines **19** through **26** of Example 1 define where the default appearance of a NIM and a home NIM are found within system **10**. Lines **27** through **31** define a collection of system NIMs. A system NIM is any type of NIM that is to be distributed to each user of system **10**. In some embodiments, system NIMs are used to provide a core functionality. In Example 1, line **28** defines a NIM that provides users with a convenient mechanism for collecting additional NIMs. Line **29** defines the location of a NIM that is invoked when the user presses a help button associated with a home NIM. Finally, line **30** defines the location of a NIM that is used to log into system **10**.

Returning to FIG. **2**, once login module **60** has created "session_config," it is sent back to requesting client **20** (**206**). When a "session_config" is received by client **20**, client parser application **38** parses the global login script in order to identify a reference to a login constructor **62** (**208**). Login constructor **62** is a server-based module that generates a construct that allows a user to log into system **10**. When client parser application **38** locates the reference to login constructor **62** in "session_config," a request for a login construct is directed to the identified reference (**210**). In Example 1 above, the reference to the login construct is provided in line **30**. On line **30**, the global variable "LOGIN TEMPLATE" is assigned the URL address "http://www.NIM.com . . . /login2.xml." Client parser application uses the URL assigned to the global variable "LOGIN TEMPLATE" to make a request for a login constructor **62** that is directed to this URL. When login constructor **62** receives a request for a login construct, it generates a login construct (**212**).

Login construct **148** (FIG. **3**B) illustrates a type of login construct that is generated in one embodiment of the present invention during processing step **212**. Before the login construct is executed on client **20**, a schematic such as logo **146** (FIG. **3**A) is displayed on output means **42**. As illustrated in FIG. **1**, login constructor **62** is a component of web server **58**. However, there is no requirement that login constructor **62** be a component of web server **58**. In fact, login constructor **62** is a standalone software program in some embodiments of the present invention whereas in other embodiments, login constructor **62** is merely a script, such as a PERL script, that is processed by an interpreter program native to server **50**. In still other embodiments, login constructor **62** is merely a simple flat file that includes a set of instructions that are interpretable by client parser application **38**. In such embodiments, login constructor **62** is the login construct. In embodiments in which a login construct is dynamically generated, it is possible to introduce last minute changes in the login construct. Thus, an advantage of the exemplary login process shown in FIG. **2** is that there are multiple stages in which updated information is used to customize the login process based on the environmental variables.

Once a login construct has been prepared by login constructor **62**, it is transferred back to client **20** (**214**) (FIG. **2**) and executed in conjunction with client parser application **38** (**216**). The login constructs of the present invention are a form of NIM. Therefore, one function of processing step **216** is to obtain the login NIM content **64** (FIG. **1**) specified by the login construct from server **64**. In login construct **148**, for example, the login NIM content includes the shape and functionality of "Exit button **160**," message **150**, the shape and functionality of "New user" button **152**, the functionality of "Forgot it?" button **154**, and login panel **156**. When processing step **216** is completed, the user uses the login construct to provide a login identifier (**218**).

In FIG. **3**B, a user has provided the login identifier "Galliani." The definition of login identifier as used in the present invention is to be broadly construed. In some embodiments, login identifiers include a unique name and a corresponding password. In other embodiments, a login identifier does not have a password. This is particularly the case when the user is a guest or a new user and there is no user profile associated with the user.

Working in conjunction with client parser application **38**, the login construct accepts the user login and sends it to server **50** for validation (**218**). As illustrated in the exemplary system of FIG. **1**, web server **58** includes a login validation module **66** to verify the login identifier provided by user (**220**). Typically, processing step **220** involves a look-up operation in which the login identifier is used to query user profile database **76** for an unprocessed or raw user profile that matches the login identifier. In embodiments that include a password, validation step **220** includes a password verification step. Successful completion of processing step requires entry of a valid login identifier sequence in processing step **218**. When processing step **220** has been successfully completed, the raw or unprocessed user profile corresponding to the login identifier is obtained from user profile database **76** (FIG. **1**) (**222**) and is processed by user profile process module **68** to produce a processed or finalized user profile that is delivered to client **26** (**226**). In sonic embodiments, a user profile **76** includes user contact information, such as the name, address, telephone number and email address of a user. Additionally, some embodiments of system **10** provide different types of access privileges. For example one embodiment of the present invention includes developer access privileges, administration access privilege, and general user access privileges. In such embodiments, the access privileges that have been granted to a user are stored in the user profile **76** associated with the user.

The processed user profile includes a reference to each NIM in system **10** that is associated with the login identifier provided in processing step **216**. One of the NIMs referenced by the processed user profile is the home NIM that corresponds to the login identifier provided in processing step **216**. When executed in conjunction with client parser application **38** in processing step **226**, the home NIM provides a mechanism for accessing each of the NIMs referenced by the processed user profile Like the login construct, the home NIM includes several components, including pull down menus and screen manipulation functionality. The reference to the home NIM in the processed user profile includes the system **10** address of each of these components. Therefore, in one

embodiment, construction of the home NIM in processing step **226** involves one or more requests to server **50** and/or server **82** for content (**228**) that is then rendered (**230**) in accordance with the home NIM description provided in the processed user profile. In some embodiments, the home NIM is distinct from other NIMs in the sense that a large proportion of the home NIM in such embodiments is pre-compiled. Such embodiments are advantageous because some of the functionality provided by the home NIM requires substantial client **30** processing resources. Therefore, to minimize such processing resource requirements, many aspects of the home NIM are pre-compiled in some embodiments. In other embodiments, however, the home NIM has a structure that is substantially the same as a regular NIM. In such embodiments, simple script commands are used to identify the NIM as a home NIM.

Upon completion of processing step **230**, the user is granted access to all of the technologies of the present invention, including the ability to view multiple NIMs simultaneously, collect new NIMs, customize NIMs, and share customized NIMs with other users. An exemplary processed user profile is provided in Example 2. Once again, each line of data is identified with a numeral, while the remaining text in the example describes select data. In some embodiments, the user is granted specific privileges and the extent to which the user is granted access to system **10** is regulated by the types of privileges that have been granted to the user.

EXAMPLE 2

```
(1) SAMPLE PROCESSED USER PROFILE
(2) <ALL CONFIG>
   NIMs AND PACKS THAT CORRESPOND TO THE USER
(3) USER
   NIM definition 1
(4) <NIM DOMAIN="ZDNet" GLOBALID="1" KND="news"
(5) <FRAME BACKGROUNDCOLOR=#FFFF00" COLLAPSED="FALSE"
(6) FIXHEIGHT="TRUE" FIXWIDTH="TRUE" NAME="ZDNet Breaking News"
(7) PIXELHEIGHT="275" PIXELWIDTH="235" X="RIGHT" Y="TOP">
(8) <TITLE COLOR=#000000" JUSTIFY="RIGHT" TEXT=" "/>
(9) <TITLEBARIMAGE DOWN=http://www.NIM.com / . . . /feed/titlebar.gif
(10) HOVER=URL address to a first GIF file <param 1> . . . <param N>
(11) INACTIVE=URL address to a second GIF file <param 1> . . . <param 2>
(12) NORMAL=URL address to a third GIF file <param 1> . . . <param 2>
(13) <BOTTOMBARIMAGE DOWN=
http://www.NIM.com / . . . /feed/bottombar.gif
(15) HOVER=URL address to a fourth GIF file <param 1> . . . <param 2>
(16) INACTIVE=URL address to a fifth GIF file <param 1> . . . <param 2>
(17) NORMAL=URL address to a sixth GIF file <param 1> . . . <param 2>
(18) </FRAME>
(19) <MENU/>
(20) <CONTROL__LAYOUT HEIGHT="1" HEIGHTSCALES="TRUE" WIDTH="1"
(21) WIDTHSCALES="TRUE"> <CONTROL CLASS="Browser" HEIGHT="1"
(22) ID="1" KIND="A" LEFT="0" TOP="0"
(23) URL=http://www.mandala.com /cgl/zdnet/zdfeedl.cgi WIDTH="1"/>
(24) </CONTROL__LAYOUT>
(25) <CATEGORIES/>
(26) <EVENTS/>
(27) </NIM>
   NIM definition 2
(28) <NIM DOMAIN=NIM DOMAIN 2 GLOBALID='2'
(29) </NIM>
   NIM definition N
(30) <NIM DOMAIN=NIM DOMAIN 2 GLOBALID='N'
(31) </NIM>
   Pack definition 1
(32) <PRESET TITLE="New DotPack">
(33) <NIM GLOBALID="1" X="RIGHT" Y="TOP"/>
(34) <NIM GLOBALID="2" X=RIGHT Y="320"/>
(35) </PRESET>
(36) </SHARE>
   Last state of the home NIM
(37) <LASTSTATE>
(38) <PRESET TITLE=" ">
(39) <NIM GLOBALID="1" X="RIGHT" Y="TOP"/>
(40) <NIM GLOBALID="2" X=RIGHT Y="280"/>
(41) </PRESET>
(42) <HOMENIM COLLAPSED="FALSE" HEIGHT="134" X=616" Y="109"/>
(43) </LASTSTATE>
(44) </ALL__CONFIG>
```

Example 2 describes a representative processed user profile in accordance with the present invention. In general, a processed user profile includes three major components: (i) a definition of each NIM associated with the user, (ii) a description of each pack associated with the user and, (iii) the last state of each home NIM associated with a user. In Example 2, the definition of each NIM associated with the user is found on lines **4** through **31**. Specifically, lines **4** through **31** describe NIM definitions I through N. In Example 2 there is only one pack associated with the user. This pack, entitled "New DotPack," is found on lines **32** through **35** of Example 2. The final major component of the processed user profile found in Example 2 is the last state of the home NIM, which is defined on lines **37** through **43**. This code stores the last state of the home NIM. Such last state information includes whether the home NIM was collapsed, and the position of the home NIM on the screen.

When the user wishes to log out of system **10**, the processed user profile is transferred from client **20** to server **50**. When web server **58** receives the processed user profile, it passes the processed user profile to user profile database manager **100**. User profile database manager **100** stores the processed user profile as the unprocessed user profile **76** corresponding to the user. In some embodiments, such a storage operation involves a conversion process. For example, advertisements or specific system NIM definitions are stripped from the processed user profile in order to convert the processed user profile to the unprocessed user profile **76** that corresponds to the user. In some embodiments, the processed user profile is periodically transferred, in its entirety or incrementally, from client **20** to server **50** and saved in the manner described in the log out procedure above. Such timed periodic or event based backup procedures are possible because NIM definitions are efficiently described, thus the absolute size of a processed user profile remains relatively small. Accordingly, timed backups of a processed user profile to user profile database **76** are possible without extensive use of system **10** bandwidth or server **50** resources.

At this stage, a system (FIG. **1**) and a login procedure (FIG. **2**) in accordance with the present invention has been disclosed. Although the system and login procedure was discussed using an example where only one home NIM was associated with a user, it will be appreciated that, in some embodiments, any number of distinctly different home NIMs are associated with a user. Furthermore, a user can simultaneously execute multiple instances of a particular home NIM on client **20** or, indeed, any number of different home NIMs. In one embodiment, a developer or merchant provides a user with a highly customized home NIM that provides specialized functionality. In such embodiments, the user collects the home NIMs and, therefore, a processed user profile includes a description of more than one home NIM.

Attention now turns to some of the advantages and features of the present invention. In FIG. **4**, a visual manifestation of the home NIM **162** is illustrated. One advantage of the home NIM, which is an advantage that is common to NIMs in general, is that the content of the NIM is not trapped in a third party viewer. In fact, the home NIM definition regulates the actual appearance of the home NIM. The home NIM definition is formed by general parameters and commands found in "session_config" as well as customized parameters and commands in the processed user profile. The division of the home NIM definition between a system level file and a user level file represents a balance in the tension between the need for a system **10** host to insure a consistent level of quality, through the proper implementation of general parameters and commands, and the desire of each user to create highly customized home NIMs. Lines **20** through **22** of Example 1 provide an example of general parameters that are defined in "session_conFIG." Lines **23** through **25** define the source location of home NIM frame images. In home NIM **162** (FIG. **4**), such home NIM frame images include the image used to represent buttons **164** through **174**, and menu tabs 1 d **178**. Furthermore, lines **20** through **22** of Example 1 define the location of other images that are used to construct default NIMs. An example of user initiated home NIM customization is found in lines **33** through **39** of Example 2, which define a "LAST-STATE" definition for the home NIM, including the dimensions of the visual manifestation corresponding to the home NIM on line **38** (HEIGHT="134" X="616" Y="109") and indicates that the home NM is not collapsed upon startup (COLLAPSED="FALSE").

The visual manifestation of home NIM **162** illustrates additional benefits and features of a home NIM in accordance with the present invention. When a user selects tab **176**, a list of the NIMs that are present in the processed user profile associated with the user is displayed in viewer **180**. As disclosed in more detail below, a user has the option to associate a collection of NIMs into an object termed a "pack". The pack references some subset of the NIMs associated with a user as well as associated state information. This arrangement includes, for example, whether a visual manifestation corresponding to each MM is displayed on output means **42** or not, the dimensions of each visual manifestation, and the position of each visual manifestation. The name of each pack is stored in the processed user profile. A user reviews packs associated with the user by selecting tab **178** (FIG. **4**). In FIG. **4**, the user only has one pack, "Customized DotPack" **182**. When the user selects pack **182**, each NIM in the pack is restored in accordance with the state information stored in the pack definition.

In total, FIG. **4** represents a typical visual experience provided by one embodiment of the present invention. In addition to home NIM **162**, visual manifestations **184** and **186**, corresponding to two additional NIMs in the processed user profile, are displayed. Visual manifestation **184** provides functionality that allows a user to manage an address book, schedule appointments, or create groups and plan activities. Visual manifestation **186** represents a NIM that provides time and date information.

FIG. **5** shows the visual manifestation of home NIM **162** of FIG. **4** with tab **176** selected. Accordingly, each of the NIMs in the processed user profile associated with the user is listed in list **188**. The user can activate any of the listed NIMs by clicking on the NIM name. In addition to the NIMs in list **188**, home NIM **162** includes core NIMs that are defined in "session_conFIG." In the "session_config" of Example 1, cores are found on lines **28** and **29**. Specifically, line **28** provides the address of an XML-based definition for the add template functionality associated with button **172** in FIGS. **4** and **5**, and line **29** provides the address of an XML-based definition for the help template functionality associated with button **174** in FIGS. **4** and **5**.

Importantly, the user can categorize NIMs using filter **190**. Categories include such topics as sports, personal, weather, etc. Furthermore, the user can add NIMs to the processed user profile associated with the user as well as delete NIMs. There are a variety of mechanisms that enable a user to add a NIM to the processed user profile. One mechanism is to receive links to NIMs from other users of system **10** (FIG. **1**), as disclosed below. Another mechanism is to toggle button **172** in order to activate a visual manifestation associated with NIM **192** (FIG. **5**).

NIM **192** provides a system that enables users to add select NIMs to their user profile with a single click or keystroke sequence. NIM **192** includes tab **194** that allows the user to select premiere NIMs and a general tab **196** that allows the user to review a general catalog of NIMs that is present in NIM templates database **74** (FIG. **1**). In one embodiment, when a user selects a NIM in list **198** (FIG. **5**), the NIM is added to list **188** and is incorporated into the processed user profile associated with the user. In this way, the user can collect NIMs of interest to the user using a single mouse click. By illustration, consider the case in which a user selects the NIM "AnyDay Calender" in list **198**. In response to this selection, a definition of the NIM "AnyDay Calender" is obtained from NIM templates database **74** and is copied directly into the processed user profile associated with the user. Furthermore, the title of the selected NIM, "AnyDay Calender" is added to list **188**. Finally, a visual manifestation that corresponds to the NIM "AnyDay Calender" is displayed

on output means **42** (FIG. **1**). As a result, the display illustrated in FIG. **5** adopts the appearance illustrated in FIG. **6**.

In FIG. **6**, the NIM "AnyDay Calendar" appears at the top of list **188**. Furthermore, a control **101** associated with the NIM "AnyDay Calendar" in list **188** is filled, indicating that the NIM is currently active. Additionally, as illustrated in FIG. **6**, a visual manifestation **103** corresponding to NIM "AnyDay Calendar" appears on the output means. The user has the ability to toggle this NIM between an inactive and active state by selecting control **101**.

In one embodiment, the user is provided with the option of (i) incorporating a NIM selected in list **198** into the processed user profile or (ii) transiently executing the NIM on client **20**. Furthermore, when the user receives NIMs from other users, the user has the option to transiently operate the received NIMs on client **20**. If the user decides to keep the transient NIMs at a later date, the user has the option to add the transient NIMs to the processed user profile at that time. Thus, in such embodiments, the user effectively has the option to "preview" NIMs before adding them to the processed user profile. This is advantageous because it reduces the chances of filling the user profile with undesirable NIMs. Such a feature is particularly advantageous in the case of novice or inexperienced users of system **10**. Furthermore, one of skill in the art will appreciate that the concept of transient NIM execution raises the possibility of executing NIMs on a client **20** during a period of time in which the user is not logged into system **10**. For example, consider a NIM that is executed on a client **20** after a user initiated response to a web page advertisement presented in web browser **36**. Although the user is not logged into web server **58** and therefore does not have a processed user profile resident on client **20**, the user can execute the NIM on client **20** on a transient basis. Furthermore, if the user wishes to add the transiently executed NIM to the user profile **76** associated with the user, the user can log into web server **58** and then add the NIM to the processed user profile that is delivered to client **20** as a function of the log in process.

Another important feature of the present invention is the ability for users to share NIMs with each other. For example, if a user wishes to share the NIM "AnyDay Calendar" that was added to list **188** in FIG. **6**, the user clicks "share" button **170** (FIG. **6**). In response, panel **105** is displayed (FIG. **7**). Because "share" button **170** is pressed while tab **176** is active in the illustration provided by FIGS. **6** and **7**, panel **105** lists each of the NIMs associated with the user. If, however, "share" button **170** is pressed while tab **178** is active rather than tab **176**, panel **105** will list each of the packs associated with the user instead of each of the NIMs. Returning to the situation illustrated in FIG. **7**, the user shares a NIM with other users by selecting the NIM to be shared from list **107** and then toggling button **109** "Share via email." In one embodiment, the user has the option to select multiple NIMs from list **107** using predefined keystroke operations. For example, in one embodiment, the user selects multiple NIMs by clicking on several of the NIMs in list **107** with a mouse button while depressing the "shift" button on the keyboard. When a user decides not to share a NIM and panel **105** is displayed, the user presses cancel button **111** and panel **105** is dismissed.

When a user toggles "share via email" button **109** at a time when one or more NIMs in list **107** have been selected, the definition of each selected NIM is copied from the processed user profile associated with the user into a container and the container is sent to server **50** (FIG. **1**). In the embodiment shown in FIG. **1**, the container is received by web server **58**. Web server **58** includes instructions for routing the container to sharelink database **78** where the container is stored. When the container is stored, a unique identifier is assigned to the container. Although a large number of different mechanisms for generating a unique identifier are practiced in accordance with this aspect of the invention, in one embodiment, the unique identifier assigned to the container upon storage in sharelink database **78** can be subsequently processed to form a URL address that specifically references the container within the context of system **10**. In one embodiment, after a unique identifier has been assigned to the container, an e-mail program is launched on client **20** and the user is requested to designate the recipients of the designated NIMs. Then, each recipient is provided with the unique identifier associated with the container in an e-mail message. When the recipient clicks on the unique identifier, a call is made for a copy of the associated container from sharelink database **78** and the container is delivered to the client **20** associated with the recipient.

As is readily apparent upon review of FIG. **7**, the user has the option to size and position the visual manifestation that corresponds to each NIM. Furthermore, by toggling controls, such as toggle button **101** (FIG. **7**), the visual manifestation of a NIM is toggled between an on state and an off state. Such functionality is highly advantageous. First, by using this functionality, the user has the option to create unique arrangements. Second, NIM developers have the ability to control the default position and size of NIMs as well, and can therefore produce an arrangement of NIMs to further specialized purposes. Finally, because the NIMs of the present invention are not trapped in third party applications that have a set of undesirable features such banner ads, the utility and overall appearance of an arrangement of NIMs is enhanced and adopts an independent value. Using the technology disclosed in the present invention, the user collects an assortment of NIMs and arranges them in a customized fashion. The user has the option to "capture" favored arrangements into constructs known as packs, which have been briefly discussed previously.

FIGS. **8** and **9**A illustrate the formation of a pack using the arrangement of NIMs illustrated in FIG. **4**. The process begins when the user toggles button **164** "Make Pack" in FIG. **4**. In the embodiment illustrated by FIGS. **8** and **9**A, panel **113** (FIG. **8**) is displayed when the user toggles button **164** (FIG. **4**). Panel **113** advises the user to open and arrange each of the NIMs that are to be included in a pack. In the case of FIG. **4**, for example, such an arrangement could include the arrangement of NIMs **184** and **186**. The user indicates that specified NIMs are in a desired arrangement by selecting button **115** "Next" (FIG. **8**). When button **115** is toggled, prompt **113** is terminated and prompt **117** is displayed to prompt the user for a name to associate with the designated pack. The user indicates that a name **119** has been provided for the pack by selecting "Done" button **121**.

In the embodiment shown in FIG. **8**, the user further has the option to return to panel **113** and rearrange the specified NIMs before committing to pack creation by selecting the "Back" button **123**. In the situation illustrated in FIG. **8**, the user has provided the name "New DotPack." FIG. **9**A illustrates the state of the visual manifestation corresponding to home NIM **162** after the user has selected "Done" button **121** (FIG. **8**). Specifically, the name "New DotPack" is added to pack list **125** and tab **178** is activated to display the user pack list rather than the user NIM list that is displayed when tab **176** is activated. Furthermore, in response to the user selection of "Done" button **121** in FIG. **8**, a reference to each NIM specified by the user is collected into a pack, along with some state information, and the pack is stored in the processed user

profile associated with the user. Representative state information for each NIM stored in a pack includes whether the NIM was collapsed and the position of the NIM. In some embodiments, the state information includes the dimensions of the last visual manifestation corresponding to the NIM to have been displayed on output means **42**.

FIG. **9** illustrates pack **139**, which is delineated with a dashed box. Pack **139** includes five NIMs **133**. Each NIM **133** includes two primary components, a viewer **135** for viewing content and a frame **137** for providing user functionality. Each viewer **135** provides a platform for reviewing machine readable information, such as HTML, dHTML, images, imbedded ActiveX and Java applications, JavaScript, CSS, Streaming Media, and/or Flash. Furthermore, in some embodiments, viewer **135** provides functionality for processing digitally recorded sound. Observe then that each NIM of the invention utilizes machine readable information that is easily retrieved from a specified address. If the content provider of this information desires to change the machine readable information, changes can be made and then delivered the next time that the machine readable information is addressed. This stands in contrast to prior art techniques in which updates to executable code can be relatively difficult to distribute.

Each frame **137** provides functions and controls for manipulating the visual manifestation of the NIM **133** corresponding to the frame. For example, some frames **137** include a dismiss button **141**. When the user clicks on dismiss button **141**, the corresponding NIM **133** is dismissed. Additionally, some frames **137** include a menu button **143**. When the user clicks on button **143**, a menu appears. In one embodiment, the menu is a pull down menu and the menu items are defined in the corresponding NIM definition. In an abstract example, the NIM definition provides a menu definition of the format:

Menu **143**-(I)-"Get more NIMs" URL

where (I) refers to the N.sup.th element of the menu that is activated when menu button **143** is pressed, "Get more NIMs" is the menu element name, and URL is the event or web address that is called when the user selects the N.sup.th element of the menu associated with button **143**.

The developer has full control over all aspects of the appearance and functionality of NIM frame **137**. Thus, a NIM developer has the ability to control, for example, the functionality located at any corner of frame **137**, functionality placed along the top or bottom of the frame, or on the sides of the frame. As an illustration, frame **137-3** includes a control button **145** that allows the user to resize the visual manifestation of NIM **133-3**. One of skill in the art will appreciate that the bottom row of NIM frame **137** could have any number of buttons, including a forward control, a backward control, and similar controls, each of which provides the user with distinct functionality.

An advantage of the present invention is that it is possible to embed commands that affect frames **137** in the content of the computer readable media delivered to frame viewer **135**. The use of embedded commands provides NIM developers with powerful tools and additional flexibility. For example, a developer can use embedded commands, such as the menu command above, to design a NIM that has a context sensitive help menu. Each time a particular event occurs, the developer changes the content of the help menu using embedded commands. For example, when a sound file is delivered to a NIM, the sound file may be attached to a set of embedded commands that provide sound manipulation functionality in the form of a pull down menu. Elements of such a pull down menu include a command for saving the sound file to disk, commands for fast forward, stop, and play, and commands for sound enhancement. After the sound file has been played, embedded commands are used in this example by the developer to reset the menu associated with button **143** to some default state. One of skill in the art will appreciate the benefits and advantages of a frame **137** that is capable of being modified based on commands embedded in the machine readable media delivered to the corresponding viewer **135**. The developer can use presentation tools, such as adding transient help buttons, resizing the visual manifestation of the NIM, changing the frame border color, changing the title of the frame, and changing the frame border patterns, to create a more effective application.

One of the advantages of the disclosed pack system is that it provides a convenient mechanism for rapidly assembling NIMs that track diverse sources of information. Furthermore, when a useful set of NIMs is collected into a pack, the user can share the pack with other users using the same procedure previously identified for sharing one or more NIMs. It will be appreciated that in some embodiments, NIMs are commercial applications and that appropriate use of packs provides an additional dimension for application development. Therefore, in some embodiments, pack recipients are subscribers to a service provided by a pack developer. In other embodiments, pack recipients are purchasers or licensees of packs. In still other embodiments, pack recipients receive packs developed by friends, family members, or business associates.

The technology of the present invention further provides a set of NIM management tools to help a user manage displayed NIMs. In some embodiments of the present invention, these management tools are provided by visual management module **114** (FIG. **1**). Two such NIM management tools are, in fact, properties that are associated with NIMs, namely magnetism and snapping. The property of magnetism is exhibited when a visual manifestation corresponding to a first NIM is dragged or moved near a visual manifestation corresponding to a second NIM. When this occurs, the two NIMs exhibit a magnetism that causes the first NIM to accelerate toward the second NIM. However, when the visual manifestations are within a predetermined distance of each other, the NIMs snap together. In one embodiment, the predetermined distance that triggers the two visual manifestations to snap together is a gap of about five pixels. While the above discussion describes the principles of snapping and magnetism based on a pair of NIMs, there is in fact no limitation on the number of NIMs that can be snapped together and furthermore, the principle of magnetism is not dependent on whether a NIM is in fact snapped to another NIM or not.

An additional management tool, illustrated in FIG. **9**A, provides a mechanism for selecting multiple NIMs and for moving the NIMs in a coordinated fashion. In FIG. **9**A, visual manifestations **162**, **184** and **186** corresponding to respective NIMs are locked together. In response, halo **127** is drawn around the selected locked visual manifestations to graphically notify the user which NIMs are locked together. As mentioned previously, the user has the option to position NIMs as a coordinated group. For example, in one embodiment, when halo **127** is displayed and the user moves mouse arrow **129** after selecting one of the visual manifestations corresponding to a locked NIM, a target manifestation **131** is displayed to indicate to the user the position that the selected NIMs will be relocated to if the user clicks a mouse key. In some embodiments, target manifestation is a shadow image of the NIMs within halo **127** rather than the box depicted in FIG. **9**.

The present technology further provides additional methods for controlling visual manifestations of NIMs. For example, in one embodiment, the user has the option to select

multiple NIMs by pressing a predefined key such as the keyboard "Ctrl" key, before selecting a specified NIM. While continuing to depress the control key, the user has the option to select additional NIMs and add the corresponding NIMs to a group. The user then has the option to move each of the NIMs in a single coordinated fashion as a group. Furthermore, by entering a designated keyboard or mouse sequence, the user has the option to move a single NIM even in situations where the NIM is in a group. In one embodiment in accordance with this aspect of the invention, the user clicks the visual manifestation corresponding to a locked NIM that the user wishes to move in an independent manner and the user does not click the visual manifestation when the user wishes to move the NIM in a manner that is coordinated with the other NIMs. Additional features of the present invention include the option to select rows or columns of NIMs using specialized control sequences. For example, in one embodiment of the present invention, the user selects a column of NIMs by clicking on a NIM while holding down the alphanumeric character "c" on the keyboard.

It will be appreciated that one advantage of the present invention is that it is possible to display multiple NIMs and that each NIM provides a specialized visual experience. Therefore, NIM alignment tools are advantageous because they allow users to quickly make room on output means **42** for additional NIMs and/or to produce highly styled NIM arrangements. Accordingly, the present technology provides a specialized feature to rapidly align NIMs. This technology is illustrated in FIG. **10**. The technology works in conjunction with the tools for selecting multiple NIMs. In FIG. **10**A, the user selects the visual manifestations **147**, **149** and **151** corresponding to respective NIMs using, for example, the column select feature disclosed above. Then, the user pushes the selected NIMs against boundary **153**. In one embodiment, boundary **153** is the horizontal or vertical edge of output means **42**. In another embodiment, all visual manifestations corresponding to NIMs are displayed in a single viewport such as a window. In such embodiments, the horizontal and vertical edges of the window each represent a boundary. FIG. **10**B illustrates how visual manifestations **147**, **149**, and **151** are automatically aligned when they are pushed against a boundary, such as boundary **153**. In some embodiments, the relative alignment between the visual manifestations is preserved even after the manifestations are moved in subsequent action by the user. The present technology further allows for the rearrangement of NIMs along a particular axis. For example, if NIMs are substantially oriented along a vertical axis as shown in FIG. **10**B and the user wishes to realign the NIMs on the horizontal axis, all the user has to do is push the collection of NIMs against a horizontal border. For example, when the user pushes visual manifestations **147**, **149** and **151** against border **155**, the visual manifestations realign to conform to border **155** thus resulting in the view depicted in FIG. **10**C.

The features of the home NIM of the invention have been fully described. Attention presently turns to the architecture and operation of individual NIMs utilized in accordance with the invention.

In one embodiment of the invention, after a user has logged into the system, as discussed above in connection with FIGS. **1** to **10**, the user interface **40** displays the home NIM **162** as shown in FIG. **5**. The home NIM typically includes a list of NIMs **188**, referred to in FIG. **5** as "MyDots." These are NIMs which have been "collected" by the user. The list of collected NIMs, along with their associated definitions, is stored on the server in the user profile database **76**, and downloaded from the application server **50** in response to a request from the client parser application **38**. The local copy of the processed user profile is then further processed when the user collects or uses NIMs.

Collected NIMs may be opened or closed by clicking on the control button next to the NIMs name or on the NIM's name itself, in list **188**, and all NIMs may be closed by clicking on the "all off button **166**. The user may place NIMs into categories in a list of categories **190**, which can be edited by clicking on the-Edit" button **168**. New NIMs may be added to the user's collection of NIMs by clicking on the "Get" button **172**.

FIG. **5** also shows a NIM **192** with a list of NIMs **198**, which may be previewed and/or collected by the client **20**. The user may preview or collect a NIM by clicking on the associated name of the NIM, e.g., "eHOW". The steps taken to provide the NIM to a user are shown in FIG. **11**. After logging in (step **240**) the user clicks on the name of a NIM, and the client parser application **38** sends a request including the NIM_ID of the NIM definition, to the applications server **50** via the transmission channel **44** (step **241**). Alternatively, the user may click on a NIM link before logging in, for example if the link has been e-mailed to the user, and then, after clicking on the link, log in. In another aspect, the user could view, but not collect, a transient NIM without ever logging in.

After the user is logged in and has clicked on the NIM, the applications server **50** retrieves the NIM definition from the NIM template database **74** using the NIM ID, in step **242**, and provides it to the client **20** in step **243**. The client **20** receives the NIM definition from the applications server **50** in step **244**, and the client parser application **38** creates a frame in the display of the user interface **42** in step **245**. In step **246**, the client **20** requests the necessary content elements **94** stored at the URLs identified in the NIM definition from the corresponding content server **82**. The content server **82** transmits the content **94** in step **247**, and in step **248** the client parser application **38** places the content in the viewer, which is enclosed by the frame, allowing the user to preview the NIM. Alternatively, the client parser application **38** may simply collect the NIM, adding it to the user's processed user profile.

The user may then view the NIM on the user interface display **42**, and may interact with the NIM much in the same way as a user may interact with Internet content or web applications. This may change the NIM from its present, "raw" state to a used state reflecting alteration or use of the NIM by the user. For example, the user may direct the NIM to different content within the NIM if the NIM content enables the user to do so. Or, the user may provide information to the content server **82** which allows the NIM to be personalized. The user may additionally be given the option of changing the size of the frame.

If the user collects the NIM, the NIM will be added to the user's list of collected NIMs such as the list **188** shown in FIG. **5**. Additionally, the client parser application will add the NIMs definition to the processed user profile, and, on logout, send the processed user profile to the application server **50**. Thus, the NIM's "state" will be preserved. Alternatively, the client parser application may collect the NIM automatically, without waiting for a user command, by adding the NIM definition directly to the processed user profile.

If the NIM's state has been altered by the user or by the content—if for example, the user has directed the NIM to Internet content other than the initially-displayed content, provided personalizing information, or changed the properties of the frame, or if the content itself has caused an alteration in the NIM—this alteration will be reflected in the NIM definition stored in the user profile database **76**. Information

which personalizes the resulting content, instead of being stored in a "cookie" on the client's hard drive, can be stored as part of the NIM definition. This advantageously permits personalization of content, such as web content that is associated with the NIM content and the user, without storing a cookie on the client **20**.

A user may also access a NIM which has been previously collected, and possibly altered by use as explained above. As previously described, the user profile **76** includes NIM definitions for NIMs which have been viewed and collected by each user. A screen shot showing NIMs **188**, which have been previously collected by a user is shown in FIG. **5**. The steps taken to provide the user with NIMs which have been previously collected are shown in FIG. **12**.

As discussed above, on login (step **250**) the user's profile is retrieved by the client parser application **38** in the client **20** (step **251** and **252**). The user's profile, stored in the user profile database **76**, includes the NIM definition for each of the NIMs previously collected, and possibly altered, by each user. The NIM definitions, as discussed above, includes the NIM frame definition and the definition of the controls for filling the viewer within the frame with content. After log in, a local copy of the processed user profile is stored on the client **20**, and this copy is further processed as the user collects new NIMs, or uses new or collected NIMs such that the NIMs are altered.

When the user clicks on the name of a collected NIM (step **253**) the client parser application **38** creates a frame in the display **42** of the user interface **40** in step **254**. At step **255**, the client **20** requests the necessary content elements **94** stored at the URLs identified in the NIM definition from the corresponding content servers **82**, which provide the content **94** in step **256**. It will be appreciated that these URLs need not be the same as the initialization URLs in the "raw" NIM definition stored in the NIM template database **74** on applications server **50**, and in fact the content servers need not be the same content servers corresponding to the initialization URLs. In step **257**, the NIM parser application **38** places the content in the NIM frame, and the NIM is then fully opened.

FIG. **13** illustrates a data structure for a NIM definition. As discussed above, a NIM is defined as a frame that contains a collection of controls, or functional units, such as a web rendering control or a GIF rendering control. The NIM frame surrounds a viewer, which displays the addressed content. The MM has a defined layout or arrangement of the controls, and defined initialization input data, e.g. data and URLS, for each control or element, in the NIM. NIM definitions are available to the client parser application via NIM links. The NIM links "point" to NIM definitions, which include all the information needed to build a NIM frame and fill the NIM with NIM content. Thus, NIMs links are easily collected, associated into packs, and shared by users.

In one embodiment, the NIM definitions are defined using Extensible Markup Language (XML), so that the NIM as a whole—the frame and the content within the viewer—is advantageously as flexible as standard web content. NIMs are extremely flexible, because the definition of the NIM is content, rather than compiled code. The NIM definition defines the structure of the NIM, and everything that is visible in a NIM is based on standard Internet content, such as HTML, dHTML, or GIFs, and is referenced or pointed to by the NIM definition. An "application"-type NIM, such as a web calendar or web mail, may be changed by the user, by the content provider, or by other content, while advantageously avoiding the need to distribute and support a hard-coded compiled application. The definition of a NIM thus includes everything that is needed for the NIM to be rendered and filled with Internet content.

As shown in the exemplary embodiment of FIG. **13**, the definition of a NIM includes tags that identify the NIM **270**, define and configure the NIM frame **271**, specify and layout the controls **273** in the NIM viewer, and specify parameters to initialize all the NIM's components with content or data.

In one embodiment, a NIM is identified by three ID strings **270**: GlobalID, Domain and Kind. A GlobaliD is used when the MM definition is within a share. It is unique with respect to other NIM tags in the share. A NIM's domain is a unique label for the owning company or developer of the NIM, such as "dodots.com." Finally, a NIM's kind, which is specified by the NIM's developer, is a helpful identifier for finding the NIM, but need not be unique. Examples of possible NIM kinds include "mp3", "scriblets," and "calculator." As discussed above, a NIM definition will typically be written in a format which facilitates sharing of data over the Internet, such as XML. An XML specification for the NIM identification strings, for one embodiment of the invention follows. The bold text identifies NIM definition data, while the remaining text describes the data.

GLOBALID="string" Used only within <SHARE> tags. This GLOBALID must be unique with respect to other <NIM> tags in this <SHARE>.

DOMAIN="string"

Unique label for the owning company of this NIM. In theory, NIMs may be limited to communicating with NIMs only from their own domain.

KIND="string"

Helpful identifier for finding such a NIM from another NIM. Does not have to be unique.

The NIM definition also includes the definition of a frame **271**, which specifies the frame size and shape, and optionally the frame orientation and/or location on the user's screen. The space within the frame is the control space or viewer; visible controls are distributed within the control space or viewer.

The NIM definition may optionally include controls for: a titlebar; a NIM menu with flexible menu entries; an exit button; and a bottombar. A typical layout for these components is: titlebar at the top of the control space, with menu on the left and exit button on the right, and the bottombar at the bottom.

The titlebar component gives the user a place to grab and drag the NIM in a windowed environment. In one embodiment, it is implemented as a GIF rendering control that can be targeted to a local or remote titlebar image. The titlebar will preferably have a fixed height and width that is a function of the NIM's width. The titlebar is preferably capable of being located at any position on the periphery of the NIM. Overlay text can also be specified to layer on top of the titlebar image. The bottombar may be implemented in a similar fashion, but typically will not include text overlay. The titlebar and bottombar may be filled in with initialization data from a fixed data file, or alternatively with Internet content from, example, an initialization URL.

In one embodiment, a menu definition **271** is also included in the NIM definition. The menu includes items and actions of the NIM provider's choosing. For example, menu items may include the title "browse" associated with the action of targeting a full-screen browser or another NIM, and retrieving content for that browser or NIM from a specified address such as a URL. Logging off, or directing the NIM to another address or URL, are also possible menu action items. Menu action items that require communication of messages between the NIM and another NIM may also be provided—

for example, opening another NIM, or changing the content of another NIM that is already open. Communication of messages between different parts of the system is discussed below.

An XML specification for a frame, titlebar, bottombar and menu, for one embodiment of the invention follows:

```
<FRAME>
<TITLE>
TEXT="string"
JUSTIFY=-"LEFT"|"CENTER"|"RIGHT"
COLOR="#XXXXXX"
PIXELWIDTH="integer"
Width in pixel units. Overrides WIDTH attribute.
PIXELHEIGHT="integer"
Height in pixel units. Overrides HEIGHT attribute.
WIDTH="integer"
Width in NIM units. Default value is 1.
HEIGHT="integer"
Height in NIM units. Default value is 1.
X="integer"|"LEFT"|"CENTER"|"RIGHT"
Initial X position in screen coordinates. Default is center.
Y="integer"|"TOP"|"CENTER"|"BOTTOM"
Initial Y position in screen coordinates. Default is center.
FIXWIDTH="TRUE"|"FALSE"
Default is false.
FIXHEIGHT="TRUE"|"FALSE"
Default is false.
BACKGROUND COLOR="#XXXXXX"
Default is white.
<TITLEBARIMAGE>
JUSTIFY="LEFT"|"CENTER"|"RIGHT"
TILELEFT="integer"
TILERIGHT="integer"
NORMAL="URL"
DOWN="URL"
HOVER="URL"
INACTIVE="URL"
<BOTTOMBARIMAGE>
JUSTIFY="LEFT"|"CENTER"|"RIGHT"
TILELEFT="integer"
TILERIGHT="integer"
NORMAL="URL"
DOWN="URL"
HOVER="URL"
INACTIVE="URL"
<MENU>
Contains zero or more <ITEM> tags.
<ITEM>
TITLE="string"
TOOLTIP="string"
ICON="URL"
ID="string"
Must be unique.
<ACTION> RECIPIENT="address" MESSAGE="string"
```

As shown in FIG. **13**, the NIM definition also includes layout and definition of the controls **273**. A control may be visible and render some sort of visual or text display, either static or dynamic. A control may be hidden, for example a functional element that is not necessarily visual such as a Java control. The control definition **273** includes identification of the types of controls, the layout of the controls, and initialization information. In one embodiment, NIM controls are specified and identified by class, kind and ID. Class defines the type of NIM control and is not unique. Kind is a useful identifier selected by the developer, and again is not unique. The NIM ID is unique within a user's processed profile.

Different classes of controls may be used. For example, a control may be a web rendering object, which can render web content such as HTML, dHTML, images, imbedded ActiveX and Java applications, JavaScript, CSS, Streaming Media, and/or Flash. Alternatively, a control may be any object capable of rendering any kind of computer readable media, such as a GIF rendering object or an custom-designed object to display a particular kind of information. Alternatively, a control may be an object capable of processing any kind of application logic, such as a Java module. For example, an on-line brokerage firm could implement a custom stock-chart-rendering control, and define a NIM to use this control.

As discussed above, the control space is located within the frame, and one or more controls can be flexibly positioned within the control space, and these controls may include a titlebar and a bottombar, as well as other controls. The location of controls is specified by the layout in the definition of the controls **273** within the NIM definition. In one embodiment, the controls are laid out or positioned within the NIM frame according to a flexible grid. In this embodiment, the NIM definition allows the control space to be subdivided into equal vertical and horizontal units, and then for the controls to be positioned and sized within the control space.

A control definition will typically include initialization data. For example, where a control is a web rendering object, the definition will include initial URLs. When the NIM is opened, the control will navigate to the initial URLs to obtain content and render the NIM. If the control is a GIF, the control could retrieve the GIF file from a content server or from the application server. A NIM definition may optionally include additional tags identifying initialization parameters for different platforms: for example, a URL is suitable for a PC, but a "P-URL" may be provided as well, pointing to content suitable for users viewing NIMs through a personal digital assistant (PDA) or similar device.

Controls are typically installed on the applications server, and may be updated after installation by the applications server. The home NIM code, downloaded when the client becomes "NIM-enabled," includes the then-existing controls. Controls are updated as new controls are installed on the server or when a user requests a NIM that requires a new control. The server may then may download such updates to the client parser application, for example on log-in. The NIM framework allows any control to be positioned and initialized in a control space in a NIM, as discussed above.

An XML specification for control definition and layout, in accordance with one embodiment of the invention, follows:

```
<CONTROL_LAYOUT> Contains zero or more <CONTROL> tags.
WIDTH="integer" Divides control space into this many evenly spaced
columns. Default is 1.
HEIGHT="integer" Divides control space into this many evenly spaced
rows. Default is 1.
WIDTHSCALES="TRUE"|"FALSE" Default is true.
HEIGHTSCALES="TRUE"|"FALSE" Divides control space into this
many evenly spaced columns. Default is 1.
<CONTROL>
CLASS="string" Class may be "Browser," "GIF reader," or other
object for rendering computer readable media.
KIND="string"
ID="string" Must be unique with other controls in this NIM.
LEFT="integer" X position of the control in container unites. Default is 0.
TOP="integer" Y position of the control in container units. Default is 0.
WIDTH="integer" Width in container units. Default is 1.
HEIGHT="integer" Height in container units. Default is 1.
URL="URL" This is read if and only if this control is of class "Browser".
This is the URL to which this control navigates. Otherwise, may include
address for other control content, e.g. GIF address in applications server.
<CATEGORIES> Contains zero or more <CATEGORY> tags.
<CATEGORY> Adds the NIM to this category. This is the only way
categories are specified; i.e. there is no master category list.
NAME="string" This is the name of the category.
```

A NIM definition may also optionally include home NIM categories **274**. A home NIM category used by home NIM **204** is a convenient way for a user to keep track of collected NEVIS. When a user adds a NIM to a category **204**, the

category is added, as a string element, to the categories element **274** of the NIM definition in the user profile. For example, a user may categorize a particular NIM as "entertainment," or "news," or "reference." This category will then be added to the categories element **274** of the MM definition.

A NIM definition may also optionally include an events element **275**, which defines actions to certain NIM events. For example, the OnClose event, when a NIM is closed, may be assigned a specific and targeted action, similar to a menu item. An XML specification for the event element in a NIM definition, in accordance with one embodiment of the invention, follows:

```
<EVENTS>
<ONCLOSE>
  Executes this action list on close.
<ACTION_LIST>
  Contains zero or more <ACTION> tags.
<ACTION>
RECIPIENT="address"
MESSAGE="string"
```

A sample NIM definition, in an XML file format in accordance with the above specification, follows:

```
<NIM DOMAIN="calculator" KIND="basic">
<FRAME CLASS="Standard" BACKGROUNDCOLOR="#FFFF00"
WIDTH="6" HEIGHT="4" FIXWIDTH="TRUE"
FIXHEIGHT="TRUE">
<TITLE TEXT="Basic Calculator" COLOR="#000OFF"
JUSTIFY="LEFT"/>
<TITLEBARIMAGE JUSTIFY="LEFT" TILELEFT="1"
TILERIGHT="1" NORMAL=" " INACTIVE=" " HOVER=" "
DOWN=" "/>
<BOTTOMBARIMAGE JUSTIFY="LEFT" TILELEFT="1"
TILERIGHT="1" NORMAL=" " INACTIVE=" " HOVER=" "
DOWN=" "/>
</FRAME>
<MENU/>
<CONTROL LAYOUT WIDTH="1" HEIGHT="1">
<CONTROL CLASS="Browser" KIND="A" ID="1" TOP="0"
LEFT="O" WIDTH="1"
HEIGHT="1"
URL="http://www.dodots.com /dots/Calc/CALCULATOR2.htm"/>
</CONTROL LAYOUT>
</NIM>
```

The first line of this definition establishes the identification of the NIM definition, as discussed above: it is in the domain "calculator," and the kind of display is "basic."

In one embodiment, the domain will be the domain name associated with the content provider. The domain name is a unique label for the provider or developer of the NIM. The NIM's "kind" is a helpful identifier for locating the NIM, and need not be unique. A NIM may also be identified using a GlobalID, when the NIM is being shared. Since the NIM defined by this XML file is not being shared, it does not have a GlobalID.

The second line of the example XML NIM definition establishes the size and appearance of the NIM frame, defining a NIM viewer in which the NIM content will be placed. The third line ensures that the height and width of the frame are fixed—that is, the size of this frame cannot be adjusted by the user. The fourth and fifth lines establish the title of the NIM—"Basic calculator"—and its location. The next four lines establish the location and placing of the titlebar and bottombar, and relevant images, e.g. mouse-over. Thus, the first part of the example NIM definition defines the NIM frame. The definition of a frame, titlebar, menu and other aspects distinguish a NIM from a browser—the content provider has control over the frame size and every aspect of the NIM's appearance, whereas when a browser is used, the content provider has to adapt to the browser display size, and browser titlebar, menu, logo and other aspects cannot be controlled by the content provider.

The rest of the NIM definition identifies, positions, and initializes the NIM's controls, which, in this case, are contained by the NIM frame. In this example, the next few lines establish that a single control will start in the upper left corner of the NIM viewer, that the control is of the type "browser," or web-rendering, and that the initialization URL for the control is www.dodots.com/dots/Calc/CALCULATOR2.htm. This URL is typically referred to as the "initialization URL," because it is where the NIM looks for NIM content when it is opened. Where the control is of type "browser," the content will typically be HTML content. However, any standard Internet content—HTML, dHTML, flash, streaming media, or Java, for example—may be used. As discussed earlier, a control, could include types other than a browser. The final two lines of the XML file close the definition file.

It will be appreciated that the NIM is designed such that content consumes the entire frame. In one embodiment, the content for the corners of the frame—the menu and the exit button—and the frame sizing images are served by an application server and referenced when the user logs in. Everything else is developed and served by a separate NIM developer. This differs fundamentally from the current approach to providing web content, in which there is a strong distinction between the viewer application—the browser—and the web page or web content. Using present browser-type technology, the content is trapped within the viewer. To obtain a cohesive application feel and access to application features, the current alternative is to develop custom client applications. NIMs allow a developer to provide an application feel without developing custom client applications.

NIMs and the client parser application have a messaging architecture—the NIM messaging architecture, or NMA—that enables NIMs, controls, and the client parser application to communicate. Messaging, in combination with the NIM definition, gives the content of a NIM access to the application/rendering program—the client parser application—and to other NIMs, allowing true application behavior. The NIM definition, discussed above, is accessible, flexible, and may be changed by a NIM or a user while the NIM is in use, even after it's been rendered.

The content provider, the user, or other NIMs can change a NIM. For example, the content—which includes the titlebar and menu elements—may be changed by the NIM provider by simply enhanced NIM content, using 20 messaging, so that NIMs can exhibit true application behavior. For example, an online brokerage firm can go beyond providing a NIM that renders stock tracking charts, and allow users to trade on-line via a second NIM that can interact with other NIMs such as the first NIM to help facilitate the activity.

A NIM can be changed by its content, or by another NIM, using messaging. This enables a NIM to, for example, notify a user of events, such as a change in content. A NIM could, for example, remain open, but in a collapsed mode, until a particular event occurs, and could then either expand to normal size, or open another related NIM. For example, if a particular stock hits a predetermined price, the stock tracking chart NIM discussed earlier can notify the user by expanding, by popping up a message, or by opening another NIM (such as a stock trading NIM). Alternatively, the NIM could notify the user of a particular event by coming into focus or changing size or content. These changes could also be made by sending messages from a content or applications server to the NIM.

As illustrated in FIG. **14**, all elements of the system can send and receive NMA messages. Message routing between NIMs, from a NIM to itself (that is, between e.g. the frame elements and a control, or one control and another), and from NIMs to the system, are handled by a message interface module **106**, which is part of the client parser application **38** in the client **20**. In one embodiment, the message interface module **106** resolves addressing queries, executes system-level commands from the NIMs such as "close all NIMs," and passes messages between NIMs. The message interface **106** may also communicate messages to controls such as the browser class controls, for example "navigate the addressed NIM's browser control to the argument URL." Alternatively, the message interface module **106** may route a message to an application or content server (not shown in FIG. **14**) for expanded functionality. In one embodiment, the message interface module **106** uses an HTTP request to access an application program interface (API) call, with data optionally being sent and received in XML format. For example, the message interface **106** could send a message providing user input, such as credit card information to a credit processing module on a web server **82**.

In one embodiment, an NMA message has two components: a recipient, or address, and the message body. Both are represented as strings. The address may specify an exact NIM, a kind of NIM or control, a domain, or the system, meaning the overall home NIM display. For example, in one embodiment, the address may be in the form:

#<NTM specifier>:<control specifier>
if the communication is between NIMs in the same domain, or
#<domain>:<NIM specifier>:<control specifier>
or
#<domain>:<NIM specifier>:<NIM id>:<control kind>:<control id>
if the communication is between NIMs in different domains, or
#system
if the communication is to the system.

In one embodiment, if the address is not properly specified it defaults to #system. The message interface **106** in the client **20** can flexibly allow, restrict MM addressing or sending messages—for example, the message interface can ensure that only certain NIMs can send message to a particular NIM. This allows NIM developers to develop coordinated NIMs that can interact, by sending messages to e.g. change content or open one another, without allowing other NIMs to interact with their coordinated NIMs.

In one embodiment, a specifier in n address may be the unique identification of the NIM, control or domain in question: <specifier>:=<ID>. Alternatively, where the kind of NIM or control receiving the message is important but the specific NIM or control is not, the specifier may address a message to the closest matching recipient using a search criteria: <specifier>:=<kind>#<search criteria>. The kind should be a type of NIM or control that is installed in the system. Search criteria may be, for example, "any," "open," or "closed." Finally, where the control is being specified, the specifier may be a symbol, such as ".", indicating that the message is addressed to the sending NIM. For example, in one embodiment, the address #system sends the message to the system. The address #. sends the message to the NIM which sent the message. The address #**7** sends the a message to the NIM with the identifier **7**. The address #B#open:**5** sends a message to the control with the identifier **5**, in the first open dot of kind B found in the system.

In one embodiment, certain NIMs will have privileges to send particular messages to certain NIMs, and not to other NIMs. For example, NIMs in a domain may be permitted to send control messages to other NIMs in the same domain, but not to NIMs in other domains. Thus, a NIM provider may have control over messaging between NIMs in his domain, and prevent NIMs in other domains from changing NIMs in his domain. Alternatively, NIM providers may coordinate with NIM providers in other domains, permitting certain messaging privileges between some of the NIMs in their respective domains. The HomeNIM and other system NIMs, such as the login NIM, which are in the system domain, may have certain messaging privileges that no other NIMs have, such as logging the user out or closing all the currently-open NIMs.

In one embodiment of the invention, the second part of the message, the body, is represented as a string of characters. Messages may be sent to the client parser application, to the frame of a NIM, or to a control. Messages may be specific, defined messages, as shown in the following examples, or may be any javascript, which may be sent in or out of NIM content. Examples of defined messages to the client parser application, in one embodiment, are:

Application Message<arg>	Function
Refresh	Refreshes the user's profile.
#have-NIM <NIM-address>	Check if the user has the specified NIM as part of the user's profile.
#delete-NIM <NIM-address>	Remove the specified NIM from the user's profile.
#get-screen-width	Returns the width of the screen.
#get-screen-height	Returns the height of the screen.
#close-all-NIMs	Closes all open NIMs.
#get-NIM-ids <NIM-address>	Returns the NIM ID of the specified NIM.

Messages can also be sent from a NIM to itself, or to another NIM, and the identified actions or functions, specified in the body, are performed on the receiving NIM. The following are examples of messages to a NIM for one embodiment of the invention:

Defined NIM Message<arg>	Function
#set-size<width><height>	Sets the size of the NIM.
#set-width<width>	Sets the width of the NIM.
#set-height<height>	Sets the height of the NIM.
#set-position<x-pos><y-pos>	Sets the position, of screen, of the NIM.
#set-title<title>	Sets the title of the NIM.
#collapse	Collapses, but does not close, the NIM.
#uncollapse	Uncollapses the NIM.
#set-user-sizable<width true/false><height true/false> sizeable	Establishes whether the NIM is by the user
#set-background-color<color>	Sets background color of the NIM.
#set-title-text-justify<justify-keyword>	

The above examples of NIM messages may, in one embodiment of the invention, be sent to a NIM by another NIM. For example, a user may provide input to a NIM, for example a stock tracking chart NIM, indicating an interest in another NIM, such as a stock trading NIM. The current NIM may then send an "open" message to the second NIM to open it, if it wasn't already open. The current NEM may then send a navigate message (see below) that may include an argument, such as a URL or other content pointer, so that the second NIM could be opened to a specific URL indicated by the first NIM.

Finally, messages may be sent to a control of a NIM, either by the NIM itself, another control, the HomeNim system, or another NIM. As examples, the following control messages are provided in one embodiment of the invention:

Defined Control Message	Function
##<any javascript>	Run any javascript in the control, e.g. javascriptin a browser-type control.
#show	Set visibility control within a NIM.
#hide	Set invisibility of a control within a NIM.
#get-size	Get size of control.
#get-address	Query for unique ID of a control by kind.
#navigate	Navigate the control.

Control messages may be used by a NIM, addressed to its own control, or to the control of another NIM. Control messages may also be sent by the HomeNIM, or by the applications server or content server.

An example of a message to a NIM is window.external.PostMessage("#.:","#collapse)" this is a message from a NIM, to itself, collapsing the NIM. Another example of a message from a NIM to another NIM is "window.external.PostMessage("#mp3#any:","#open")", which is addressed to a NIM with the kind "mp3," but only if it is in the same domain as the sending NIM. The body of the message instructs the message of type mp3 to open. An example of a javascript message is "window.external.PostMessage ("#mp3#any:**1**", "**33**play( )")", which sends a message to the mp3 NIM control that calls the javascript function play( ).

The operation of the home NIM and individual NIMs has been described. Attention presently turns to different techniques used in accordance with the invention **25** to host NIMs.

As shown and described in relation to FIG. **1**, the application server **50** includes a NIM Management Modulel **12**, a NIM Templates Database **74**, a ShareLink Database **78**, and a user profile database **76**.

FIG. **15** is a diagrammatic illustration of an embodiment of the NIM Management Modulel **12**. NIM Management Modulel **12** may contain the NIM Templates Database **74** and the ShareLink Database **78**, discussed in further detail below. NIM Management Module **112** may also contain content **402** for filling in a NIM frame or for rendering Internet pages. Alternatively, content **402** may be stored elsewhere, such as on a Web server similar to the server **82** shown in FIG. **1**. As discussed above, the content preferably contains Internet content such as HTML (Hypertext Markup Language), dHTML, and images.

In addition, Management Modulel **12** preferably contains executable procedures **403** for controlling and managing the NIM system. These procedures **403** may include: a Collection procedure **404** for obtaining new NIMs; a Sharing procedure **405** for sharing NIMs with others; Development procedures **406**, such as a DevZone procedure **407** and a DevNIM procedure **408**, for creating, modifying or deleting NIMs; Publishing procedures **409**, such as a PubZone procedure **410** and a PubNIM procedure **411**, for publishing NIMs so that they are publically accessible; and Administration procedures **412**, such as an AdminZone procedure **413** and an AdminNIM procedure **414**, for administering the system. It should be noted that the NIM sharing procedure, discussed in further detail below, may be processed by either the Server module (**102** of FIG. **1**), or the Sharing procedure (**405** of FIG. **15**). Control and management of the server and the NIM Management Modulel **12** components will now be discussed in further detail.

FIG. **16** is a diagrammatic illustration of an embodiment of the NIM Templates database **74**. In this embodiment, NIM Templates database **74** primarily stores XML NIM definitions in their initial unmodified state as initially designed by a partner and which have not been altered by a user in any way. The unmodified NIMs are also referred to as "Raw NIMs". The NIM Templatesdatabase **74** is used as the starting point for the development of NIMs which may later be customized or modified by a user, developer, or system administrator, as discussed below.

For ease of explanation, the individual or organization that controls the server (**50** of FIG. **1**) will hereinafter be referred to as the system provider and the individual or organization who supplies the content will hereinafter be referred to as a partner. It should be understood that a provider, partner, user, developer, and administrator of the system may be distinct entities, the same entity, or a combination of both. Furthermore, as discussed above, each of the above entities is assigned access rights or privileges which permit or forbid that entity from performing different actions on the system.

FIG. **16** illustrates an embodiment of a NIM Template database **74**. A NIM identification number (NIM_ID) **416** is stored in a NIM Template table **41.5** in the NIM Templates database **74**. A Raw NW, identified by its NIM_ID, includes a plethora of RAW NIM characteristics, including, but not limited to, a Raw NIM creation date **417**, which indicates when the NIM was created and is useful when searching for NIMs created during a specific time, a Raw N.I.M definition module **418**, and the NIMIndex categories **422** in which the NIM has been categorized. Each NIM when created in typically classified into one or more NIM Index categories by the developer, such as "Applications", "Business", "Entertainment" and "News".

Each NIM is fully configurable and definable. The NIM definition module **418** contains details defining the NIM, such as the look-and-feel **419**, of the Raw NIM, initialization URLs (Uniform Resource Locators) **420**, and a location **421** of where the developer would like the NIM to open on a users's computer screen. The look and feel of the NIM is the appearance and function of the NIM interface. The look and feel may characterize the frame or skeleton layout, the graphics used to represent certain functions, such as opening and closing the NIM, whether the frame is sizable, and the appearance and operation of menus in the frame.

The definition module **418** may also contain Initialization URLs **420** which reference resources containing content. The content resources may be HTML (Hypertext Markup Language), dHTML, images, programs such as Java applets, or any other file supported by HTTP. The Initialization URLs **420** contains the name of the protocol required to access the resource, a domain name that identifies a specific computer on a network, such as the Internet, and a hierarchical description of a file location on that specific computer. These files or resources are then used by the home NIM to fill in the frame and controls with content. In addition, the definition module **418** may contain other details such as the location **421** on a user's computer screen where the NIM should initially open.

The NIMIndex may be used to search for, learn about, and collect NIMs. The NIMIndex is typically accessed from either a Web browser, such as Internet Explorer® or Netscape Navigator® or from the home NIM, **108** of FIG. **1**. A user may search for NIMs by, or according to, any field of the NIM Templates table **415** via the NIMIndex.

A user accessing the NIMIndex from a Web Brow typically navigates to a main NIMIndex web page such as that shown in FIG. **17**. FIG. **17** is an illustration of the main NIMIndex Web page **423**. A user typically navigates to main NIMIndex Web

page **423** from a NIM home-page (not shown), or while anywhere within the NIM Web site by clicking on the "Collect the DOtS™" link **424** in a menu **425**. A user may search the NIMIndex by entering a search term in a form **426** and clicking on the "Search" button **428** which implements the Collection Procedure (**404** of FIG. **15**) to search the NIM Templates database (**74** of FIG. **15**) for NIMs that match the query. In one embodiment, the NLMIndex may be searched by NIM title, NIM description, or partner, as shown in the pull down menu **430**. The user may also browse the NIMIndex by clicking on a link **432** to a NIMIndex category **434** which will navigate the user to a NIMIndex category Web page **440**, as shown in FIG. **18**.

FIG. **18** is an illustration of a single NIMIndex category, the "Applications" category **435**, shown in FIG. **17**. A list of sub-categories (not shown), as well as a list of NIMs **442** and their short descriptions are shown in FIG. **18**. A user may click on the "more detail" link **444** to be taken to a page displaying a full description (discussed later in relation to FIG. **29**B) of the NIM, shown in FIG. **19**.

Navigation of the Internet generally occurs through the use of URLs (Uniform Resource Locators), which are the addresses of files or resources accessible on the Internet. The type of resource depends on the Internet application protocol. Using the World Wide Web's protocol, the Hypertext Transfer Protocol (HTTP), the resource can be an HTML (Hypertext Markup Language) page, an image file, a program such as a Java applet, or any other file supported by HTTP. The URL contains the name of the protocol required to access the resource, a domain name that identifies a specific computer on the Internet, and a hierarchical description of a file location on the computer and usually takes the form: "URL=protocol://machine.name[:port]/directory/document-.name'?[&argu-ments]" The "protocol" is the Internet protocol used to reach the document or resource. On the Web, the "protocol" is typically HTTP, but it can take any number of forms, such as ftp (file transfer protocol), file (a local file), gopher (gopher protocol), mailto (electronic mail address), news (Usenet news), telnet and tn3270 (interactive sessions), wais (wide area information servers), or the like.

The "machine.name" is the name of the host come where the document resides (such as www.NIM.com). The ":port" portion of the address is optional and is only necessary when a resource is accessible through a non-standard TCP port number. Although the standard port number for HTTP is 80, there are numerous Web servers on the Internet that use non-standard ports, such as port 8000.

The NIM system, however, may also utilize a proprietary NIM protocol. An example of a URL using the proprietary NIM protocol is: "NIMS:?NIMTemplate=<N/M_ID>"

The NIM protocol URL is used to collect, distribute, and share NIMs. When collecting NIMs the NIM protocol URL is referred to as a NIMLink. When distributing or sharing NIMs the NIM protocol URL is referred to as a ShareLink.

The "NIMs:" term defines the NIM protocol or scheme and is always followed by a colon. The "?NIMTemplate=<NIM_ID" is an argument, where a dollar sign (5) and a question mark (?) are used to denote path and/or search elements. It should be noted that no path is supplied (i.e.: "//path/to/something"). The argument instructs the client parser application (**38** of FIG. **1**) how to handle a user's selection of a NIM protocol URL and what the NIM protocol URL must do. For example, to obtain a NIM, the argument might read "NIMTemplate=123", to obtain a Share (discussed below) the argument might read "Share=123", to obtain a Pack (discussed below) the argument might read "Pack=123", etc. The argument can be used to cause the client parser application to do anything within it's system of functionality by specifying new argument sets to build new types of special client parser application links.

In one embodiment, the address for where the client parser application (**38** of FIG. **1**) searches the system (**10** of FIG. **1**) for the NIM Template or ShareLink database (**74** and **78** of FIG. **1**) is specified within the processed login script or session_config, although it could alternatively be specified within the NIM protocol URL. When a user clicks on a NIM protocol URL (from any where you can place and click on a link, for example in a browser, in a NIM, in email, in a document, etc.), the client parser application processes the NIM protocol URL in the same manner as a browser processes HTTP links and an email program processes mailto links.

When a protocol URL takes the form of a NIMLink, the client parser application responds by obtaining the NIM definition from the NIM Template database, optionally adds the NIM to the user's processed user profile (unless the NIM has been specified to be opened in transient mode, which may be specified in the argument), and optionally opens the NIM on the user's display screen.

In one embodiment, by default, unless specified otherwise, a NIM will be added to a users collection (transient—false) and will be opened (open—true). A NIMLink with arguments may look as follows: "NIMS:?NIM-Template&transient=true" or "NIMS:?NIM-Template&open=false". More than one additional argument could be added by appending another argument to the URL which may read as "&argument=value".

A ShareLink (discussed below) is similar to a NIMLink and may read "NIMS:?share=123", where 123 is the SHARE ID referencing the share module within the ShareLink Database. Pack Links (discussed below) typically read as "NIMS:?pack-123", where 123 is the PACK_ID referencing a pack module within the NIM Template Database.

FIG. **19** is an illustration of a full description of NIM content **446**. A graphic of the opened NIM may also be displayed **448**.

Once the user decides that he would like to add a NIM to his home NIM, the user clicks on the "get it now" or "Get This Dots™ NIMLink **450** (FIGS. **18** and **19**) which either runs the Collection procedure (**404** of FIG. **15**) which obtains that NIM's NIM definition module (**418** of FIG. **16**) from the NIM Template table (**415** of FIG. **16**), or opens another Web page as shown in FIG. **20**.

FIG. **20** is an illustration of a Web page **452** which might be displayed to the user once the user has clicked on the NIM-Link **450**. The user is presented with an option of either collecting the NIM **456**, or if the user does not have the home NIM application, the user may first download the home NIM by clicking on "Get the homeDotIm" **454**. Once the user clicks on the download the NIM button **456**, the Collection procedure (**404** of FIG. **15**) obtains that NIM's NIM definition module (**418** of FIG. **16**) from the NIM Template table (**415** of FIG. **16**).

The Collection procedure (**404** of FIG. **15**) transmits the NIM definition to the user's home NIM, which optionally opens the MM and saves the NIM definition module (**418** of FIG. **16**) on the user's local processed user profile. All NIM definition modules (**418** of FIG. **16**) on the user's computer may subsequently be saved to the user profile database, as discussed earlier in this writing. Alternatively, a "preview" button may be provided which transiently displays the NIM on the user's computer screen without adding the NIM to the user's local processed user profile. The user may also search the NIMIndex from their home NIM.

FIG. **21** is an illustration of the main home NIM graphical user interface (GUI) **464**, similar to that shown in FIG. **5**. The home NIM displays a list of all NIMs **466** that the user has collected. Furthermore, any NIMs that the user has collected in groups or packs, can be accessed by clicking on the "My Dotpacks'" tab **468**. One way to obtain new NIMs is to click on the "Get" button **470**, which opens the NIM shown in FIG. **22**.

FIG. **22** is an illustration of a get new NIM GUI **474**. A list of all NIMs **476** (or a featured subset) that may be collected by the user are displayed. Clicking on the "More Dots" tab **478** (shown in FIG. **23**) displays further NIMs which may be collected.

When a user selects or clicks on any of the NIMLinks **480**, NIMLink **480** references the NIM_ID (**416** of FIG. **16**) for that NIM in the NIM Templates database (**74** of FIG. **16**). The collection procedure (**404** of FIG. **15**) receives the NIM_ID (**416** of FIG. **16**) from the user, locates the NIM definition module (**418** of FIG. **16**) corresponding to that NIM_ID in the NIM templates database, and transmits the NIM definition module to the user's computer. That NIM may automatically be opened on the user's computer screen. The NIM is saved to the user's list of NIMs on their home NIM (**466** of FIG. **21**), and the NIM definition module is saved in the user's local processed user profile. Alternatively a "preview" button may be provided which transiently displays the NIM on the user's computer screen without adding the NIM to the user's local processed user profile, as discussed above in relation to the NIMLink. All the NIM definitions that the user has listed on their home NIM are saved to the user profile database either periodically, at a set time, by event, or when the user closes their home NIM. The technique of the invention facilitates a viral distribution architecture. In other words, the technique of the invention facilitates rampant distribution of generated NIMs, as described below.

Users (or developers) may share NIMs they have collected, and perhaps even modified, with other users (or developers) in accordance with this viral distribution architecture. Because the NIM definition contains basic reference information, such as data to instantiate the NIM and URLs and other references to where the NIM content is located, a NIM is easily and quickly distributed, collected, and shared. By-packaging Internet content and applications as NIMs and referencing the NIMs by NIMLinks, the system advantageously gives Internet content viral characteristics as the NIMs can easily be distributed or shared between users.

Each NIM definition contains just enough information to define and initialize the NIM's components (NIM frame, controls, etc.). For example, this information may contain data to configure the skeleton or frame that is filled in by NIM content from a developer's server. The NIM definition is therefore fairly small in size (~2K), and is therefore easily distributable as an XML file or Blob (binary large object), which is communicated using the same mechanisms (HTTP/HTTPS requests) as regular Web pages.

This is especially useful where a user has collected a NIM or a group of NIMs (Packs) that he would like to send to another user. For example, a user may have an online trading NIM, calculator NIM, and stock research MM all set up in various positions on his screen, and would like to share the entire Pack with a friend who is remotely connected to the Internet.

To share NIMs with others, the system utilizes the Sharelink database **78** of **25** FIG. **15** and the Sharing procedure **405** of FIG. **15**. FIG. **24** is a diagrammatic illustration of the Sharelink database **78**. NIM Sharelink database **78** stores a list of all NIMs shared by users, developers, or administrators, in a share table **484**. Each NIM or group of NIMs shared is assigned a Share ID **486** which points to a Share module **488**. Each Share Module **488** may also include a creation date **490**, multiple **30** individual MM definition modules **492**, or multiple packs of NIMs that have been shared (Sharepack module **494**) containing multiple NIM definition modules **496** and **498**.

FIG. **25** is an illustration of a Share NIM's GUI **500**. All dots collected by the user (**466** of FIG. **21**) can be shared with other users by clicking on the "Share" button **502** shown in FIG. **21**. Once the user has clicked on the "Share" button **502**, the GUI **500** shown in FIG. **25** is launched. The user may then highlight any of the NIMs or packs of NIMs **504** he has collected or created and thereafter share the NIMs or packs of NIMs **504** by clicking on the "Share via email" button **506**. It should be noted that other means of distributing the NIMs may be used together with, or instead of, email.

When users share NIMs or NIM packs, their home NIM application generates a 10 share module, which may for example be an XML Blob containing the NIM definition or Sharepack modules shared. The shared NIM XML is then sent to, and saved in, the Sharelink database (**78** of FIGS. **1** and **24**). The Sharing procedure **405** of FIG. **15** then automatically generates a shared link (ShareLink) that references or points to the address of the shared XML on the Sharelink database. This ShareLink is then sent or 15 distributed (via email or posted on a Web site) to other users.

If a user receives shared NIM(s) or pack(s) and has a home NIM installed on his client computer, then clicking on the Sharelink adds the NIM(s) to the user's home NIM and opens the shared NIM(s) on the user's screen. If a recipient of a shared NIM does not have the home NIM installed on his computer, then the home NIM is downloaded and installed (with the user's cooperation), the shared MM is added to his local processed user profile, and the NIM is opened.

The NIM management module (**112** of FIG. **15**) may also be responsible for controlling and managing the development of new NIMs via the DevZone and the DevNIM discussed below.

Because NIM content is based on existing Internet content standards (HTML, DHTML, GIFs, etc.) developers can create MM content using their existing Internet content development tools and methodologies. Therefore, no special hardware or software is required to develop or serve NIM content.

Furthermore, as the application server (**50** of FIG. **1**) hosts and delivers NIM definitions from the NIM Templates database (**74** of FIG. **1**) developers merely define and package the NIM content without directly authoring, hosting, or serving the XML NIM definitions. Therefore, no special hardware or software is required on the developer-side to host and serve the NIM content, other than required for their regular Internet content.

Two means are provided for creating NIMs. First, a Developer Zone Web site (DevZone) and second, a set of developing NIMs (DevNIMs). Both means enable NIM developers to create, define, and modify NIM definitions, and to support the NIM development process which results in XML NIM definitions being added to the NIM Templates database and NIMLinks generated.

The DevZone is a Web site where NIM developers can view a list of NIMs they have defined and/or published, add new NIMs, and categorize, view, modify, or delete their existing NIMs. The DevZone is preferably rendered in a Web browser, is hosted on the Web server (**82** of FIG. **1**), and is implemented with a DevZone procedure (**406** of FIG. **15**). To

access the DevZone, the developer may typically pass through a secure portal, such as by supplying a login identity and password.

FIG. **26** is an illustration of the main DevZone Web page **510**. All NIMs created by the NIM developer appear in a customized NIM list **514** that may only be accessed by that NIM developer. All NIMs created by a developer appear on the NIM list **514**, unless they have been deleted by the developer or by a system administrator. The NIM list may contain the NIM name **518**, the date the NIM was created **520**, and an indication **516** of whether the NIM is in development or accessible by the public in the NIMIndex (i.e. "in-development" or "published").

To access the NIM definition (for modification or review) the developer clicks on a "modify" or "preview" link **524** as transient (e.g. to add the NIM to their home NIM for previewing and testing). By clicking on the "modify" link, the developer is taken to the NIM modification web page, as shown in FIG. **27**. Alternatively, by clicking on the "Create a Dot" button **522**, the developer is taken to a web page similar to the NIM modification web page shown in FIG. **27**, where the developer may create a new NIM.

FIG. **27** is a partial view of a NIM modification web page **530**. To modify an existing NIM, or create a new NIM definition, a developer preferably utilizes Web forms, such as **534** to **542**, or any area that contains objects that capture user input, such as text entry spaces, check boxes, and selection buttons. Developers typically fill in forms with information which defines the NIM, where the details might include the NIM's name **534**, the URL for any image associated with NIM **540** (as shown in FIG. **18**), the URL for a detailed image **542** (**448** of FIG. **19**), and such details as NIM frame (e.g., size of NIM, sizeable), layout of the controls (e.g., WebConduit control), and to specify the initial MM content (e.g., the initial target URLs for the WebConduit control, TitleBar, BottomBar), and any categories in which the developer would like the NIM to be listed in the NIMIndex. Once the developer has completed or modified the forms, he may either save or delete the NIM **532**. If the developer selects either the development check box **536** or the public check box **538**, and then saves the NIM, the DevZone procedure (**407** of FIG. **15**) generates a XML NIM definition, stores the XML NIM definition in the NIM Templates database (**74** of FIGS. **1** and **15**) and returns a NIMLink pointing to that NIM which is listed on the NIM list (**514** of FIG. **26**) on the developer's home NIM. The only difference being that once the developer selects the public check box **538** and saves the NIM, the NIM definition is published utilizing the PubZone publishing procedure (**410** of FIG. **15**) to a publically accessible portion of the NIM Template Table (**415** of FIG. **16**), from where users can access, download, and collect the NIM. If the developer selects the development check box **536**, the NIM can only be viewed and or modified by the developer and system administrator. It should be noted that the DevZone only allows control of certain characteristics of each NIM. Other characteristics may be set to default while still other characteristics can only be altered by an administrator. In an alternative embodiment, the DevZone may be rendered in a NIM or group of NIMs just as it was rendered in a Web browser. In either embodiment, NIM developers fill out one or more forms specifying NIM definition parameters, an XML NIM definition gets created and stored in the NIM Templates. Database, and a NIMLink gets generated that points to the new NIM. The Developer can then view or debug this NIM by clicking on the NIMLink to add it to his home NIM, or preview as transient, and thereafter render it on his screen. NIM definitions may also be developed using NIMs and NMA messages. A **30** developer may create Raw NIMs from empty NIM Templates using a development NIM (the DevNIM) on the developer's home NIM.

FIG. **28**A to **28**D are GUIs of a development NIM (the Dev NIM). A developer may obtain a DevNIM by either collecting the DevNIM in the usual manner, as discussed above, or the system, via the system administrator, may share the NIM with the developer, also as discussed above. The DevNim contains a DevNIM procedure (**408** of FIG. **15**) which is transmitted to the developer's home NIM, as discussed above.

To create a new NIM, the developer launches the DevNIM and enters a NIM name **550** into the DevNIM. The server then obtains an empty NIM (a NIM with default or no initialization data and with only basic characteristics) from the NIM Template Database using the procedure for collecting NIMs described above, and saves the empty NIM under the supplied new NIM name **550** locally in the developer's processed user profile. The developer may then modify the empty NIM to the required form using the DevNIM. In the preferred embodiment a pull down menu **552** is provided where the developer can select which feature to modify, such as the frame characteristics (FIG. **28**B), the titlebar (FIG. **28**C), or initialization URLs for different frame or control elements (FIG. **28**D).

Each time the developer modifies a setting, the DevNIM, using the DevNIM procedure, sends NMA messages to the newly saved NIM to modify its definition parameters. For example, modifying the NIMs name, size, TitleBars, BottomBars, or WebConduits (as shown in FIGS. **28**A to **28**C).

Unlike the DevZone, a new XML NIM definition and NIMLink is not generated every time a modification is made. All modifications (during the development cycle) are made locally to the NIM definition and are stored in the developer's processed user profile. The DevNIM embodiment, therefore, requires a separate publishing step that promotes the newly created NIM definition from the developer's user profile, to the NIM templates database on the application server.

To publish the NIM, the NIM developer categorizes the NIM and the NIM definition is copied from the developer's processed user profile to the NIM Templates database.

A publishing NIM (PubNIM), implemented with Publishing procedures **409**, is provided to handle these functions. The PubNIM may therefore be shared or transmitted to the developer along with the DevNIM. The PubNIM contains a PubNIM procedure (**411** of FIG. **15**) which controls the publication of the NIMs to the NIM template database, as discussed above. The PubNIM procedure sends a NIM definition module to the application server which receives the NIM definition module, extracts the NIM definition from the share module, stores it in the NIM Templates database, and associates the NIM with the developer so that the NIMLink shows up on the developer's NIM list (in their DevZone account).

Alternatively, the new NIM may be published directly from the DevNIM. Once a user is satisfied with the NIM, he may select an option which publishes that NIM definition to the NIM Templates database. It should be noted that a developer may modify his NIMs at any time from the DevNIM.

As mentioned earlier, when a developer is first authorized to create and/or modify NIMs, or at any time thereafter, information about that developer is saved in that developer's user profile (**76** of FIG. **1**) on the application server.

A developer may also create application programs using NIMs, which a user may access from his client computer. Just as client-side application characteristics (sizing, position, menus) are accessible to content via NMA, the system may offer server-side application functionality, or toolkits, which are accessible through the NMA.

A developer can build a NIM application without implementing, hosting, or supporting complex server or client

applications. By using the server toolkits, a developer can develop NIMs that exhibit server-application behavior by focusing on implementing NIM content (just like standard Internet content).

For example, a NIM's content (an HTML page) may send a message to the system (or server) to request a credit card to be processed. Other toolkit examples may include credit card billing, user profiling, targeted advertising, email, chat rooms, Internet telephony applications, or calendars.

Any server-side application can be made accessible through the NMA, as a toolkit, just as client-side application behaviors are made accessible. In the current implementation, server-side application functions could be offered by a NIM (exposed via javascript functions on a page in a hidden frame). Other NIMs could access this functionality by sending NMA messages to this "Toolkit NIM" calling the functions. A NIM developer may therefore focus on Internet content development while accessing the features, behaviors, and functionality of an application just as if he had developed custom client and server side applications. The NIM management module (**112** of FIG. **15**) may also be responsible for controlling and managing the administration of the system via the AdminZone and the AdminNIM discussed below.

A system administrator has the power to create, modify or delete users, developers, NIMs, other administrators, or NIMIndex categories, depending on that administrators access privileges. In a similar manner to the DevZone and DevNIM, system administrators may utilize either a Web browser administration zone (AdminZone), or an administration NIM (AdminNIM) which both make use of Administration procedures (**412** of FIG. **15**).

To access the AdminZone, an administrator typically passes through a secure portal, such as by supplying a login identity and password. Once within the AdminZone, the administrator may search for a NIM by NIM name or title **552**, category, developer, developer contact name, or status, as shown in FIG. **29**A. The administrator may also selectively search for NIM's in development or publically accessible NIMs **554**.

Utilizing an AdminZone procedure (**413** of FIG. **15**), once the required NIM **20** is located the administrator may modify or delete the NIM in a similar manner to a developer as shown in FIG. **29**B, and described above.

Also utilizing the AdminZone procedure, the administrator may manage NIMIndex categories by creating new categories, modifying or deleting existing categories, and/or adjusting the layout of the NIMs within those categories as shown in FIGS. **30**A and **30**B. For example, an administrator may change a category's name **558**, designate the category active or inactive **556**, or create sub-categories **560**. The system administrator may also select a category or categories for the NIM to appear in, where each NIM may be registered in more than one category.

Finally, utilizing the AdminZone procedure (**413** of FIG. **15**), an administrator may search for users, providers, or developers and adjust their details, as shown in FIGS. **31**A and **31**B. The system administrator may, for example, change a users contact details. In addition to adding, modifying or deleting NIMs, system administrator may have the task of reviewing NIM submissions from developers and promoting NIMs to the public. A submissions list of newly submitted NIMs may be displayed to an administrator, who may promote the NIM to the public or view the NIM. Once promoted, changes are made to the NIM Templates database and the NIM is automatically removed from the submissions list (again by utilizing the AdminZone procedure (**413** of FIG. **15**)).

The foregoing discussion has explored the inherent nature of NIMs. Attention now turns to different techniques that may be used to exploit information that is associated with the use of NIMs. In particular, the following discussion is directed toward the accumulation of statistical information that is only available in view of the architecture of the present invention.

Currently, the predominant method of tracking and collecting user online behavior is severely limited for a number of reasons. First, most Internet use or visitor statistics are single-dimensional (linear, sequential) because Internet content is presented to users one full-screen page at a time. Second, users visit and leave sites so rapidly their visits are barely meaningful. Third, user's browsing habits are often discontinuous (browsers give users navigational bypass controls—back, forward, home, refresh, stop, etc.). Fourth, user behavior tracking is limited from a single site's server point of view. Current use statistics are plagued with the challenge of tracking continuous user behavior (especially from a cross-company perspective), with more than a single dimension of use context. Finally, because a computer may have multiple users, or a single user may use multiple computers, tracking continuous user on-line behavior is extremely difficult.

One of the advantages of the NIM system as illustrated in FIG. **1**, is that the Server **50** is able to track continuous, long-term NIM use information about each user. This is because the NIM server, through communication with the home NIM, can track each NIM event performed by each user. Therefore, it is possible to track each individual user's entire NIM use activity from the moment the user downloads the home NIM.

Referring to FIG. **32**, in one embodiment of the invention the following events may be tracked by the Event Log Module **98** (within the client computer **20** of FIG. **1**): home NIM Download Event **704** NIM Download Event **706** NIM Display Event **708** Web Click-Through Event **710** Page-View Event **712** First NIM Installation Event **714** First home NIM Startup Event **716** Transient Mode Event **718** Share NIM Received Event **720** NIM Pack Received Event **722**.

A home NIM Download Event **704** is logged when the user clicks on a link to request the home NIM user application. Preferably, the start time **752**, and the end time **754** are recorded for this event. Also recorded is the provider ID **750** which is a parameter (generally, an integer) that represents the content provider partner who provided the link to the user.

NIM Download Event **706** is logged when the home NIM acquires a NIM via a NIMLink. The start time **752**, the end time **754**, and the provider ID **750** are recorded for this event. Also recorded is the NIM ID **756** which is a parameter (generally, an integer) that represents the NIM that was just downloaded.

NIM Display Event **708** is logged when a user activates a NIM. The NIM ID **756**, the start time **752**, and the end time **754** are recorded for this event. Web Click-Through Event **710** is logged whenever a user links from a NIM to a full-screen browser. This can occur when a user clicks on a link in the NIM, or it can occur automatically through the NIM messaging, or directly through the content provider. The NEM ID **756**, the start time **752**, and the Internet address **758** of the link are recorded. Page-View Event **712** is logged whenever a user views a page of content within a NIM. The NIM ID **756**, and the start time **752** are recorded for this event.

First NIM Installation Event **714** is logged the first, a NIM or NIM Pack is installed from a web site. This event is logged only once for each user account. The NIM ID **756**, start time **752**, and end time **754** are recorded for this event.

First home NIM Startup Event **716** is logged when the home NIM runs for the first time. This event is logged only once for each user account. The start time **752** is recorded for this event.

Transient Mode Event **718** is logged when the home NIM runs in transient mode. Transient mode occurs when the home NIM runs before the user has logged in. The start time **752**, and the end time **754** are recorded for this event.

Shared NIM Received Event **720** is logged for each NIM a user receives as part of a share. If a NIM Pack is shared, this event will be recorded for each NIM in the shared pack. The NIM ID **756**, the start time **752**, and the end time **754** are recorded for this event.

Shared NIM Pack Received Event **722** is logged for each NIM Pack a user receives as part of a share. Thus, when a NIM Pack is shared, an Event **720** will be logged for each NIM in the NIM Pack, while an Event **722** will be logged once for the NIM Pack itself. The start time **752**, and the end time **754** are recorded for this event.

The events listed above are tracked in one particular embodiment. Other embodiments may track more or perhaps fewer events. This comprehensive event tracking is possible because each user event can be identified by the NIM Server through communication with the home NIM. Additional events may include tracking when a user sends a share or tracking when a user sends a NIM or a NIM Pack.

FIG. **33** shows a typical series of user actions **800** as they are tracked by the Event Log Module **98**. First, a user may request to download the home NIM application (step **802**) from either a partner's web site or the NIM Server **50**. The Event Log Module **98** records a home NIM Download Event, as shown with field **704** of FIG. **32**. The start time **752**, and the end time **754** are preferably recorded. Also, the provider ID **750** of the site from where the home NIM download request was received is recorded.

Returning to FIG. **33**, the user subsequently activates the home NIM for the first time (step **804**). The Event Log Module **98** records a First home NIM Startup Event **716**, as shown in FIG. **32**. The start time **752** is preferably recorded. In addition, the home NIM is activated and the user lot yet logged in, a Transient Mode Event **718** is logged and the start time **752** is recorded.

As shown in step **806** of FIG. **33**, the user logs into the home NIM. When this occurs, the end time **754** may be recorded for the Transient Mode Event **718**.

A user download of a new NIM (step **808**) may be from a partner's web site or the NIM Server. When this occurs, the raw NIM definition is copied into the user's User Profile **76**. The event log **98** records two events. First, because this is the first NIM the user has installed, a First NIM Installation Event **714** is recorded. The start time **752**, the end time **754**, and the provider ID **750** of the download site are preferably recorded. The second event recorded is a NIM Download Event **706**. The Event Log Module **98** preferably tracks the NIM ID **756**, the provider ID **750**, the start time **752**, and the end time **754** for this event. The next thing a user may do is open the NIM (step **810**). This consists of retrieving the NIM definition from the user's User Profile and getting NIM content from the provider **82**, as discussed above. The NIM is displayed for the user and the Event Log Module **98** records a NIM Display Event **708**. However, at this point, the Event Log Module **98** can only record the start time **752**, and the NIM ID **756** for this event. The end time **754** is recorded when the NIM is closed.

For every page of content a user views within a NIM **812**, a Page-View Event **712** is recorded. Some page views may require content from the provider **82**. The NIM ID **756**, and the start time **752** are recorded for this event.

The NIM may also enable the user to click on a link that results in navigating to a full screen web browser (step **814**). When a user does this, a Web Click-Through Event **710** is recorded. The Event Log Module **98** records the NIM ID **756**, the start time **752**, and the URL of the web site that is passed from the NIM content to the browser **758**.

When the NIM closes (step **816**), the end time **754** for the NIM Display Event **708** is recorded. When the user logs out of the home NIM (step **818**), the event log is uploaded to the Server **50** (of FIG. **1**).

In one embodiment of the invention, the previously described Event Log Module **98** (within the client computer **20** of FIG. **1**) tracks user events in the home NIM user application and uploads the information to the Statistics Database **80** (of the server computer **50** of FIG. **1**) at predetermined intervals alternate embodiments, the Event Log **700** (in FIG. **32**) may be processed by the NIM Server before it is stored in the Statistics Database **80**. For example, the NIM Server may process NIM use status information for each user that is currently logged in.

The Statistics Database **80**, illustrated in FIG. **34**, preferably lists every event **1002** by every user of home NIMs along with the corresponding fields associated with each event. For-example, if a NIM Display Event is recorded, the User ID **1004** of the user that performed the event is listed, the start time **1006** is listed, the end time **1008** is listed, and the NIM ID **1010** is listed. If a Web Click-Through Event is recorded, the User ID **1004** is listed, the NIM ID **1010** is listed, the start time **1006** is listed, and the URL of the web site **1014** is listed. The Statistics Database **80** therefore allows the list of events to be easily referenced and searched by each event or by each of the fields associated with the events.

Referring to FIG. **35**, the Statistical Analysis Module **900** uses the Statistics Database **80** in order to provide various services for the content provider partners **82**. Preferably, the Statistical Analysis Module **900** includes a Multi-Dimensional Consumer Profile Module **902**, a Real-Time Advertising Module **904**, and a Pack Building Module **906**, as discussed below.

A primary advantage of the present invention is that, because NIMs are used in groups and are used more often and for longer periods of time than web pages or web sites, real-time multi-dimensional NIM use data (that's a function of which NIMs are activated simultaneously) can be accumulated. In accordance with an embodiment of the invention, this accumulated data is used to generate a multi-dimensional consumer profiling database. The Multi-Dimensional Consumer Profile Module **902** uses information from the Statistics Database **80** to examine, for each user, the start time, and the end time of each NIM Display Event It then determines the NIMs (using the NIM IDs) that are opened simultaneously for each user. The Module **902** determines, for every selected NIM, the other NIMs that a given user may use in conjunction with the selected NIM. The Module **902** also determines how often these other NIMs are used simultaneously with the selected NIM. For example, Company X provides a NIM for selling its books. The Multi-Dimensional Consumer Profile Module **902** determines for Company X that a particular user has a NIM related to finance activated 30% of the time the user has the book-selling NIM acted, a NIM related to computers 20% of the time the user has the book-selling NIM activated, and a NIM related to wedding gifts 5% of the time the user has the book-selling NIM activated. This will provide Company X with a more complete profile of the user's interests.

The Real-Time Advertising Module **904** determines the NIMs that each user has displayed at any given moment. This

information is used by a content provider partner or by the NIM Server to target advertising information. For example, if a user has a NIM related to sports displayed simultaneously with Company X's book-selling NIM, Company X uses this information to stream an advertisement for a sports book. In one embodiment, this is accomplished by associating each NIM with a context keyword. This is done by incorporating the context keyword into the NIM definition or, alternatively, by maintaining a table of NIMs and their corresponding context keywords. For example, the NIM related to sports is associated with the context keyword "sports." Moreover, the Real-Time Advertising Module **904** may combine the real-time user information with the historical user information from the Statistics Database **80** to provide advertisers with a complete picture of a user's interests.

The Pack Building Module **906** uses the Statistics Database **80** to determine which NIMs are being used simultaneously. The Module **906** also determines which NIMs are being shared as NIM Packs. From this, the Module **906** provides information to content provider partners about which NIMs should be bundled together. In alternate embodiments, the Module **906** builds a NIM Pack based upon the information it processes. For example, if the Pack Building Module **906** determines that an airline NIM is being used with a hotel NIM and a car rental NIM, the Module **906** may build a NIM Pack with a restaurant NIM.

Additionally, in one embodiment of the present invention, the NIM Server **82** may track the content within a NIM in a Content Database **1050**, as illustrated in FIG. **36**. A content descriptor **1052** which may be a string describing the content that is shown within the NIM is recorded for content shown in the NIM. For example, if a NIM displayed an advertisement for an automobile followed by an advertisement for a restaurant, the two recorded content descriptors might say "automobile ad" and "restaurant ad." In addition, the NIM ID **1054**, the start time at which the content is displayed **1056**, and the end time **1058** are all preferably recorded for each content descriptor.

Referring to FIG. **35**, The Content Analysis Module **950** is able to correlate, at any moment, the content displayed to the user as recorded in the Content Database with the user's NIM activity recorded in the Statistics Database. For example, if one NIM displays to a user an advertisement for a travel book, the user may open a NIM related to Florida, a NIM owned by a specific airline, and a NIM owned by a car rental company. This pattern of user behavior will allow the company that provides the travel book advertisement to better understand the effect of the advertisement on the user. The company may use this information to make cross-promotions with other NIM providers, or, simply to provide more effective targeted advertisements.

In an alternative embodiment, each of the content providers may track its own content information. The content providers could then compare its content information with the user information provided by the Statistics Database of the NIM-Server.

Finally, referring to FIG. **37**, all of the user event information may be used in conjunction with user information provided at login. During the login process, the user may be required to enter demographic information such as age, marital status, etc. In one embodiment, this information is stored in a User Account Database **1100**. Each User ID **1102** is listed along with the corresponding user information **1104**. Therefore, it is possible to match the user events with personal information about the particular user to give advertisers or NIM content providers a more complete behavior profile of each user.

The foregoing description, for purposes of explanation, used specific nomenclature to provide a thorough understanding of the invention. However, it will be apparent to one skilled in the art that the specific details are not required in order to practice the invention. In other instances, well known circuits and devices are shown in block diagram form in order to avoid unnecessary distraction from the underlying invention. Thus, the foregoing descriptions of specific embodiments of the present invention are presented for purposes of illustration and description. They are not intended to be exhaustive or to limit the invention to the precise forms disclosed, obviously many modifications and variations are possible in view of the above teachings. The embodiments were chosen and described in order to best explain the principles of the invention and its practical applications, to thereby enable others skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the following claims and their equivalents.

What is claimed is:

1. A computer-implemented method of obtaining content over a network and displaying the content to a user, the method being implemented in a client computing device in operative communication with a server over a network, the client computing device including electronic storage, a display, and one or more processors configured to execute one or more computer program modules, the method comprising:

transmitting a request to the server over the network, the request requesting networked information monitor template;

receiving the requested networked information monitor template from the server over the internet, the requested networked information monitor template having been transmitted from the server over the network responsive to the transmitted request, the networked information monitor template comprising:

a definition of a viewer graphical user interface within which content in a web browser-readable language may be presented on the display of the client computing device; and

a definition of a first content element for the networked information monitor template, the definition of the first content element referencing a first network location from which the first content element for the networked information monitor template is served over the network;

responsive to instructions included in the requested networked information monitor template, presenting the viewer graphical user interface defined by the networked information monitor on the display of the client computing device separate from and outside of any other graphical user interface that includes user controls for specifying the first network location from which the first content element for the networked information monitor is served over the network;

responsive to instructions included in the requested networked information monitor template, transmitting over the network a first content request to the first network location referenced by the definition of the first content element for the networked information monitor template;

receiving, over the network, the first content element transmitted responsive to the first content request;

presenting the received the first content element in the viewer graphical user interface defined by the networked information monitor template, wherein the definition of

the viewer graphical user interface and/or the first content element define all controls for enabling a user to interact with the first content element through the viewer graphical user interface.

2. The method of claim **1**, wherein the definition of the viewer graphical user interface includes a definition of at least one control effective for controlling an aspect of the viewer graphical user interface, and wherein presenting the viewer graphical user interface comprises presenting the at least one control.

3. The method of claim **2**, wherein the aspect is the location at which the user interface is rendered on a display device.

4. The method of claim **2**, wherein the aspect is the size of the user interface when presented on the display of the client computing device.

5. The method of claim **1**, wherein the networked information monitor template includes instructions in extensible markup language (XML) format, and further wherein the method further comprises parsing the definition of the viewer graphical user interface.

6. The method of claim **1**, wherein the definition of the viewer graphical user interface defined by the networked information monitor template comprises an image, and wherein presenting the viewer graphical user interface defined by the networked information monitor template on the display of the client computing device comprises presenting the image included in the definition as part of the viewer graphical user interface.

7. The method claim **6**, wherein presenting the received the first content element in the viewer graphical user interface defined by the networked information monitor template comprises presenting the received the first content element in association with the image.

8. The method of claim **6**, wherein presenting the received the first content element in the viewer graphical user interface defined by the networked information monitor template comprises presenting the received the first content element at least in part within the image.

9. The method of claim **1**, wherein the definition of the viewer graphical user interface defines at least one control permitting the modification of the appearance of the viewer graphical user interface defined by the networked information monitor template, and wherein presenting the viewer graphical user interface defined by the networked information monitor template on the display of the client computing device comprises presenting the at least one control.

10. The method of claim **1**, wherein the first content element includes audio content, video content, graphics content, still image content, textual content and/or interactive content.

11. The method of claim **1**, further comprising:

executing a client parser application that is separate from the networked information monitor template, the client parser application being configured to present the viewer graphical user interface defined by the networked information monitor template on the client computing device, to generate the request for the content that is transmitted to the first network location referenced by the definition of the first content element for the networked information monitor template, and to present the first content element received from the first network location within the viewer graphical user interface defined by the networked information monitor template.

12. The method of claim **1**, further comprising causing the networked information monitor to communicate with another networked information monitor.

13. The method of claim **1**, wherein the networked information monitor is configured to cause the client computing device to periodically request the first content element from the first network location referenced by the definition the first content element for the networked information monitor template and to present the received first content element within the viewer graphical user interface in an ongoing manner.

14. The method of claim **1**, wherein the network is the internet.

15. The method of claim **1**, wherein the networked information template further comprising a definition of a second content element for the networked information monitor template, the definition of the second content element referencing a second network location from which the second content element for the networked information monitor template is served over the network, and wherein the method further comprises:

responsive to instructions included in the requested networked information monitor template, transmitting over the network a second content request to the second network location referenced by the definition of the second content element for the networked information monitor template;

receiving, over the network, the second content element transmitted responsive to the second content request; and

presenting the received second content element in the viewer graphical user interface defined by the networked information monitor template, wherein the definition of the viewer graphical user interface and/or the second content element define all controls for enabling a user to interact with the second content element through the viewer graphical user interface.

16. A computer-implemented method of obtaining and displaying content obtained over a network to a user on a client computing device, the method being implemented in the client computing device, the client computing device including electronic storage, a display, and one or more processors configured to execute one or more computer program modules, the method comprising:

executing on the client computing device client parser, wherein the execution of the client parser results in the client computing device:

transmitting over the network a first content request to a first network location referenced by a definition of a first content element in a networked information monitor template that is readable by the client parser;

receiving, over the network, the first content element transmitted responsive to the first content request;

presenting a viewer graphical user interface defined by the networked information monitor template on the display of the client computing device and wherein the viewer graphical user interface defined by the networked information monitor template is presented separate from and outside of any other graphical user interface that includes user controls for specifying the first network location from which the first content element for the networked information monitor is served over the network; and

presenting the received content element within the viewer graphical user interface defined by the networked information monitor template, wherein the definition of the viewer graphical user interface and/or the first content element define all controls for enabling a user to interact with the first content element through the viewer graphical user interface.

17. The method of claim **16**, wherein execution of the client parser further results in the client computing device determining that the first element content modifies the networked

information monitor template and responsive to such determination, modifying the networked information monitor template prior to the step of presenting the graphical user interface.

18. The method of claim **16**, further comprising presenting an icon representing networked information monitor associated with the networked information monitor template on the display of the client computing device, and wherein execution of the client parser to perform one or more of the transmitting operation, the presenting the viewer graphical user interface operation, or presenting the first content element operation is responsive to reception of selection by a user of the icon.

19. The method of claim **16**, further comprising invoking the client parser an initiating one or more of the transmitting operation, the presenting the viewer graphical user interface operation, or presenting the first content element operation automatically during a start-up process of the client computing device.

20. The method of claim **19**, wherein execution on the client computing device of the client parser further results in the client computing device:

transmitting over the network a second content request to a second network location referenced by a definition of a second content element in a second networked information monitor template that is readable by the client parser;

receiving the second content element transmitted responsive to the second content request;

presenting a second viewer graphical user interface defined by the second networked information monitor template on the display of the client computing device, wherein the second viewer graphical user interface lacks user controls, wherein the definition of the second viewer graphical user interface and/or the second content element define all controls for enabling a user to interact with the second content element through the second viewer graphical user interface; and

presenting the second content element within the viewer second graphical user interface,

wherein execution of the client parser to perform the transmitting the second content request, the presenting the second viewer graphical user interface operation, and the presenting the second content element is responsive to reception of a selection to initiate a second networked information monitor associated with the second networked information monitor template.

21. The method of claim **16**, wherein the network is the internet.

* * * * *

US008020083B1

(12) **United States Patent**
Kembel et al.

(10) **Patent No.: US 8,020,083 B1**
(45) **Date of Patent: *Sep. 13, 2011**

(54) **SYSTEM AND METHODS FOR CREATING AND AUTHORING INTERNET CONTENT USING APPLICATION MEDIA PACKAGES**

(75) Inventors: **John Albert Kembel**, Palo Alto, CA (US); **George Andrew Kembel**, Menlo Park, CA (US); **Daniel Kim**, Palo Alto, CA (US); **John Russell**, Palo Alto, CA (US); **Jake Wobbrock**, Palo Alto, CA (US); **Geoffrey Kembel**, Menlo Park, CA (US); **Jeremy Kembel**, Palo Alto, CA (US); **Joseph Bella**, San Francisco, CA (US); **Sridhar Devulkar**, Mountain View, CA (US); **Mark Wallin**, Mountain View, CA (US)

(73) Assignee: **Mainstream Scientific, LLC**, Mountain View, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 938 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **11/932,585**

(22) Filed: **Oct. 31, 2007**

Related U.S. Application Data

(63) Continuation of application No. 09/558,922, filed on Apr. 26, 2000, now Pat. No. 7,756,967.

(51) **Int. Cl.**
G06F 17/00 (2006.01)
G06F 15/173 (2006.01)

(52) **U.S. Cl.** **715/201**; 715/741; 715/762; 709/224

(58) **Field of Classification Search** 709/203, 709/223, 224; 715/201, 741, 762
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,625,781	A	4/1997	Cline et al.
5,649,186	A	7/1997	Ferguson
5,740,549	A	4/1998	Reilly et al.
5,761,662	A	6/1998	Dasan

(Continued)

FOREIGN PATENT DOCUMENTS

WO WO0180086 A2 10/2001

OTHER PUBLICATIONS

Alexa 1.4.1 Support Pages, 9 pages, www.alexa.com/support/index 1.html, Jan. 1999.

(Continued)

Primary Examiner — Chau Nguyen
(74) *Attorney, Agent, or Firm* — Pillsbury Winthrop Shaw Pittman, LLP

(57) **ABSTRACT**

An Application Media Package is a software component for accessing and displaying Internet content which includes a definition for rendering a graphical user interface and a URL pointing to Internet content to be downloaded and presented within said user interface. An Application Media Viewer may be used in association with an Application Media Package to manage the collection, organization, sharing, and rendering of a plurality of such Packages. A development server supports the community of Application Media Package developers, providing developer tools, including Package templates which provide an expedient method of re-purposing existing internet media into a new presentation package by choosing from existing examples. Access to tools and information may be controlled at the development server. The development server may also provide a development and test zone for Package verification, authentication and acceptance before posting.

16 Claims, 14 Drawing Sheets

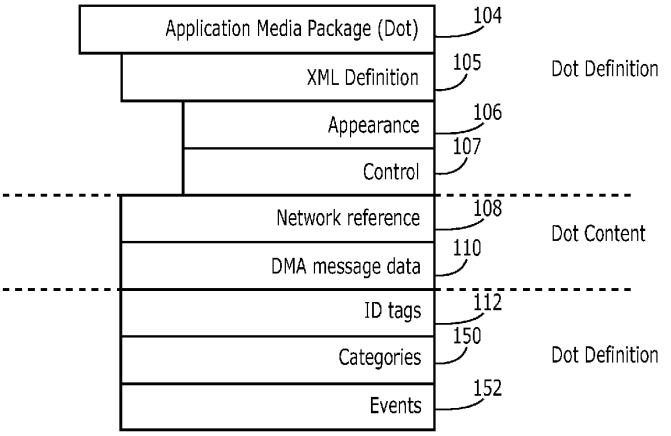


Exhibit No.
6

U.S. PATENT DOCUMENTS

5,794,230	A	8/1998	Horadan et al.	
5,796,393	A	8/1998	MacNaughton et al.	
5,796,952	A	8/1998	Davis et al.	
5,801,702	A	9/1998	Dolan et al.	
5,809,248	A	9/1998	Vidovic	
5,838,906	A	11/1998	Doyle et al.	
5,890,172	A	3/1999	Borman et al.	
5,893,091	A	4/1999	Hunt et al.	
5,948,061	A	9/1999	Merriman et al.	
5,974,446	A	10/1999	Sonnenreich et al.	
5,977,964	A	11/1999	Williams et al.	
5,983,227	A	11/1999	Nazem et al.	
5,987,513	A	11/1999	Prithviraj et al.	
5,995,756	A	11/1999	Hermann	
6,006,252	A	12/1999	Wolfe	
6,012,090	A	1/2000	Chung et al.	
6,012,098	A	1/2000	Bayeh et al.	
6,018,344	A	1/2000	Harada et al.	
6,026,433	A	2/2000	D'Arlach et al.	
6,065,044	A	5/2000	Ogasawara	
6,101,510	A *	8/2000	Stone et al.	715/234
6,115,040	A *	9/2000	Bladow et al.	715/741
6,133,916	A	10/2000	Bukszar et al.	
6,161,112	A	12/2000	Cragun et al.	
6,177,936	B1	1/2001	Cragun	
6,199,082	B1	3/2001	Ferrel et al.	
6,230,173	B1	5/2001	Ferrel et al.	
6,237,030	B1	5/2001	Adams et al.	
6,268,856	B1	7/2001	Bruck et al.	
6,275,854	B1	8/2001	Himmel et al.	
6,292,185	B1	9/2001	Ko et al.	
6,297,819	B1	10/2001	Furst	
6,314,451	B1	11/2001	Landsman et al.	
6,341,305	B2	1/2002	Wolfe	
6,369,840	B1 *	4/2002	Barnett et al.	715/853
6,393,407	B1	5/2002	Middleton et al.	
6,411,992	B1	6/2002	Srinivasan et al.	
6,418,440	B1	7/2002	Kuo et al.	
6,434,563	B1	8/2002	Pasquali et al.	
6,460,029	B1	10/2002	Fries et al.	
6,484,149	B1	11/2002	Jammes et al.	
6,487,566	B1	11/2002	Sundaresan	
6,496,203	B1 *	12/2002	Beaumont et al.	715/762
6,537,324	B1	3/2003	Tabata et al.	
6,538,673	B1	3/2003	Maslov	
6,549,612	B2	4/2003	Gifford et al.	
6,560,639	B1	5/2003	Dan et al.	
6,571,245	B2	5/2003	Huang et al.	
6,594,682	B2	7/2003	Peterson et al.	
6,662,341	B1 *	12/2003	Cooper et al.	715/234
6,751,606	B1	6/2004	Fries et al.	
6,784,900	B1	8/2004	Dobronsky et al.	
6,816,880	B1	11/2004	Strandberg et al.	
6,834,302	B1 *	12/2004	Harvell	709/224
6,842,779	B1	1/2005	Nishizawa	
7,107,548	B2	9/2006	Shafron	
7,216,300	B2	5/2007	Dang	
7,356,569	B1	4/2008	Kembel et al.	
2002/0089536	A1 *	7/2002	Dang	345/749
2002/0091697	A1 *	7/2002	Huang et al.	707/10

OTHER PUBLICATIONS

Alexa general faqs, 4 pages, www.alexa.com/whatisalexa/faq.html#general, Jan. 1999.

"Custom Explorer Bars Give Sites an Edge," 2 pages, www.microsoft.com/Windows/le/IE5/custom.asp, Jan. 1999.

"Flexibility Across the Web," 2 pages, www.microsoft.com/Windows/le/IE5/choice.asp, Jan. 1999.

"Web Accessories Overview," 2 pages, www.microsoft.com/workshop...er/accesory/overview/overiew.asp, Jan. 1999.

"Browser Extensions Overview," 2 pages, www.microsoft.com/workshop/browser/ext/overview/overview.asp, Jan. 1999.

Alexa Technology, 4 pages, www.alexa.com/support/technology.html, Jan. 1999.

"Creating Custom Explorer Bars and Desk Bands," 13 pages, www.microsoft.com/workshop/browser/ext/overview/Bands.asp, Jan. 1999.

Alexa Internet Tour, 1 page, www.alexa.com.whatisalexa/index/html, Jan. 1999.

"Revolutionary Ad Model," Advertise on Alexa, 1 page, www.alexa.com/company/advertise.html, Jan. 1999.

"The Alexa Service Appears on Your Desktop in Its Own Window," 1 page, www.alexa.com/tour/overview.html, Jan. 1999.

"Know More About the Sites You Visit," 1 page, www.alexa.com/tour/site_stats.html, Jan. 1999.

"Find Related Web Sites," 1 page, www.alexa.com/tour/related_links.html, Jan. 1999.

500,000 Sites and Growing,: 1 page, www.alexa.com/tour/archive.html, Jan. 1999.

"Research Tools at Your Fingertips," 1 page, www.alexa.com/tour/eb.html, Jan. 1999.

"Reporting," 1 page, www.alexa.com/company/reporting.html, Jan. 1999.

"Alexa Internet's Related Links Integrated Into Netscape Browser," 1 page, www.alexa.com/company/netscape.html, Jan. 1999.

"Demographics," alexa.com/company/demographics.html, Jan. 1999.

"Ads Appear in the Pop-up and on the Bar," 1 page, www.alexa.com/company/adspecs.html, Jan. 1999.

"Alexa Why Crawl," 1 page, www.alexa.com/support/why_crawl.html, Jan. 1999.

GIF Image 590x329 pixels, Alexa, 1 page, www.alexa.com/tour/images/alexa_overview.gif, Jan. 1999.

"It's X-treme!," Alexa, PC Magazine: The Best of 1998, 1 page, www.zdnet.com/pcmag/special/bestof98/internet5.html, Jan. 1999.

"Search While You Surf," PC Magazine: Search the Web, 1 page, www.zdnet.com/pcmag/features/websearch98/surf.html, Jan. 1999.

MindSpring, MyYahoo, pp. 1-16, www.mindspring.com/myyahoo/contents.htm, Dec. 1997.

Morrison, XML Unleashed, Sams Publishing, Dec. 21, 1999.

Flanagan, JavaScript; The Definitive Guide, 3rd Ed., O'Reilly, Jun. 1998.

* cited by examiner

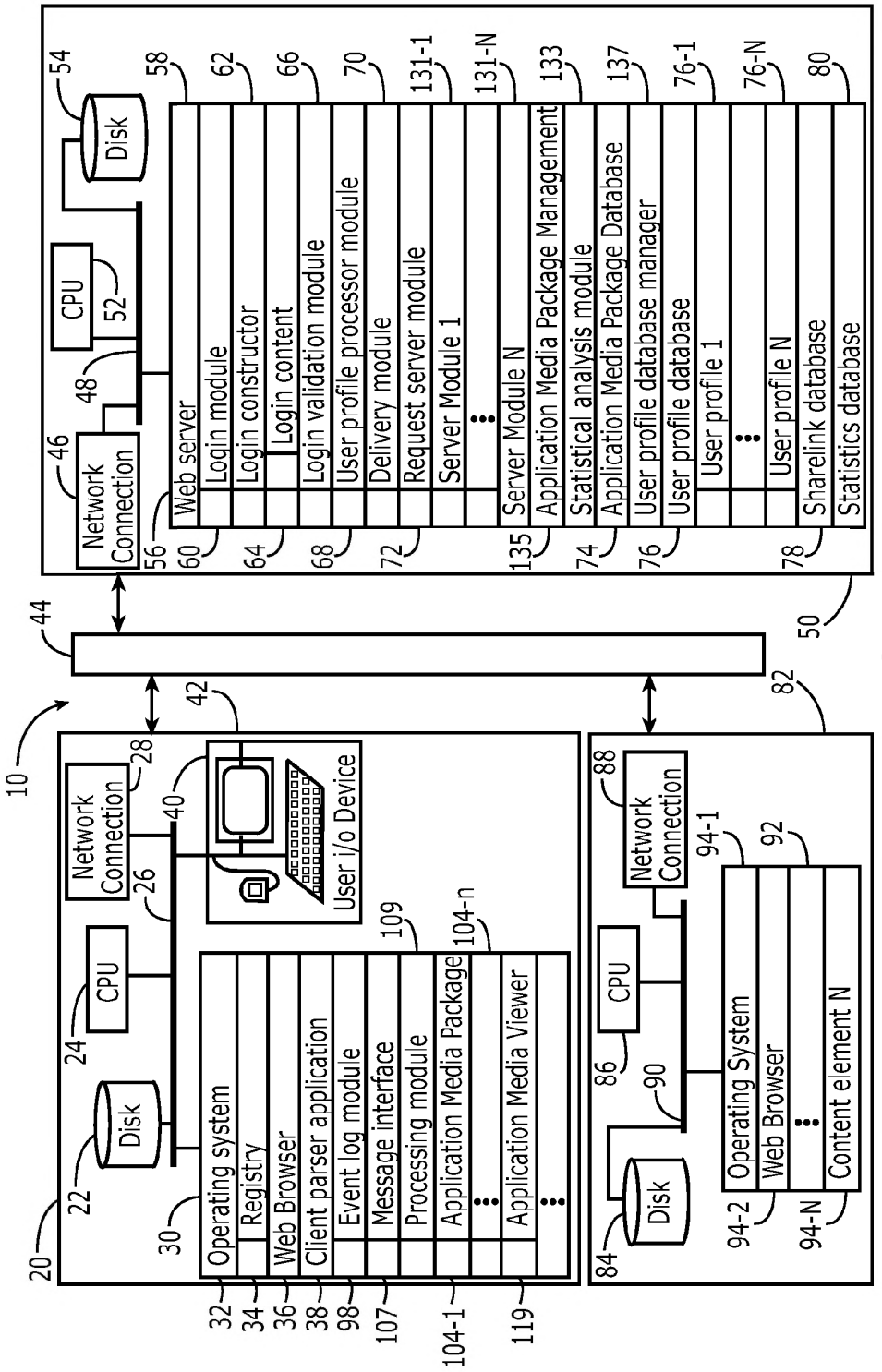

10
20
22
Disk
24
CPU
26
Network Connection
28
30
Operating system
Registry
Web Browser
Client parser application
Event log module
Message interface
Processing module
Application Media Package
Application Media Viewer
32
34
36
38
98
107
104-1
109
104-n
119
40
User i/o Device
42
44
46
Network Connection
48
CPU
52
Disk
54
56
Web server
Login module
Login constructor
Login content
Login validation module
User profile processor module
Delivery module
Request server module
Server Module 1
Server Module N
Application Media Package Management
Statistical analysis module
Application Media Package Database
User profile database manager
User profile database
User profile 1
User profile N
Sharelink database
Statistics database
58
60
62
64
66
68
70
72
131-1
131-N
135
133
74
137
76
76-1
76-N
78
80
50
82
84
Disk
86
CPU
88
Network Connection
90
92
94-1
94-2
94-N
Operating System
Web Browser
Content element N

FIG. 1

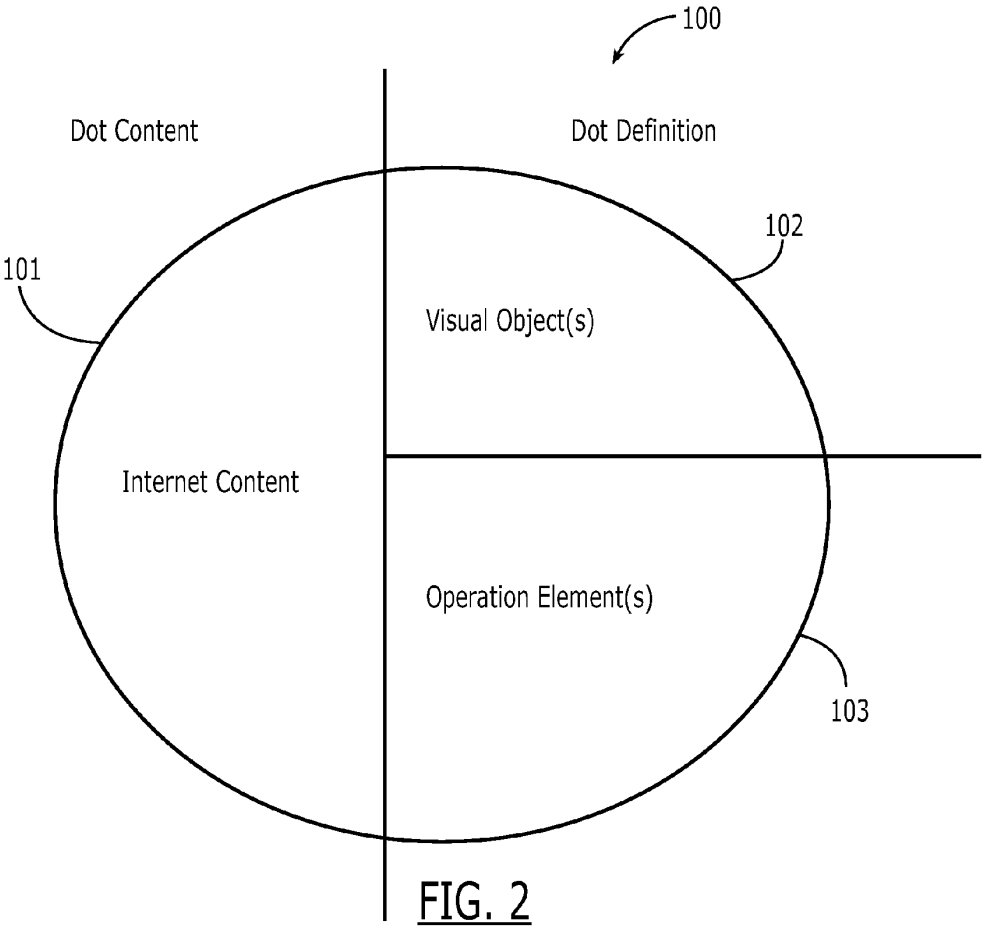

100
Dot Content
Dot Definition
101
102
Visual Object(s)
Internet Content
Operation Element(s)
103

FIG. 2

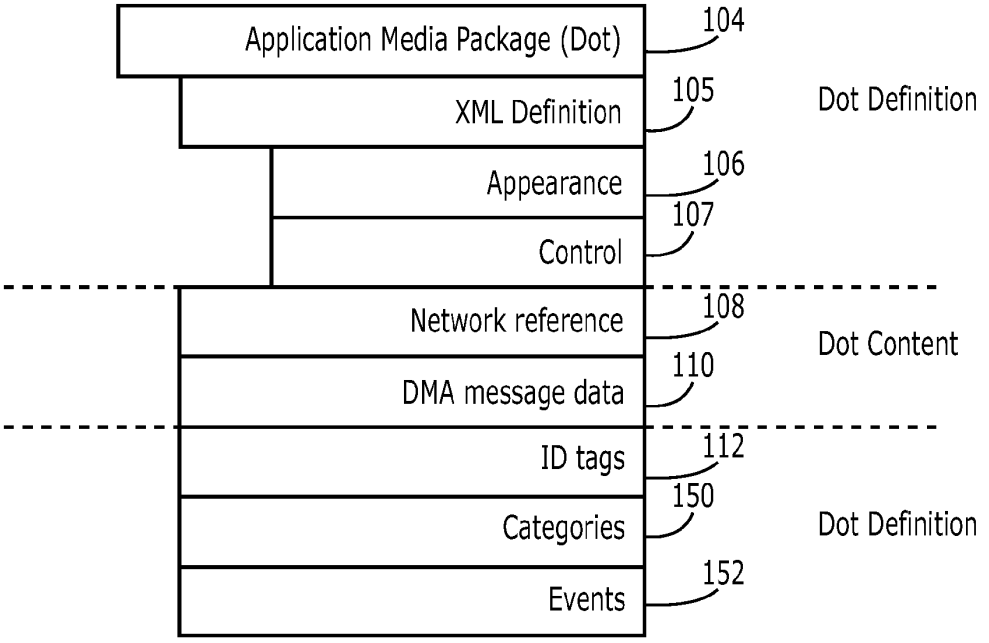

Application Media Package (Dot)
104
XML Definition
105
Dot Definition
Appearance
106
Control
107
Network reference
108
Dot Content
DMA message data
110
ID tags
112
Categories
150
Dot Definition
Events
152

FIG. 3

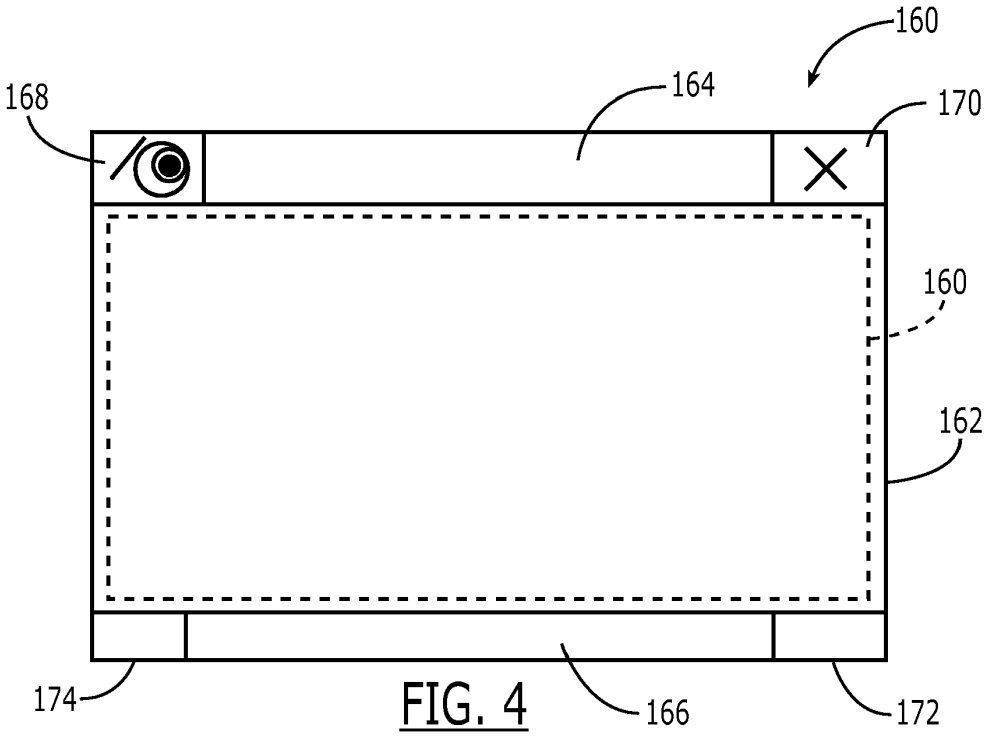


FIG. 4

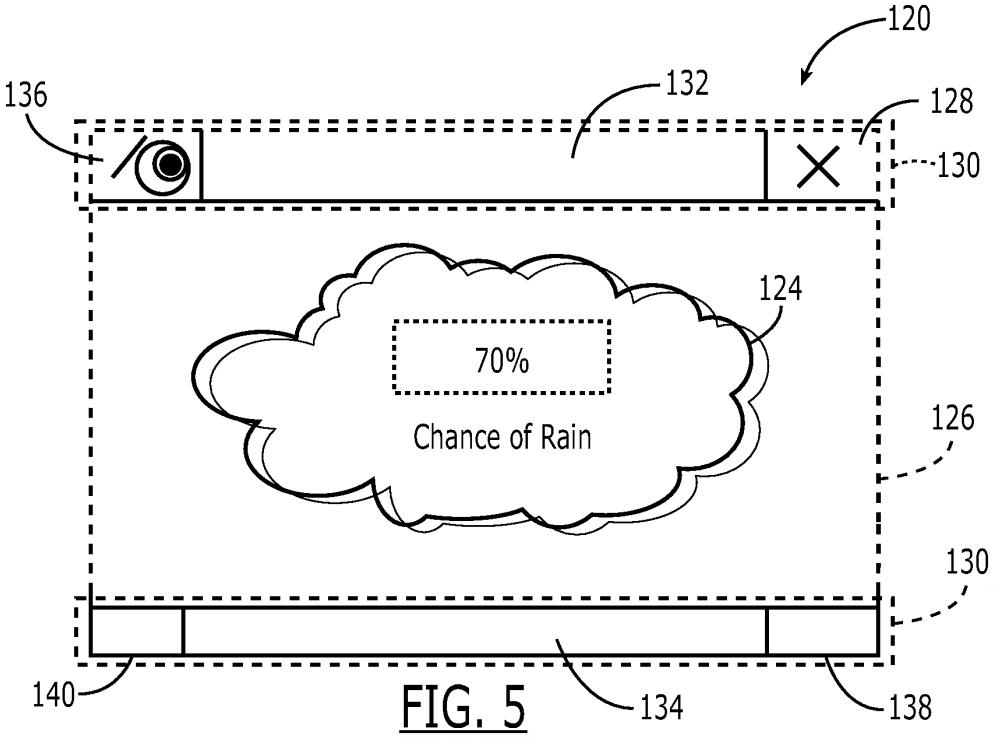


FIG. 5

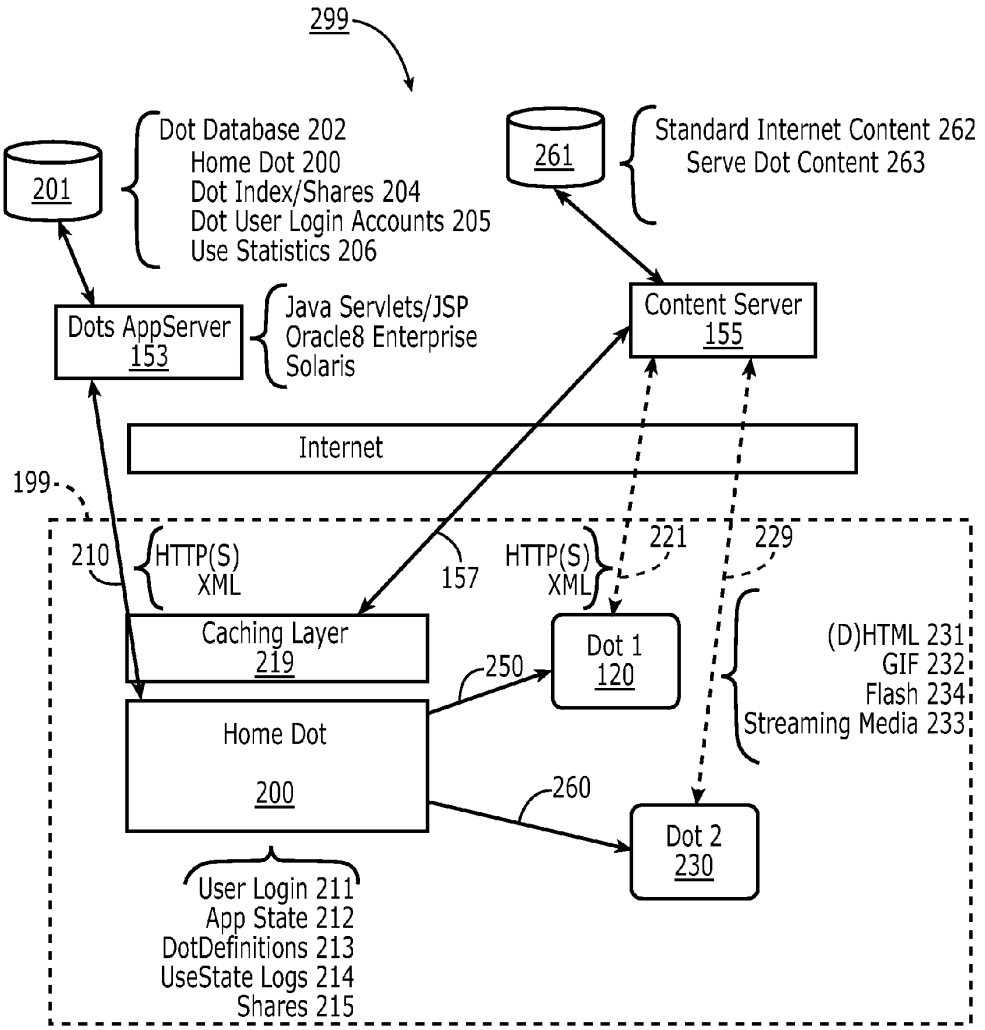

299
201
Dot Database 202
Home Dot 200
Dot Index/Shares 204
Dot User Login Accounts 205
Use Statistics 206
261
Standard Internet Content 262
Serve Dot Content 263
Dots AppServer 153
Java Servlets/JSP
Oracle8 Enterprise
Solaris
Content Server 155
Internet
199
210
HTTP(S)
XML
157
HTTP(S)
XML
221
229
Caching Layer 219
250
Dot 1 120
(D)HTML 231
GIF 232
Flash 234
Streaming Media 233
Home Dot 200
260
Dot 2 230
User Login 211
App State 212
DotDefinitions 213
UseState Logs 214
Shares 215

FIG. 6

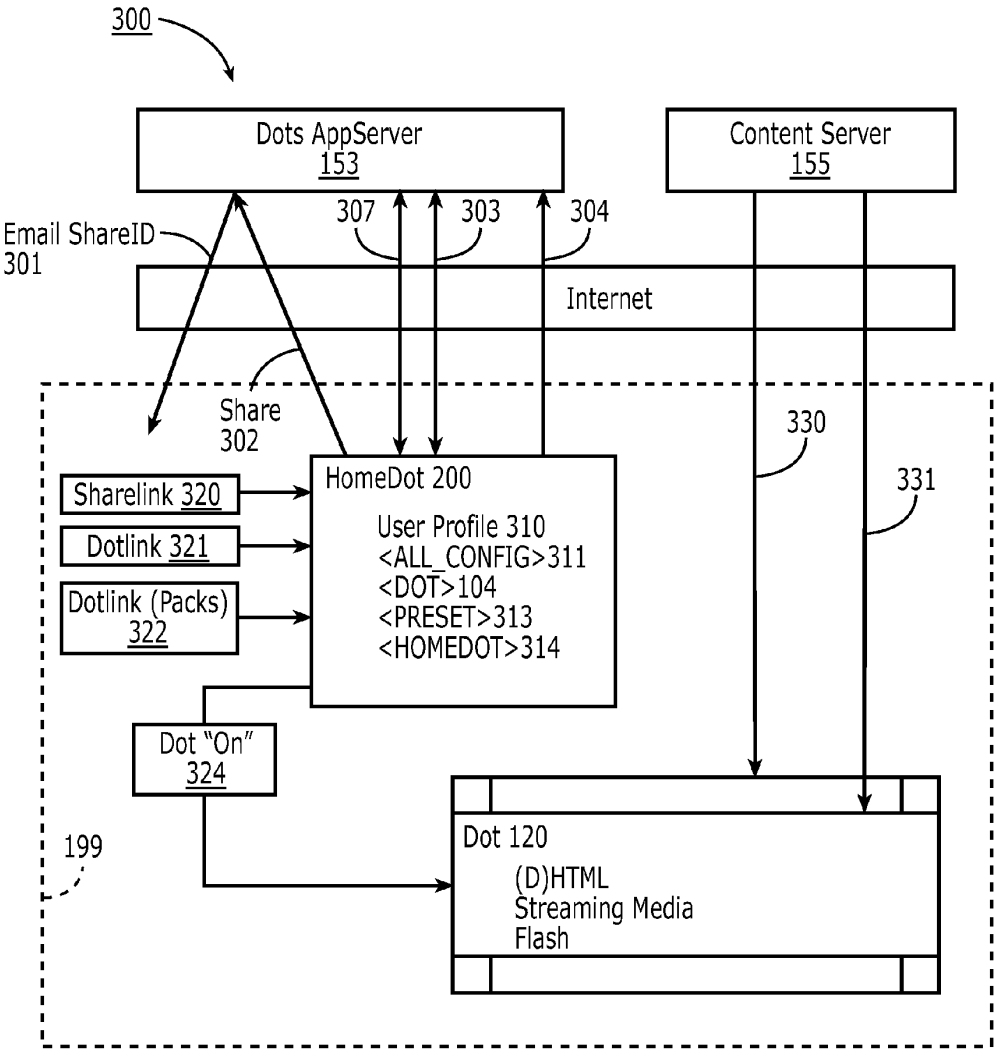

300
Dots AppServer
153
Content Server
155
307
303
304
Email ShareID
301
Internet
Share
302
330
331
Sharelink 320
Dotlink 321
Dotlink (Packs)
322
HomeDot 200
User Profile 310
<ALL_CONFIG>311
<DOT>104
<PRESET>313
<HOMEDOT>314
Dot "On"
324
199
Dot 120
(D)HTML
Streaming Media
Flash

FIG. 7

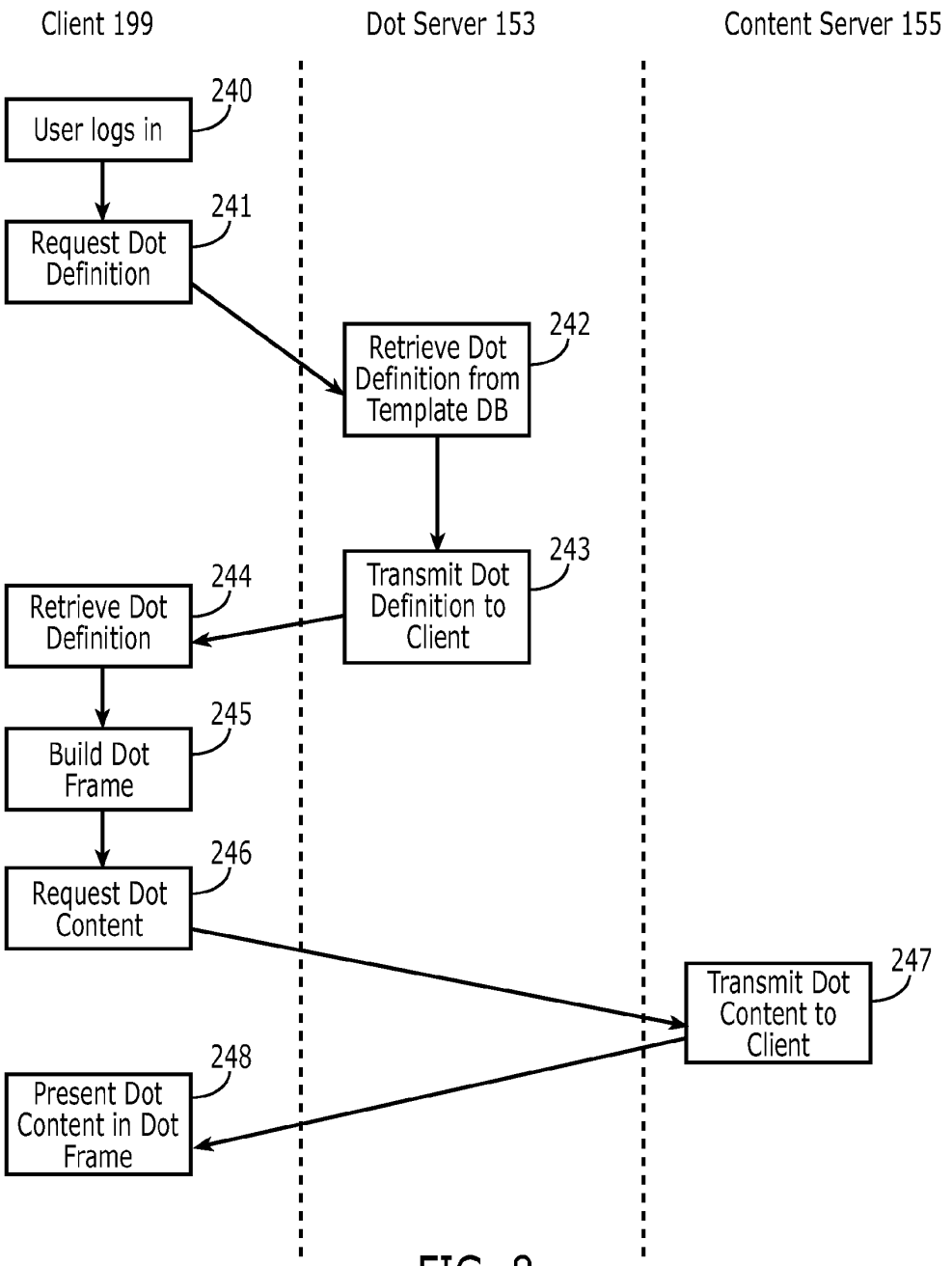

Client 199
Dot Server 153
Content Server 155
User logs in
240
Request Dot Definition
241
Retrieve Dot Definition from Template DB
242
Transmit Dot Definition to Client
243
Retrieve Dot Definition
244
Build Dot Frame
245
Request Dot Content
246
Transmit Dot Content to Client
247
Present Dot Content in Dot Frame
248

FIG. 8

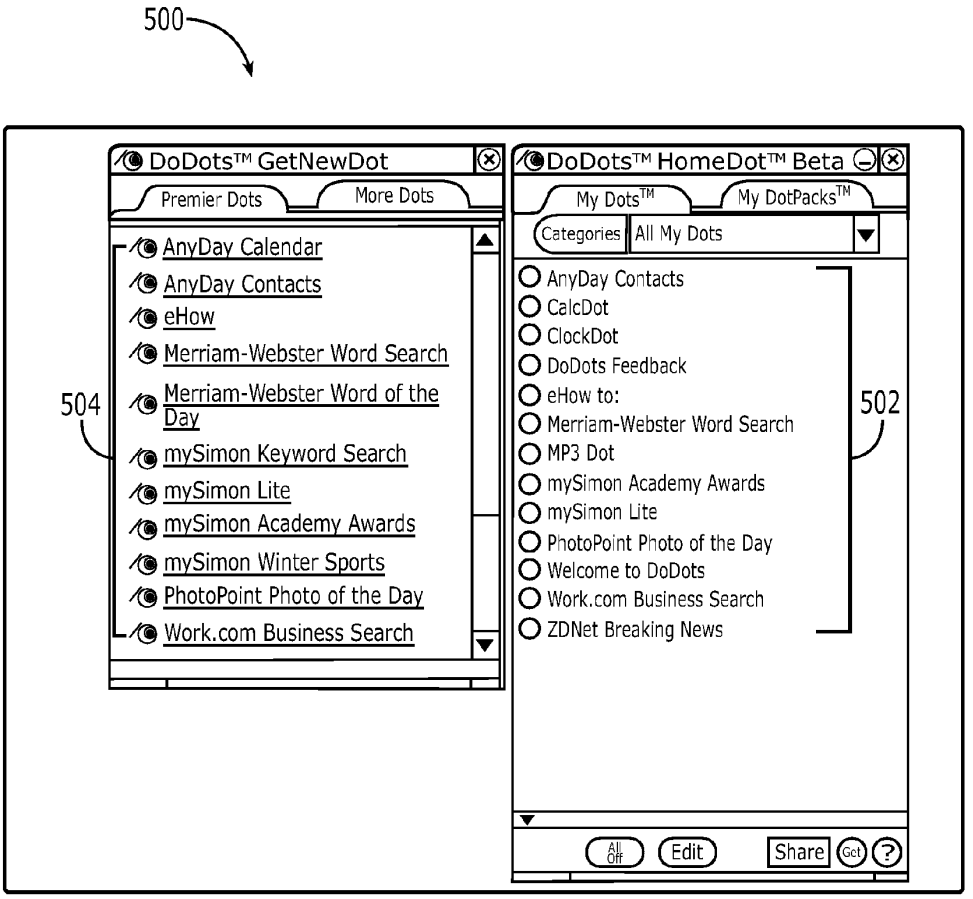

500
DoDots™ GetNewDot
Premier Dots
More Dots
AnyDay Calendar
AnyDay Contacts
eHow
Merriam-Webster Word Search
Merriam-Webster Word of the Day
mySimon Keyword Search
mySimon Lite
mySimon Academy Awards
mySimon Winter Sports
PhotoPoint Photo of the Day
Work.com Business Search
504
DoDots™ HomeDot™ Beta
My Dots™
My DotPacks™
Categories
All My Dots
AnyDay Contacts
CalcDot
ClockDot
DoDots Feedback
eHow to:
Merriam-Webster Word Search
MP3 Dot
mySimon Academy Awards
mySimon Lite
PhotoPoint Photo of the Day
Welcome to DoDots
Work.com Business Search
ZDNet Breaking News
502
All Off
Edit
Share
Get
?

FIG. 9

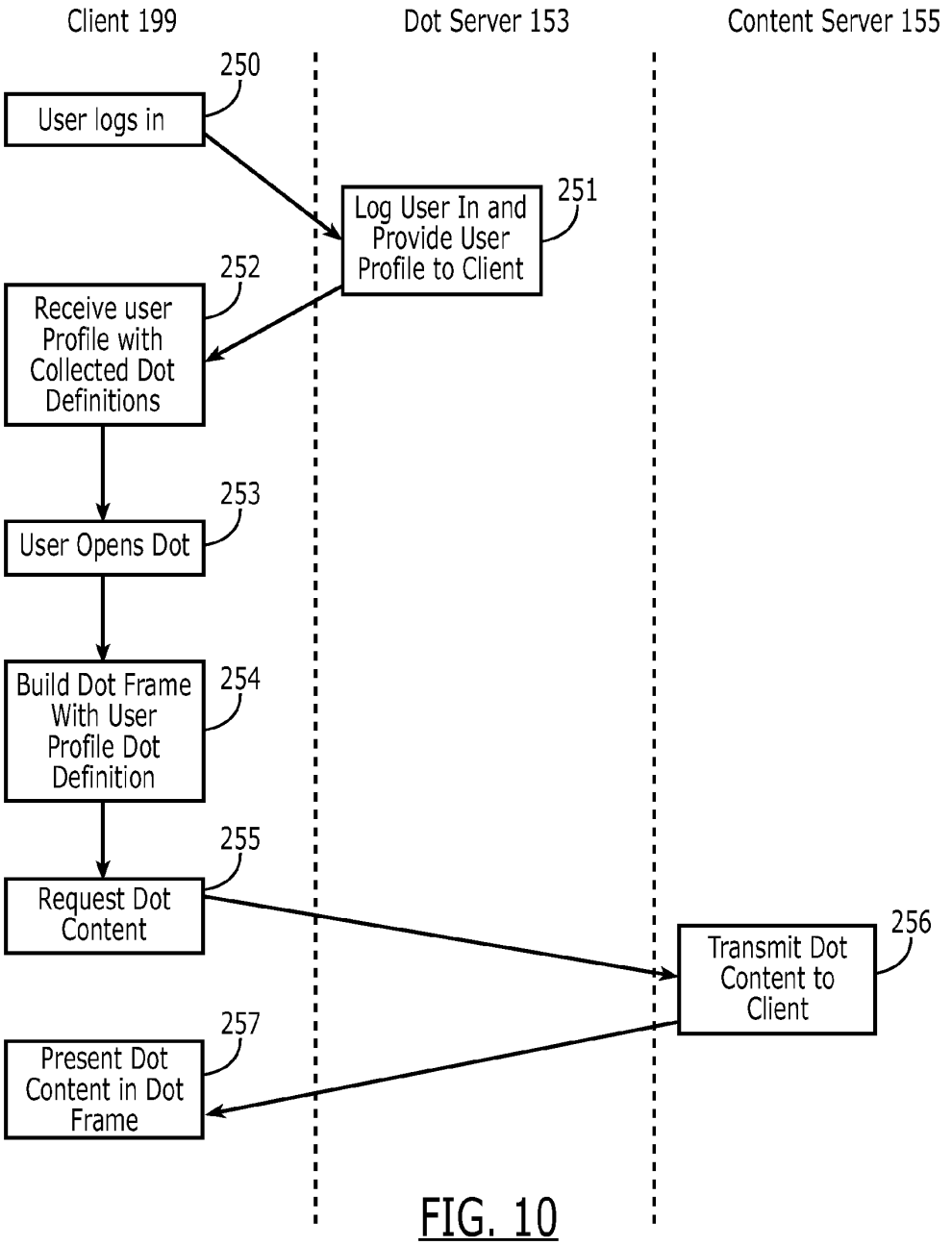

Client 199
Dot Server 153
Content Server 155
250
User logs in
251
Log User In and Provide User Profile to Client
252
Receive user Profile with Collected Dot Definitions
253
User Opens Dot
254
Build Dot Frame With User Profile Dot Definition
255
Request Dot Content
256
Transmit Dot Content to Client
257
Present Dot Content in Dot Frame

FIG. 10

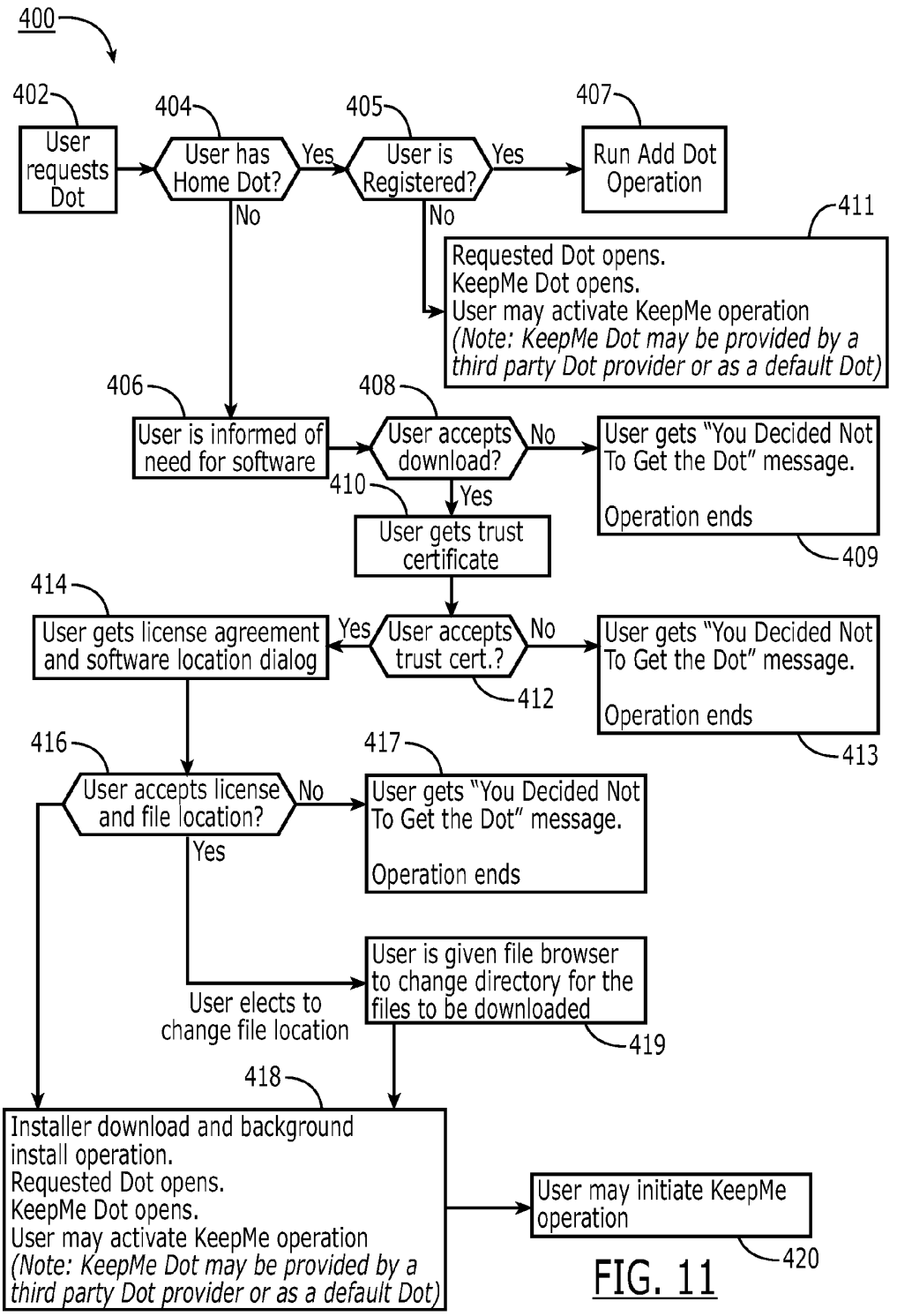

400
402
User requests Dot
404
User has Home Dot?
Yes
No
405
User is Registered?
Yes
No
407
Run Add Dot Operation
411
Requested Dot opens.
KeepMe Dot opens.
User may activate KeepMe operation
(Note: KeepMe Dot may be provided by a third party Dot provider or as a default Dot)
406
User is informed of need for software
408
User accepts download?
No
Yes
User gets "You Decided Not To Get the Dot" message.
Operation ends
409
410
User gets trust certificate
414
User gets license agreement and software location dialog
Yes
User accepts trust cert.?
No
412
User gets "You Decided Not To Get the Dot" message.
Operation ends
413
416
User accepts license and file location?
No
Yes
417
User gets "You Decided Not To Get the Dot" message.
Operation ends
User elects to change file location
User is given file browser to change directory for the files to be downloaded
419
418
Installer download and background install operation.
Requested Dot opens.
KeepMe Dot opens.
User may activate KeepMe operation
(Note: KeepMe Dot may be provided by a third party Dot provider or as a default Dot)
User may initiate KeepMe operation
420

FIG. 11

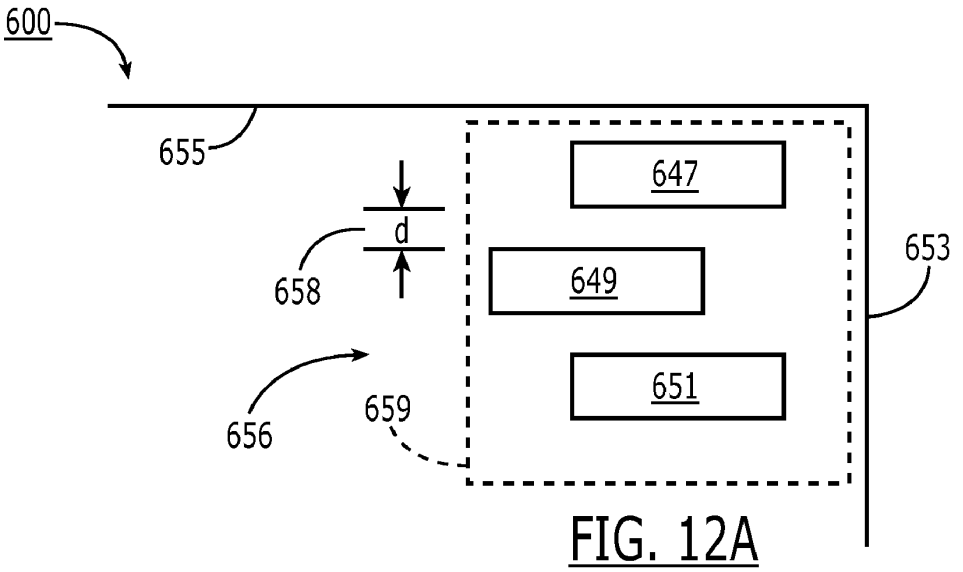


FIG. 12A

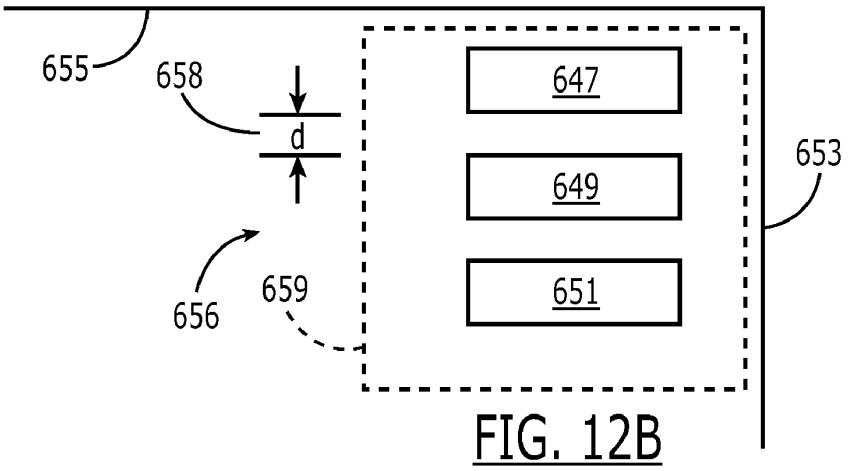


FIG. 12B

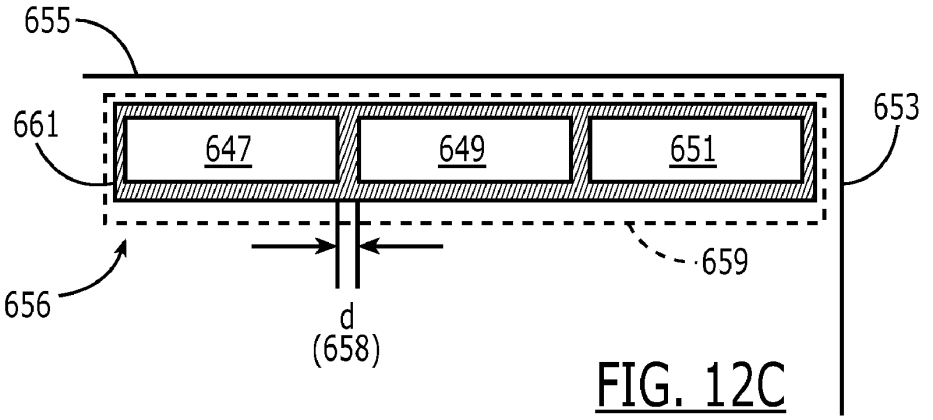


FIG. 12C

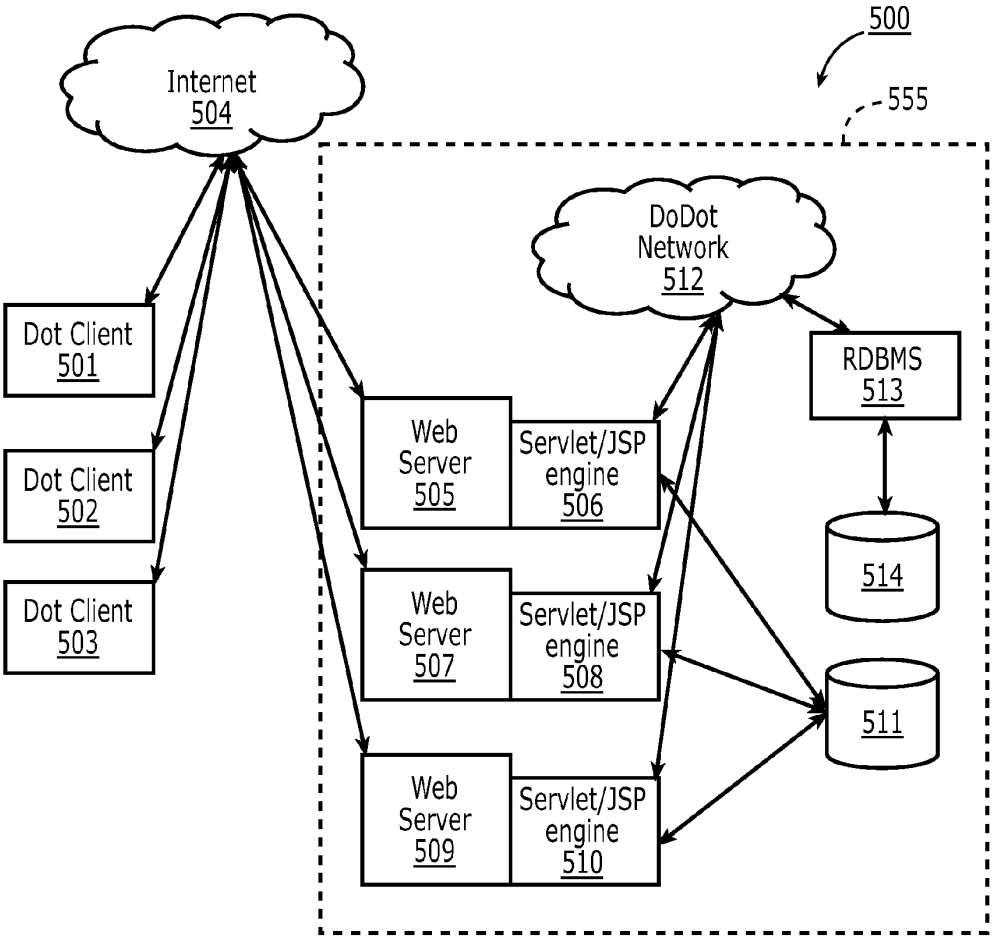

500
555
Internet
504
DoDot
Network
512
Dot Client
501
Dot Client
502
Dot Client
503
RDBMS
513
Web
Server
505
Servlet/JSP
engine
506
514
Web
Server
507
Servlet/JSP
engine
508
511
Web
Server
509
Servlet/JSP
engine
510

FIG. 13

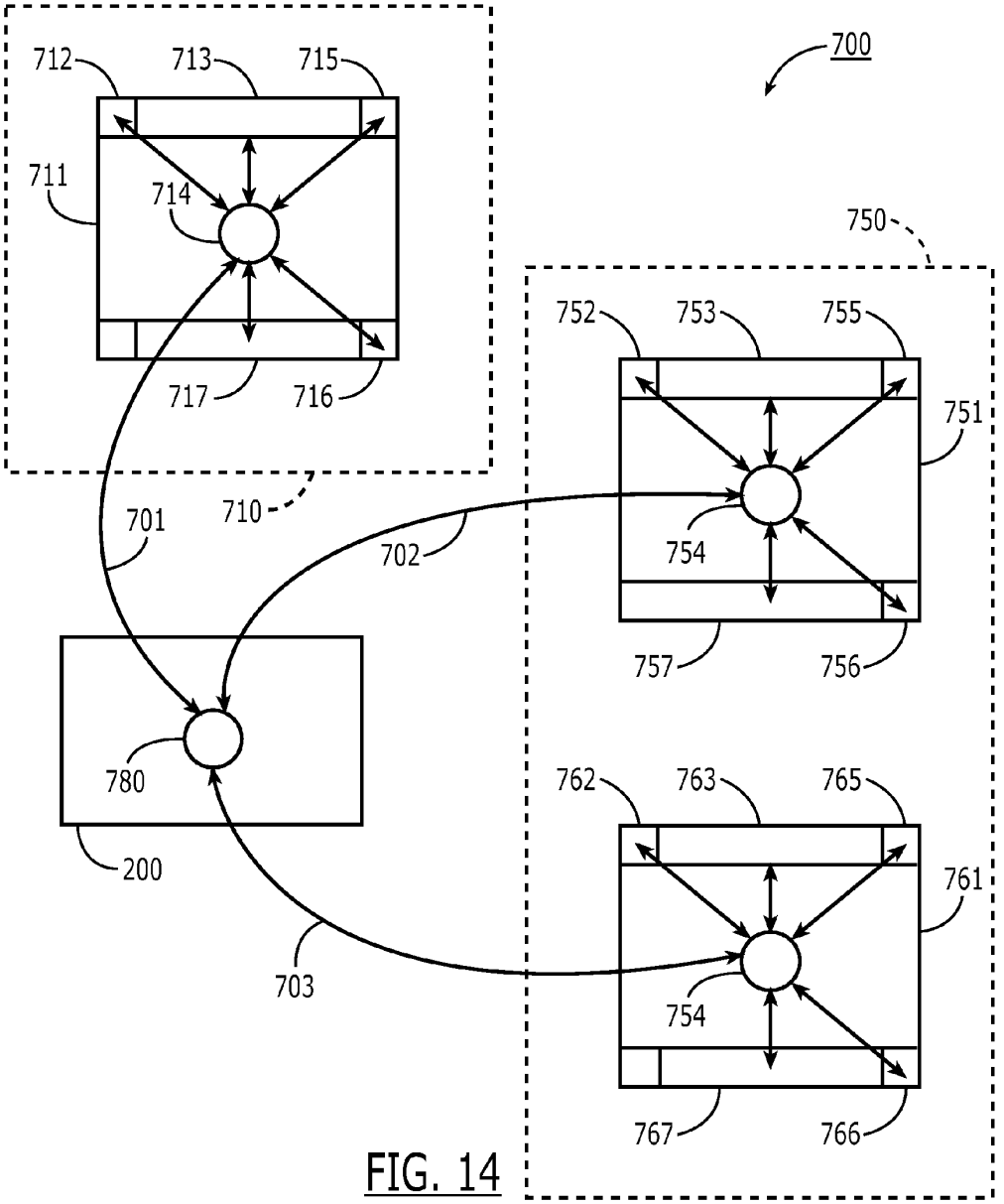

700
712
713
715
711
714
717
716
710
701
702
750
752
753
755
751
754
757
756
780
200
762
763
765
761
754
703
767
766

FIG. 14

SYSTEM AND METHODS FOR CREATING AND AUTHORING INTERNET CONTENT USING APPLICATION MEDIA PACKAGES

CROSS-REFERENCE TO RELATED APPLICATIONS

The present application is a continuation of and incorporates by reference U.S. Non-Provisional patent application Ser. No. 09/558,922, filed Apr. 26, 2000, which claims priority from and incorporates by reference U.S. Provisional Application ser. Nos. 60/131,083, filed Apr. 26, 1999, 60/131,114, filed Apr. 26, 1999, 60/131,115, filed Apr. 26, 1999, 60/176,687, filed Jan. 18, 2000, and 60/176,699, filed Jan. 18, 2000. The present application claims priority to U.S. Non-Provisional patent application Ser. No. 09/558,925, filed Apr. 26, 2000 and each of the aforementioned applications to which it claims priority.

The present application is also related to and incorporates by reference the following U.S. patent applications: Non-Provisional application Ser. No. 09/558,923, filed Apr. 26, 2000; Non-Provisional application Ser. No. 09/558,924, filed Apr. 26, 2000; Non-Provisional application Ser. No. 09/558,925, filed Apr. 26, 2000; Non-Provisional application Ser. No. 11/932,286; filed Oct. 31, 2007, titled "Component For Accessing And Displaying Internet Content"; Non-Provisional application Ser. No. 11/932,340, filed Oct. 31, 2007, titled "Server Including Components For Accessing And Displaying Internet Content And For Providing Same To A Client"; Non-Provisional application Ser. No. 11/932,392; filed Oct. 31, 2007, titled "Method For Accessing And Displaying Internet Content"; Non-Provisional application Ser. No. 11/932,427, filed Oct. 31, 2007, titled "Component For Coordinating The Accessing And Rendering Of An Application Media Package"; Non-Provisional application Ser. No. 11/932,456, filed Oct. 31, 2007, titled "Tracking and Tracing User Activity with Application Media Packages"; Non-Provisional application Ser. No. 11/932,553, filed Oct. 31, 2007, titled "Displaying Time-varying Internet based Data using Media Application Packages"; Non-Provisional application Ser. No. 11/932,630, filed Oct. 31, 2007, titled "Methods of Obtaining Application Media Packages"; Non-Provisional application Ser. No. 11/932,663, filed Oct. 31, 2007, titled "Indexing, Sorting, and Categorizing Dots"; Non-Provisional application Ser. No. 11/932,692, filed Oct. 31, 2007, titled "System and Methods of Messaging between Application Media Packages"; and, Non-Provisional application Ser. No. 11/932,763, filed Oct. 31, 2007, titled "Component For Accessing And Displaying Internet Content In Association With a Web Browser Application".

COPYRIGHT NOTICE

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

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention is related to accessing and viewing Internet content, and more specifically to a method and apparatus for providing a unique frame, independent of a Web Browser application and window, for the retrieval and display of such content.

2. Description of the Prior Art

A user operating a client computer typically accesses the Internet by using a viewer application, such as a browser to view Internet content provided at a destination address, typically a web page. In this context, Internet content and web applications are designed to fill the entire web page. It is known to divide the Internet content into different regions of a single web page. For example, personalized web pages can be specified, such that a user views a variety of content sources in a single page, such as stock information, weather information, and sports information, which is aggregated at the server that delivers the web page to the user, who then views the aggregated content in a single web page. Observe that even when disparate content is aggregated, in this manner, it is reassembled into a full web page and is served through a full-screen browser.

Users and application developers therefore have limited control over the presentation of internet content: content is typically trapped within the frame of the browser. A developer's only alternative to engaging a user page-by-page in a browser is to develop, distribute, and support custom client software. In the Web browser scenario, it is the content provider, not the user that aggregates the information that is viewed by the user. Thus, the user is not in a position to separately aggregate the content at a client computer, instead the user is constrained to view the content that has been delivered in the manner provided by the server computer hosting the web page. There is a growing desire for individual users to fully control the aggregation and presentation of content and web applications that appears on a client computer.

A user who wishes to view multiple web pages or applications can open multiple instances of a browser. However, the user will not be able to view each "full-screen" page at the same time. Instead, the user must adjust the windows corresponding to each browser instance and view only part of each page. The information appearing in each browser is not designed for viewing in this manner. Thus, the user cannot create an optimized display of content from multiple sources.

Currently, content providers and end users have limited tools to alter the browser in which content appears. That is, the controls associated with a browser are not fully configurable. Thus, the vendor of a browser is in a position to brand the browser and regulate the controls associated with the browser. There is a growing desire for content providers to not only fill a browser with their content, but to also fully brand and control the frame in which the content appears. Further, in some instances, content providers desire to limit the controls associated with a browser or viewer, so that a user is more inclined to view a single set of content, for example, by having limited access to previously viewed content. However, the current browser applications provide very limited control to a user or content provide to alter the frame and controls provided by the browser window.

In summary, therefore, the current model of the Internet has the following attributes and limitations:

a) Internet content is typically viewed one page at a time, with each page displayed serially displayed in a browser application window which typically takes up the majority of the user's computer screen real estate.

b) Internet content is designed for delivery in web pages. Even if content is modularized, it is reassembled into a

full web page and viewed serially in the window of browser application taking up a significant portion of the user's desktop.

c) There is a distinction both visually and architecturally between the "viewer application" (browser) and the "content/document" (web page) such that a browser window is not tailored to the content being displayed, but rather is capable of displaying any web content.

d) Internet content is effectively limited and trapped within the "frame" of the browser (viewer application). Therefore, content developers, users and web-application developers are limited in how the user experience is controlled.

e) Although internet programming technologies (such as Java script, CSS, layers, flash, etc.) are giving web pages more functionality, the pages have limited access to application functionality such as access and control of the window and frame, the size of the frame, branding, application behavior such as size and menu items, etc.

f) "Web-applications" such as web-mail and web calendars are being packaged and viewed through the page-by-page web model. Even though web-applications are being implemented by many online companies, the web is currently a destination page-based model where, for example, a user visits one page, then goes to another page and so on. It is therefore a sequential, linear experience, e.g., one full page at a time.

In view of the foregoing, there is a need in the art to provide a technique for accessing multiple instances of distributable computer readable web content in which these instances are typically smaller than the full pages used in current web pages and web applications, and which may be displayed in user- or content provider-controlled frames. Preferably, such techniques allow such access to be done simultaneously. There is a further need for providing the user with flexibility in selecting, collecting, relating and viewing such web content and for giving the content provider flexibility in directing media to a specific user and controlling the framework in which media, such as web content, is presented. Finally, there is a need to gather more accurate information regarding the type of content that a user enjoys, so that the user can be automatically provided with this content.

SUMMARY OF THE INVENTION

The present invention is directed to systems and methods by which internet content may be authored and distributed. The invention provides for a structure defined herein as an Application Media Package. Application Media Packages are web browser-readable code that is executed on a non-browser-based installed client application. The client application, referred to herein as an Application Media Viewer or Home Dot, executes independently from a web browser. The Application Media Viewer parses and executes the Application Media Package code to create the user experience. The terms Application Media Package and Dot are use synonymously herein.

In addition to the Home Dot, the Dot Server provides a central collection and distribution point for Dot related data. The Dot server is communicated to directly by the Home Dot application, by Dot users, and by third party Dot developers. Internet content that is referenced within a Dot may or may not reside on the Dot server. In general, it does not.

The Dot server supports the community of Dot developers, providing developer tools, including Dot template which provide an expedient method of re-purposing existing internet media into a new presentation package by choosing from existing examples.

The Dot server supports posting and categorization methods for Dot developers to place their Dots in a manner that is effective for Dot users, a manner that promotes Dot developers, a manner that generates revenue by preferential placement of Dots, etc. The Dot server supports developer accounts. These accounts may control access to tools and information as well as provide a development and test zone for Dot verification, authentication and acceptance before posting.

According to the present invention, the Dot is supported by a flat file format that enables a simple file download process that is completed by the Home Dot. The flat file format may be zipped and may also include a method that protects the contents of the Dot from source level viewing. Decoding and parsing is executed by the Home Dot.

In one embodiment, the present invention provides for standard or base controls that are common to all Dots. These controls include size and placement for title bar, exit button, resize button, and menu button. In addition, standard images or content are provided as a default for each of these controls.

According to the present invention, each Dot developer is provided with a unique developer ID by which all of his Dots is identified. Dots may also have a unique Dot ID, a Dot kind (or type attribute), and category.

The above is a summary of a number of the unique aspects, features, and advantages of the present invention. However, this summary is not exhaustive. Thus, these and other aspects, features, and advantages of the present invention will become more apparent from the following detailed description and the appended drawings, when considered in light of the claims provided herein.

BRIEF DESCRIPTION OF THE DRAWINGS

In the drawings appended hereto like reference numerals denote like elements between the various drawings. While illustrative, the drawings are not drawn to scale. In the drawings:

FIG. **1** is an illustration of the overall architecture of a network in which the present invention may operate.

FIG. **2** is an illustration of the elements of an Application Media Package (Dot) according to one embodiment of the present invention.

FIG. **3** is an illustration of the elements of an Application Media Package (Dot) definition according to one embodiment of the present invention.

FIG. **4** is an illustration of an instantiation of a generic GUI according to one embodiment of the present invention.

FIG. **5** is an example of an instance of an Application Media Package (Dot) according to one embodiment of the present invention.

FIG. **6** is an illustration of the communication flow associated with the acquisition and instantiation of an Application Media Package (Dot) according to one embodiment of the present invention.

FIG. **7** is all illustration of the communication pathways associated with the building of a user profile, and the acquisition of Dots through links, packs, and sharelinks according to one embodiment of the present invention.

FIG. **8** is an illustration of the communication pathways associated with the acquisition and instantiation of an Application Media Package (Dot) according to one embodiment of the present invention.

FIG. **9** is an example of a display showing Application Media Packages (Dot) available for downloading as well as Application Media Packages (Dots) already downloaded according to one embodiment of the present invention.

FIG. **10** is an illustration of the communication pathways associated with the instantiation of an already-acquired Application Media Package (Dot) according to one embodiment of the present invention.

FIG. **11** is a flow chart illustrating the download process of an Application Media Package (Dot) and Application Media Viewer (Home Dot) according to an embodiment of the present invention.

FIGS. **12**A, **12**B, and **12**C are illustrations of Application Media Packages arranged in groups, and arranged in vertical and horizontal Blocks (position-justified groups), respectively, according to an embodiment of the present invention.

FIG. **13** is an illustration of the client-server model of a system according to the present invention.

FIG. **14** is an illustration of the message routing paths and elements of a Dot Messaging Architecture according to the present invention.

DETAILED DESCRIPTION OF THE INVENTION

The present invention discloses a technology that is capable of processing distributable computer readable media. Distributable computer readable media includes, but is not limited to, standard Internet content, such as HTML, dHTML, images, imbedded ActiveX and Java applications, JavaScript, CSS, Perl scripts, Streaming Media, and/or Flash. The present invention is advantageous relative to prior art systems and methods because it provides improved mechanisms for simultaneously interacting with several independent sources of distributable computer readable media, collecting references to such media, and sharing such references with other users. The disclosed technology is further advantageous because it provides improved systems and methods for on screen management of distributable computer readable media.

Central to the present invention is the concept of an Application Media Package. As used herein, the term Application Media Package refers to a component which includes a fully configurable frame with one or more controls; the frame through which content is optionally presented. The fully configurable frame utilized in accordance with the invention stands in contrast to present web browsers, which are branded by the browser vendor and which have limited means by which to alter the controls associated with the browser.

Absence of Web Browser

The Application Media Package is a file that is comprised of web browser readable language. According to the preferred embodiment, the present invention renders, displays, and updates Internet data without the use of a browser. In fact, no browser need be installed on the client computer on which the Application Package is instantiated. The present invention produces a user experience by parsing and rendering the Application Media Package through the Application Media Viewer. The Application Media Viewer is an installed client application which renders the Application Media Package as well as provides additional functionality to the user experience (hence, it is more than simply a viewer application). The Application Media Viewer may render web browser readable content (of the type typically supported by a browser application) due to it being programmed utilizing the Microsoft IE4 object for receiving, parsing and rendering web content. It will be understood by one skilled in the art that, despite its name, the Microsoft IE4 object is not a web browser application. One apparent difference between the Application Media Viewer and a typical web browser application is that the Application Media Viewer of the present invention is not designed to provide user web navigation and page control typically provided by a web browser.

FIG. **1** is a general illustration of a system in accordance with one embodiment of the present invention. In FIG. **1**, a network **10** is operated in accordance with the present invention. Network **10** includes at least one user or client computer **20**, at least one server computer of class **50**, and optionally one or more server computers of class **82**. User computer **20** as well as server computers of class **50** and **82** are each connected by transmission channel **44**, which is any wired or wireless transmission channel.

User computer **20** is any device that includes a Central Processing Unit (CPU) **24** connected to a random access memory **30**, a network connection **28**, and one or more user input/output ("i/o") devices **40** including output means **42**. Thus, the term "computer," as used herein, is intended in its broadest sense to include not only traditional workstations, laptops and similar devices, but any device containing a CPU of sufficient operability to utilize Internet data, such as web-enable cell phones, personal digital assistants (PDAs), and so forth.

Output means **42** is any device capable of communicating with a user and includes, for example, a video monitor, a liquid crystal display, voice user interfaces, and/or integrated graphic means such as mini-displays present in web-enabled cell phones, PDAs, etc.

Typically, user computer **20** includes a main non-volatile storage unit **22**, preferably a hard disk drive, for storing software and data. Further, user computer **20** includes one or more internal buses **26** for interconnecting the aforementioned elements. In a typical embodiment, memory **30** includes an operating system **32** for managing files and programs associated with user computer **20**. In some embodiments, operating system **32** includes a registry **34** that has one or more references to specified locations in system **10**.

The exemplary memory **30** of FIG. **1** further includes a web browser **36** for viewing Internet content and a separate collection of items, referred to herein as a client parser application **38**. In some embodiments, client parser application **38** uses the one or more references in registry **34** to obtain a login construct from server **50**. In various embodiments, in accordance with the present invention, client parser application **38** runs in conjunction with one or more software modules, such as an event log module **98**, which tracks user activity, a message interface module **107**, which serves as a communication interface between the client parser application **38** and web server **58** and/or external web servers, an application media processing module **109**, one or more Application Media Packages **104-1** to **104-*n***, and an Application Media Viewer **119** which, among other functions, regulates the characteristics of visual manifestations of Application Media Packages **104-1** to **104-*n*** when displayed on output device **42**.

Server computer **50** includes standard server components, including a network connection device **46**, a CPU **52**, a main non-volatile storage unit **54**, and a random access memory **56**. Further, server computer **50** includes one or more internal buses **48** for interconnecting the aforementioned elements. Memory **56** stores a set of computer programs, modules and data to implement the processing associated with the present invention.

The embodiment of memory **56** illustrated in FIG. **1** includes a web server **58** for processing requests received from client computer **20**. Web server **58** has many components, including a variety of modules and data structures to

assist users that want to log into system **10**. Namely, login module **60** handles an entry request from a client computer **20** and accepts a login identifier that corresponds to a user from client computer **20**.

Once a user has successfully logged into system **10**, request server module **72** handles requests for specified Application Media Packages **104-1** to **104-*n*** from client **20**. When such a request is received, request server module **72** routes the request to an address that corresponds to the specified Application Media Packages **104-1** to **104-*n*** and transmits the specified Application Media Packages **104-1** to **104-*n*** to client **20**. One class of specified networked information handled by request server module **72** is requests for Application Media Packages **104-1** to **104-*n***. When such a request is received, request server module **72** searches Application Media Packages database **74** for the specified Application Media Package. Application Media Package templates database **74** includes a large number of Application Media Package templates. Each Application Media Package template defines the characteristics of a specific Application Media Package, including fully configurable frame characteristics, viewer and control characteristics, and Application Media Package content references.

The web server **58** illustrated in FIG. **1** further includes additional modules **131** to handle specialized features of the present invention. For example, one embodiment of the present invention provides a mechanism that allows users to distribute Application Media Packages to each other. In such embodiments, a special server module **131** provides instructions for storing the Application Media Packages, which are to be distributed, using a sharelink database **78**. Advantageously, Application Media Packages that are distributed to other users are customizable. A user can, for example, resize and position a particular Application Media Packages prior to sharing it with another user. Indeed, it is possible, in such embodiments, for a user to arrange a series of Application Media Packages in a unique arrangement and then distribute the collection of Application Media Packages in the designated arrangement. As an illustration, a user arranges a first Application Media Package that represents a scrolling stock ticker at the bottom of an output means, such as a computer screen, a second Application Media Package that tracks the NASDAQ top ten most heavily traded stocks in the upper left corner of the output means, and a third Application Media Package that tracks headline news on the upper right hand corner of the output means. Then, the user distributes the three Application Media Packages in this customized arrangement to other users. Observe that in this example a user of a client computer is aggregating separate sets of information in different Application Media Packages. This stands in contrast to prior art approaches where a web server running on a server computer aggregates information in a single page.

System **10** is highly scalable and thus supports a large number of users. This scalability stems from the fact that the server **50** is delivering the definition associated with an Application Media Package. The content displayed in the Application Media Package may be located on a separate computer. Memory **56** may provide a statistical analysis module **133** for tracking key events associated with users. This information is stored in statistics database **80**. The information collected by statistical analysis module **133** is used for a wide variety of purposes, including server load optimization and directed advertising, as discussed below. As described below, the statistical information gathered in accordance with the invention includes fully traced events defining the type of content and the duration over which all content is viewed by a user. This type of comprehensive information is not available using present techniques. Much of the distributable computer readable media that is available for processing is stored as content elements **94** on server **82**. Server **82** is a standard web server that includes components such as a network connection device **88**, a CPU **86**, a main non-volatile storage unit **84**, a random access memory (RAM) **92**, and one or more internal buses **90** for interconnecting the aforementioned elements. RAM **92** includes some of the content elements **94** stored by server **82**. Other content elements **94** are stored in storage unit **84**. In some embodiments, a single web server **58** is capable of directly accessing content elements **94** located on one or more servers **82**. In other embodiments, each server **82** has a resident web server module that works in conjunction with server **50** to identify, optionally dynamically generate, and serve content elements **94** upon demand.

With the general architecture of a system within which the present invention may operate provided with reference to FIG. **1**, we next turn to a number of definitions of key terms used herein. Terms not specifically defined herein shall be understood to have their broadest, generally accepted meaning. Other terms may be defined elsewhere in the present disclosure. (See also Appendix A and B herein.)

Application Media Package—An Application Media Package, also referred to herein as a Dot, is computer software component, such as XML code and data, representing the definition of a client-side mini-application, which displays information and/or provides functionality to an end user. The Application Media Package combines the packaging, application behavior, and the presentation of Internet content with the content itself, creating one integrated unit—a Dot. Therefore, Dots are the integration of application and media. A Dot may be viewed as a free-floating mini-site. It is frame in which Internet content is presented (although lacking the most common functionality typically associated with a browser, navigation).

In one embodiment, Dots are programmed with browser readable language that is parsed to the Microsoft Internet Explorer (IE) rendering object (referenced within a control, so named the Dot Web Conduit) and are capable of displaying any Internet content supported by IE 4 (however, not necessarily requiring use of Internet Explorer or any browser for such display). This browser readable language defines the appearance of a Dot, its functionality, and what content is presents. This browser readable language includes XML, streaming media, dHTML, etc.

Structurally, the Dot or media application package comprises initialization data, control calls, and a frame. Content, such as that obtained from a network reference, is rendered within the frame. The control calls may include the web conduit, base controls such as "close" and resize", base GUI elements such as a title bar, Dot menu, and Dot bottom bar, as well as XML Dot calls that are made by the Dot Definition and executed by the Home Dot. Unlike a downloadable, custom, client-application (e.g., for delivering custom web applications to users' desktops), an Application Media Package is entirely content. That is, the entirety of a Dot package, referred to as its definition, is in a browser-readable language.

All that is required to instantiate an Application Media Package (on the client computer) is its definition assuming that the client computer has an Application Media Viewer (discussed below) installed. That is, no browser is needed to render an Application Media Package. No browser need be running or be present on the client machine in order to render an Application Media Package.

Application Media Package Content—Application Media Package content, also referred to as Dot Content, is Internet content served off of partner/Dot Developer servers or

another third party server. While in some cases a party developing and making a Dot available (from a partner/Dot Developer server) may also host (on the server) content for that Dot, the developer is free to cause the Dot to point to content from another party/site for rendering as part of an instantiated Dot. It is an attribute of the present invention that any internet content can be accessed, processed, and/or displayed as Dot Content or Application Media Package Content.

Application Media Viewer—The Application Media Viewer, also referred to herein as the Home Dot, is a network enabled, client application that loads and displays an Application Media Package on a client computer. Data contained within the Application Media Package is parsed by the Application Media Viewer and rendered within the extent of the Application Media Package's frame as defined therein. Data is web browser readable language including media and Internet references thereto, as well as control calls. These control calls contained within an Application Media Package are also parsed and executed. The Application Media Viewer is required to instantiate an Application Media Package (i.e., to create a Dot and provide any associated functionality on a client display device). The Application Media Viewer effectively takes the place of a browser application when rendering Internet content via an Application Media Package. However, an Application Media Package contains Internet data and as such, this data may also be referenced by a browser. In the preferred embodiment, a browser is not employed to instantiate an Application Media Package, nor to invoke any functionality they may provide. Also in the preferred embodiment, the Application Media Viewer is a compiled application, programmed using MFC (Microsoft Foundation Class) or the like. The Application Media Viewer need only be downloaded to the client computer one time and may be used to instantiate any number of Application Media Packages. More than one Application Media Package may be instantiated at a time, i.e. the Application Media Viewer is capable of executing and displaying multiple instantiated Application Media Packages. The Application Media Viewer is also capable of checking for updated versions of itself (and any Application Media Package) on the server. The Application Media Viewer supports a number of facilities including support for standard internet protocols (http, https, etc.) Additionally, the Application Media Viewer also supports collecting and organizing Application Media Packages, user login, user activity statistics collecting and reporting, and messaging between Application Media Packages.

Block—A group of Dots. Blocks are created and destroyed by the user through the dragging and positioning of individual Dots or other blocks.

Computer—as used herein, the term computer is meant to refer to any device that includes a Central Processing Unit (CPU) connected to a random access memory, a network connection, connected and rendering to one or more output devices, which has sufficient operability to utilize Internet data, such as web-enable cell phones, personal digital assistants (PDAs), etc. Thus, the term is intended in its broadest sense to include not only traditional workstations, laptops and similar devices, but web-enabled cell phones, PDAs, etc.

Content Server—Also called a Partner Web Server, used to deliver Internet content to a Dot on a client machine.

Controls—A Dot comprises of at least one control, each of which is arranged within the frame according to the Dot Definition. In the preferred embodiment, this minimum control is the Web Conduit which is the Internet data rendering portion of the frame. Controls are calls that are contained within the Dot content that enable a Dot, for example with application behavior amongst other attributes and explained further herein. The Dot Definition may contain base control (close, resize) overrides such that the controls are not rendered or are rendered only upon mouse over events rather than persistent display. Additional media elements and network referenced media may also be contained within the Dot Definition.

DoDots—The working product name of the invention described herein.

Dot Class—An instance of a particular set of binaries to create a particular type of Dot.

Dot Definition—This software entity contains enough information to instantiate an Application Media Package (a Dot) and pointers to location of Application Media Package content (Dot content). A Dot is defined by its dimensions and a set of four attributes called Dot components, as will be later described. A Dot Definition also defines access to Dot Web Conduit, which is a control element of a Dot and is a fully functional rendering element that can access available Internet content.

Dot Link—Hyperlink or URL on a server for downloading a Dot. Based on standard Internet links, these point to or reference particular Dot Definitions and enable the Home Dot to quickly access (using standard HTTP/HTTPS requests, for example) the frame and initialization pointers/URLs in the Dot Definition to instantiate the Dot and render it.

Dot Messaging Architecture (DMA)—Elements within Dots and the Home Dot application create a messaging architecture (the Dot Messaging Architecture) that enables elements of the system including Dots, controls within Dots, and the Home Dot application, to communicate with one another. The DMA enables these elements to exchange information, request actions or functionality, and respond to system, element, or content events.

Dot Pack—A group of Dot templates specified by a particular Dot developer, provider or aggregator that complement each other in some way.

Dot Server—Also called an Internet Application Server, used to deliver Dots to a client machine.

Dot Template—The XML definition of a particular Dot provider's Dot in its default state. Information includes the Dot size, Dot class, starting URL, etc. The Dot template may be defined by a third party Dot provider. The Dot template is used as the starting point for a particular Dot, which may be customized by the user in some way after the first instantiation.

Frame—The frame, also referred to as a Graphical User Interface (GUI), defines that area upon a display of the Dot instantiation in which data may be rendered. As previously described, the frame may contain rendered Internet data. A Dot is comprised of a frame or GUI, initialization input data including data and network references to data displayed in said frame. The appearance of the frame (if any) together with data that is displayed therein is the visible instantiation of the Dot. Any Internet data or media data may be contained within the frame such as an image, Internet content, etc. Additionally the frame shape and content may be wholly defined by the Dot developer.

Lead Dot—The controlling Dot within a block. A lead Dot is the only Dot within a group that presents a control and close box. The lead Dot is the leftmost, topmost Dot (in that order).

Share—A Share is a XML file that represents a collection of one or more

Dots and/or one or more Snapshots.

Snapshot—A point-in-time data set describing an individual user's overall visible Dot configuration. This includes location and configuration information on the user's overall configuration, and all blocks and Dots.

Web Browser Readable Language—standard Internet content that is capable of being parsed by a browser, such content including HTML, Java script, XML, CSS, streaming media, Flash, HTTPS, cookies, etc.

Web Conduit—a control that accesses and renders standard Internet content. Dots in conjunction with the Web Conduit control can render or process anything a browser application can render or process. That is, the web conduit is an entity which processes or handles standard Internet content for rendering the GUI (graphical user interface).

The present invention discloses a technology that is capable of distributing standard Internet content without the use of and restrictions imposed by a browser as detailed hereinabove, in a manner that enables developer control of the user interface. The present invention includes a method and system in which Internet content is developed, packaged and delivered from a server. The present invention further describes methods and systems in which the user obtains content, interacts with it, and is presented with updates to the content either in part or including the entire Application Media Package.

The present invention provides an alternative model of viewing content obtained from the Internet. Unlike the web page model requiring a web browser, the present invention is founded on the premise that the basic unit of the Web and its media should not be limited or restricted to a webpage-based display for presentation in a conventional browser nor should the user's experience of the Web be limited to one page at a time as defined by a conventional browser (such as Microsoft's Internet Explorer™).

Furthermore, the present invention enables efficient placement of web applications directly onto users' desktops. The present invention enables content providers to: break their Internet-based content up into smaller pieces; package this content as Application Media Packages (Dots); give Dots an application behavior through a Dot Messaging Architecture; and to distribute Dots from one user to another (Viral Distribution Architecture).

The ability to break web-based internet content into smaller packages (e.g., Dots) for transfer independent of a conventional browse enables content providers to distribute their media to: fully brand and control a user's experience; obtain direct access to consumers; secure longer on-screen presence; give application-media viral characteristics; and eliminate the need to maintain a software application in order to achieve these characteristics.

With regard to viral characteristics, for example, the present invention allows content providers to enable users to: collect Dots; use Dots in groups (Dot Packs); and share Dots with others. Therefore, Dots are passed from server to user and also from user to user via email links. Not only are may single Dot links be virally distributed, but also collections of Dots and their links may also be virally distributed from user to user. Because Dots reference web data including media, Dots enable the viral distribution of said media although the media may reside on a server.

Software application download is eliminated because Dot content is based on standard Internet content (HTML, dHTML, flash, streaming media, Java, etc.), providing Dots with certain browser-like functionality. By this virtue, Dots may be distributed not as software application downloads but as Internet media downloads, similar to web page downloads. As a Dot may define the frame within which content is to be rendered, the visual extent of a Dot is not bound within the borders of a browser window. Unlike a web page in a third party browser, a Dot's graphical user interface may be wholly defined and/or branded by the developer.

Because Dots are instantiated and rendered upon a client computer as separate and independent graphical user interfaces, and because they may be continuously refreshed with new internet content, Dots may persist on a user's screen. Therefore Dots may possess a longer desktop presence (i.e., remain in place, or "On", while a user attends to other tasks on a device, such as interacting with different applications, making phone calls, etc.

The present invention therefore also allows for persistence of displayed Internet content. For example, a dedicated frame may be rendered on a user's display, Internet content obtained from a server, the obtained Internet content presented in the frame (and possibly periodically updated), and the frame and content positioned and sized such that it may remain, unobtrusively, open on the user's desktop, even when other windows are open. This is fundamentally different than today's model where users visit websites and leave them due to size of the browser window, full use of the content displayed, etc.)

Dots provide Internet content developers and web application developers: a unique way to package their content as custom client application experiences without having to develop, distribute, and support their own custom client application. Developers may repurpose the Internet content that they have developed and used in web pages to form Dots. In doing so, according to the present invention, developers may also add application behavior to Dots such that the user experience includes interactive graphical user interface elements, likened more to a custom desktop client application rather than to a web page. Thus, Dots enable Internet media to posses application behavior without the use of a browser or custom application.

From a Dot provider's perspective, Dots provide direct access to users; and a persistent branded presence on users' desktops beyond and without the browser, thus providing full control of the user's experience. That is, a Dot that is instantiated on a user's client computer has the user's display real estate. Conversely, a web page typically must be navigated to through the use of a web browser each time a user views the web page media, and once the user navigates to a new page, the display space (browser window) effectively belongs to the new web page content. As previously described, the branded experience may be wholly defined by the Dot developer, thus removing the rendering and branding limitations of a web page in a browser. Without the present invention, an alternative available to the Internet content developer is to develop a custom application that must be downloaded each time it is changed or alternate content is desired to be displayed.

In summary, Dots give users a unique way to experience web media by virtue of its: (1) flexible frame definition; and (2) a Dot's persistence when instantiated. Because of their ability to persist on a user's desktop apart from a browser, Dots also provide a convenient method to access content and services (without separate user selection of a URL). With a Dot for each of a user's specific Internet data needs, Dots eliminate the need to download custom software from multiple sites for various content. Rather than implementing custom client applications in Java, C++, etc., Dots are defined using XML and filled with standard Internet content such as HTML, GIFs, and the like.

The present invention also provides for the ability to collect and share favorite content and application-like behavior (i.e., sharing Dots). Links to Dots may be easily emailed between friends. Dots may be collected according to common interest, usage requirements or the like and defined as groups called Dot Packs. These Dot Packs may also be shared as links via email or server.

A fundamental aspect of the present invention is the creation of a Dot, including packaging of Internet content. The goal is to create an Application Media Packages (Dots) which combine the packaging, application behavior, and presentation of Internet content with the content itself (or typically a pointer thereto), as a single integrated unit (a Dot). Therefore, Dots are the integration of application and media.

This is fundamentally different than rendering internet content in a viewer application or alternatively, developing, distributing, and supporting a custom client-side application. Typically, a Dot definition includes details regarding a window "frame," and the Dot content has access to the entirety of this frame. The details regarding the frame is itself Internet content, so that the entire Dot "package" (including the definition of the package) is Internet content. Thus, the Internet content is not trapped in a third party viewer (e.g., Stock Trading site's web page in Microsoft's browser).

Dot content has access to application behaviors of the window frame (size, position, look and feel) and of the Dot application system (show other Dots, delete Dots, etc.) For example, a Dot may change color according to ongoing data updates that it receives over the Internet. A Dot designed to display weather information may become increasingly red as temperature data received for a particular city or region increases.

Furthermore, a Dot may communicate with other Dots. Communication between Dots is typically carried out by two or more Dots that are instantiated on the same client and enabled such that they may participate in inter-Dot communication. In one embodiment, communication is carried out by messages that are passed between Dots via the Home Dot that instantiated them. These messages may pass data, alter controls, or result in behavior change for example. Not only may Dots of the same kind communicate but in one example, Dots formed between different developers may also communicate and affect one another. Such communication may be apparent or may be a background function supporting some aspect of a Dot's functionality.

Dot and Dot Definition

With the above general description in mind, we now turn to a description of a Dot and its definition. With reference to FIG. **2**, the primary components of a Dot **100** are: Internet content **101**, a visual object(s) **102** within which the Internet Content **101** may be rendered, and operation element(s) **103**, which perform certain functions, each discussed further below.

A Dot **100** is an instantiation of a Dot Definition **104**, the components of which are illustrated in FIG. **3**. At its broadest, Dot Definition **104** comprises an XML (or similar) definition **105** (which may include initialization data, control calls, and a frame definition, discussed further below) and a Network Reference **108** (e.g., a URL) to Internet content **101**. (See, for example, Appendix A—DoDots XML Specification.)

XML definition **105** includes an appearance definition **106** for defining the appearance of a GUI (discussed further below), and a control definition **107** for defining controls associated with the GUI. Definitions **106** and **107** will typically include initialization data, control calls, and other elements. Optionally, a Dot may include message data **110** (e.g., access to the Dot Messaging Architecture). Application Media Package **104** (or "Dot Definition") may also include tags **112** to identify the Dot.

The XML definition **106** of the appearance of the GUI typically includes a frame definition, specifying size, location, etc. FIG. **4** is an illustration of an instantiation of a generic GUI **160**, which comprises a frame **162** typically divided into a number of control regions. For example, Frame **162** may include a title bar **164**, a bottom bar **166**, a menu control region **168**, and other definable control regions **170**, **172**, and **174**. One critical control region, referred to herein as the Web Conduit **176**, is a regions in which many Dots will present Internet content. Returning to FIG. **3**, definition **106** is responsible for providing the dimensions and general shape of the GUI.

The control definition **107** defines the layout (arrangement of the controls) and functional routines or pointers to functional routines (what the controls do when accessed by a user). Control definition **107** may also include XML Dot calls that are made by the Dot Definition and executed by the Home Dot (not shown here).

Network reference **108** will typically be a URL containing the address of a server having stored thereon data (Internet content) for retrieval and display within the GUI. Functionality typically provided by an associated Application Media Viewer controls the accessing of the server at the URL and the retrieval and processing of the Internet content for display. Importantly, retrieval and display of the Internet content may take place without resort to a Web browser application. Indeed, no Web browser application need be executing or even present on the client to support the Dot functionality (although embodiments in which Dot functionality is operational in conjunction with browser functionality is within the scope of the present invention). Internet content obtained from network reference **108** may be rendered within the frame defined by and according to the layout definition dictated by the frame appearance definition **106**. Internet content obtained from reference **108** may itself include XML calls providing certain functionality.

As previously mentioned, Dot Definition **104** is typically XML code. These definitions are quite simple to author, and to edit if needed. The definition is content, rather than compiled code, which provides additional flexibility at the client side should it be desired to modify the definition (e.g., allow a user to edit the network reference). As will be described further below, when an Application Media Viewer is operating on the client computer, the Dot is effectively both content and instructions. This is therefore a method of packaging internet content as an application (as compared to a hard coded custom client application) for operation on Internet content.

FIG. **5** illustrates a rendering or instantiation of a Dot Definition as a Dot **120**, in this case a visual indicator of likelihood of rain, expressed as a percentage **122**, and rendered against an image of a cloud **124**. According to the above, a definition for Dot **120** includes a definition of the size, configuration, and location of a first control space **126**, a specification of the layout and functionality of control interfaces (or simply controls) located in a second control space **130**. Examples of such controls include a "close" button, a "resize" handle, etc.

The image of a cloud **124** against which the likelihood of rain is rendered is a static image. The data representing the instructions for rendering the cloud may be found in the appearance definition **106**, and thus resides within the Dot Definition **104**. Alternatively, the data representing the image of the cloud **124** may be obtained from the location to which network reference **108** points. In either case, the data representing the cloud is utilized by the Application Media Viewer **104** to render the cloud image.

The actual data representing the likelihood of rain may be standard Internet content, and will change periodically as controlled by the source (e.g., the Internet content itself obtained from a server at network reference **108**) of the data. Since this data changes as controlled by the source, the data is

considered dynamic. This dynamic data is obtained by the Application Media Viewer from the source, and rendered atop the static image of the cloud. The frequency of updating and re-rendering of the dynamic data may be under the control of the Dot Definition or the source of the data, as a developer deems appropriate. The data to be rendered with the image may for example overlay, or be rendered in a layer on top of the image, beside the image, in the same layer or otherwise, in a layer under the image layer, etc. However, there is generally an intended relationship between the position of rendered Internet content and other items rendered in the graphical user interface. For this reason, we say that the Internet content is rendered "in association with" the image within the graphical user interface.

It will be appreciated that Dot **120** is merely on illustration of a Dot, and many other types of content, format, layout, controls, functionality, etc. are contemplated by the present invention. For example, additional display of information and/or control may also be presented in top/bottom bars **132**, **134**, respectively, as will be understood by one skilled in the art.

Thus, a Dot is defined using, for example, an XML file which is the embodiment of a Dot Definition **104**. The definition contains enough information that, when instantiated, the Dot may be rendered and filled with Internet content from a source. The Dot Definition contains data used to define and configure a frame and its elements, specify and lay out the controls, and specify parameters that initialize all the Dot's components with content and data.

Frame Definition

A frame can be defined to have a configuration (e.g., base shape), size (e.g., dimensions), and initial location (which may be changed by a user upon instantiation). In additional, a simple Dot may be defined to have four default functional frame definition controls, for example to encourage user interface consistency between different Dots. These four default functional controls include a title bar **132** (Gif rendered with title bar properties), a Dot menu **136** (with flexible menu entries), an exit control **128**, and a bottom bar **124** (Gif rendered with bottom bar properties) with corner elements **138**, **140** (for sizing and consistent user interface). A default layout of these functional components may be set, such as positions for the title bar **132** at the top of a Dot (Dot Menu **130** on its left, exit control **128** on its right) and the bottom bar **134** at the bottom of a Dot (with corner elements on either side). It will be appreciated that this is one form of layout, of which many others are within contemplated within the scope of the present invention.

Title Bar

The title bar **132** may contain a reference title for the Dot and may provide for a place for a user to grab and drag the Dot in a windowed environment. It may be implemented as a GIF rendering control that can be targeted to a local/remote title bar **132** image (an embodiment supports four title bar images—normal, mouse-down, mouse-over or hover, inactive). In this embodiment, the title bar **132** has a fixed height and width that is a function of the frame's width. The Dot Definition allows the title bar **132** image to be justified left, right, or center and for specified number of repeat-pixels, the title bar image may be tiled the full width of the Dot. Overlay text can also be specified to layer on top of the title bar **132** image.

The default size and the default position of the Dot are defined as part of the appearance definition **106** in the Dot Definition **104**. The Dot size can be specified in pixels, or Dot Units (1 Dot Unit is 40 pixels), etc. Dot Units can define a grid that keeps Dots sized on the same units so that they more easily and neatly align as well as cleanly snap to each other when Dots are used together. Dots can be sizeable or fixed-size in either dimension or both. The default screen position for a Dot can also be specified in pixels, or in relative position—top, left, center as provided for in the appearance definition **106** in the Dot Definition **104**.

Menus

One embodiment of the present invention reserves the four corners of a Dot for functional branding elements. The upper left corner is currently reserved for the menu control **136**. The upper right corner is currently reserved for the Dot exit **128** control. The lower corners **138**, **140** are for sizing the Dot if the Dot is resizable. Menu entries for branded menu control **136** (upper-left) can be specified in the Dot Definition **104**, for example at **112**. The menu items can be named, assigned images and tooltips, and assigned a specific and targeted action, for example a DotMessage **110** sent to a specified element (address). (See DMA messages hereinbelow). The Dot exit **128** button (upper-right) closes the Dot. An on-close action (such as a message and address pair) can be assigned to the Dot close event.

Bottom Bar

The bottom bar **134** may be implemented much like the title bar **132**, including supporting text overlay and additional functionality. (See Appendix A—DoDots XML Specification for a list of controls with XML call and their definitions)

Alternative Embodiments for Controls

Dots may be defined so as not to require a rectangular title bar **132** or bottom bar **134** (at top and bottom). Indeed, either or both of title bar **132** and bottom bar **134** may be omitted in a Dot Definition. Other controls (**128**, **130**, **138**, **140**) may also be omitted, provided certain functionality (e.g., exit or close) is otherwise provided. Furthermore, every functional element in a Dot may be defined as a control element that is relatively positioned and layered (in z order)—much like layers in HTML. Transparency may also be a given property. Different layers and controls may be given Dot properties (such as a title-bar-move-property or Dot-menu-property). Very flexible Dot interfaces may therefore be provided. (See the specification for the <DOT> and <CONTROL>DTDs in Appendix A—XML Specification.)

Control Space

Between the title bar **132** and the bottom bar **134** is a control space **126** where one or more controls, images, data, etc., can be flexibly positioned. For example, control space **126** may include a web rendering control referred to herein as a Web Conduit (described further below). The Dot framework supports any Active-X based control which may be positioned and initialized in a Dot (e.g., an on-line stock trading company may implement a custom stock chart-rendering control and define a Dot to permit a user to interface with this control).

A Dot may be defined to include a static image over which dynamic data may be displayed, such as the example of FIG. **3** in which a static image of a cloud has rendered thereover dynamic (changing) data relating to likelihood of rain. Alternatively, the Web Conduit may render in control space **126** purely static data obtained from the Dot Definition, a URL, or the client device, as well as dynamic data resulting from client-run processes, pushed or pulled Internet content, etc.

The Web Conduit control is just one of many controls that may be included and positioned in a Dot's control space **126**. For example, Active-X based control can also be referenced and inserted. This capability allows a Dot developer to implement a custom control. For example, a stock ticker display may be implemented as a custom secure chart control that renders stock tracking charts with small streams of secure

xml data. Such a custom control can be laid out with a Web Conduit control next to it if the Dot developer chooses to do so. Also, just like a browser, a Dot developer may embed an active-x control in an HTML page rendered in the Web Conduit control as is the practice for use with full screen browsers.

Web Conduit Control

The Web Conduit (mentioned above) control can render Internet content in a fashion similar to Microsoft's Internet Explorer™ (IE). This control functionality may be provided, for example, by utilizing Microsoft's Internet Explorer's (IE4+) WebBrowser object. Note that the IE4+ object does not constitute a browser. The Web Conduit merely uses rendering tools of the object—it does not invoke or require the Internet Explorer application or any other web browser application. Rather, functionality is provided to support rendering of Internet content so as to integrate this control with the Dot framework and to receive and transmit messages in and out of the HTML rendered in the control.

Dot Identification

A Dot can be identified (in the Dot Definition **104**) by three ID strings: GlobalID, Domain, and Kind reference. These are defined as follows:

GlobalID—A Dot's GlobalID is used when a Dot Definition is within a Share (described further below); this ID is unique with respect to other Dot tags in the Share.

Domain—A Dot's Domain is a unique label for the owning company Dot developer of the Dot.

Kind—A Dot's Kind (specified by the Dot developer) is a helpful identifier for finding the Dot; A Dot's Kind does not have to be unique.

Categories

In an embodiment of the present invention, categories are used as an organizational tool. A Dot Definition **104** (FIG. **3**) may be provided with an element **150** defining the category (ies) with which that Dot is associated. A mechanism is provided to allow identification of categories of Dots, useful for selecting, sorting, organizing, etc. The categories that a Dot belongs may be edited by editing the string elements in the Categories element **150** of the Dot Definition **104**. (See the specifications for the <ALL-CONFIG> and <DOT>DTDs in Appendix B.)

Events

A Dot Definition may also include an Events element **152** in which actions can be assigned to certain Dot Events such as the Dot's ONCLOSE event. DotEvents can be assigned a specific and targeted action much like a menu control **136** item; currently this action is implemented as a DotMessage sent to a specified recipient (see DMA messages, below). When the specified DotEvent occurs, for example ONCLOSE when the Dot closes, the specified message is sent to the specified recipient.

Hosting and Serving Dots

Referring again to FIG. **1**, web server **50** may serve the Dot Definitions **104** as an XML file to client computer **20**. The Dot Definition **104** may be served elsewhere such as a third party or partner server (not shown) along with the Dot content that fills in the Dot. In one embodiment, this XML specification may be kept proprietary and Dot developers define and package Dots indirectly without having to author XML Dot Definitions. In another embodiment, the xml specification is open, and content providers (and others) have complete control of the authoring of Dots.

Dot Definitions **104** are indexed and stored in server **50** in database **74**, and are accessible to Dot-rendering and Dot-management applications (the Application Media Viewer) via "DotLinks". These DotLinks, based on standard Internet links, point to particular Dot Definitions and enable the Application Media Viewer (using standard HTTP/HTTPS requests) to quickly access the frame and initialization pointers or URLs in the Dot Definition to instantiate the Dot and render it, filling with DotContent (Internet content served by a partner Dot Developer servers **82**).

Dot developers (e.g., content and/or service providers) serve the Dot content obtained from source or reference **108** (the standard internet content that fills in the Dot). According to one embodiment, Dots may be served my numerous different non-proprietary servers **50**. Content may also be provided by a non-proprietary server, as specified by the Dot developer. Of course, it is possible that in other embodiments specific Dots may only be obtained from certain sources.

Because the Dot Definition **104** contains enough information to instantiate the Dot as well as the reference **108** that address the location at which the Dot content is located, a Dot is easily and quickly distributed, as well as collected and shared by users. (This is discussed further herein with regard to Viral Distribution Architecture.) It is therefore easier and faster to get information delivered in a Dot than in a web page.

This Dot-based architecture is very different than the current model of Internet content. Today, users visit web sites, following a destination-based model of content access. The process of accessing media therefore is sequential or linear, resulting in user viewing of one full-screen web page at a time. Furthermore, there is no simple to use or inherent mechanism in place today to share sites short of sending URL's.

Enhanced Dot Content

The Dot Definition **104** code is accessible, flexible and dynamic, enabling it to be modified at any time, even after it is rendered. For example, Internet content **101** (HTML) can access and modify its associated Dot Definition **104** by sending messages to other portions of the definition, such as the definitions of appearance **106** or control **107**. Other application system-level functionality is also available by sending messages to the Application Media Viewer **199**, discussed further below.

By using an architecture, referred to as Dot Messaging Architecture, for communication between Dots, and between a Dot and the Home Dot, standard Internet content can access and enable Dots to exhibit application behavior. That is, content can provide functionality typically associated with applications, such as dynamic refresh, contextual presentation, interactive response between user actions and changes in the graphical user interface or frame, etc. (See "Dot Messaging Architecture" below.)

EXAMPLES

The following examples are used to demonstrate key features of the present invention. Some of these features include: a new way to view standard internet content, a fully branded experience for the user, and a web experience that delivers true application behavior. Sample companies are used in these examples to demonstrate branding together with the use of internet content to produce a novel user experience that transcends the web page. Trademarks are those of the respective companies.

E*Trade

E*Trade's primary business is to enable users to execute securities trading online. Therefore stock information and notification are essential elements to their business, as are the transactions themselves. Persistent display is an important aspect of Dots as is the ability to provide ongoing data updates. Below is a list of aspects that illustrate novel advantages that Dots provide for a possible E*Trade Dots system.

E*Trade can leverage their existing content through Dots (Dots are built on top of standard HTML]

E*Trade can break their content into smaller pieces and package them into Dots.

E*Trade can give their Dots application behavior. Instead of being trapped inside a viewer window, content providers have access to the entire frame, which enables them to brand the Dot, access to application and system features. Content providers can also enable users to resize the Dot click, to open E*Trade Dot, ability to execute operations in the Dot such as trading stocks as opposed to only monitoring stock prices.

Because Dots can be left on or actively running and displayed on a client computer, they can notify users of coming/pending/immediate content events. A Dot can notify a user in many ways, including: popping up a message box or another Dot. A Dot can resize itself and show new content and/or bring itself into focus, as a window application may also do.

Merriam-Webster

In this example, Miriam-Webster has word definition content that is useful for users to access. Users may be better served by the Dot format of presenting as opposed to traditional web pages. Below is a list of features that illustrate this point.

Miriam-Webster can leverage their existing content to provide a dictionary "application" in a Dot; the content is HTML delivered from their servers. In a Dot, Miriam-Webster can provide direct access to just the information users are interested in viewing without requiring them to disengage in other operations such as word processing work in order to actively seek information from Miriam-Webster's servers. Additionally, content providers can add new functionality to their Dot applications just by changing the HTML files on their servers that are delivered to Dots or as Dot Definitions **104**. They do not need to revise and redistribute custom client software.

eBay

A single company such as eBay can use multiple Dots to engage users from many points at the same time to facilitate a more complex process, such as online auctions or shopping. Many companies have an enormous amount of assets (content, services, and applications) that can only be exposed to users one page at a time. Featuring and exposing new or important functionality is often difficult to do if they are buried several pages deep.

Dots enable content providers to feature and expose functionality and services in a new form, leveraging and featuring all of a company's assets in a consistent way. eBay, for example, might provide a "MyBids" Dot that allows users to more closely follow those items they have bid on (the current asking price and when the auction will expire).

eBay can also provide a Gallery Dot. Instead of wading through thousands of thumbnails in the gallery section to find what their looking for, users can identify categories they are interested in and window shop these items, click on a particular item to add new item to MyBids Dot. When a user finds something that interests him, it's added to the MyBids Dot with a single click.

There is a messaging architecture (DMA) that may be enabled by the Dots architecture disclosed herein which enables Dots to communicate with each other, thus facilitating operations between Dots without user intervention. For example, a user finds an item of interest in a "Gallery" Dot. He clicks on that item and it is added to the "MyBids" Dot with a single click. Additionally, a "BidDot" may be caused to pop up for that single item, allowing a user to directly bid and monitor the auction process. He clicks on a second item in the "MyBids" Dot and another "BidDot" pops up.

MP3 Dot

By using an embedded MP3 playing control, an MP3 playing Dot can access, play, and manipulate MP3 files and playlists from a local hard drive or the web, just like a custom MP3 client application. The significant difference is that the GUI & controls (the Dot Definition **104**) are entirely Internet content (HTML, javascript, etc.) and can change simply by modifying HTML files online.

Mobile/Other Platforms

Dots can be viewed as a smaller-format package for internet application media. This package is more portable than executable applications as well as full screen formatted media for browsers. Portability across platforms requires consideration of diverse screen sizes and resolutions as well as operating systems and user interface controls. This diversity requires companies or users who require cross platform media distribution to scale down their existing content for portability to mobile devices. No such scaling down is required when deploying Dots and the Dot architecture.

Application Media Viewer (Home Dot)

As previously mentioned with regard to FIG. **1**, a client computer **20** will ideally have stored thereon a software component referred to as an Application Media Viewer (or Home Dot) **119**. As Application Media Viewer **119** performs a number of functions in the process of acquiring, instantiating, modifying, etc. Dots, in addition to simply viewing Dot content, we also refer to an Application Media Viewer as Home Dot herein. The Home Dot **119** is effectively a client application that contains the procedures or calls to procedures for rendering and managing Dots on the client computer. It therefore has attributes of an application. The Home Dot is designed to operate in conjunction with a Dot Definition, and vice versa.

Data contained within an Application Media Package (Dot) is parsed by the Home Dot. Control calls contained within the Dot are also parsed and executed. The Home Dot may then instantiate a Dot without relying on functionality provided by a browser. In fact, in a preferred embodiment, a browser is not employed to instantiate or operate an Application Media Package.

The Home Dot is a compiled application, and can be programmed using, for example, MFC (Microsoft Foundation Class) or the like. The Home Dot need only be downloaded to the client computer one time and may be used to instantiate any number of Dots. The Home Dot is capable of executing and displaying multiple Dots simultaneously.

The Home Dot may be provided with the ability to automatically or manually check for updated versions of itself on a server. If a newer version is detected, the user may be provided with the opportunity to download and install the updated version of the Home Dot.

The Home Dot supports a number of functionalities, including support for standard internet protocols (http, https, etc.) Additionally, the Home Dot supports collecting and organizing Dots, user login to a Dot server, user activity statistics collecting and reporting, and messaging between Dots, as discussed further below.

Network Distribution of Dot and Home Dot

Referring now to FIG. **6**, a communication flow diagram **299** is shown for an exemplary system enabling the distribution and use of Dots between a client computer **199**, a server (Dot server) **153** making available one or more Dots including a Home Dot, and content server **155**, each with a network

address or IP address. It will be assumed for this particular example that a Home Dot **200** has previously been downloaded and installed on client computer **199**. It will also be assumed for this example that Dots **120** and **230** have been previously requested (the mechanism for this request is described further below).

Accordingly, Home Dot client application **200** utilizes Dot Definitions to instantiate Dots **120**, **230**. There may be ongoing communication **250**, **260** between Home Dot **200** and Dots **120**, **230**, respectively. This communication may include data for the presentation of Internet content, messages, and/or state information. Furthermore, the Home Dot client application **200** executing on a client computer **199** communicates with Dot server **153** through a channel **210** (such as a broadband Internet connection) as well as Content server **155** through channel **157**. Internet content is received by Home Dot **200** and rendered into the instantiated Dots **120**, **230**, as appropriate. The Internet content may be one or more of many formats, such as XML, HTML, GIF, Streaming Media, Flash, HTTP, HHTP(S), etc.

The Dot server **153** is communicatively connected to a physical memory device **201** which holds a database **202** containing software objects for downloading to or access by a client device such as client computer **199**. This physical memory device **201** may be a RAID hard drive system, a standard hard drive, removable media, or any other type of volatile or non-volatile memory known in the art. Database **202** may contain one or more Dot Definitions **104**, the Home Dot client application **200**, available for download to the client device, as well as user account data **205**, state data including use statistics **206** and Dot index/shares **204** (each described in further detail herein).

The content server **155** is communicatively connected to a physical memory device **261** which holds the Internet content **262** as well as other forms of data **263**. Content/data **262**, **263** is communicated to the instantiated Dots **120**, **230** where it is rendered on the client computer **199**. In general, content passes first through the Home Dot application **200** or a similar client computer **199** program that is capable of receiving and parsing Internet content. Therefore, the connections between content server **155** and Dots **120**, **230** are shown as dashed lines, indicating that for this figure the connection is indirect.

Unlike a downloadable custom client application, the Dot Definition **104** is comprised entirely of Internet content in a web browser readable language. To instantiate a Dot on a client device or computer, only its definition is required (assuming that the computer is Dot-enabled, i.e., that the Home Dot client application **119** has been installed). The Dot Definition **104** contains just enough information to define, layout, and initialize a Dot's components (frame, controls, etc.); this information configures the graphical user interface which may then present Internet content therein. Consequently, a Dot Definition typically has a small file size (~2 KB), and is therefore a quickly accessible and loaded XML file.

One aspect of software distribution according to the present invention may proceed as illustrated in FIG. **7**. A user first requests a Dot, or more specifically its Dot Definition **104**, by clicking on a Dot link **321** which may be a hyperlink on the web page of a Dot server **153**, a web page of a partner or content server **155**, or selectable menu item on the client computer **199**. A Dot link **321** is an Internet link to a Dot Definition **104**. A Dot link operates in a manner similar to an HTML link. One added feature associated with a Dot link is that the server that serves the Dot Definition **104** typically has a sensing mechanism that can determine whether or not a user has the Home Dot application **200** installed on his computer or device **199**.

If the user does not have the Home Dot application **200**, then it is downloaded and installed with a first set of Dots (e.g., **120**, **230**) that may be user-selected or part of an initialization set. This download process is discussed in further detail below. From that point forward, the client computer **199** is Dot-enabled and does not require a subsequent download and install of the Home Dot. In short, only a single application-like software product need be downloaded and installed on a client device to enable a variety of different Dots, as opposed to requiring a user to download a different custom client application for different types of Internet content.

Once a client machine **199** is Dot enabled, (Home Dot application **200** installed), Dot **120** may be rendered by simply clicking on Dot link **321**. This rendering is typically faster than it takes to load and render a small web page. Furthermore, once the Home Dot application is installed on the client no additional executable software need be downloaded or install to view and interact with a Dot. The Home Dot application **200** accomplishes this by retrieving the Dot Definition **104** specified by the Dot Link **321**, instantiating the Dot **120**, obtaining the Internet content specified by the Dot Definition **104**, and presenting the Internet content (standard Internet content from DoDots' partner servers **155**) within the Dot.

Once a user has received a Dot Definition **104**, the Home Dot application **200** remembers the Dot's Dot Definition **104** as part of a user-profile **310** so that it can instantiate the Dot and begin filling it with Internet content immediately. Because a Dot Definition **104** becomes part of a user-profile **310**, it can be modified by use (e.g., the image of its default title bar **125** may be changed, its Web conduit control may be navigated to a different URL using DMA messages, etc.) and the Dot will initialize from its last state stored in user profile **310** the next time the Dot is to be instantiated (or "turned on") at **324**.

A Dot Link **321** represents a Dot. Therefore, this Dot Link **321** (and hence the Dot **120**) can be easily and instantly distributed; the Dot Link **321** can be posted on web sites to promote the Dot, or shared with friends via email. This is viral distribution of a Dot, via distribution of the Dot Link **321**—the Dot server hosts and serves the Dot Definitions **104** to which the Dot Links **321** point. Dots may also be aggregated into packs by content providers, or other aggregator, grouped for example by like subject or perceived common user interest, and the packs of links acquired by a user at **322** as if they were single Dots. A user may also provide others with access to Dots when the actual Dot Definition is located on a remote third party Dot server. This may be accomplished by way of a Sharelink, which may be provided by a user at **302**. Others may then access the Dot, including setting changes that may have been made by the user, by accessing the Sharelink provided at **320**.

From the client perspective, one example of a process of acquiring and instantiating a Dot is illustrated in FIG. **8**. According to this process, a Home Dot is employed to acquire a new Dot, however it is within the scope of the present invention to acquire new Dots through other mechanisms, such as through a web browser application (with utilization of the Home Dot ultimately required in order to instantiate and populate the Dot). After logging in at step **240** the user clicks on the Dot Link signaling a request to acquire the Dot. The Home Dot application then sends a request to download the Dot to the Dot server **153** at step **241**. Alternatively, the user may click on a Dot link before logging in, for example if the link has been e-mailed to the user, and then, after clicking on the link, log in. In another aspect, the user could view, but not collect, a transient Dot without ever logging in.

After the user is logged in and has clicked on the Dot, the Dot server **153** retrieves the Dot template from its Dot database, and provides it to the client in step **243**. The client receives the Dot template from the Dot server **153** in step **244**.

For display of the new Dot, the Home Dot creates a frame in the display of the user interface (**42**, FIG. **1**) in step **245**. In step **246**, the client **199** requests the necessary Internet content stored at the URL(s) identified in the Dot Definition from the corresponding content server **155**. The content server **155** transmits the content in step **247**, and in step **248** the Home Dot presents the content in the viewer. As the type of content may vary greatly, the content may appear within the frame (such as text or images) or may be independent of the frame (such as audio) but the frame including controls for controlling aspects of the presentation of the content. In one embodiment, Internet content is presented such that it is enclosed by the frame, allowing the user to preview the Dot. Alternatively, the Home Dot may simply collect the Dot, adding it to the user's processed user profile **310**.

The user may then view the Dot on the user interface display **42**, and may interact with the Dot much in the same way as a user may interact with typical Internet content or web applications. This may change the Dot from its present, "raw" state to a used state reflecting alteration or use of the Dot by the user. Thus, the user has stored on client **199** a Dot Definition rather than a Dot template. For example, the user may direct the Dot to display different content within the Dot if the Dot content enables the user to do so. Or, the user may provide information to the content server **155** which allows the Dot to be personalized. The user may additionally be given the option of changing the size or location of the frame, etc.

If the user collects the Dot, the Dot will be added to a user's list of collected Dots. For example, the Home Dot may add the Dot's definition to a processed user profile (discussed further below). Thus, the Dot's "state" will be preserved. Alternatively, the Home Dot may collect the Dot automatically, without waiting for a user command, by adding the Dot Definition directly to the processed user profile.

If the Dot's state has been altered by the user or by the content—if, for example, the user has directed the Dot to Internet content other than the initially-displayed content, provided personalizing information, or changed the properties of the frame, or if the content itself has caused an alteration in the Dot—this alteration will be reflected in the Dot Definition stored in the user profile. Information which personalizes the resulting content, instead of being stored in a "cookie" on the client device, can be stored as part of the Dot Definition. This advantageously permits personalization of content, such as Internet content that is associated with the Dot content and the user, without storing a cookie on the client **199**. One advantage this provides is that the state of a Dot can be returned for a user no matter which computer the user accesses the dot from.

A user may also access a Dot which has been previously collected, and possibly altered by use as explained above. As previously described, the user profile includes Dot Definitions for Dots which have been viewed and collected by the user. A screen shot showing Dots **502**, which have been previously collected by a user is shown in FIG. **9**. Also shown are Dot Links **504** available from a Dot server for download to the client.

One embodiment of the steps taken to provide the user with Dots which have been previously collected are shown in FIG. **10**. According to this process, acquired Dot Definitions are maintained on a central server, and a user profile identifies which Dot Definitions are associated with particular users. This is useful when a user may be using more than one computer and wishes to have access to her collection of Dots on any machine she is using. However, it is equally within the scope of the present invention that Dot Definitions may remain resident on a user's computer and not be stored for that user on a central server or the like.

As discussed above, on login at step **250** the user's profile is retrieved by the Home Dot stored on the client (step **251** and **252**). The user's profile, stored in the user profile database, includes the Dot Definition for each of the Dots previously collected, and possibly altered, by each user. The Dot Definition, as discussed above, includes the Dot frame definition and the definition of the controls for filling the viewer within the frame with content. After log in, a local copy of the processed user profile is stored on the client **199**, and this copy is further processed as the user collects new Dots, or uses new or collected Dots such that the Dots are altered.

When the user clicks on the name or icon of a collected Dot at step **253**, the Home Dot creates a frame in the display of the user interface (**40**, FIG. **1**) in step **254**. At step **255**, the Home Dot causes the client **199** to request the Internet content from the URL identified in the Dot Definition from the corresponding content server **155**. This content is provided in step **256**. It will be appreciated that the URL need not be the same as the initialization URL in the Dot template stored in the Dot template database **202** on Dot server **153**, and in fact the content server need not be the same content server corresponding to the initialization URL. In step **257**, the Home Dot places the content in the Dot frame, and the Dot is then fully instantiated with content.

Hosting Dot Definitions

Dot web servers **153** host and serve the XML Dot Definitions **104** and provide the Dot Links **321**, Dot Packs **322**, and sharelinks **320** that point to the Dots so that a user can easily and instantly add the Dot **104** to their Home Dot application **200** (adds Dot **104** to their user-profile **310**) simply by clicking on the Dot Link **321**, pack link **322**, or sharelink **320**. The Home Dot application **200** registers with the local browser/computer so that Dot Links **321** are handled by the Home Dot application **200** (and not by the browser).

Dot Templates and the Dot Index

New Dot Definitions **104** get published to a Dot index **204**. A Dot Definition can be modified by a user once downloaded. Since these new Dot Definitions are resident on the server for downloading, they are also referred to herein as Dot templates. The Dot index stores Dots definitions or templates **104** in database **202** on Dot server **153** by category (such as sports, finance, games, etc.) with descriptions and images for each Dot. In the preferred embodiment, Dot Definitions **104** are published to categories specified by the Dot developer. Categories can also be automatically created to support indexing Dots that are most popular (most commonly accessed, most commonly shared, or other sorting criteria).

Dots as well as their index are formatted as Internet content. Therefore, users can browse or search the Dot index for new Dots using either a browser or using their Home Dot application **200** application (via an AddNewDot Dot or like functionality).

Each of these Dot templates (Dots definitions **104**) are pointed to by a Dot link **321**; the Dot link **321** for each Dot is generated automatically by the server **153** (when the Dot is published or previewed) to reference that Dot. Clicking this Dot link **321** adds the Dots definition **104** to the user's Home Dot application **200** (via the user profile **310**) and turns the Dot on (instantiates it and fills it in with internet Dot content).

Once a user receives a new Dot (Dots definition **104**) by clicking on a Dot link **321**, the Dot template is downloaded

and becomes a Dot Definition **104** forming part of their user profile **310** (the XML data that defines which Dots the user has, which Dots they left open, what the state of the Home Dot application **200** last was, etc.).

The Dot server **153** may optionally host Dot Definitions **104** as part of user profiles **310**. Every Home Dot application **200** user has a user profile (expressed in XML) that stores the Dot Definitions **104** that the user has collected as well as the last state of the Home Dot application **200**. (See also the specification of the <ALL-CONFIG> DTD in Appendix B.)

In an additional optional embodiment, when a user successfully logs into the Home Dot application **200**, state restoration may be provided by Dot server **153**. In this embodiment, authentication may be handled by the Dot application server **153** (as opposed to the user device and Home Dot). The Home Dot application **200** requests the user profile's <ALL-CONFIG> file **311** by communicating with Dot server **153** at **303**. The Home Dot application **200** then receives the <ALL-CONFIG> file **311** from Dot server **153** at **303**, and restores its state (from the <ALL-CONFIG> **311**) presenting last user state, i.e., which Dots were ON, where the Home Dot application **200** was positioned on the screen, etc.)

The Home Dot application user interface (UI) enables a user to turn a Dot ON **324**; in this case the Home Dot application **200** has the Dot Definition **104** (part of the user profile **310** that was retrieved on login) that contains enough initialization data to instantiate the Dot and fill it with content just as the user left it.

As previously mentioned, Dot Definitions **104** that are part of a user's profile **310** may differ with use. In other words, the Dot Template from which they were originally created may have a different state than a Dot Definition **104** that has been modified through use. The user profile could also be implemented to point to Dot Definitions **104** that are always hosted remotely (and/or not entirely part of the user profile).

Centralization of Dots within the Home Dot **200** enables users to collect Dots. This feature is significantly different than today's model of the Internet in which users visit a page one at a time, and then leave. Users collect Dots, keep them running, and share them with others. By packaging Internet content as a Dot and referencing it by a Dot link **321**, Internet content is given viral characteristics (i.e., Dots can be instantly distributed). (See also the Session_Config Example found in Appendix B.)

Shares

Dot servers **153** also host Dot Definitions **104** as part of Shares. Because a Dot link **321** represents a Dot, this Dot link **321** (and hence the Dot) can be easily shared (distributed via email) from one user to other users.

If a user receives a Home Dot application share and has the Home Dot application **200** installed (their machine is Dot enabled) then clicking on the Share Link Dot link **320** in the share will add the Dot(s) in the share to the user's Home Dot application **200**. If a recipient of a Share Link doesn't have the Home Dot application **200** installed, then the Home Dot application **200** is downloaded and installed (with the user's cooperation) with the first Dot(s).

When a user shares Dots, their Home Dot application **200** generates a share file (XML file) that contains the Dot Definitions **104** of the Dot(s) included in that share. The Share XML is then sent to Dot server **153**; the Dot server **153** automatically generates the Share Link **320** that references the Share XML. This Share Link **320** rather than the Share XML is sent or distributed (via email or posted on a web site) to other users. (See also the specification of the <SHARE> DTD found in Appendix B.)

Software Product Download Process to Client

Reference is now made to FIG. **11**, which is a flow chart **400** illustrating the software product client download process. According to one embodiment, to begin, a user makes a request for a Dot at step **402**. At step **402** it is determined that the user does not have the Home Dot application **200** installed. Step **406** comprises of the following:

User is informed of need to download software to view Dot. The message could be from the site owner (content provider) or from a Dot server.

"Do you trust [provider/Dot server host]?" dialog is presented to user at **405**. A dialog then tells the user the size of download and approximate time for download.

In step **408**, the user clicks the "Yes" button. (Alternatively, if the user clicks the "No" button, step **409** executes with an exit message such as "You decided not to get the Dot"). Proceeding now to step **410**, the user gets a Trust Certificate (a security process well understood in the art). The user accepts the Trust Certificate in step **412**. (Alternatively, if the user declines, step **413** executes with an exit message such as "You decided not to get the Dot"). In step **414**, the user accepts and receives the license agreement and is presented with a dialog that asks the user to confirm the directory for download location. (Alternatively, if the user declines, step **417** executes with an exit message such as "You decided not to get the Dot"). In step **416**, the user accepts the license and confirms the file location, and proceeds now to step **418**. The download process begins, followed by the installer download and background install operation. The requested Dot opens, and the "KeepMe" Dot opens. Note that the "KeepMe" Dot may be provided by a third party such as a content provider. If no third party-provided "KeepMe" Dot exists, a default "KeepMe" will open.

In step **420**, if the user decides to keep the Dot, the KeepMe (Dot) Operation initiates (see below). If the user decides to close the requested Dot before selecting "KeepMe" and then decides to select "KeepMe" the requested Dot will close, the Keep Dot operation will initiate and the requested Dot will be added to the Dot list. Alternatively, if the user closes the requested Dot and the "KeepMe" Dot without selecting "KeepMe" the Home Dot application **200** will remain installed but the Home Dot application **200** UI will not be available to the user. Lastly, if the user decides to close the "KeepMe" Dot before closing the requested Dot the user will not be able to initiate the Keep Dot operation.

The other branches of the flow chart **400** will now be described. Beginning once again with step **402**, the user requests Dot. According to the alternate branch leaving step **404**, the Home Dot application **200** is found on user's machine. In step **405**, it is indicated that the user is not registered. (Alternatively, if the user is registered, the Add Dot operation is executed step **407** and the process terminates). Proceeding therefore to step **411**, the requested Dot opens. The "KeepMe" Dot opens ("KeepMe" Dot may be a default Dot or a "KeepMe" Dot provided by the Dot/content provider). If the user decides to keep the Dot, the Keep Dot operation then initiates (see below). Alternatively, if the user decides to close the requested Dot before selecting "KeepMe" and then decides to select "KeepMe" the requested Dot will close, the Keep Dot operation will initiate and the requested Dot will be added to the Dot list. In another scenario, if the user closes the requested Dot and the "KeepMe" Dot without selecting "KeepMe" the Home Dot will remain installed but the Home Dot UI will not be available to the user. If the user decides to close the "KeepMe" Dot before closing the requested Dot the user will not be able to initiate the "KeepMe" operation.

The following is a summary of the "KeepMe" registration operation (assuming an unregistered user has a partner's Dot open):

User clicks "KeepMe" button of "KeepMe" Dot.

User registration form launches.

User form displays in "KeepMe" Dot.

User form includes link to privacy statement.

User completes user registration form and submits.

Home Dot appears.

Welcome Dot appears.

In one embodiment of the present invention, it is possible to track the referring partner, for example for awarding incentives for referring a user. The form of the incentive is a matter of business choice. However, in such an embodiment, the registration operation will comprise the additional step of:

Crediting the referring party (e.g., partner) with the referral and/or converting user.

For an alternative software download process, refer to U.S. Provisional

Patent Application Ser. No. 60/176,687, Appendix F—Do-Dots Feature Priority List PPA.

It will be appreciated that the present invention addresses scaling issues by breaking content up into smaller, more focused software components. These smaller software components (Dot Definitions **104**) may thus be served to mobile devices to compensate for bandwidth and content feature support issues, to overcome a lack of browser functionality and processing overhead, etc. Furthermore, the architecture supporting Application Media Packages or Dots offer a consistent experience with application media or Dot content **104** across device types, e.g., similar experiences as between desktop, laptop, web-enabled phone, PDA, etc. As Dots are distributed as easily as web pages are viewed/visited, Dots can be instantly distributed and users can stay connected with content providers of their choice without having to download custom client software from each provider, and without regard to the device type being operated.

For further details on the structure of the XML structure used for Dots according to one embodiment of the present invention, see Appendix A hereof. For further details on the functions and design of Dot server **153**, see Appendix B hereof.

Application Media Viewer—the Home Dot

One unique aspect of the present invention is the interaction between the application media package **104** and the application media viewer **119**. This aspect is now discussed.

As mentioned, the application media viewer, or Home Dot, is a network enabled client application. With respect to the user experience, the Home Dot provides the parsing and rendering function of the application media packages or Dots. It also provides for the application behavior of Dots by executing calls or methods that are parsed from the Dot by the Dot Definition, user events, system events, or the like. That is, at a minimum, a Dot comprises a definition of a graphical user interface (frame) and content to be rendered within or associated with that frame. Thus, a critical function of the Home Dot is to parse the Dot definition, render the frame, obtain the content, and render the content in or associated with the frame.

As previously discussed, the Home Dot comprises Internet content. Thus, the Home Dot is in part a content parser, providing rendering of the graphical user interface (GUI) from Internet content.

Furthermore, Dots originate on remote network devices. Thus, another function of the Home Dot (client-side) is to obtain and organize Dots on the computer on which the Home Dot resides.

The Home Dot is a part of a client/server system. As a client installed in one of a multitude of client computers, all communicating with a Dot server, the Home Dot maintains communication with that server to perform system functions for the client. These system functions require a defined command set or functions calls between the client and server. Such functions may include user login, user account status, use statistics, Dot downloads, individual Dot configurations or user customizations, Dot organization, revision updates for both Dot and Home Dot installations.

Because the present invention enables versatility in the manner of packaging and operating with Internet media (forming application media packages), and because the breadth of internet media and the scope of available information is expansive, the present invention also provides for methods of Dot organization on a user's client computer. More specifically, the Home Dot provides for Dot management for a large number of Dots. Such methods of management include grouping Dots into an organized layout for persistent display, categorizing and grouping Dots into Dot Packs, opening and closing sets of Dots or Dot Packs according to a user's current information requirements, and configuring individual Dots either by user defined categorization or customization parameters that have been enabled by a Dot's developer.

The Home Dot according to the present invention provides for methods for grouping Dots, sharing information between Dots, and sharing these groupings with others. In other words, the present invention provides for methods by which a user or other third party or collectively, third parties, may build upon the utility of Dots as an atomic media element by adding their context through grouping and configuration. In doing so, the present invention enables the addition of third party knowledge that may or may not include the participation of the original Dot developers. Furthermore, the present invention provides for mechanisms by which this knowledge is shared. These mechanisms include saving the customization states of individual Dots, grouping of a plurality of Dots and saving them to a Dot server, and emailing links to other users or installing these links in a web page. Additionally, the present invention provides for third parties to share collaborative efforts by posting such groupings and customizations on a Dot server for general availability.

In addition to grouping and customization, sharing information and events between Dots is provided for by the present invention. The Home Dot routes messages between Dots and the system. Messaging enables Dots to act upon one another such that an action or event that affects one Dot may also be routed to another Dot, thus affecting a change to it or an action to be taken by it. The resulting actions or changes of the individual Dots may not be the same. To support messaging generally, levels of restrictions are also provided for. It is therefore possible to control the scope of messages received by a Dot from other Dots according to membership or domains. Likewise, it is possible to control transmission of messages to other Dots. Therefore, the present invention provides for cooperative messaging and information sharing between Dots as enabled by Dot developers.

In one embodiment, the server is a high availability system comprising a plurality of individual servers functioning together on a server network. Such a configuration advantageously services a large number of simultaneously executing client Home Dots. Whereas the client application or Home Dot provides for those services surrounding enabled or instantiated Dots, the server network and its applications as embodied in a web site, Java servlets, an RDBMS, Dot and Home Dot distribution support, provides for the development

and distribution of Dots. Additionally, as the Home Dot performs the optional features of collecting statistics surrounding Dot usage and posting these statistics to the server, the server may aggregate these statistics for reporting. According to one aspect of the invention, the server operating together with the Home Dot, may source or enable messages to specific Dots based upon real-time context of user interaction with Dots. This function enables, for example, real-time context based advertising. (See also the Client-Server Protocols section of Appendix B.)

Method of Delivering Dynamic Web Data without Web Browser

With reference again to FIG. **6**, according to one embodiment of the present invention, Home Dot application **200** enables users to collect Dots **120**, **230**, etc., organize those Dots into collections, manage the collections with organizational and workspace management functionality (e.g., categories, Snapshots, group-move), and easily share Dots, Dotpacks, and Snapshots with others. Also, the Home Dot application **200** implements and enables a Dot Messaging Architecture (DMA, discussed further below) providing messaging between Dots and the Dot server system.

When Dots are instantiated on a client machine, the Home Dot application **200** collects usage statistics (**211**, **212**, **213**, **214**, **215**) and keeps an event log (in XML format) that is intermittently sent to Dot application servers **153** and stored, for example as part of the Dot database **202** as use statistics **206**. This enables the Dot server to track how the Home Dot application **200** and the Dots are being used and shared. In one embodiment this is implemented as a local MFC (Microsoft Foundation Class) application on the Windows platform, thus enabling rapid user response (no Internet delay for functional UI components). Home Dot application **200** functionality may also be provided across platforms (MAC O/S, Linux, unix, mobile, wap, etc.)

Home Dot Operation

Each user of a Home Dot application **200** has a Home Dot application login account **205** that accesses (step **303** of FIG. **7**) a personal user profile stored on a Dot server **153** that stores the last state **212** of the Home Dot application **200** along with which Dots a user has collected **213**, and how the user has organized their collection according to categories, Snapshots or packs.

When the Home Dot application **200** is launched, it queries the network for configuration information, enables a user to login, retrieves the user's profile from the Dot application server **153**, and restores the Home Dot application **200** to the last state that was stored to the application server **153**.

The method used by Home Dot **200** to access remote configuration information (again, step **303** of FIG. **7**) creates a very flexible application that can be configured to support different application looks, different login sequences, dynamically by session and for each for each user. The user profile retrieved at login **303** enables the application to be personalized for a user, and allows that user to access that personalized application state on different client computers or devices **199**.

Launching the Home Dot Application

When the Home Dot application **200** is launched, it reads the registry on the client machine to find the location of a configuration file (session-config). The session-config (XML file) directs the Home Dot application **200** to an application server using a URL as the server address that will handle its servlet requests. The URL points the Home Dot application **200** to a default graphic element (skin) and generic Dot graphic elements for controls such as the Title Bar **164** (FIG. **4**), Bottom bar **166** (FIG. **4**), etc. and it points to Dot definitions **213** for several default Dots, such as a Help Dot, Add New Dot, and Login Dot (i.e., System Dots that support the Home Dot application **200**, as described further below).

For security, the session-config is served by a servlet method operating on the Dot application server **153** that only responds to authorized Home Dot applications **200**. The session-config can also be served as a flat file from a standard web/local directory.

The session-config file contains a mechanism to redirect the Home Dot application **200** to another session-config file. This mechanism may look like:

<REDIRECT>="URL" attribute:

When the Home Dot application **200** reads the redirect attribute, it ignores the current configuration (session-config file) and attempts to retrieve the configuration stored at the redirect URL. The Home Dot application **200** will not redirect if this tag is omitted, if the URL is empty, or if the URL is the same URL used to retrieve this configuration in the first place. (This last state is recursive; therefore if redirection were carried out, the two configuration files redirect to each other, resulting in an endless loop within the Home Dot application **200**.)

The session-config file contains a mechanism to force or provide an option to upgrade the Home Dot application. This mechanism employs a version tracking, such as:

<VERSION>="string" attribute:

This attribute is required and must match the version of the Home Dot application **200**. According to one embodiment, if this attribute is omitted or differs from the Home Dot application's **200** version, the Home Dot application **200** will download the executable specified in the UPGRADE attribute and execute it. The mechanism for performing the upgrade may look something like the following:

<UPGRADE>="URL" attribute:

This is used only if the VERSION attribute differs from the Home Dot application's **200** version. This URL specifies the location of an executable that will upgrade the Home Dot application. The Home Dot application **200** will quit, then run the upgrade executable automatically.

System Dots

According to one embodiment of the present invention three system-level Dots are provided with a Home Dot: the AddNew Dot; the Help Dot; and the Login Dot. Dot definitions for these System Dots are referenced in the session-config. System Dots are Dot definitions that are not explicitly listed in the Home Dot application **200** or a user's ALL-CONFIG, but are accessible via the Home Dot user interface.

The AddNew Dot enables users to access the Dot Index through the Home Dot application **200** and find new Dots and Snapshots to collect. The Help Dot contains help content for the Home Dot application **200**.

The Login Dot is what first comes up when the Home Dot application **200** is launched. It enables the user to login to the Home Dot application **200**.

Sampling Dots—Trial Dots

One feature of the present invention is the ability to provide a user with the option to try, or sample Dots. According to one implementation of this feature, if a user doesn't have a Home Dot application login account (or is not logged in), then the Home Dot application **200** is in trial mode. Dots can be sampled (by clicking on Dot links **321**) but not kept.

Keeping Dots

A user may decide to keep or not keep a Dot. According to one embodiment, to keep a Dot, a KeepMe Dot is displayed with Dots that are sampled (unique to Dot developer or provider) that informs the user that a Home Dot application **200** login account is required to keep the Dots. The KeepMe Dot

has a mechanism (link) that enables users to create a new Home Dot application **200** login account **205** and keep the Dots by adding them to the new user's Dot definitions **213** or user-profile **310** (FIG. **7**).

User Log In

Each Home Dot application user has a Home Dot application login account that accesses a personal user profile (stored server **153**). When a user launches and logs in to the Home Dot application **200**, it retrieves the user profile from server **153** and restores the Home Dot application **200** to the state in which the user left it, thus recovering their personal application experience. When the user logs out, their user profile is updated on server **153**.

This mechanism enables different users to use the same client computer **199** and receive different application experiences or the same user to use different client computers or devices **199** and receive the same coherent experience. The application **200** restores itself to its last saved state no matter where a user may log in.

After the Home Dot application **200** retrieves its configuration information, it renders the Login Dot from the Dot definition specified in the SESSION-CONFIG. The Login Dot's Dot content asks the user for login and password, sends the login request to the Dot server servlet (also specified in the SESSION_CONFIG), and retrieves the user's profile required to restore the Home Dot application.

User Profile (<All_Config>)

A Home Dot user's profile holds the state of the user's Home Dot application **200** (size, position on the screen, which Dots were open) along with which Dots **213** the user has collected, and how the user has organized their collection (categories, Snapshots or packs). The profile is stored in an XML file called the user's <ALL_CONFIG>. The ALL_CONFIG file contains a SHARE and a LASTSTATE element as described below. The Share may look something like the following:

ALL_CONFIG's<SHARE> attribute:

Since a Share can contain Dots and Snaphsots, the Home Dot **200** uses the SHARE XML DTD to represent the user's collection of Dots and Snapshots. The LASTSTATE element may look something like the following:

ALL CONFIG's<LASTSTATE> attribute:

It contains a PRESET element **313** and a HOMEDOT **314** element (FIG. **7**). The Home Dot **200** uses the PRESET XML DTD (Snapshot) to represent the Dot's that are left open. The Home Dot application element holds the position/size/state of the Home Dot itself. (See also the All_Config Example found in Appendix B.)

The Home Dot Application Server

The Dot application server **153** support much of the functionality of Home Dot application **200** (such as logging in/out, collecting, adding, and sharing Dots and Snapshots, Home Dot usages logging, etc.)

All communication **210** between Home Dot application **200** and the Dot application server **153** is secure and occurs over standard communication protocols (HTTPS). HTTP/HTTPS is chosen for the socket level client/server communication protocol because of its simplicity and more importantly, because most firewalls typically leave the default handling of the HTTP/HTTPS communication with the client.

The Home Dot **200** communicates to the server **153** via an HTTP request. The HTTP request URL contains an API call. Any API calls that require the uploading of data to the server place the data in the message body of the request. API calls that require uploading/downloading of data send/receive that data in XML format. All API calls are the end part of a complete URL that begins with:

http://<someservername>.DoDots.com/DoDots/

where <someservername> is variable and DoDots is an alias for the DoDots servlet/JSP servlet or servlets directory. Arguments to any of the methods are passed in as name value pairs in the query string portion or the URL.

The preferred embodiment of the Dot application server application that supports the Home Dot application **200** currently supports a number of servelt methods including the following:

AddUser

The adduser method is used to create a new Dot user account. Note that this is available as an end-user API to allow new users to add themselves rather than wait for an administrator to do it for them.

GetUser

The getuser method retrieves the user's own demographic data. This method supports the ability of the Home Dot application **200** to then allow updates to the user's demographic information using the setUser method (see below).

SetUser

The setUser method is used to update an existing user account. Note that this is available as an end-user API to allow existing users to update themselves rather than wait for an administrator to do it for them.

GetSession Config

The getsession Config method is used to retrieve general Home Dot application configuration information.

Set Password

The setPassword method provides a mechanism for the user to change their associated Dot server account password.

getMasterDotList

The getMasterDotlist method requests the list of all Dot templates known by the server **153**. The server **153** returns a message body in predefined XML format (see the DOT_MASTERLIST DTD and example) of all possible Dots and their default template values. Note that included in each Dot element are a version stamp and dotclass ID's which allows the Home Dot application **200** to determine if it has the most up-to-date Dot template and Dot class binaries for that Dot. The Home Dot application **200** may then request the updated Dot template (see the getDotTemplate method) and/or dotclass binary zip file (see the getDotClassBinaries method).

setAllConfiq

The setAllConfig method sends, at **303**, the entire body of user-specific client-side configuration information **310** including all Snapshots, Dots, general settings, etc. to the server **153** for persistent storage on behalf of the user. The user profile data **310** is sent as the message body in a predefined XML format (see the ALL-CONFIG DTD and example). The response returned by the server **153** indicates success or failure.

getAllConfig

The getAllConfig method retrieves, at **303**, the last user-specific, server-side saved client side configuration information **310** (all Snapshots, Dots, general settings, etc) from the server **153**. The data is sent as the message body in a predefined XML format (see the ALL-CONFIG DTD and example).

SetAll Events

The setAllEvents method sends, at **307**, usage information (**211**, **212**, **213**, **214**, **215**, FIG. **6**) that has accumulated since the last call to setAllEvents to the server **153** for later use in statistical usage report generation (for DoDots's own use). The data is sent as the message body in a predefined XML

format (see the ALL-EVENTS DTD and example). The response returned by the server indicates success or failure.

GetDotTemplate

The getDotTemplate method requests from the server **153** a specific Dot Template (Dot definition) from the Dot database. The server **153** returns the XML that describes the Dot. This method is called when the Home Dot application **200** has determined that it does not have the current version of the Dot and the user wants the most recent version. (NOTE: this includes when the user is downloading the Dot for the first time)

Addshare

The addshare method stores at **302** a specific shared Dot(s) or Snapshot(s). The server **153** takes the XML that was uploaded by the Home Dot application **200** for sharing and returns the unique id of the share. The Home Dot application **200** puts the XML stream to be shared in the request body.

GetShare

The getshare method requests from the server **153** a specific shared Dot(s) or Snapshot(s). The server **153** returns the XML that was stored for sharing by the original sending user. The format of the returned XML stream is dependent on what has been shared.

Caching Layer

Returning to FIG. **6**, the Home Dot application **200** may advantageously implement a caching layer **219** between its network requests and the network **210**. The Home Dot application **200** sends network requests to the caching layer and the caching layer is responsible for completing those requests (retrying if necessary).

This is used in the event that the Dot application server **153** does not respond. If a user is attempting to log in and the server **153** does not respond, then the Home Dot application **200** logs in using a locally cached <ALL-CONFIG> (the last written user application state can be saved in the caching layer **219**). If a user attempts to log out and the server **153** does not respond, then the caching layer **219** retries at a later time to write the logout application state to the server **153**.

Add New Dots/Snapshots (Packs)

To get a Dot and invoke it—that is, turn it on—the Home Dot application **200** requires the Dot's definition (XML file with the information necessary to instantiate a Dot and point it to Internet content). With reference again to FIG. **7**, a user collects Dots from multiple sources: by clicking on Dot links **321** (Internet link to a Dot's Dot definition) and/or by receiving a Share **320** (Dots and Snapshots shared from other users). In both cases, the Dot definitions for the Dots involved are hosted and served by a Dot application server **153**.

Users can find Dots (Dot links) in the Dot server **153** DotIndex **204** (online Database of Dots via a web browser or the AddNew Dot) and/or promoted on a content provider server **155**, e.g., via a web site. When a user clicks on a DotLink **321** and the Home Dot application **200** is installed, the Home Dot application **200** retrieves the Dot definition that the DotLink **321** points to, adds it to the current user's ALL-CONFIG file **311** of user profile **310**, and turns the Dot on **324**. The Dot instantiates the UI and begins filling it with the Dot content (Internet content) as defined in the Dot definition.

Once a user receives a Dot, the Home Dot application **200** saves the Dot's definition as part of the user's ALL-CONFIG file **311**. Because a Dot definition becomes part of a user profile **310**, it can be modified by use e.g., its default TitleBar image **132** may be changed or its WebConduit control may be navigated to a different URL using DMA messages. The Dot will pick up where it left off next time it is turned on.

Dots can also be received in the form of Shares. A Share **320** is a XML file that represents a collection of one or more Dots and/or one or more Snapshots. A Share **320** may be shared, for example by attaching a Share Link **320** to a standard email message. When a user receives an emailed Share Link **320** from another user and clicks on the Share Link **320** in the email:

The Home Dot application **200** retrieves the share XML file from the Dot application server **153** using the getshare servlet method to which the Share Link **320** points.

The Home Dot application **200** parses the share package and adds the new Dots and Snapshots to the Share recipient's ALL-CONFIG **311**. The Share recipient can now turn on any Dot and Snapshot received as part of the share.

According to one embodiment of the present invention, every Dot that a user gets is added to and becomes part of the user's ALL-CONFIG **311** and becomes accessible to the user anywhere they can login to a Home Dot application **200**. According to another embodiment, a user is provided with the option to keep or discard a sampled Dot prior to adding it the user's ALL-CONFIG **311**.

If the Home Dot application **200** is not installed on a client computer **199** when a user first attempts to get/receive Dots, then the Home Dot application **200** is downloaded and installed with the first Dots that are received. From that point forward, the client machine **199** is Dot-enabled and the user is not required to download and install the Home Dot **200** again (subject to updates). This method removes the alternative approach known today in the art which is to download a custom client application from each company's web server **105** that is visited.

Once a client machine **199** is Dot-enabled, (i.e. the Home Dot application **200** technology installed), then every time that a user clicks on a Dot link **321**, the Dot **120** pops up immediately (faster than it would take to load a small web page). Rather than download and install software for different Dots, only the Dot definitions **102** which are packaged web readable content is collected.

Dot Categories

The Home Dot application **200** enables users to organize Dots that they have collected (in their ALL-CONFIG **311**) by categories. A Dot can be a member of more than one category. Alternatively, a Dot is not required to be categorized (i.e., set attribute categorized=uncategorized). The Home Dot application **200** enables the user to Add/Remove/Rename categories and edit a category's contents (those Dots that are members of that category). Categories may also be assigned by the Dot creator (which may or may not be overwritten.)

In the preferred embodiment, Dots keep track of the categories to which they are members (e.g., there is no master category list). The categories to which a Dot belongs are added to the Dot's definition (in the user's ALL-CONFIG **311**).

The Home Dot application **200** has one unique category: the ALL MY DOTS category (users cannot rename or remove this category). This category contains all the Dots that the user currently has in their collection (in their ALL-CONFIG **311**). Removing Dots from the ALL MY DOTS category removes it from ALL-CONFIG **311**. A user can duplicate a Dot from within the ALL MY DOTS category in the event they want two of the same Dot (e.g., 2 stock watching Dots—one for monitoring a first security, and the other for tracking a second security).

Managing Dots

The Home Dot application **200** provides for several features enabling visually organizing, collecting, and working with Dots **102** individually and as a system. Referring to

FIGS. **12**A, **12**B, and **12**C, a series of drawings depicting a user interface **600** for the organization of Dots on a computer desktop is shown.

To improve ease of use when working with Dots in groups, the Home Dot application **200** enables users to capture an image, or Snapshot, of a workspace that they have defined with Dots. Such Snapshots may include, for example, which Dots are on and where they are positioned on a display screen **656**. Users can instantly recall these Snapshots to restore the previously defined set of Dots to their captured locations on screen **656**.

Snapshots can be created to support different work tasks (e.g., morning news pack, web development pack), different user groups (e.g., small business pack, runners health pack, school study pack), or different activities (e.g., day-trading, dream team sports tracking). Within a Snapshot may be user-created Share Packs **659** (groupings) of Dots (created, for example, using Share Links **320**). In addition, Dot developers can create and publish Packs, which are groups of Dots designed to work together or which may be of similar/related interested to a user, and online content providers can offer their users pre-made Packs.

Users can easily and quickly create and configure/reconfigure their own aggregation of content and functionality as Dots, thus providing a user-aggregated content/application system. This is significantly different than current Internet functions of offering users "My-" versions of a full-screen Internet site in which the choice of content, arrangement, presentation style, etc. are quite limited.

Dots may be aggregated together and manipulated as a block. Blocks are defined as linear clusters of Dots (snapped together vertically or horizontally). An example of a group **659** of Dots **647**, **649**, **651** not in a block is shown in FIG. **12**A. That is, in FIG. **12**A the group **659** of Dots are not aligned with one another with reference to either a horizontal or vertical axis. Two examples of the Dots **647**, **649**, **651** comprising group **659** arranged in a blocks are shown in FIGS. **12**B and **12**C, respectively.

Blocks of Dots have an orientation: vertical (Dots snapped to each other in a vertical column as in FIG. **12**B) or horizontal (Dots snapped to each other in a horizontal row as in FIG. **12**C). In one embodiment, the Home Dot application **200** or Dots (**647**, **649**, or **651**) expose a UI mechanism (e.g., CTRL+O, see slamming) to enable users to change the orientation of Blocks of Dots.

Blocks of Dots can have a justification. When a block of Dots are "justified", they share a common edge such that all edges on one side of the Dots are lined up. For example, a vertical block of Dots all snapped toward the right side of the screen **653** is right justified as shown in FIG. **12**B. In one possible embodiment, the Home Dot application **200** or the Dots exposes a UI mechanism (e.g., SHIFT+CTRL, CTRL+J, see slamming) to enable users to change the justification of Blocks of Dots (left, center, right).

Slamming blocks of Dots against screen edges (**653**, **655**) or other window edges is used as a mechanism to alter the justification and/or orientation of a Block of Dots. For example, a user may slam a horizontal block against the left edge of the screen and the Block's orientation could swing vertical and justified to the left with all Dots snapped to the left of the screen. Likewise, a horizontal block that is bottom justified may be slammed against the top edge **655** of the screen changing it to a top justified Block.

One feature provided by an embodiment of the present invention to assist a user with the organization and presentation of Dots is the snap feature. A Dot (**649** for example) "snaps" to other Dots (**647** and **651** for example), other windows (not shown), and screen boundaries (edges) **653**, **655**. When Dots **120** are dragged (moved) near an edge (**653** or **655**), a magnetism behavior is exhibited and the Dot **649** accelerates towards and "snaps" to that edge (**653** or **655**). The present invention provides for a magnetic gap **658** such that even when Dots are snapped to edges, there still exists a gap (~5 pixels). This gap is supported so that the user may easily visually recognize independent Dots.

The snapping feature simplifies the task for users to quickly and neatly align Dots. (See FIGS. **12**B and **12**C). When Dots (**647**, **649**, and **651**) are snapped together and a Dot is collapsed or resized, then a Dot that is snapped to another Dot can move accordingly; for example, staying snapped or not, according to user preference.

The user interface of the present invention supports the movement of Dots (**647**, **649**, or **651**) in clusters. All Dots (**647**, **649**, or **651**) that share an edge (snapped together=cluster/group) can be grabbed and moved as a group **659** (which may, but need not be, a block).

The present invention teaches multiple methods of selecting a cluster of Dots **659** (as opposed to a single Dot **651** for example). The simplest method comprises of pressing the CTRL key before grabbing a member Dot (any one of **647**, **649**, or **651**) of the cluster **659**. This method selects the entire cluster and moves all Dots **120** as one unit.

Another way is to change the user's selection, that is which Dots (**647**, **649**, or **651**) in the cluster **659** are selected for moving, by the number of times a user clicks before grabbing a member Dot (**647**, **649**, or **651**) of the cluster **659**. For example:

0 clicks before grabbing a member Dot (**647**, **649**, or **651**) selects and moves the entire cluster **659** of Dots.

1 click before grabbing a member Dot (**647**, **649**, or **651**) selects and moves just the member Dot.

Additional clicks could select all vertical, all horizontal, etc.

The present invention provides for a unique method of indicating which Dots (**647**, **649**, **651**) in a cluster **659** are selected (for a group action such as move, or minimize). When multiple Dots are selected (by either method: CTRL+Select or Click+Select), a halo **661** appears around the selected Dots. In one embodiment halo **661** (shown only in FIG. **12**A) is a contrastingly shaded or brightly colored line (~3 pixels wide) floating around the outer perimeter edges of the Dots selected (**647**, **649**, **651**) in a cluster **659**. The halo remains visible as the Dots (**647**, **649**, **651**) are moved.

Dot Sharing

The Home Dot application **200** provides methods for users to easily and quickly Share the Dots that they have collected and the Snapshots that they have created with others. An exemplary delivery mechanism is email. The user's default mail client application is used (a user can use their existing address book and add a detailed message); a web-based mail service extended through the Home Dot application **200** or Dots may also be used. To share and convey shared Dots or Snapshots in an email, a Share Link **320** is required (an Internet link that points to stored Share data on the Dot application server **153**).

When a user sends a Share or Snapshots to another user:

1) A Share XML file **302** is constructed employing the shared Dots and Snapshots from the sharing user's ALL-CONFIG **311**.

2) The Share XML file **302** is stored in Dot database **202** on Dot application server **153** in exchange for a unique shareID (using the addshare servlet method).

3) An email is constructed with a Sharelink **301** (an Internet link that points to the stored Share) and placed in a new email using the user's default mail client.

When a user receives a Share of Dots/Snapshots from another user and clicks on the ShareLink **320** in the email:

1) The recipient's Home Dot application **200** retrieves the Share XML file from the Dot application server **153** (using the getshare servlet method) to which the Share Link **320** points.
2) The Home Dot application **200** parses the Share package and adds the new Dots and Snapshots to the Share recipient's ALL-CONFIG **311**.

The Share recipient can now turn on any Dot and Snapshot received as part of the Share. Because the Share was constructed from Dot definitions that were part of the Sharing user's ALL-CONFIG **311**, the recipient receives the Dots just as the Sharing user had configured them at the time of building the Share.

Preferred Embodiment of Client/Server System

The client/server model **500** of the preferred embodiment is described to according to FIG. **13**. Paramount for the successful implementation of a consumer or high volume system is the proper separation of responsibilities between Dot clients (**501**, **502**, **503**) and the Dot server system **555**. The architecture presented herein supports the primary operations of Dot distribution, client installation, use statistics collection, as well as Dot developer activities. These operations, as described above, are supported in part by the XML structure and calls as listed in the attached appendices A and B. It will be appreciated by one skilled in the art that the list is not exhaustive but descriptive of an implementation of certain features of the present invention.

The Dot server system **555** which is required for support of the general Dot functionality provides for persistent storage and retrieval of configuration and statistical (usage) information and for intelligent software upgrade service for the Dot clients **501**, **502**, **503**. Users are required to logon to the Dot server system **555** with a username and password to access this information. The server **555** also provides for generating statistical analysis reports. The server **555** is used to provide small, persistent data storage areas for third-party Dots. The server **555** may be maintained by a single administration entity and does not require third-party Dot providers for its support. The content contained by the Dot however, may be provided by the third party developers, with the initial content optionally being supplied by existing or re-purposed web pages served by the third party web servers. An administrative interface is therefore provided according to the present invention which is used by the administration entity to maintain the software upgrade information, manage users and generate statistical usage reports.

The software portion of Dot server system **555** consists of several architectural components, including:

Web servers (**505**, **507**, **509**), Servlet JSP Engines (**506**, **508**, **510**) which maintain a Java based XML Parser with SAX (Simple API for XML) and DOM (Document Object Model) interfaces, all of which share a common file system, and at least one Relational Database Management System (RDBMS) Server **513** which supports the RDBMS file system **514**.

According to one embodiment of the present invention, there are no specific operating systems, application server, or database server constraints placed on the server **555** (e.g. the operating system (O/S) may be NT, Solaris, HP-UX, Linux or FreeBSD, or any other viable server O/S. The RDBMS may be Oracle, Sybase, Informix, SQLServer etc.).

The client and server transmit and receive data on the internet **504** in XML format over standard HTTP/HTTPS. An XML parser residing on the Servlet JSP Engine (**506**, **508**, **510**) assists in deconstructing and reconstructing the XML into and out of the RDBMS Server **513** when the XML stream contains information that is needed for report generation. XML parsers with DOM and SAX interfaces are freely available for most major programming languages. HTTP/HTTPS, and therefore a web server (**505**, **507**, **509**), is chosen for the socket level client/server communication protocol because of its simplicity and more importantly, because most firewalls typically leave the default TCP ports for HTTP/HTTPS (ports **80** and **443** respectively) unblocked.

The Web servers (**505**, **507**, **509**) provide the default handling of the HTTP/HTTPS communication with the Dot client **501**, **502**, **503**. The Servlet JSP Engine (**506**, **508**, **510**) functions as an in-process extension of the Web servers (**505**, **507**, **509**) and provides the infrastructure for the application logic layer (servlets) and the presentation layer (Java Server pages). Servlet/JSP is chosen over the architecturally similar ASP due to consideration in performance, maintenance costs, and the variety of O/S and application server vendor choices.

The persistent storage mechanism for everything except the binary software components (client component updates) is Relational Database Management System (RDBMS) **513**. The binary software component for client updates are stored in the file system **514** with a pointer (full pathname) which is stored in the database to each component file. The binary software components are not stored directly in the RDBMS in so as to improve performance.

Since the configuration data need not be manipulated or reported on by the server, the configuration data is stored directly in the RDBMS **513** as a small text file, in the form by which it is transmitted from the client **501**, **502**, **503**. The configuration data is not deconstructed and reconstructed into its constituent parts by the XML parser on the servers (**505**, **507**, **509**). The statistical data is used to generate reports on the server-side by a servlet JSP-based administrative interface. SQL queries are used for statistical report generation to provide simplicity and flexibility. Therefore, a single statistical data XML stream is deconstructed upon receipt by an XML parser residing on the Servlet JSP Engine (**506**, **508**, **510**) and stored as discrete RDBMS columns (the entire stream instance is the RDBMS row) rather than storing the entire stream in a single text column.

Development and Testing of Dots

It is possible to provide for the development and testing of new Dots, Categories, Packs, etc. in a physically separate system that is a superset (duplicate with additional testing support) of the production system **500**. The development and testing (dev/test) system has additional server methods and interfaces, not present in the production system **500** which requires higher security and simplified maintenance, to facilitate the development and testing process. The interfaces to these additional server methods are HTML/JSP pages to allow for easy accommodation of new, geographically distributed Dot content providers.

The development and test system allows the Dot content providers to easily build and test new Dots and Dot Packs. When tested to their satisfaction, the Dot content providers can then, through the interface, submit their Dots/packs for inclusion in the production system. The Dot administrator will be able to periodically run a report to see which Dots/packs were submitted. The Dot administrator is then able to decide whether or not to approve a Dot/pack for import into

the production system. Migration of an approved Dot/pack will involve exporting the appropriate data from the RDBMS on the development and test system, moving the export file from the development and test system and importing the export file into the production system.

Dot Messaging Architecture (DMA)

Dots and the Home Dot application have a messaging architecture (Dot Messaging Architecture—DMA) that enables elements of the system including Dots, controls within Dots, and the Home Dot application, to communicate with one another. This enables these elements to exchange information, request actions or functionality, and respond to system, element, or content events.

FIG. **14** illustrates the message routing paths and elements **700** of the Dot Messaging Architecture. The Dot Messaging Architecture (DMA) has a messaging addressing and routing scheme, defined messages, and an extensible message format (as defined in further detail below) that provides each component with access to component, application, and system features and true application behavior. Also, Dot content has access to the application rendering system within the Home Dot **200** and other Dots (**711**, **751**, **761**) via the DMA.

All elements of the system can send and receive (via paths **701**, **702**, and **703**) DMA messages. In this embodiment, these elements fall into three groups: the Dots (**711**, **751**, **761**); the Controls in the Dot's control space including the WebConduit control (**714**, **754**, **764**) and base controls that encompass menu controls and window operations controls; and the system control **780** embodied within the Home Dot application **200**.

The DMA enables controls to send and receive messages. Controls can exchange messages with other elements in the same Dot (controls in the control space, Dot frame, System) or in a different Dot (the addressing scheme supports addressing elements in other Dots. All Dot controls share a set of common messages. They also can provide messages that are unique to that control (see WebConduit Control). Common control messages include messages such as:

#show
#hide
#get-width
#get-height
#get-size
#is-open
#get-address

These messages query/effect properties/methods of a control within its Dot's control space such as layout, size, etc.

The Web Conduit control (**714**, **754**, **764**) supports additional messages (in addition to the common control messages explained previously). These unique Web Conduit messages enable other elements in the Dot System (Controls, Dot frames, Home Dot application **200**) to interact with WebConduit functionality (request actions or functions, and respond to events). Messages unique to the WebConduit control which is a wrapped Microsoft IE web control in the preferred embodiment, include:

##<any javascript>
#navigate.

One of the most significant features of the WebConduit control is that DMA enables messages to flow IN and OUT of standard HTML rendered within a control. The preferred embodiment currently extends DHTML and javascript to send and respond to DMA messages. The present invention supports SendMessage (a synchronous method which is used if return result is required) and PostMessage (an asynchronous method in which no return result required) methods that can be called using a window.external.<method> call from DHTML in the Web Conduit control (**714**, **754**, **764**).

The preferred embodiment of the message format (explained in more detail in the following sections) allows for two types of messages to be sent IN and OUT of HTML with respect to the WebConduit control. Specific messages and functions can be called (e.g., #navigate) that cause a WebConduit to perform a specific action or function. In addition, any javascript (e.g., ##<any javascript>) can be called within a WebConduit's document as well.

This provides the messaging architecture with exceptional flexibility and extensibility in which data can be passed, functions can be called, and variables can be set. Integrating DMA with standard Web content by enabling DMA messages to call into HTML and for DMA messages to be initiated from HTML within the WebConduit control is the basis that enables Dots to exhibit true application behavior.

Any DMA message that is defined can be sent or called from DHTML. This, in conjunction with the application logic capability that is provided by javascript (and other HTML scripting languages), provides the application media development environment (which includes application development platform and language). A Dot developer can thus author a new Dot application by developing web content (HTML, GIF files, etc.) and by packaging that content in a Dot Definition, eliminating the need for compilers and consequently, downloading executables. Therefore, if a client device **199** has been Dot-enabled (the Home Dot application **200** is installed) then that device can instantaneously view, open, run Dots as well as modify and save their configurations.

DMA messages can be sent to and received from a Dot. This enables elements of the system that can send and receive DMA messages to interact with a Dot. Elements of a Dot's definition can be accessed/modified using Dot messages. Examples of Dot Properties/Dot methods that can be accessed and modified via the DMA include:

#set-title
#get-title
#set-size<width><height>
#is-open
#close
#collapse
#uncollapse
#set-title-images <up-URL><down-url><over-url><inactive-url>

Since a message can originate from DHTML content which is rendered in the Web Conduit control as Dot content, standard web content when rendered in a Dot can therefore access properties and behaviors of its packaging (the Dot). Internet content can therefore do such things as modify the size of the Dot that is rendering it. It can move it or collapse it.

The present invention therefore enables properties and behaviors to be tied to and between any Dot content event, even outside of a Dot affecting another Dot (e.g., an internet content (DHTML) mouse-over event can change the size and position of another Dot, etc.)

Most of the DMA examples thus presented are of other elements sending messages to the Dot frame (**711** for example). An example of a Dot frame **711** sending messages can be seen in the case of the menu control **712** (discussed further below). The Dot frame **711** may also send messages (as can the menu control **712**) based upon the occurrence of certain events such as Dot-moving, Dot-collapsed/expanded, Dot-infocus/inactive, etc.

The preferred embodiment has the menu control **712** implemented as part of the Dot Frame **711**. Menu control **712** entries are given/tied-to actions using the DMA. Each menu entry may consist of, inter alia, a text-title, an icon, a tooltip, an ID, and an action. The action is simply a DMA message that has a recipient specified address. Dot menu control **712** items are defined as part of a Dot definition.

When a menu entry within a menu control **712** is selected by a user, the specified Dot Message is sent to the specified recipient (DMA address). This illustrates an example of the true application behavior that is exhibited by Dot Content having access to DMA.

As with the functionality described in the previous sections, the following are examples of types of additional application behaviors that are possible:

A menu control **712** entry (e.g., refresh) can send a message (to refresh) to a Web Conduit control and cause an action (e.g., refresh content).

A menu control **712** entry can send a message to the Dot (itself) and cause it to collapse, or resize, or exit.

A menu control **712** entry can call any javascript (##<any javascript>) in the HTML of Dot content rendered in the same Dot or a different Dot to set variables, change images, call functions, etc.

DMA messages can also be sent to and received (**701**, **702**, **703**) from the system (**720**)—the functional layer above and between individual Dots. Some aspects for which the system is responsible include DMA message routing, adding, removing, opening, closing Dots, etc. In the preferred embodiment, the system includes the Home Dot application **200** or equivalent Dot-rendering and Dot-management client application.

Examples of system functions and behaviors that can be accessed and called using DMA messages include:

#refresh
#install-dot <dot-url>
#have-dot <dot-address>
#delete-dot <dot-address>
#quit
#save-to-server
#get-screen-width
#get-screen-height
#close-all-dots
#open-preset <preset-name>
#take-preset <preset-name>
#get-dot-ids <dot-address>

Messages can be sent to/from controls (WebConduit control—in and out of HTML), to and from the Dot (DotFrame—menu control **712**, Titlebar **713**, bottom bar **717**, flexible Dot Definition), to and from the system (Home Dot application **200**—open and close Dots, system variables and data). The DMA addressing scheme also provides for messages to be sent to any Dot, to any control in any Dot, to any HTML, in any control, in any Dot. This enables Dots to work together as an application system, enables Dot developers to share functionality and leverage and build on the functionality of other Dots.

In the preferred embodiment, a DMA message has two components: a recipient address and the message body itself. Both are represented as strings. The addressing scheme is explained in the next section.

The body of a DMA messages is, at its simplest, a text string which may represent any javascript, for example, sent to a Web Conduit control. For defined messages, there is a method element to define the function/behavior call/request [e.g., #set-size] followed by arguments if the method element requires them (e.g., height in DotUnits; width in DotUnits).

The present invention provides for two functions that enable messages to be generated from within DHTML in the WebConduit control: SendMessage and PostMessage. SendMessage, which is synchronous, is used if a return value is required. PostMessage, which is asynchronous, is used otherwise.

Below are examples of messages generated with DHTML within the WebConduit control:

window.external.PostMessage ("#.:", "#set-position **350 500**")

This message is sent to a Dot and causes it to move to a new screen position, for example 350×500 pixels.

window.external.SendMessage ("#system, "#get-screen-width")

This message asks the system for the current screen's width in pixels.

Messages are routed according to three pieces of information: Domain, Dot Specifier, Control Specifier.

One of the ways that a Dot is identified is by a Domain. In the preferred embodiment, the Domain is the same as a Dot provider's ID (developer ID). Generally this is a unique identifier for each company or developer and is specified in a Dot's definition.

Dots in the same domain, by default, can communicate to one another without explicitly specifying the domain in the address. When no messaging access restrictions are placed on addressing a Dot according to its domain (by default there are restrictions), then a Dot can also send messages to another Dot belonging to a different domain by explicitly addressing that Dot by its domain and Dot specifier.

Dots are further identified with a kind attribute as a part of the Dot Definition, and by an ID attribute. The ID is volatile and is not typically hard-coded into DHTML scripts. The addressing scheme allows for a Dot to be specified by explicitly referencing or querying a Dot's kind or ID identifier attribute.

Similarly, controls are identified by kind and by ID (also a part of the Dot Definition). Controls can also be specified by explicitly referencing or querying its kind or ID.

In the preferred embodiment, there are several permitted forms for the address of a message recipient. The fundamental one is:

#<dot specifier>:<control specifier>

Other accepted address forms are:

#<domain>:<dot specifier>:<control specifier>
#<domain>:<dot kind>:<dot id>:<control kind>:<control id>
#system

If an address does not match any of these forms, the address defaults to #system.

Specifiers have four different forms: The first form specifies the unique ID of the Dot/control/domain in question. An example of this first form is:

<specifier>:=<ID>

The second form allows the sender to address a message to the closest matching recipient by some form of search criteria. Multiple Dots or controls might be of the same kind, enabling broadcasting a message to these elements. An example of this second form is:

<specifier>:=<kind><#search criteria>

Search criteria can be one of:

<search criteria>:=any
<search criteria>:=open
<search criteria>:=closed

Controls that are hidden are considered closed, and controls that are visible are considered open.

The third form enables reference to a specific Dot or control, allowing messages to be sent within a Dot. If a particular control is specified, then it must be specifically referenced in the Dot. An example of this third form is:

<specifier>:=dot

The final form is specifically for the control specifier. If a message is to be directed to a Dot and not a control, the control specifier must be empty. An example of this last form is:

<specifier>:=

The current embodiment supports messages to a specific recipient: the addressing/routing scheme could support messages addressed to multiple simultaneous recipients (broadcast). Similarly, Dots can address Dots on the same platform or client (computer or device); the addressing or routing scheme may also support messages addressed to Dots/Controls/Home Dot applications **200** on other devices or by user. The Home Dot application **200** routes messages sent between Dots (**711**, **751**, **761**) and resolves addressing queries (e.g., address: "#A#Any:"=first Dot of kind "A" found in the Home Dot application **200**).

Since the Home Dot application **200** routes messages between Dots (**711**, **751**, **761**), it can allow or restrict Dots from addressing and sending messages to other Dots outside their own domain (the Home Dot application **200** could also restrict messages to within the same Dot). For example, Dot**2** (**751**) is a member of domain B (**750**) as is Dot **3** (**761**), whereas Dot **1** (**711**) is a member of Domain A. Home Dot **200** can permit or deny messaging between Dot **1** (**711**) and Dots **2** and **3** (**751**, **761**) and vice versa.

The Home Dot application **200** acts to enforce the access rights to and between published messages built on the DMA API. In one embodiment, the Home Dot application **200** accesses a database of published message methods implemented by Dot Developers and restricts/allows messages to pass based on access criteria data posted to the data base.

For example, a Dot developer could specify 2 public functions, 2 functions with access restricted by partner (Domain **710** or **750** for example), and 2 functions with access restricted by Dot address. The Home Dot application **200** may allow or restrict messages to be sent to one Dot from other Dots according to access criteria specified by the Dot developer.

The Home Dot application **200** also responds to messages sent to #system. Access to system features/functions/behaviors is one aspect of the present invention that enables extensibility of the Home Dot application system.

System messages include:

#REFRESH—Refreshes the user's ALL-CONFIG **311**.

#INSTALL-DOT<DOT-URL>—Adds the specified Dot to the user's Home Dot application **200** (ALL-CONFIG) using the same mechanism that Dot definitions are added to the system.

#HAVE-DOT<DOT-ADDRESS>—Checks if the user currently has the specified Dot in their Home Dot application **200** (as part of the ALL-CONFIG).

#DELETE-DOT<DOT-ADDRESS>—Removes the specified Dot from the Home Dot application **200** (and the user's ALL-CONFIG).

#QUIT—Quits the Home Dot application **200**.

#SAVE-TO-SERVER—Saves the user's ALL-CONFIG to the Dot application server **153**.

#GET-SCREEN-WIDTH—Returns the width of the screen in pixels.

#GET-SCREEN-HEIGHT—Returns the height of the screen in pixels.

#CLOSE-ALL-DOTS—Closes all open Dots.

#OPEN-PRESET<PRESET-NAME>—Opens the specified user's Snapshot.

#TAKE-PRESET <PRESET-NAME>—Capture a Snapshot.

#GETDOT-IDS <DOT-ADDRESS>—Returns the DotID of the specified Dot.

Statistics Collection/Analysis

The Home Dot application/Dot system and method of collecting use statistics from Home Dot applications (and application servers) enables the present invention to build and query a multidimensional use-profiling database. Because Dots are used in groups and used more often for longer periods of time than web pages/sites, the present invention may collect real-time multidimensional use statistics **206** (according to which Dots are ON, for example together or simultaneously) that over time becomes a valuable multi-dimensional user behavior profiling database. The Dot server **153** collects use statistics **206** from its Home Dot application **200** on clients **199** and manages and tracks Home Dot application downloads, access to Dot Index **204**, etc.

Information on a Dots user can be monitored in a multidimensional fashion. Instead of classifying a user based solely on their demographic characteristics and linear use of the internet, the present invention enables tracking on an additional dimension. A user can be classified according to which Dots they use at the same time, instead of only how they navigate within an individual viewer (the browser). This user information can be packaged and sold to content providers so that they can better provide services to their users.

Because the present invention enables multi-dimensional profiling, a service may be provided by the operator of this system and its partners to customize advertisements and offerings to users in a more efficient and targeted manner. Tracking this profiling data in real-time allows the operator to tailor these offerings to users of specific Dots in a way that is not otherwise possible today. For example, one content provider partner can be informed that users of a certain Dot are also disproportionately monitoring content about a specific topic through another Dot. This information is then used to target ads at the moment the peripheral interests of that content providers Dot users are known. The present invention enables this information to be tracked as a complete system. The system can be utilized to facilitate serving advertisements and offerings based on the information that it collects.

The Home Dot application **200** records application events in an XML log that intermittently gets uploaded to the Dot application server **153**. Application events that are logged include the following (by timestamp and by session):

When a Dot is opened and when it is closed (particularly, when Home Dot **200** is opened and closed)

When a Dot is added from a Dot Template or a Share

The number of times a user clicks through from a Dot to a full-screen web page (and to which URLs)

The number of page views in a Dot-by-Dot session

When and from which URL did a user download a Home Dot application **200**

The following outlines the types of questions the use statistics of the present invention can answer regarding Home Dot application **200** and Dot use:

Duration and frequency of opening Home Dot applications and Dots—How often is the Home Dot application **200**/Dot used on average (optionally, by user group) during the parts of a day (e.g., morning)?

Duration: What was the average amount of time that users leave the Home Dot application **200**/Dot ON **324** during the day

Distribution of Home Dot application **200** and Dots (Point of Distribution including Shares)—How many users have downloaded the Home Dot application during a set period of time? From which URL were they referenced?

How many times has this Dot's DotTemplate been installed (from the DotIndex)?

From which web site were users referred? How many times has the Dot been installed from a Share?

Page views in Dots, Click-throughs/URL's—How does the page-view performance of a Dot compare to the comparable content on a full-screen web site?

List the full-screen URLs most often clicked through to (in ranking order) from this Dot.

Dots used in Groups—Which Dots are used most frequently together?Which Dots do users most often use with Dot.

The Home Dot application **200** intermittently sends its event logs to the Dot application server **153** using the setAllEvents servlet method. The Home Dot application's events are logged and sent as an XML file in a format specified by the CALL-EVENTS.

Categorizing Dots and/or Dotcontent by "context keywords" (e.g., CNN Dot and FoxNews Dot="News"; CBS Sportsline Dot="Sports") enables the Home Dot application **200** to build a real-time (accessible via the DMA) multi-dimensional use context on-the-fly, based on which Dots the user currently has ON (e.g., assemble the keywords of the currently open Dots into a multi-dimensional keyword string). This highly resolved view of a user's behavior enables the Home Dot application **200** to source a very targeted offering (commerce opportunity or ad) and/or enable Dot developers to do the same. In one embodiment, the Home Dot application **200** supports a DMA message (pay to access) that provides Dot developers with access to this type of information (e.g., SendMessage "#system" "#get-use-context-string")

The Home Dot application **200** supports a developer toolkit service that sources ads targeted according to this multi-dimensional use context generated by the Home Dot application **200** (more targeted than currently possible with singular contexts such as text strings, e.g., searched on "toys"=serve a toy ad). The Home Dot application **200** and/or Dot application server **153** monitors/tracks, and handles incremental billing for all parties (e.g., anonymously).

Variations

Many variations on the above description are contemplated and within the scope of the present disclosure. For example, in an alternate embodiment, the Home Dot application **200** interface is implemented entirely as Dot content served from Dot application server **153**. Furthermore, the Home Dot application **200** itself may be packaged as a Dot. The Home Dot application Dot has special responsibilities and rights, but otherwise, it could be as much a Dot as any other Dot. This embodiment enables the application executable to be significantly smaller and the Home Dot application **200** UI to be significantly more flexible.

In yet another alternate embodiment, the Home Dot application **200** is implemented as a plug-in to the web browser.

In still another alternate embodiment, the Home Dot application **200** is implemented in Java, thus enabling it to be more easily portable to other platforms.

Additionally, in another embodiment, a subset of the Home Dot application **200** functionality is implemented through the browser using pop-ups; this would be particularly valuable to mobile Home Dot application **200** users who don't always have easy access to a computer with the Home Dot application **200** installed.

The present invention enables the definition of an interface that groups Dots together from different content providers and offers them to end users as part of a unique system of content. Content from different internet companies can be made to interact with each other using aspects of the present invention provided for in the Dot messaging architecture (DMA). The Dot server **153** may therefore be provided with functionality to broker these relationships and facilitate this interaction in Internet content between companies and between Dot-enabled internet sites.

Because Dots can work together (via DMA and packs), the present invention enables companies or content providers to enable their Dots to work together. The present invention provides for another layer built upon the DMA API (application programming interface) that specifies how different kinds and types of Dots from different companies or developers communicate/work together. This layer is optionally open source so that a large portion of it is self-published by the Dot Developers themselves.

Dot Developers implement, specify, and publish in a database DMA message methods that other Dot Developers implement through calls made within their published Dots. These Dot developer methods may be aggregated and published for use in a database provided by the Home Dot publisher for example and sourced within a developer zone, or as part of a Dot Definition.

Dot developers may implement these DMA message methods as Java script functions that other Dot Developers calls with the DMA's ##<any Javascript> message. Dot Developers would need to be able to address a particular Dot's Web Conduit control that supports a page implementing the specified Javascript function and address by domain if different from their domain. Restricted or variable permissions may be provided via a web server or defined in a Dot's Dot Definition to restrict/enable routing of messages.

A Dot Developer has the option to restrict access to certain methods according to some criteria e.g., public, partners, domain. Different types of restriction criteria schemes may be implemented. A Dot may send a message to another Dot and query for methods accessible to it. An example of access levels for functions is provided below.

Public functions—Dot Developer implements and publishes DMA message methods that any other Dot Developer could call (e.g., what-time-is-it for a Clock Dot).

Reserved functions—Dot Developer implements, specifies, and publishes a set of DMA message methods that may be called only by certain types of Dot Developers (e.g. premier partner), specific Dot Developers, or specific Dots.

Private functions within the same domain—A Dot Developer restricts access to certain DMA message methods that may only be implemented by other Dots in the same domain. The preferred embodiment provides for the option to either allow all messages to flow between domains, or to fully restrict messages to within a domain.

Local Private functions within Dot—At the most restricted level, access is specified such that DMA messages may only be accessed from within the same Dot.

A published API that specifies ways for Dots to extend functionality to other Dots enables Dot Developers to leverage the development work and functionality of other Dots. As an example, a Dot Developer may implement a credit card processing Dot. In one embodiment, this Dot is implemented as a javascript function that checks the credit available on a credit card. This developer could specify this as a public function (in the Dot Definition for example) and publish this

function in a Server Dot Index. Other Dot Developers send a message to this Dot and call "check-credit", thus leveraging the development carried out from the first Dot Developer.

In the above example, the messaging architecture may provide revenue generation by way of monthly fees, per-access fees, etc. The Home Dot application **200** and/or Dot application servers could monitor/track and automatically handle incremental billing for all parties. By applying this method to the previous example, a Dot Developer who calls a credit card processing Dot for a credit check may be incrementally billed for each such call.

It will be appreciated that the methods, in the form of instructions having a sequence, syntax, and content, of the present invention may be stored on (or equivalently, in) any of a wide variety of computer-readable media such as magnetic media, optical media, magneto-optical media, electronic media (e.g., solid state ROM or RAM), etc., the form of which media not limiting the scope of the present invention. A data processor reading said media is operable to either transfer (e.g., download) said instructions thereto and then operate on those instructions, or cause said instructions to be read from the media and operate in response thereto. Furthermore, devices (e.g., a reader) for accessing the instructions on said media may be contained within or connected directly to the data processor residing on a device on which those instructions operate, or may be connected via a network or other communication pathway to said data processor.

While a plurality of preferred exemplary embodiments have been presented in the foregoing detailed description, it should be understood that a vast number of variations exist, and these preferred exemplary embodiments are merely representative examples, and are not intended to limit the scope, applicability or configuration of the invention in any way. Rather, the foregoing detailed description provides those of ordinary skill in the art with a convenient guide for implementation of the invention, by way of examples, and contemplates that various changes in the functions and arrangements of the described embodiments may be made without departing from the spirit and scope of the invention defined by the claims thereto.

What is claimed is:

1. A client device, the client device comprising:

electronic storage having stored thereon a plurality of networked information monitor templates defining a plurality of networked information monitors, the plurality of networked information monitor templates comprising a first networked information monitor template defining a first networked information monitor, wherein the first networked information monitor template comprises:

(1) a content reference that comprises a network location at which content for the first networked information monitor is accessible via a TCP/IP protocol;

(2) a definition of a graphical user interface of the first networked information monitor that lacks controls for manually navigating a network, and that includes a frame within which content received from the network location can be displayed, and frame characteristics defining one or more a color, a size, or a position on the electronic display of the frame; and

(3) instructions configured (i) to cause the first networked information monitor to request content from the network location in the content reference via the TCP/IP protocol, and (ii) to cause the first networked information monitor to generate the graphical user interface of the first networked information monitor with the content received from the network location via the TCP/IP protocol within the frame;

an electronic display; and

one or more processors configured to access the first networked information monitor template, and to execute the first networked information monitor template such that the graphical user interface of the first networked information monitor is presented to a user on the electronic display having content received from the content reference therein.

2. The client device of claim **1**, wherein the first networked information monitor template further comprises control characteristics that define one or more controls that are usable by the user to interact with or control the first networked information monitor.

3. The client device of claim **2**, wherein the control characteristics define visual representations of the controls that are included in the graphical user interface of the first networked information monitor.

4. The client device of claim **1**, wherein the plurality of networked information monitor templates further comprises a second networked information monitor template defining a second networked information monitor, wherein the second networked information monitor template comprises:

(1) a second content reference that comprises a second network location, which is different from the network location in the content reference of the first networked information monitor template, at which content for the second networked information monitor is accessible via the TCP/IP protocol; and

(2) a definition of a graphical user interface of the second networked information monitor that lacks controls for manually navigating a network, and that includes a second frame within which content received from the second network location can be displayed; and

(3) instructions configured (i) to cause the second networked information monitor to request content from the second network location in the second content reference via the TCP/IP protocol, and (ii) to cause the second networked information monitor to generate the graphical user interface of the second networked information monitor with the content received from the second network location via the TCP/IP protocol within the frame.

5. The client device of claim **4**, wherein the one or more processors are further configured to execute the second networked information monitor template such that the graphical user interface of the second networked information monitor is presented to the user on the electronic display separately and discretely from the user interface of the first networked information monitor, and having content therein received from the second content reference.

6. The client device of claim **1**, wherein the one or more processors are further configured to transmit a request to a server for a further networked information monitor template responsive to reception of a user request for a further networked information monitor defined by the further networked information monitor template.

7. The client device of claim **6**, wherein the electronic storage is configured to electronically store the received further networked information monitor template, responsive to reception of the further networked information monitor template from the server.

8. The client device of claim **7**, wherein the one or more processors are further configured to execute the further networked information monitor template.

9. A computer-implemented method of presenting Internet content to a user, the method being implemented in a client device comprising a electronic storage, an electronic display, and one or more processors, the method comprising:

storing to the electronic storage a plurality of networked information monitor templates defining a plurality of networked information monitors, the plurality of networked information monitor templates comprising a first networked information monitor template defining a first networked information monitor, wherein the first networked information monitor template comprises:

(1) a content reference that comprises a network location at which content for the first networked information monitor is accessible via a TCP/IP protocol; and

(2) a definition of a graphical user interface of the first networked information monitor that lacks controls for manually navigating a network, and that includes a frame within which content received from the network location can be displayed, and frame characteristics defining one or more a color, a size, or a position on the electronic display of the frame; and

(3) instructions configured (i) to cause the first networked information monitor to request content from the network location in the content reference via the TCP/IP protocol, and (ii) to cause the first networked information monitor to generate the graphical user interface of the first networked information monitor with the content received from the network location via the TCP/IP protocol within the frame;

accessing the first networked information monitor template in the electronic storage; and

executing the first networked information monitor template on the one or more processors such that the graphical user interface of the first networked information monitor is presented to the user on the electronic display having therein content received from the content reference.

10. The method of claim **9**, wherein the first networked information monitor template further comprises control characteristics that define one or more controls that are usable by the user to interact with or control the first networked information monitor.

11. The method of claim **10**, wherein the control characteristics define visual representations of the controls that are included in the graphical user interface of the first networked information monitor.

12. The method of claim **9**, wherein the plurality of networked information monitor templates further comprises a second networked information monitor template defining a second networked information monitor, wherein the second networked information monitor template comprises:

(1) a second content reference that comprises a second network location, which is different from the network location in the content reference of the first networked information monitor template, at which content for the second networked information monitor is accessible via the TCP/IP protocol; and

(2) a definition of a graphical user interface of the second networked information monitor that lacks controls for manually navigating a network, and that includes a second frame within which content received from the second network location can be displayed; and

(3) instructions configured (i) to cause the second networked information monitor to request content from the second network location in the second content reference via the TCP/IP protocol, and (ii) to cause the second networked information monitor to generate the graphical user interface of the second networked information monitor with the content received from the second network location via the TCP/IP protocol within the frame.

13. The method of claim **12**, further comprising executing the second networked information monitor template on the one or more processors such that the graphical user interface of the second networked information monitor is presented to the user on the electronic display separately and discretely from the user interface of the first networked information monitor, and having content therein received from the second content reference.

14. The method of claim **9**, further comprising transmitting a request to a server for a further networked information monitor template responsive to reception of a user request for a further networked information monitor defined by the further networked information monitor template.

15. The method of claim **14**, further comprising, responsive to reception of the further networked information monitor template from the server, storing the received further networked information monitor template to the electronic storage.

16. The method of claim **15**, further comprising executing the further networked information monitor template on the one or more processors.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 8,020,083 B1
APPLICATION NO. : 11/932585
DATED : September 13, 2011
INVENTOR(S) : John Albert Kembel et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

On the Title page, Item (63) should read:

(63) Continuation of application No. 09/558,922, filed on Apr. 26, 2000, now Pat. No. 7,756,967, and application No. 09/558,925, filed on Apr. 26, 2000, now Pat. No. 7,660,868.

On the Title page, insert Item (60) as follows:

-- (60) Provisional application No. 60/131,083, filed on Apr. 26, 1999, provisional application No. 60/131,115, filed on Apr. 26, 1999, provisional application No. 60/131,114, filed on Apr. 26, 1999, provisional application No. 60/176,687, filed on Jan. 18, 2000, provisional application No. 60/176,699, filed on Jan. 18, 2000. --

Signed and Sealed this
Fourth Day of September, 2012

David J. Kappos

David J. Kappos
Director of the United States Patent and Trademark Office

US008510407B1

(12) **United States Patent**
Kembel et al.

(10) **Patent No.: US 8,510,407 B1**
(45) **Date of Patent: *Aug. 13, 2013**

(54) **DISPLAYING TIME-VARYING INTERNET BASED DATA USING APPLICATION MEDIA PACKAGES**

(75) Inventors: **John Albert Kembel**, Palo Alto, CA (US); **George Andrew Kembel**, Menlo Park, CA (US); **Daniel S. Kim**, Palo Alto, CA (US); **John Russell**, Palo Alto, CA (US); **Jake Wobbrock**, Palo Alto, CA (US); **Geoffrey S. Kembel**, Menlo Park, CA (US); **Jeremy L. Kembel**, Palo Alto, CA (US); **Lynn D. Gabbay**, Sunnyvale, CA (US)

(73) Assignee: **Mainstream Scientific, LLC**, Mountain View, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 928 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **11/932,553**

(22) Filed: **Oct. 31, 2007**

Related U.S. Application Data

(63) Continuation of application No. 09/558,925, filed on Apr. 26, 2000, now Pat. No. 7,660,868.

(60) Provisional application No. 60/176,699, filed on Jan. 18, 2000, provisional application No. 60/176,687, filed on Jan. 18, 2000, provisional application No. 60/131,115, filed on Apr. 26, 1999, provisional application No. 60/131,114, filed on Apr. 26, 1999, provisional application No. 60/131,083, filed on Apr. 26, 1999.

(51) **Int. Cl.**
G06F 15/16 (2006.01)

(52) **U.S. Cl.**
USPC **709/217**; 709/219

(58) **Field of Classification Search**
USPC 709/217, 219
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,599,756	A	2/1997	Matsuo	501/127
5,625,781	A	4/1997	Cline et al.	
5,649,186	A	7/1997	Ferguson	
5,682,511	A	10/1997	Sposato et al.	715/716
5,740,549	A	4/1998	Reilly et al.	

(Continued)

FOREIGN PATENT DOCUMENTS

WO WO0180086 A2 10/2001

OTHER PUBLICATIONS

Alexa 1.4.1 Support Pages, 9 pages, www.alexa.com/support/index1.html, Jan. 1999.

(Continued)

Primary Examiner — Kenny Lin
(74) *Attorney, Agent, or Firm* — Pillsbury Winthrop Shaw Pittman, LLP

(57) **ABSTRACT**

A software component for accessing and displaying time-varying Internet content includes a definition for rendering a graphical user interface and a URL pointing to the time-varying Internet content to be downloaded and presented within said user interface. A component provider may create displays of Internet content which vary as a function of the varying of the time-varying Internet content. In one example, the graphical user interface includes an image, and the image varies in order to illustrate the varying of the time-varying data.

24 Claims, 37 Drawing Sheets

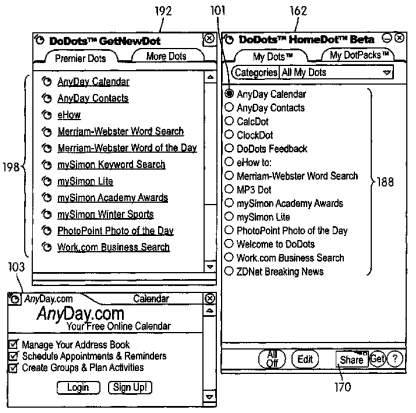


Exhibit No.
7

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,745,718	A		4/1998	Cline et al.	715/777
5,761,662	A		6/1998	Dasan	
5,774,670	A		6/1998	Montulli	395/200.57
5,794,230	A		8/1998	Horadan et al.	
5,796,393	A		8/1998	MacNaughton et al.	
5,796,952	A		8/1998	Davis et al.	
5,801,702	A		9/1998	Dolan et al.	
5,802,530	A		9/1998	Van Hoff	715/513
5,805,829	A		9/1998	Cohen et al.	395/200.32
5,809,248	A		9/1998	Vidovic	
5,818,446	A		10/1998	Bertram et al.	715/746
5,835,088	A		11/1998	Jaaskelainen, Jr.	715/803
5,838,906	A	*	11/1998	Doyle et al.	715/205
5,860,068	A		1/1999	Cook	705/26.81
5,864,676	A		1/1999	Beer et al.	709/229
5,864,868	A		1/1999	Contois	707/104
5,890,172	A		3/1999	Borman et al.	
5,893,091	A		4/1999	Hunt et al.	
5,896,533	A		4/1999	Ramos et al.	395/680
5,918,237	A		6/1999	Montalbano	715/206
5,919,247	A		7/1999	Van Hoff et al.	709/217
5,922,044	A		7/1999	Banthia	709/203
5,923,845	A		7/1999	Kamiya et al.	709/206
5,923,885	A		7/1999	Johnson et al.	717/176
5,948,061	A		9/1999	Merriman et al.	
5,959,621	A		9/1999	Nawaz et al.	345/329
5,966,715	A		10/1999	Sweeney et al.	707/203
5,973,692	A		10/1999	Knowlton et al.	345/348
5,974,446	A		10/1999	Sonnenreich et al.	
5,974,546	A		10/1999	Anderson	713/2
5,977,964	A		11/1999	Williams et al.	
5,983,227	A		11/1999	Nazem et al.	
5,987,513	A		11/1999	Prithviraj et al.	
5,995,756	A		11/1999	Herrmann	
6,006,252	A		12/1999	Wolfe	
6,012,090	A		1/2000	Chung et al.	
6,012,098	A		1/2000	Bayeh et al.	
6,018,344	A		1/2000	Harada et al.	
6,023,698	A		2/2000	Lavey, Jr. et al.	707/10
6,026,433	A		2/2000	D'Arlach et al.	
6,031,904	A		2/2000	An et al.	379/201
6,034,652	A		3/2000	Freiberger et al.	715/730
6,044,403	A		3/2000	Gerszberg et al.	709/225
6,061,695	A		5/2000	Slivka et al.	707/513
6,061,696	A	*	5/2000	Lee et al.	715/209
6,065,044	A		5/2000	Ogasawara	
6,088,717	A		7/2000	Reed et al.	709/201
6,091,411	A		7/2000	Straub et al.	715/747
6,091,412	A		7/2000	Simonoff et al.	345/335
6,101,510	A	*	8/2000	Stone et al.	715/234
6,104,391	A		8/2000	Johnston, Jr. et al.	715/745
6,105,063	A		8/2000	Hayes, Jr.	709/223
6,115,040	A		9/2000	Bladow et al.	345/335
6,128,655	A	*	10/2000	Fields et al.	709/219
6,133,916	A		10/2000	Bukszar et al.	
6,161,112	A		12/2000	Cragun et al.	
6,177,936	B1		1/2001	Cragun	
6,192,407	B1		2/2001	Smith et al.	709/229
6,199,082	B1		3/2001	Ferrel et al.	
6,215,490	B1		4/2001	Kaply	715/788
6,216,141	B1		4/2001	Straub et al.	707/513
6,230,173	B1		5/2001	Ferrel et al.	
6,237,030	B1		5/2001	Adams et al.	
6,240,555	B1		5/2001	Shoff et al.	725/110
6,268,856	B1		7/2001	Bruck et al.	
6,275,854	B1		8/2001	Himmel et al.	
6,278,448	B1		8/2001	Brown et al.	345/333
6,278,449	B1		8/2001	Sugiarto et al.	715/826
6,286,034	B1		9/2001	Sato et al.	709/204
6,289,362	B1		9/2001	Van Der Meer	715/273
6,292,185	B1		9/2001	Ko et al.	
6,292,186	B1		9/2001	Lehman et al.	345/335
6,297,819	B1		10/2001	Furst	
6,314,451	B1		11/2001	Landsman et al.	
6,317,759	B1		11/2001	Osmond	715/513
6,339,826	B2		1/2002	Hayes, Jr. et al.	713/166
6,341,305	B2		1/2002	Wolfe	
6,342,907	B1		1/2002	Petty et al.	345/762
6,343,377	B1		1/2002	Gessner et al.	717/10
6,356,905	B1	*	3/2002	Gershman et al.	705/26.8
6,369,840	B1		4/2002	Barnett et al.	
6,370,552	B1		4/2002	Bloomfield	715/513
6,374,273	B1		4/2002	Webster	707/513
6,385,596	B1		5/2002	Wiser et al.	705/51
6,393,407	B1		5/2002	Middleton et al.	
6,401,134	B1		6/2002	Razavi et al.	709/310
6,411,992	B1		6/2002	Srinivasan et al.	
6,414,677	B1		7/2002	Robertson et al.	345/419
6,418,440	B1		7/2002	Kuo et al.	
6,434,563	B1		8/2002	Pasquali et al.	
6,434,598	B1		8/2002	Gish	709/203
6,452,609	B1		9/2002	Katinsky et al.	345/716
6,453,348	B1		9/2002	Barnier et al.	709/225
6,460,029	B1		10/2002	Fries et al.	
6,476,833	B1		11/2002	Moshfeghi	345/854
6,484,149	B1		11/2002	Jammes et al.	
6,487,566	B1		11/2002	Sundaresan	
6,487,663	B1		11/2002	Jaisimha et al.	713/193
6,496,203	B1		12/2002	Beaumont et al.	345/762
6,510,466	B1		1/2003	Cox et al.	709/229
6,537,324	B1		3/2003	Tabata et al.	
6,538,673	B1		3/2003	Maslov	
6,549,612	B2		4/2003	Gifford et al.	
6,560,639	B1		5/2003	Dan et al.	
6,571,245	B2		5/2003	Huang et al.	
6,594,682	B2		7/2003	Peterson et al.	
6,606,657	B1		8/2003	Zilberstein et al.	709/224
6,629,143	B1		9/2003	Pang	709/226
6,662,341	B1		12/2003	Cooper et al.	715/513
6,681,368	B1		1/2004	Kawabata	715/501.1
6,687,745	B1		2/2004	Franco et al.	709/219
6,691,130	B2		2/2004	Kawasaki et al.	707/102
6,694,484	B1		2/2004	Mueller	715/513
6,718,015	B1		4/2004	Berstis	379/88.17
6,724,403	B1		4/2004	Santoro et al.	345/765
6,751,606	B1		6/2004	Fries et al.	
6,757,716	B1		6/2004	Blegen et al.	709/217
6,766,454	B1		7/2004	Riggins	713/185
6,784,900	B1		8/2004	Dobronsky et al.	
6,816,880	B1		11/2004	Stent et al.	
6,819,343	B1		11/2004	Sobeski et al.	715/848
6,819,345	B1		11/2004	Jones et al.	345/856
6,834,302	B1		12/2004	Harvell	709/224
6,842,779	B1		1/2005	Nishizawa	
6,879,994	B1		4/2005	Matsliach et al.	709/204
6,938,041	B1	*	8/2005	Brandow et al.	1/1
7,039,857	B2		5/2006	Beck et al.	715/500.1
7,039,859	B1		5/2006	Sundaresan	715/513
7,076,737	B2		7/2006	Abbott et al.	715/744
7,107,548	B2		9/2006	Shafron	
7,216,300	B2		5/2007	Dang	
7,222,303	B2		5/2007	Oren et al.	715/744
7,356,569	B1		4/2008	Kembel et al.	
7,574,649	B1		8/2009	Safars et al.	715/200
7,660,868	B1		2/2010	Kembel et al.	709/217
7,756,967	B1		7/2010	Kembel et al.	709/224
7,792,947	B1		9/2010	Kembel et al.	709/224
8,020,083	B1		9/2011	Kembel et al.	715/201
8,346,887	B1		1/2013	Kembel et al.	709/217
2001/0042107	A1		11/2001	Palm	709/218
2002/0065896	A1		5/2002	Burakoff et al.	709/206
2002/0078136	A1		6/2002	Brodsky et al.	709/203
2002/0089526	A1		7/2002	Buxton et al.	345/700
2002/0089536	A1		7/2002	Dang	345/749
2002/0091697	A1		7/2002	Huang et al.	707/10
2002/0130900	A1		9/2002	Davis	345/744
2002/0161879	A1		10/2002	Richard	709/223
2003/0051027	A1		3/2003	Aupperle et al.	709/224
2003/0069944	A1		4/2003	Barlock et al.	709/220
2004/0041836	A1		3/2004	Zaner et al.	345/751
2004/0165007	A1		8/2004	Shafron	715/781
2005/0273718	A1		12/2005	Naas	715/745
2008/0040681	A1		2/2008	Synstelien et al.	715/765
2008/0134018	A1		6/2008	Kembel et al.	715/234

2008/0163202	A1	7/2008	Kembel et al.	717/178
2008/0229217	A1	9/2008	Kembel et al.	715/760
2010/0235757	A1	9/2010	Kembel et al.	715/745
2010/0257442	A1	10/2010	Kembel et al.	715/234
2012/0117479	A1	5/2012	Kembel et al.	715/736

OTHER PUBLICATIONS

Alexa general faqs, 4 pages, www.alexa.com/whatisalexa/faq.html#general, Jan. 1999.
"Custom Explorer Bars Give Sites an Edge," 2 pages, www.microsoft.com/Windows/le/IE5/custom.asp, Jan. 1999.
"Flexibility Across the Web," 2 pages, www.microsoft.com/Windows/le/IE5/choice.asp, Jan. 1999.
"Web Accessories Overview," 2 pages, www.microsoft.com/workshop...er/accesory/overview/overiew.asp, Jan. 1999.
"Browser Extensions Overview," 2 pages, www.microsoft.com/workshop/browser/ext/overview/overview.asp, Jan. 1999.
Alexa Technology, 4 pages, www.alexa.com/support/technology.html, Jan. 1999.
"Creating Custom Explorer Bars and Desk Bands," 13 pages, www.microsoft.com/workshop/browser/ext/overview/Bands.asp, Jan. 1999.
Alexa Internet Tour, 1 page, www.alexa.com.whatisalexa/index/html, Jan. 1999.
"Revolutionary Ad Model," Advertise on Alexa, 1 page, www.alexa.com/company/advertise.html, Jan. 1999.
"The Alexa Service Appears on Your Desktop in Its Own Window," 1 page, www.alexa.com/tour/overview.html, Jan. 1999.
"Know More About the Sites You Visit," 1 page, www.alexa.com/tour/site_stats.html, Jan. 1999.
"Find Related Web Sites," 1 page, www.alexa.com/tour/related_links.html, Jan. 1999.
500,000 Sites and Growing,: 1 page, www.alexa.com/tour/archive.html, Jan. 1999.
"Research Tools at Your Fingertips," 1 page, www.alexa.com/tour/eb.html, Jan. 1999.
"Reporting," 1 page, www.alexa.com/company/reporting.html, Jan. 1999.
"Alexa Internet's Related Links Integrated Into Netscape Browser," 1 page, www.alexa.com/company/netscape.html, Jan. 1999.
"Demographics," alexa.com/company/demographics.html, Jan. 1999.
"Ads Appear in the Pop-up and on the Bar," 1 page, www.alexa.com/company/adspecs.html, Jan. 1999.
"Alexa Why Crawl," 1 page, www.alexa.com/support/why_crawl.html, Jan. 1999.
GIF Image 590×329 pixels, Alexa, 1 page, www.alexa.com/tour/images/alexa_overview.gif, Jan. 1999.
"It's X-treme!," Alexa, PC Magazine: The Best of 1998, 1 page, www.zdnet.com/pcmag/special/bestof98/internet5.html, Jan. 1999.
"Search While You Surf," PC Magazine: Search the Web, 1 page, www.zdnet.com/pcmag/features/websearch98/surf.html, Jan. 1999.
MindSpring, MyYahoo, pp. 1-16, www.mindspring.com/myyahoo/contents.htm, Dec. 1997.
Morrison, XML Unleashed, Sams Publishing, Dec. 21, 1999.
Flanagan, JavaScript; The Definitive Guide, 3rd Ed., O'Reilly, Jun. 1998.
Patent Application entitled "Parallel Web Sites", U.S. Appl. No. 09/192,633, filed Nov. 16, 1998.
Microsoft Computer Dictionary, Fifth Edition, 2002, Definition of "Web Browser".
Williams, Margot, "Cyberspace Calendars: The Web's Growing Date Base", Nov. 30, 1998, The Washington Post, p. 1.
Bott, Ed., et al., "Special Edition Using Windows 95 with Internet Explorer 4.0", Publisher: Que, Feb. 17, 1998, pp. 585 and 435.
McFedries, Paul, "Windows 98 Unleashed", Publisher: Sams, May 12, 1998, pp. 594-596.
McFedries, Paul, "Windows 98 Unleashed", Publisher: Sams Publishing, May 12, 1998, pp. xix, xx, xxi, xxiv, 2-4, 44-46, 69, 70, 79, 80, 97-116 ,148, 158-163, 251, 551, 787-792, 799-807, 885, 899, 900, and 904-906.
McCrickard, D. Scott, et al., "Supporting Information Awareness Using Animated Widgets", Proceedings of the 7th USENIX Tel/Tk Conference, Feb. 14-18, 2000, 12 pages.
"Streaming Internet Technologies", *PR Newswire*, May 18, 1999, 2 pages.
"NewsEdge Delivers the Power of Real-Time News in a Browser", *Business Wire*, Nov. 9, 1998, 3 pages.
McCartney, Terrance Paul, "End-User Construction and Configuration of Distributed Multimedia Applications", *ProQuest Dissertations and Theses*, 1996, 197 pages.
"Microsoft Eyes Marimba's Castanet", by CNET News.com Staff, Dec. 24, 1996, printed from <http://news.cnet.com/Microsoft-eyes-Marimbas-Castanet/2100-1001_3-257491.html>, 2 pages.
Williams, Dennis, "Application Delivery on a Grand Scale", *Network World Fusion*, Mar. 22, 1999, printed from <http://www.networkworld.com/reviews/0322revmarimba.html>, 4 pages.
Whitehead et al., "WEBDAV: IETF Standard for Collaborative Authoring on the Web", Sep. and Oct. 1998, Retrieved from the Internet <URL:ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=722228>, pp. 1-7 as printed.

* cited by examiner

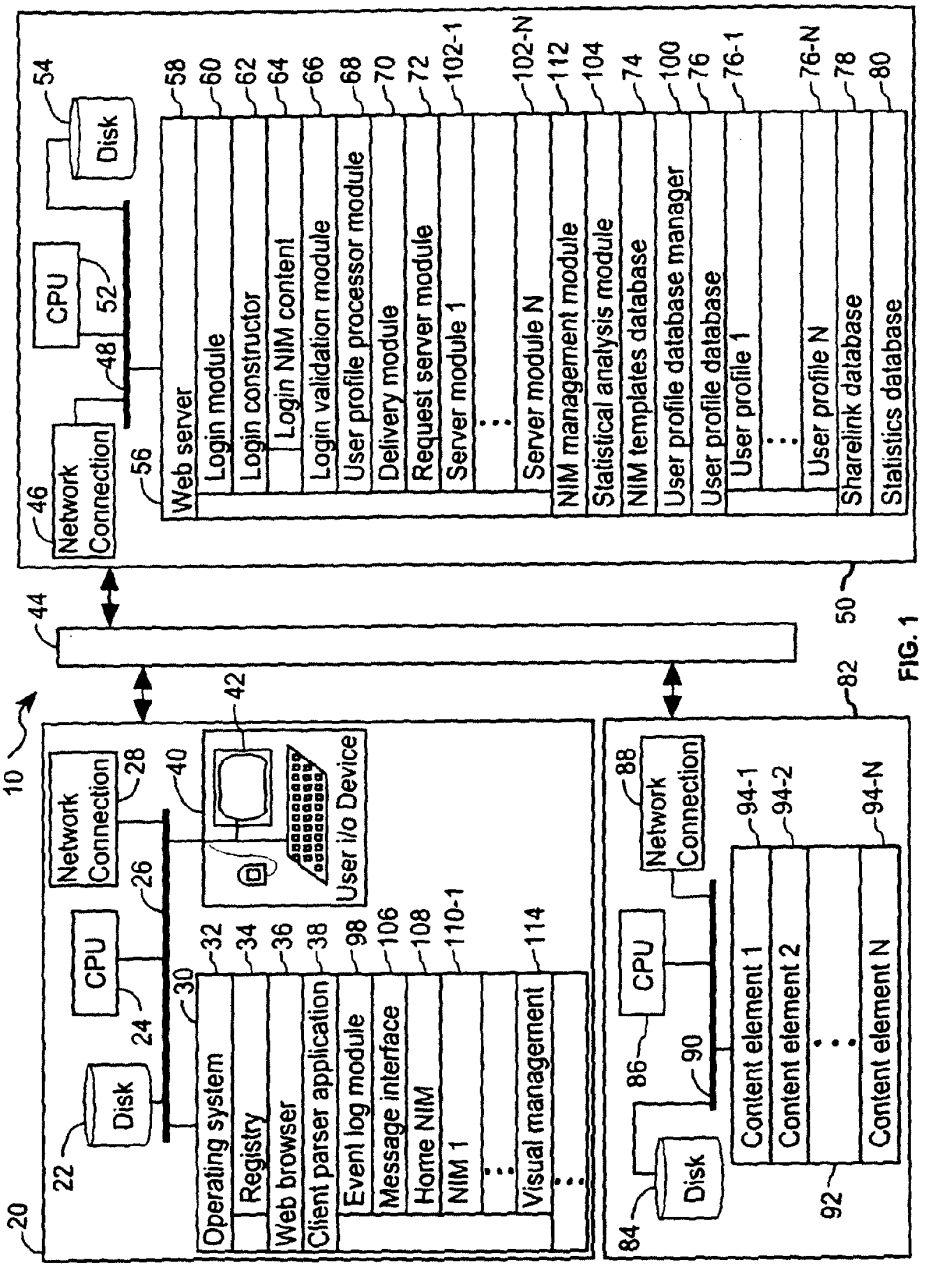

10
20
22
Disk
24
CPU
26
Network Connection
28
30
Operating system
32
Registry
34
Web browser
36
Client parser application
38
Event log module
98
Message interface
106
Home NIM
108
NIM 1
110-1
Visual management
114
40
42
User i/o Device
44
46
Network Connection
48
CPU
52
Disk
54
56
Web server
58
Login module
60
Login constructor
62
Login NIM content
64
Login validation module
66
User profile processor module
68
Delivery module
70
Request server module
72
Server module 1
102-1
Server module N
102-N
NIM management module
112
Statistical analysis module
104
NIM templates database
74
User profile database manager
100
User profile database
76
User profile 1
76-1
User profile N
76-N
Sharelink database
78
Statistics database
80
50
82
84
Disk
86
CPU
88
Network Connection
90
Content element 1
94-1
Content element 2
94-2
Content element N
94-N
92

FIG. 1

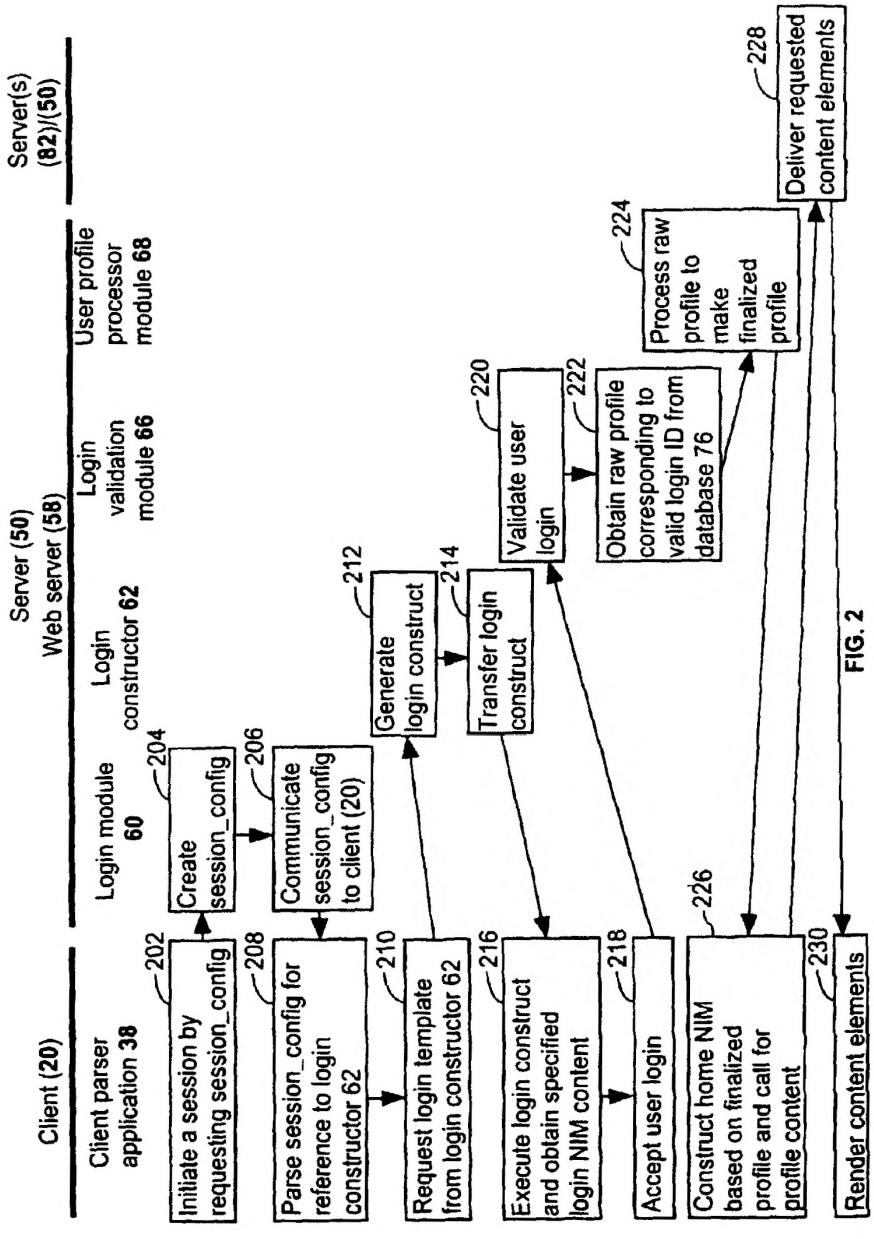

Client (20)
Server (50)
Web server (58)
Server(s) (82)/(50)
Client parser application 38
Login module 60
Login constructor 62
Login validation module 66
User profile processor module 68
202
Initiate a session by requesting session_config
204
Create session_config
206
Communicate session_config to client (20)
208
Parse session_config for reference to login constructor 62
210
Request login template from login constructor 62
212
Generate login construct
214
Transfer login construct
216
Execute login construct and obtain specified login NIM content
218
Accept user login
220
Validate user login
222
Obtain raw profile corresponding to valid login ID from database 76
224
Process raw profile to make finalized profile
226
Construct home NIM based on finalized profile and call for profile content
228
Deliver requested content elements
230
Render content elements

FIG. 2

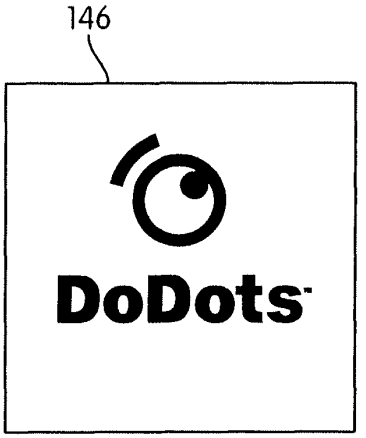

146
DoDots

FIG. 3A

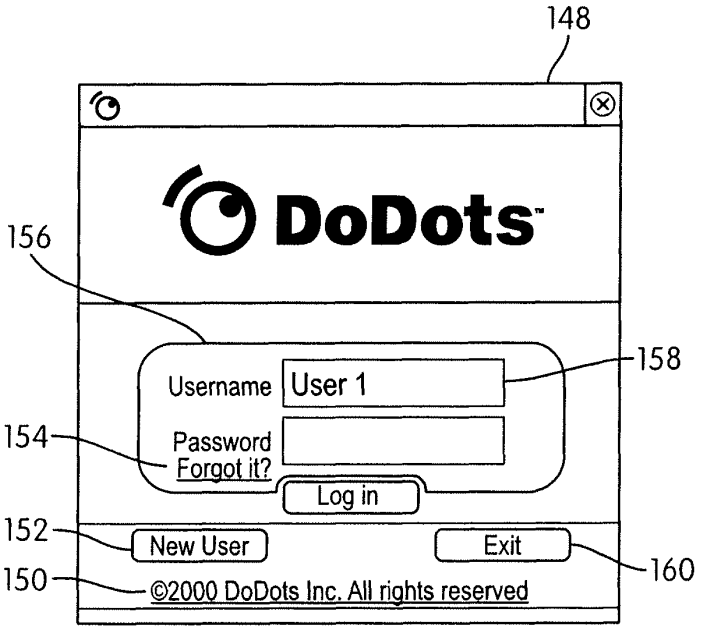

148
DoDots
156
158
Username
User 1
154
Password
Forgot it?
Log in
152
New User
Exit
160
150
©2000 DoDots Inc. All rights reserved

FIG. 3B

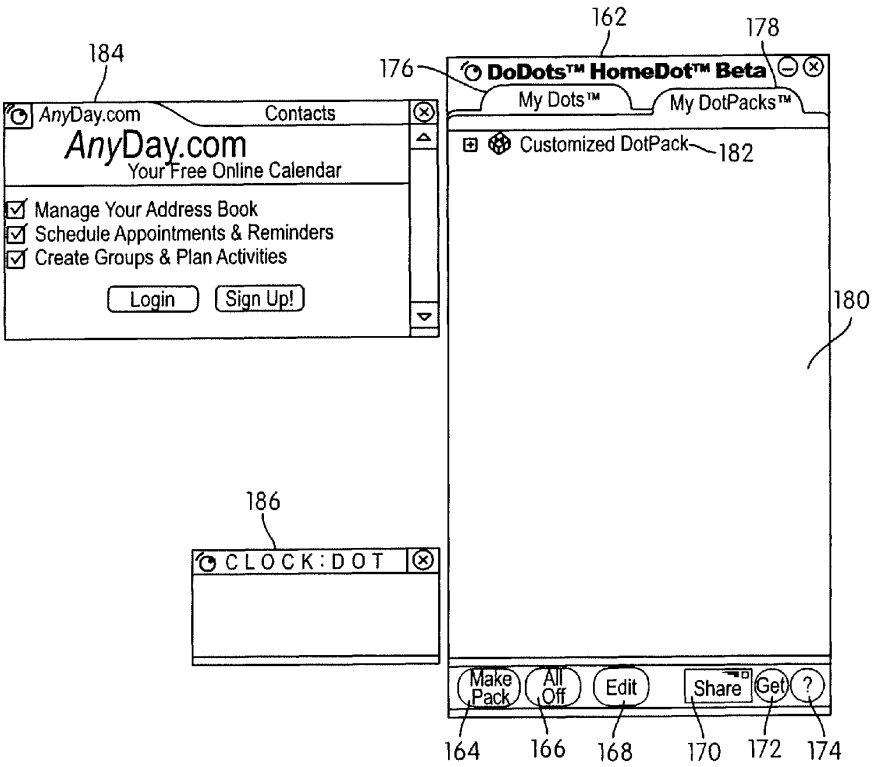

184
AnyDay.com
Contacts
AnyDay.com
Your Free Online Calendar
Manage Your Address Book
Schedule Appointments & Reminders
Create Groups & Plan Activities
Login
Sign Up!
176
162
178
DoDots™ HomeDot™ Beta
My Dots™
My DotPacks™
Customized DotPack
182
180
186
CLOCK:DOT
Make Pack
All Off
Edit
Share
Get
?
164
166
168
170
172
174

FIG. 4

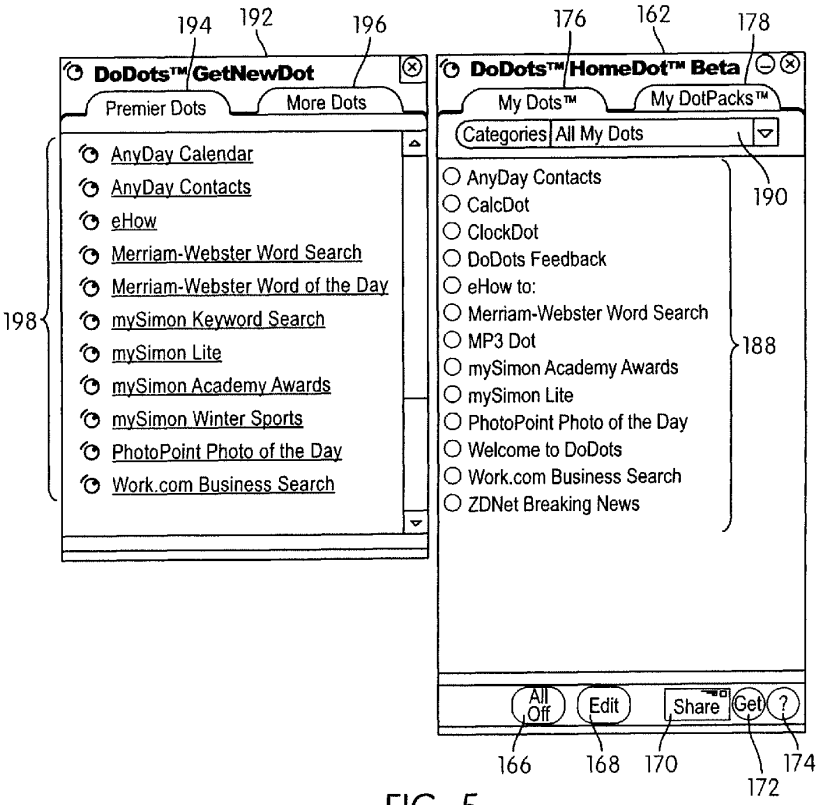

194
192
196
176
162
178
DoDots™/GetNewDot
Premier Dots
More Dots
AnyDay Calendar
AnyDay Contacts
eHow
Merriam-Webster Word Search
Merriam-Webster Word of the Day
mySimon Keyword Search
mySimon Lite
mySimon Academy Awards
mySimon Winter Sports
PhotoPoint Photo of the Day
Work.com Business Search
198
DoDots™/HomeDot™ Beta
My Dots™
My DotPacks™
Categories
All My Dots
190
AnyDay Contacts
CalcDot
ClockDot
DoDots Feedback
eHow to:
Merriam-Webster Word Search
MP3 Dot
mySimon Academy Awards
mySimon Lite
PhotoPoint Photo of the Day
Welcome to DoDots
Work.com Business Search
ZDNet Breaking News
188
All Off
Edit
Share
Get
?
166
168
170
172
174

FIG. 5

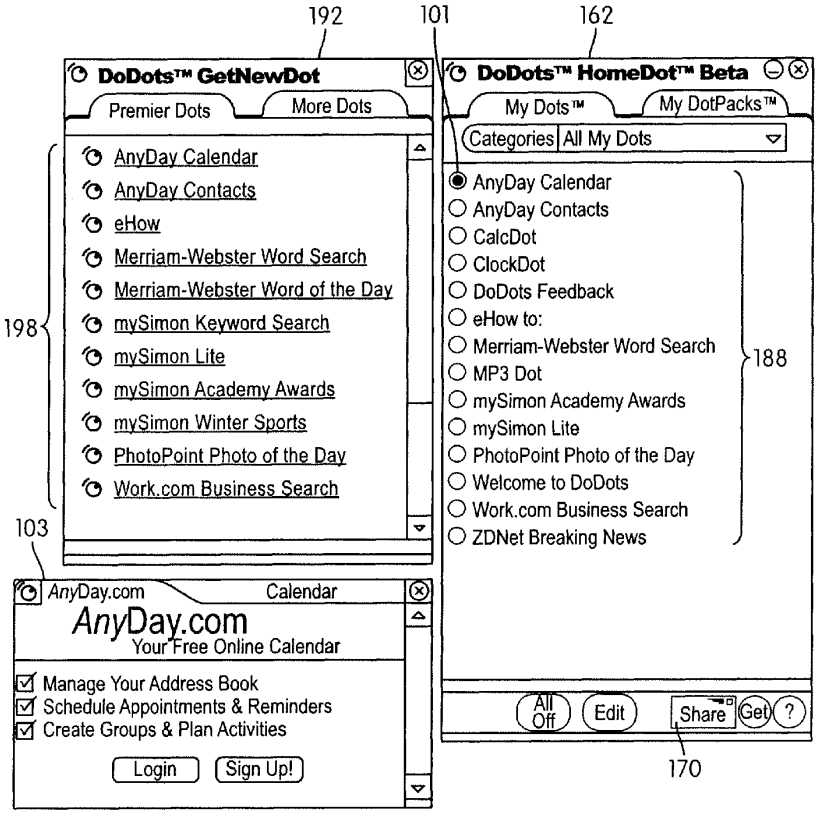

192
101
162
DoDots™ GetNewDot
Premier Dots
More Dots
AnyDay Calendar
AnyDay Contacts
eHow
Merriam-Webster Word Search
Merriam-Webster Word of the Day
mySimon Keyword Search
mySimon Lite
mySimon Academy Awards
mySimon Winter Sports
PhotoPoint Photo of the Day
Work.com Business Search
198
103
AnyDay.com
Calendar
AnyDay.com
Your Free Online Calendar
Manage Your Address Book
Schedule Appointments & Reminders
Create Groups & Plan Activities
Login
Sign Up!
DoDots™ HomeDot™ Beta
My Dots™
My DotPacks™
Categories
All My Dots
AnyDay Calendar
AnyDay Contacts
CalcDot
ClockDot
DoDots Feedback
eHow to:
Merriam-Webster Word Search
MP3 Dot
mySimon Academy Awards
mySimon Lite
PhotoPoint Photo of the Day
Welcome to DoDots
Work.com Business Search
ZDNet Breaking News
188
All Off
Edit
Share
Get
?
170

FIG. 6

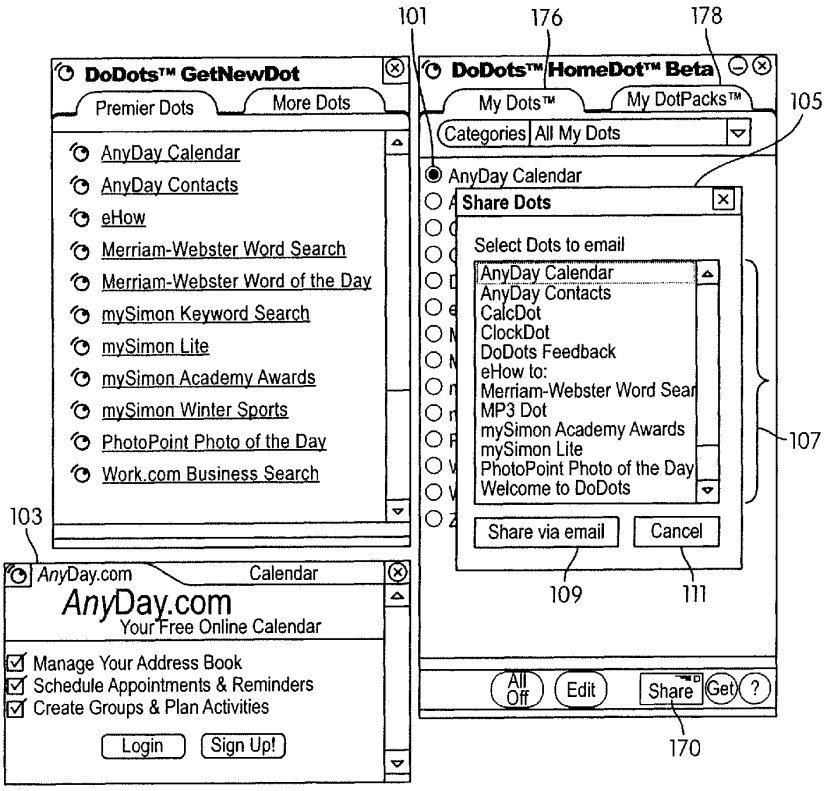

101
176
178
DoDots™ GetNewDot
Premier Dots
More Dots
AnyDay Calendar
AnyDay Contacts
eHow
Merriam-Webster Word Search
Merriam-Webster Word of the Day
mySimon Keyword Search
mySimon Lite
mySimon Academy Awards
mySimon Winter Sports
PhotoPoint Photo of the Day
Work.com Business Search
DoDots™ HomeDot™ Beta
My Dots™
My DotPacks™
105
Categories
All My Dots
AnyDay Calendar
Share Dots
Select Dots to email
AnyDay Calendar
AnyDay Contacts
CalcDot
ClockDot
DoDots Feedback
eHow to:
Merriam-Webster Word Sear
MP3 Dot
mySimon Academy Awards
mySimon Lite
PhotoPoint Photo of the Day
Welcome to DoDots
107
Share via email
Cancel
109
111
103
AnyDay.com
Calendar
AnyDay.com
Your Free Online Calendar
Manage Your Address Book
Schedule Appointments & Reminders
Create Groups & Plan Activities
Login
Sign Up!
All Off
Edit
Share
Get
?
170

FIG. 7

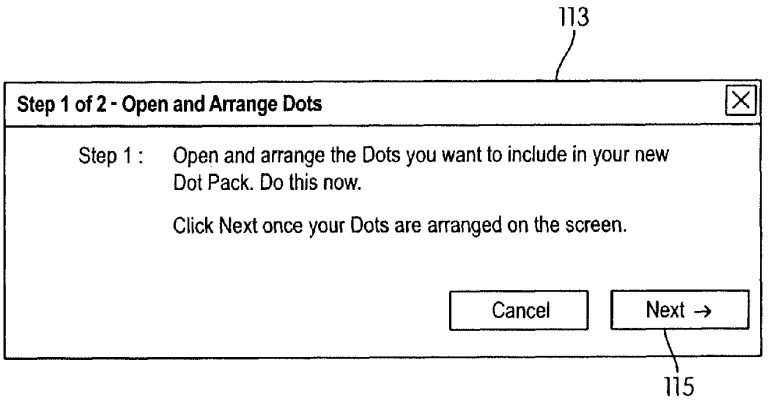

113
Step 1 of 2 - Open and Arrange Dots
Step 1 :
Open and arrange the Dots you want to include in your new Dot Pack. Do this now.
Click Next once your Dots are arranged on the screen.
Cancel
Next →
115

FIG. 8A

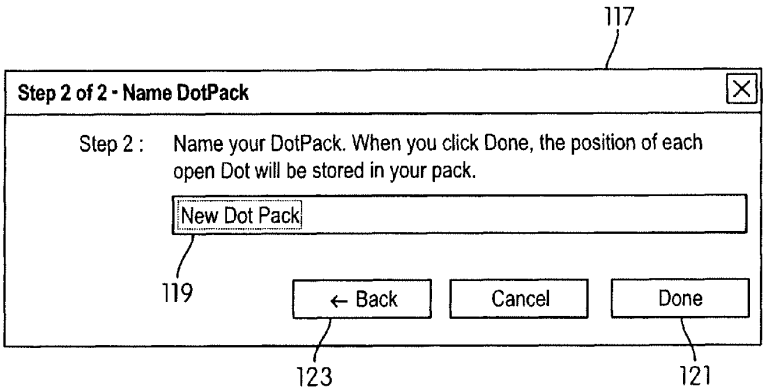

117
Step 2 of 2 - Name DotPack
Step 2 :
Name your DotPack. When you click Done, the position of each open Dot will be stored in your pack.
New Dot Pack
119
← Back
Cancel
Done
123
121

FIG. 8B

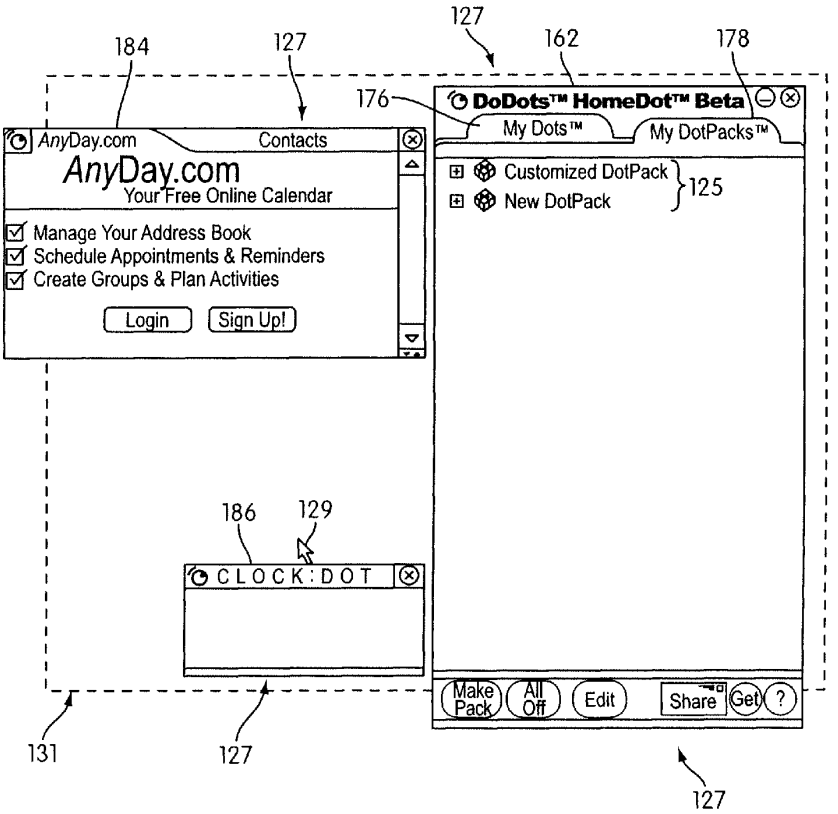

127
184
127
162
178
176
DoDots™ HomeDot™ Beta
My Dots™
My DotPacks™
AnyDay.com
Contacts
AnyDay.com
Your Free Online Calendar
Manage Your Address Book
Schedule Appointments & Reminders
Create Groups & Plan Activities
Login
Sign Up!
Customized DotPack
New DotPack
125
186
129
CLOCK:DOT
Make Pack
All Off
Edit
Share
Get
?
131
127
127

FIG. 9A

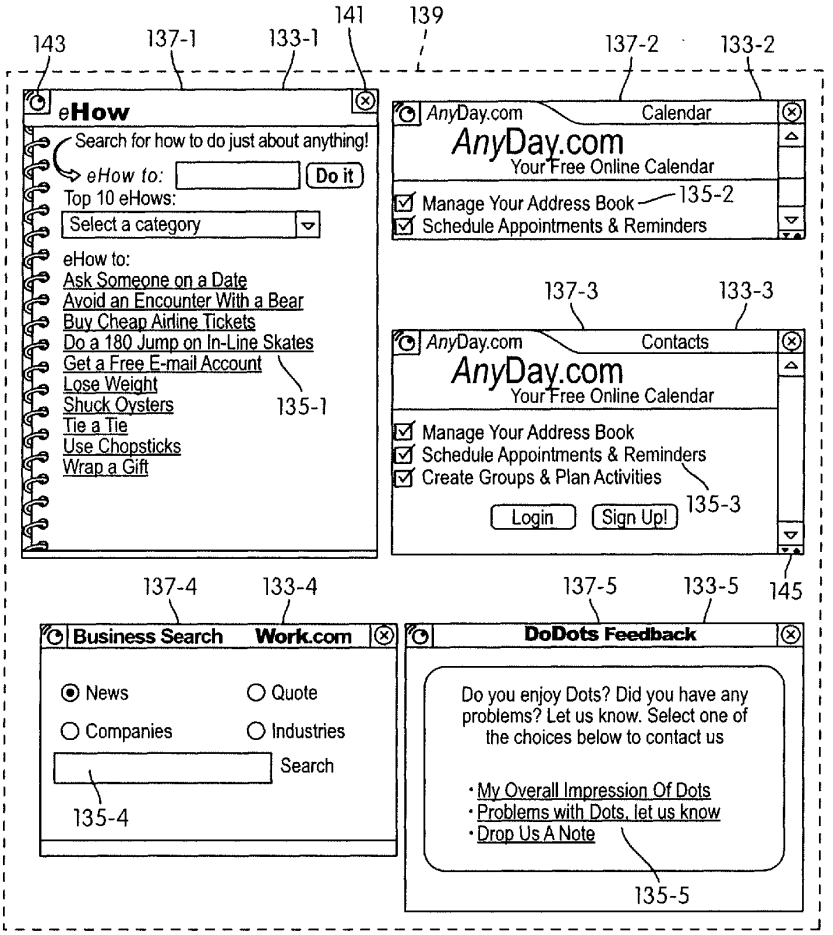

143
137-1
133-1
141
139
137-2
133-2
eHow
Search for how to do just about anything!
eHow to:
Do it
Top 10 eHows:
Select a category
eHow to:
Ask Someone on a Date
Avoid an Encounter With a Bear
Buy Cheap Airline Tickets
Do a 180 Jump on In-Line Skates
Get a Free E-mail Account
Lose Weight
Shuck Oysters
Tie a Tie
Use Chopsticks
Wrap a Gift
135-1
AnyDay.com
Calendar
AnyDay.com
Your Free Online Calendar
Manage Your Address Book
Schedule Appointments & Reminders
135-2
137-3
133-3
AnyDay.com
Contacts
AnyDay.com
Your Free Online Calendar
Manage Your Address Book
Schedule Appointments & Reminders
Create Groups & Plan Activities
Login
Sign Up!
135-3
145
137-4
133-4
Business Search
Work.com
News
Quote
Companies
Industries
Search
135-4
137-5
133-5
DoDots Feedback
Do you enjoy Dots? Did you have any problems? Let us know. Select one of the choices below to contact us
My Overall Impression Of Dots
Problems with Dots, let us know
Drop Us A Note
135-5

FIG. 9B

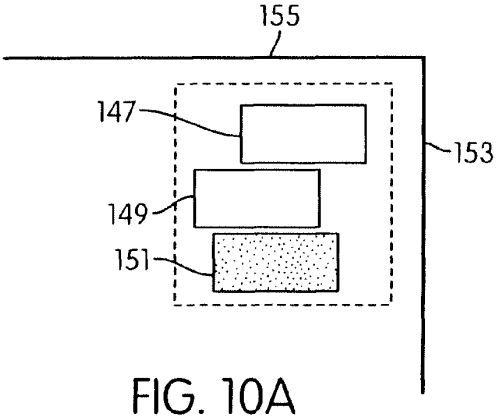


FIG. 10A

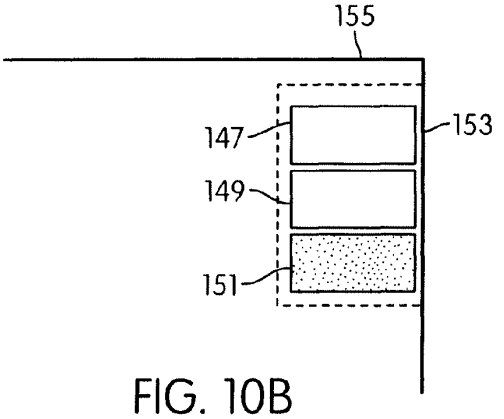


FIG. 10B

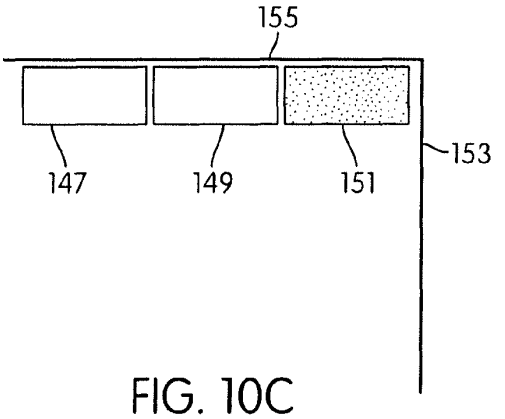


FIG. 10C

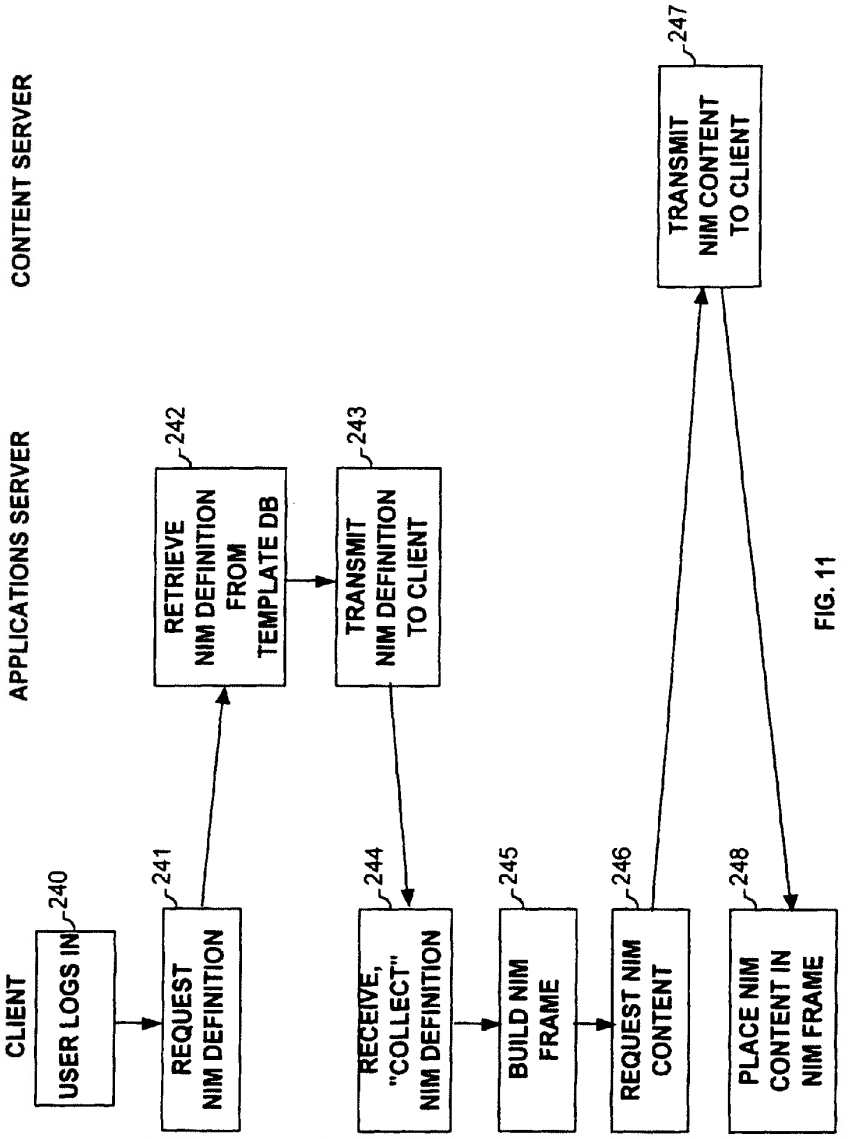

CLIENT
APPLICATIONS SERVER
CONTENT SERVER
USER LOGS IN
240
REQUEST NIM DEFINITION
241
RETRIEVE NIM DEFINITION FROM TEMPLATE DB
242
TRANSMIT NIM DEFINITION TO CLIENT
243
RECEIVE, "COLLECT" NIM DEFINITION
244
BUILD NIM FRAME
245
REQUEST NIM CONTENT
246
TRANSMIT NIM CONTENT TO CLIENT
247
PLACE NIM CONTENT IN NIM FRAME
248

FIG. 11

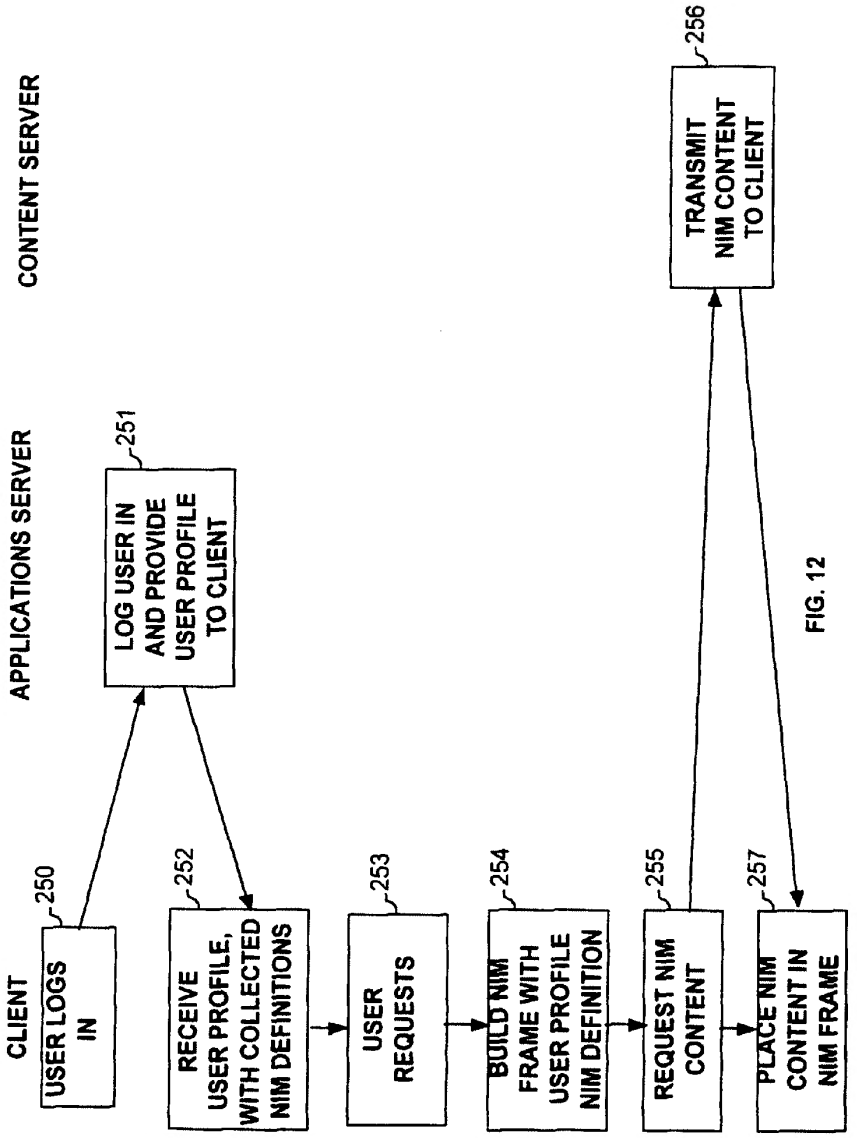

CLIENT
APPLICATIONS SERVER
CONTENT SERVER
250
USER LOGS IN
251
LOG USER IN AND PROVIDE USER PROFILE TO CLIENT
252
RECEIVE USER PROFILE, WITH COLLECTED NIM DEFINITIONS
253
USER REQUESTS
254
BUILD NIM FRAME WITH USER PROFILE NIM DEFINITION
255
REQUEST NIM CONTENT
256
TRANSMIT NIM CONTENT TO CLIENT
257
PLACE NIM CONTENT IN NIM FRAME

FIG. 12

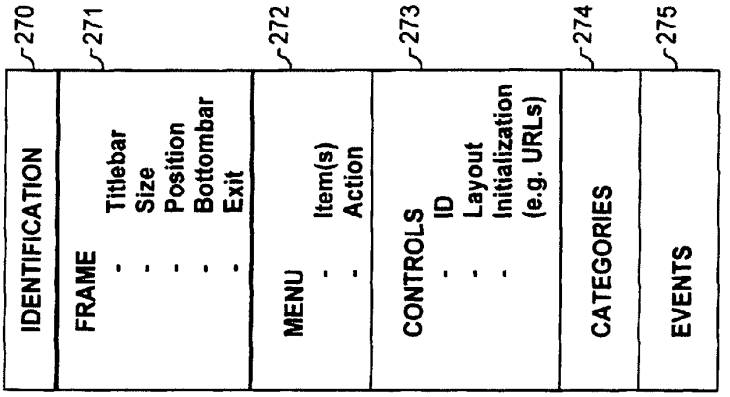

270
IDENTIFICATION
271
FRAME
- Titlebar
- Size
- Position
- Bottombar
- Exit
272
MENU
- Item(s)
- Action
273
CONTROLS
- ID
- Layout
- Initialization (e.g. URLs)
274
CATEGORIES
275
EVENTS

FIG. 13

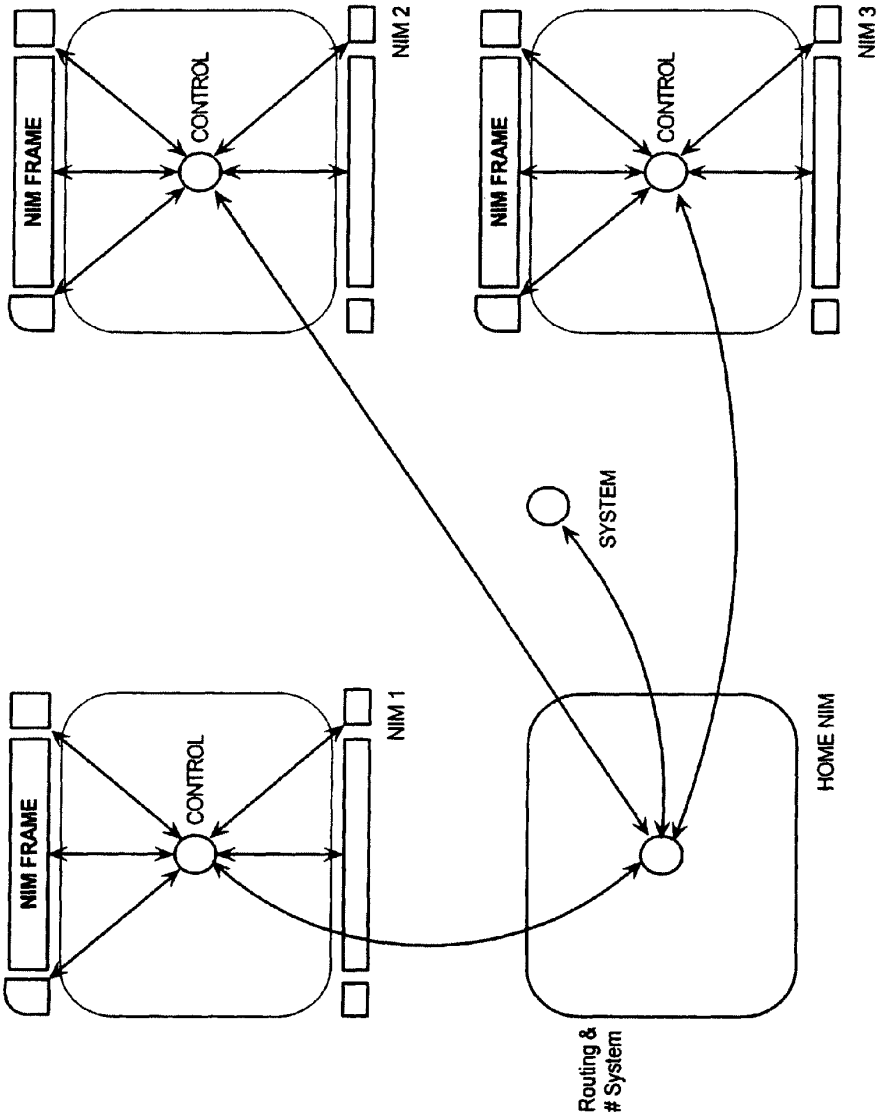

NIM FRAME
CONTROL
NIM 1
NIM FRAME
CONTROL
NIM 2
NIM FRAME
CONTROL
NIM 3
SYSTEM
HOME NIM
Routing &
System

FIG. 14

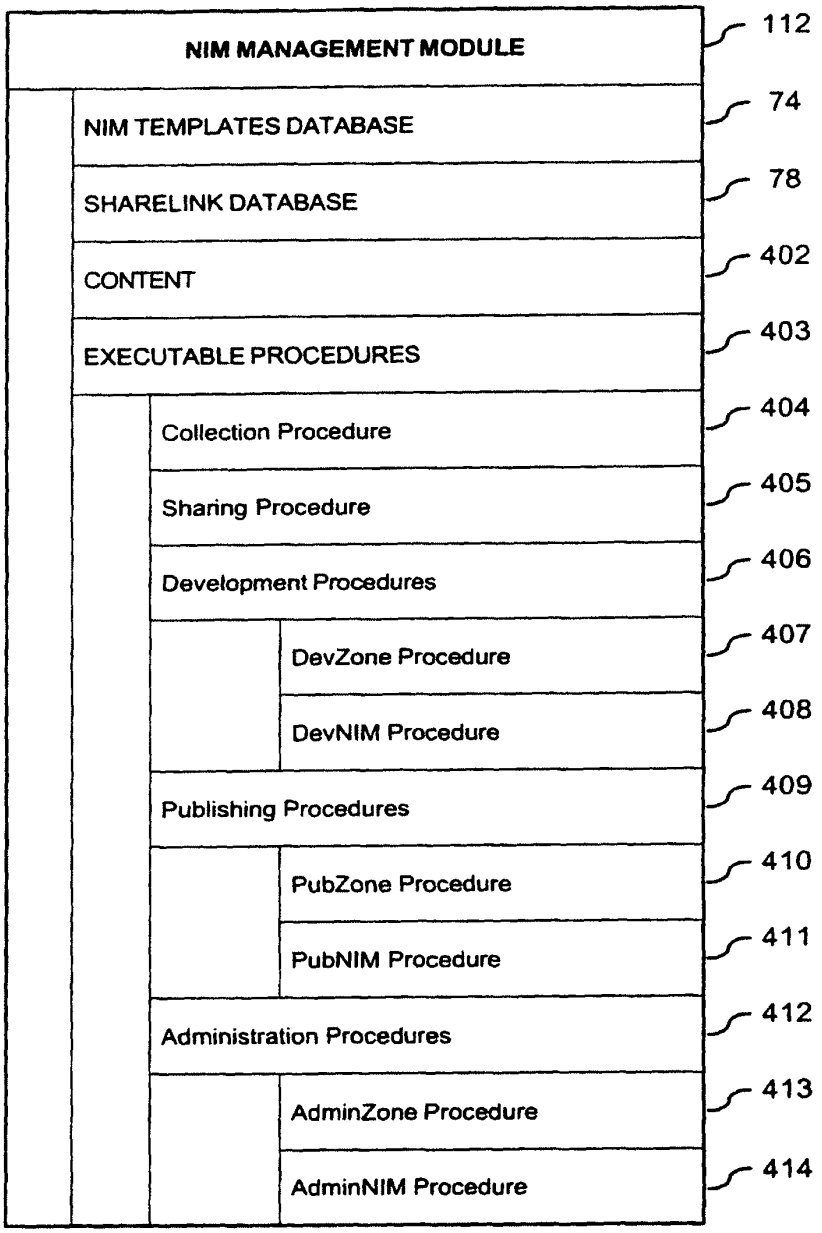

NIM MANAGEMENT MODULE
112
NIM TEMPLATES DATABASE
74
SHARELINK DATABASE
78
CONTENT
402
EXECUTABLE PROCEDURES
403
Collection Procedure
404
Sharing Procedure
405
Development Procedures
406
DevZone Procedure
407
DevNIM Procedure
408
Publishing Procedures
409
PubZone Procedure
410
PubNIM Procedure
411
Administration Procedures
412
AdminZone Procedure
413
AdminNIM Procedure
414

FIG. 15

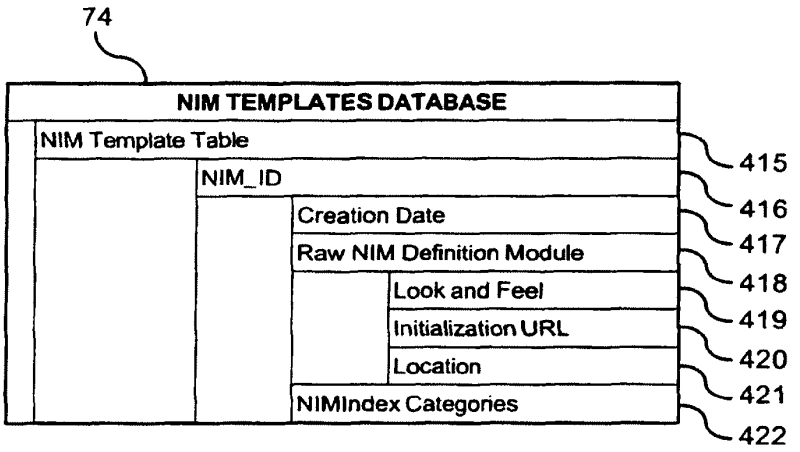

74
NIM TEMPLATES DATABASE
NIM Template Table
NIM_ID
Creation Date
Raw NIM Definition Module
Look and Feel
Initialization URL
Location
NIMIndex Categories
415
416
417
418
419
420
421
422

FIG. 16

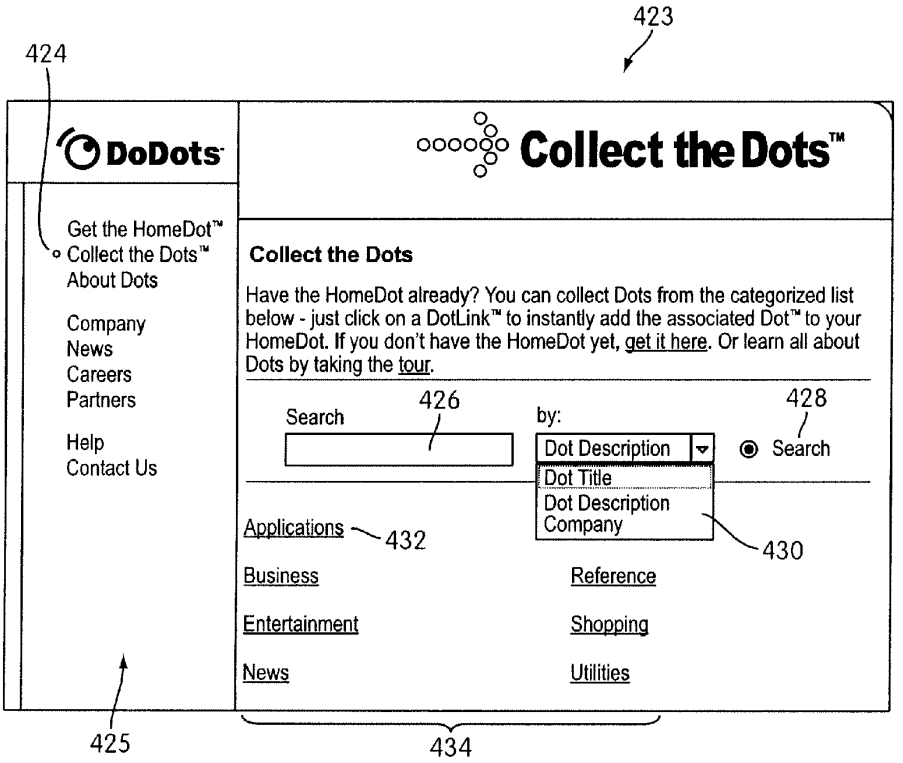

423
424
DoDots
Collect the Dots™
Get the HomeDot™
Collect the Dots™
About Dots
Company
News
Careers
Partners
Help
Contact Us
Collect the Dots
Have the HomeDot already? You can collect Dots from the categorized list below - just click on a DotLink™ to instantly add the associated Dot™ to your HomeDot. If you don't have the HomeDot yet, get it here. Or learn all about Dots by taking the tour.
Search
426
by:
428
Dot Description
Search
Dot Title
Dot Description
Company
430
Applications
432
Business
Reference
Entertainment
Shopping
News
Utilities
425
434

FIG. 17

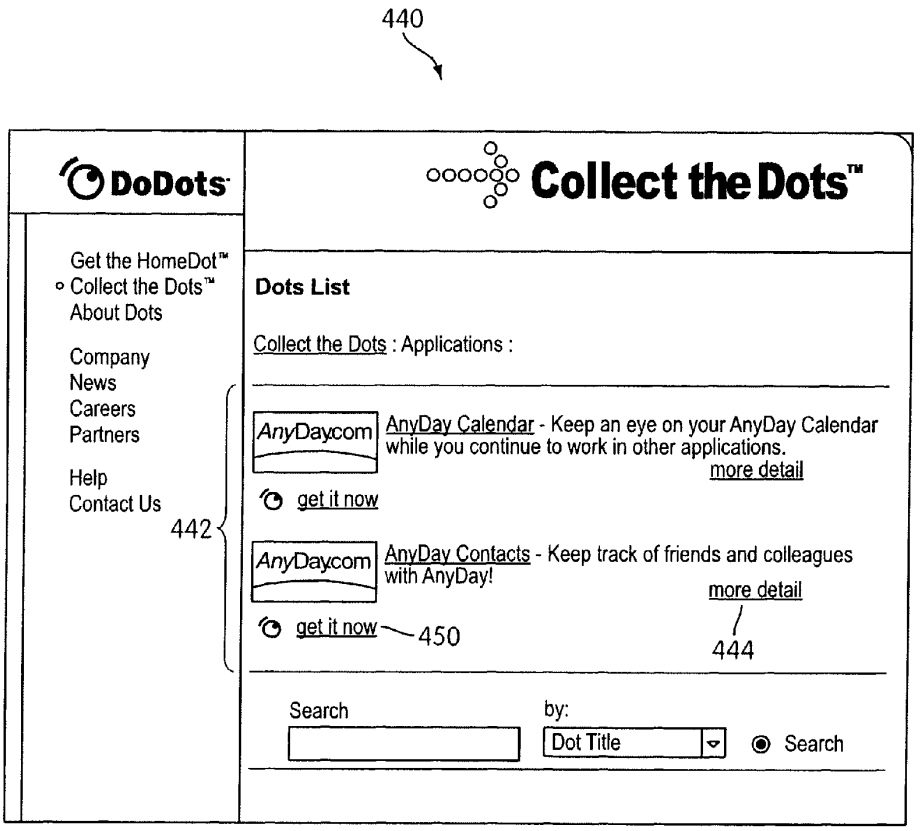

440
DoDots
Collect the Dots™
Get the HomeDot™
Collect the Dots™
About Dots
Company
News
Careers
Partners
Help
Contact Us
442
Dots List
Collect the Dots : Applications :
AnyDay.com
AnyDay Calendar - Keep an eye on your AnyDay Calendar while you continue to work in other applications.
more detail
get it now
AnyDay.com
AnyDay Contacts - Keep track of friends and colleagues with AnyDay!
more detail
get it now
450
444
Search
by:
Dot Title
Search

FIG. 18

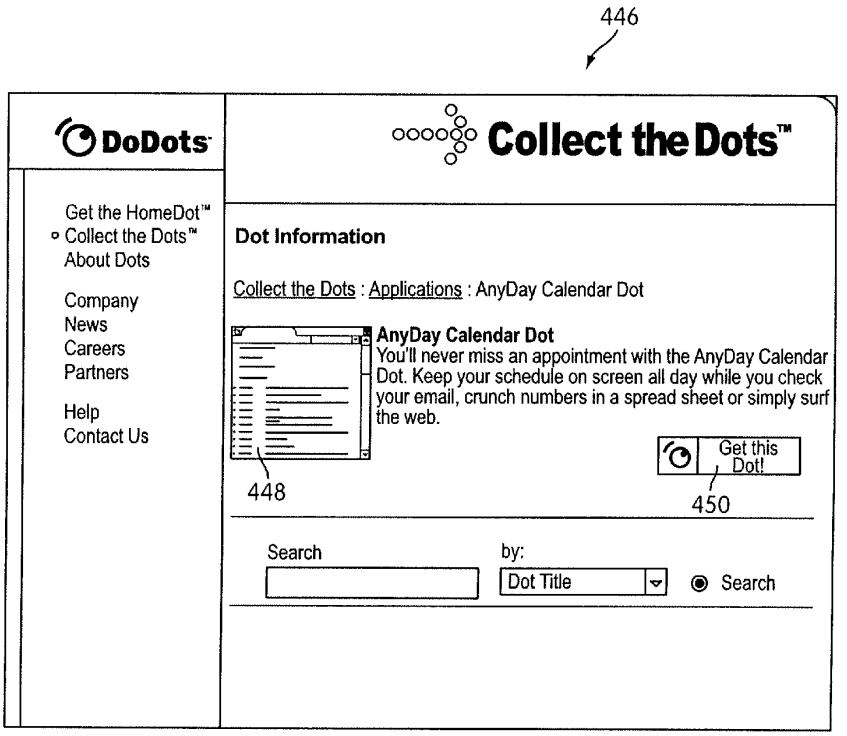

446
DoDots
Collect the Dots™
Get the HomeDot™
Collect the Dots™
About Dots
Company
News
Careers
Partners
Help
Contact Us
Dot Information
Collect the Dots : Applications : AnyDay Calendar Dot
AnyDay Calendar Dot
You'll never miss an appointment with the AnyDay Calendar Dot. Keep your schedule on screen all day while you check your email, crunch numbers in a spread sheet or simply surf the web.
Get this Dot!
448
450
Search
by:
Dot Title
Search

FIG. 19

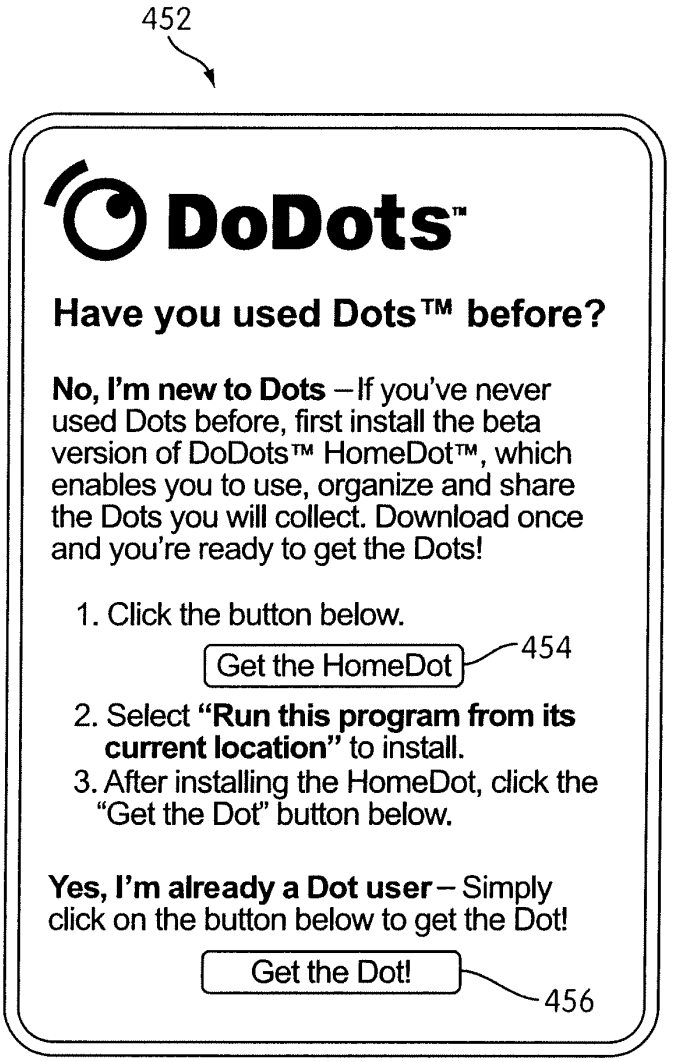

452
DoDots™
Have you used Dots™ before?
No, I'm new to Dots – If you've never used Dots before, first install the beta version of DoDots™ HomeDot™, which enables you to use, organize and share the Dots you will collect. Download once and you're ready to get the Dots!
1. Click the button below.
Get the HomeDot
454
2. Select "Run this program from its current location" to install.
3. After installing the HomeDot, click the "Get the Dot" button below.
Yes, I'm already a Dot user – Simply click on the button below to get the Dot!
Get the Dot!
456

FIG. 20

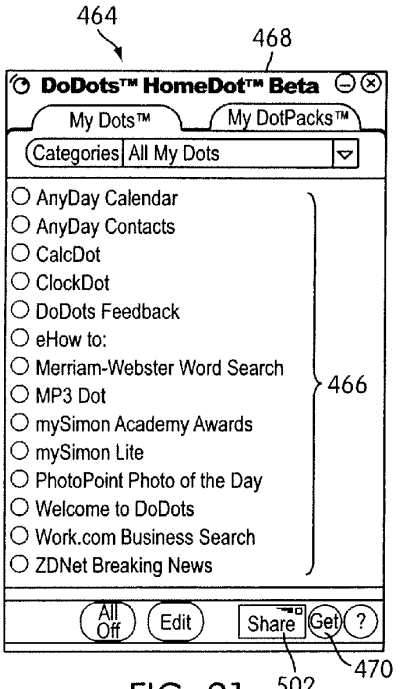


FIG. 21

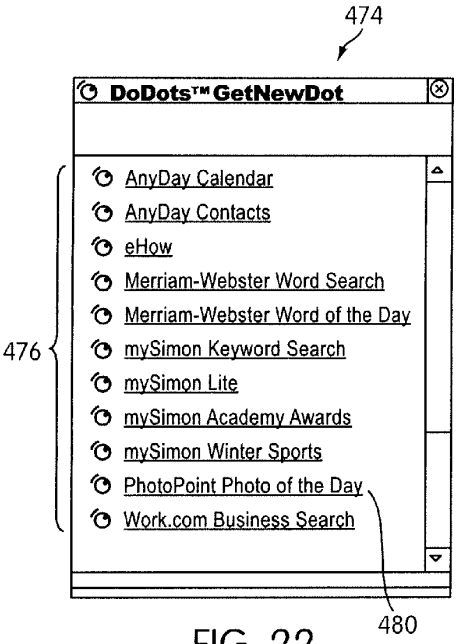


FIG. 22

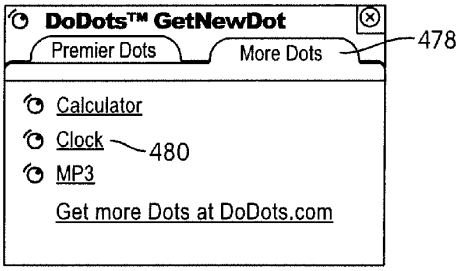


FIG. 23

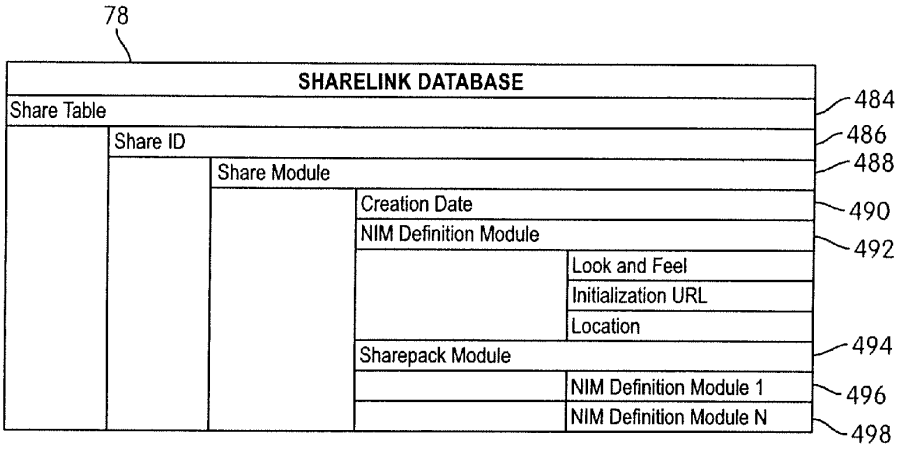

78
SHARELINK DATABASE
Share Table
Share ID
Share Module
Creation Date
NIM Definition Module
Look and Feel
Initialization URL
Location
Sharepack Module
NIM Definition Module 1
NIM Definition Module N
484
486
488
490
492
494
496
498

FIG. 24

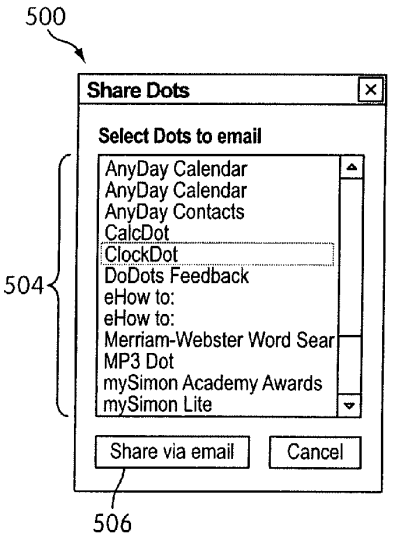

500
Share Dots
Select Dots to email
AnyDay Calendar
AnyDay Calendar
AnyDay Contacts
CalcDot
ClockDot
DoDots Feedback
eHow to:
eHow to:
Merriam-Webster Word Sear
MP3 Dot
mySimon Academy Awards
mySimon Lite
504
Share via email
Cancel
506

FIG. 25

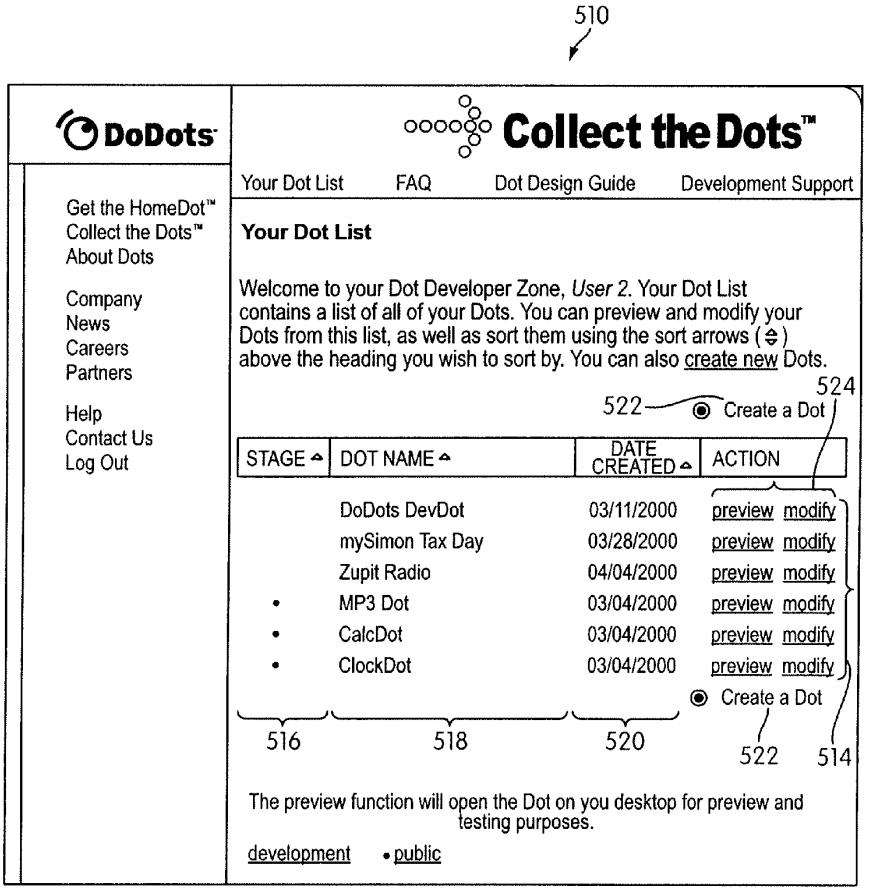

510
DoDots
Collect the Dots™
Your Dot List
FAQ
Dot Design Guide
Development Support
Get the HomeDot™
Collect the Dots™
About Dots
Company
News
Careers
Partners
Help
Contact Us
Log Out
Your Dot List
Welcome to your Dot Developer Zone, User 2. Your Dot List contains a list of all of your Dots. You can preview and modify your Dots from this list, as well as sort them using the sort arrows ( ) above the heading you wish to sort by. You can also create new Dots.
524
522
Create a Dot
STAGE
DOT NAME
DATE CREATED
ACTION
DoDots DevDot 03/11/2000 preview modify
mySimon Tax Day 03/28/2000 preview modify
Zupit Radio 04/04/2000 preview modify
MP3 Dot 03/04/2000 preview modify
CalcDot 03/04/2000 preview modify
ClockDot 03/04/2000 preview modify
Create a Dot
516
518
520
522
514
The preview function will open the Dot on you desktop for preview and testing purposes.
development
public

FIG. 26

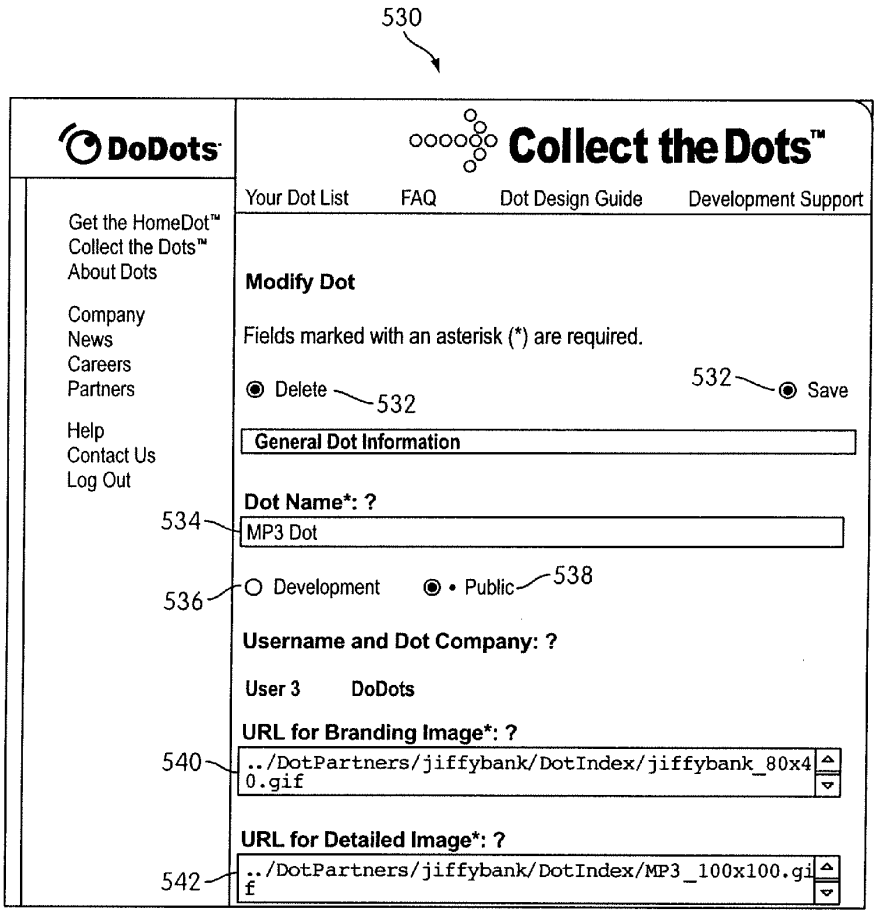

530
DoDots
Collect the Dots™
Your Dot List
FAQ
Dot Design Guide
Development Support
Get the HomeDot™
Collect the Dots™
About Dots
Company
News
Careers
Partners
Help
Contact Us
Log Out
Modify Dot
Fields marked with an asterisk (*) are required.
Delete
532
532
Save
General Dot Information
Dot Name*: ?
534
MP3 Dot
536
Development
Public
538
Username and Dot Company: ?
User 3
DoDots
URL for Branding Image*: ?
540
../DotPartners/jiffybank/DotIndex/jiffybank_80x40.gif
URL for Detailed Image*: ?
542
../DotPartners/jiffybank/DotIndex/MP3_100x100.gif

FIG. 27

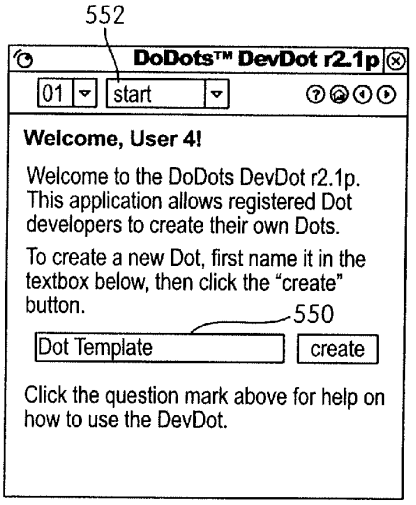


FIG. 28A

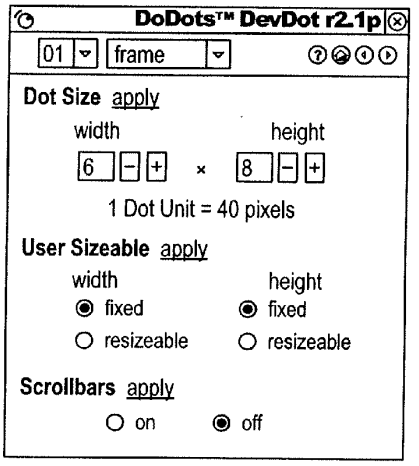


FIG. 28B

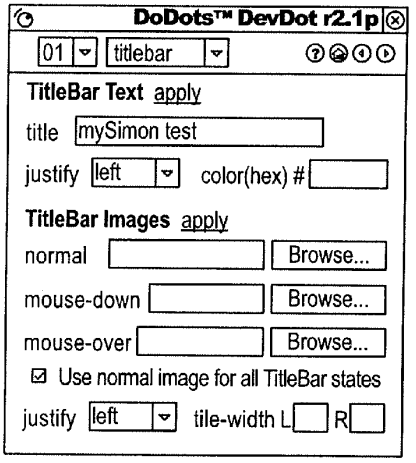


FIG. 28C

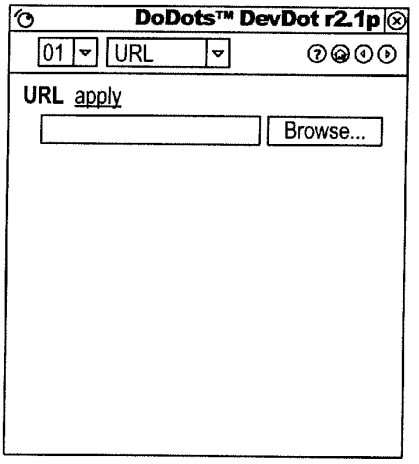


FIG. 28D

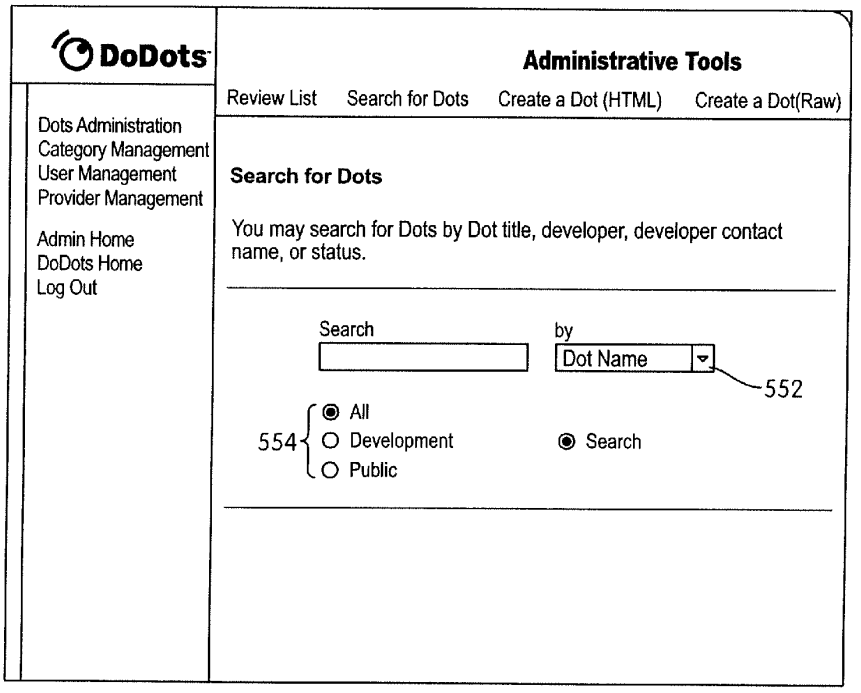

DoDots
Administrative Tools
Review List
Search for Dots
Create a Dot (HTML)
Create a Dot(Raw)
Dots Administration
Category Management
User Management
Provider Management
Admin Home
DoDots Home
Log Out
Search for Dots
You may search for Dots by Dot title, developer, developer contact name, or status.
Search
by
Dot Name
552
554
All
Development
Public
Search

FIG. 29A

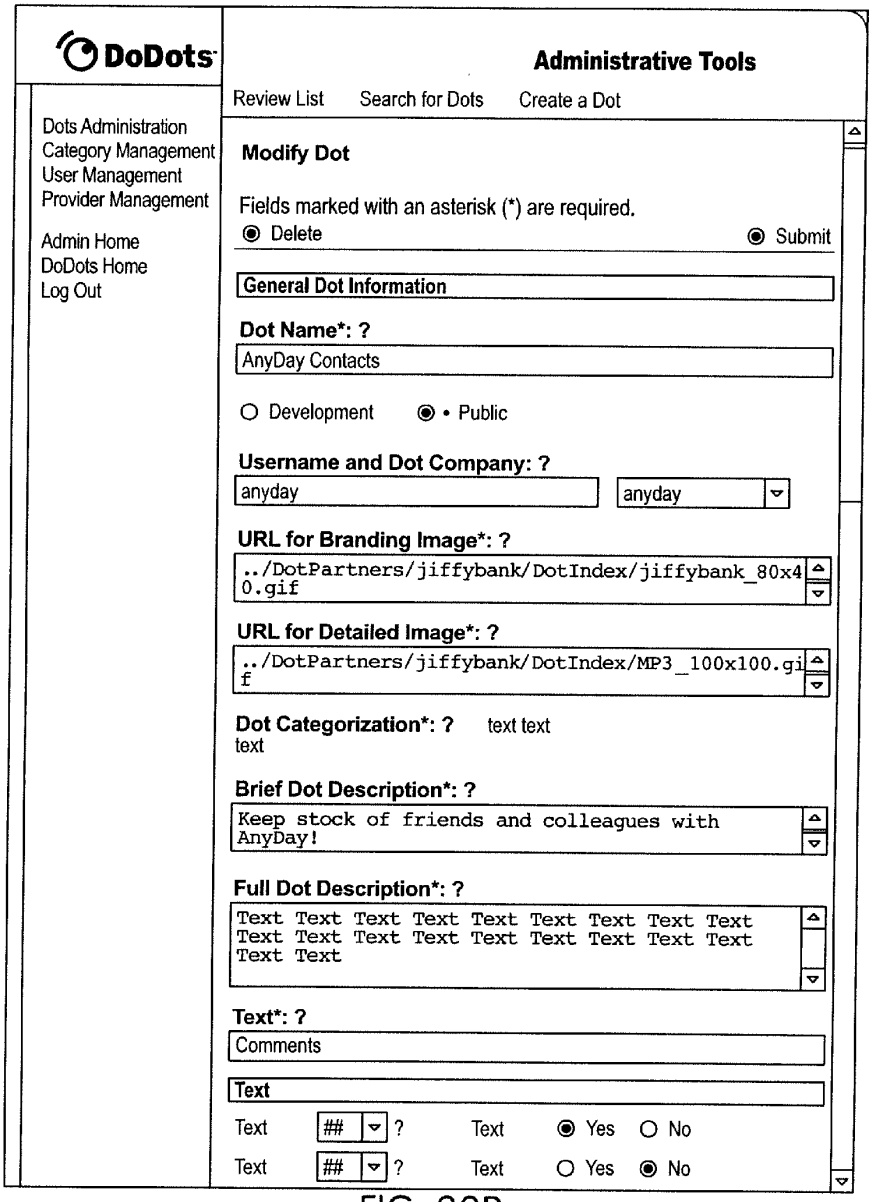

DoDots
Administrative Tools
Review List
Search for Dots
Create a Dot
Dots Administration
Category Management
User Management
Provider Management
Admin Home
DoDots Home
Log Out
Modify Dot
Fields marked with an asterisk (*) are required.
Delete
Submit
General Dot Information
Dot Name*: ?
AnyDay Contacts
Development
Public
Username and Dot Company: ?
anyday
anyday
URL for Branding Image*: ?
../DotPartners/jiffybank/DotIndex/jiffybank_80x40.gif
URL for Detailed Image*: ?
../DotPartners/jiffybank/DotIndex/MP3_100x100.gif
Dot Categorization*: ?
text text text
Brief Dot Description*: ?
Keep stock of friends and colleagues with AnyDay!
Full Dot Description*: ?
Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text
Text*: ?
Comments
Text
Text
##
?
Text
Yes
No
Text
##
?
Text
Yes
No

FIG. 29B

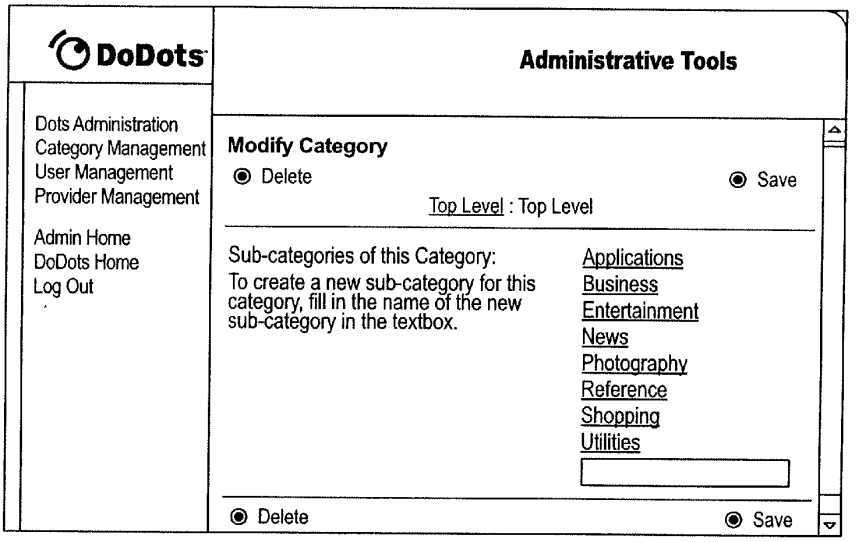


FIG. 30A

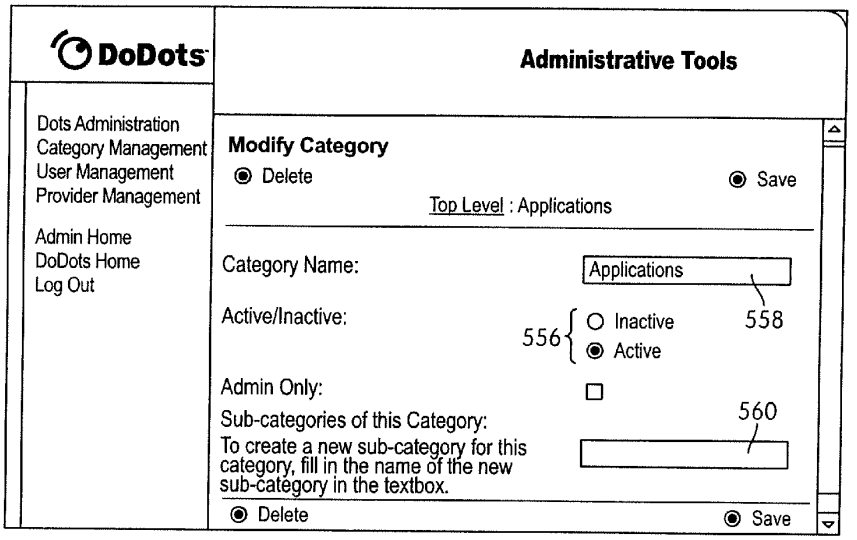


FIG. 30B

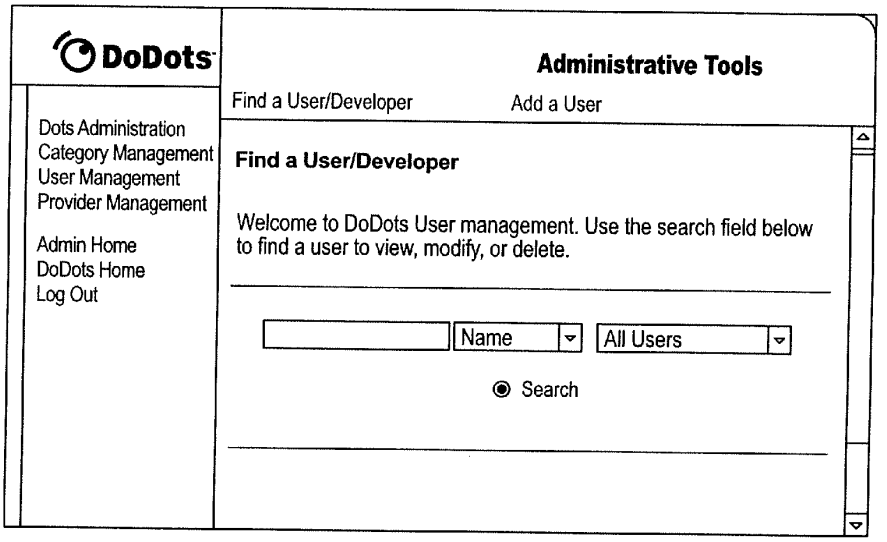

DoDots
Administrative Tools
Find a User/Developer
Add a User
Dots Administration
Category Management
User Management
Provider Management
Admin Home
DoDots Home
Log Out
Find a User/Developer
Welcome to DoDots User management. Use the search field below to find a user to view, modify, or delete.
Name
All Users
Search

FIG. 31A

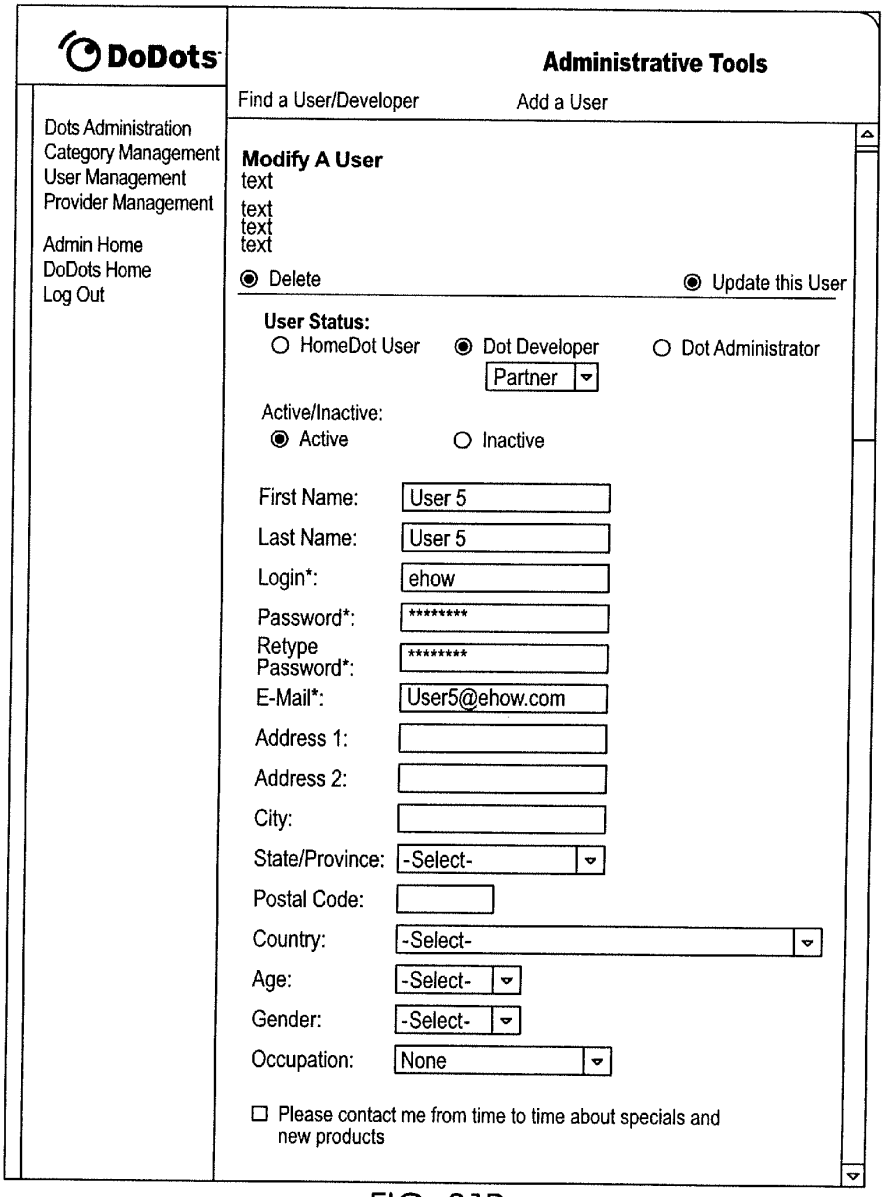

DoDots
Administrative Tools
Find a User/Developer
Add a User
Dots Administration
Category Management
User Management
Provider Management
Admin Home
DoDots Home
Log Out
Modify A User
text
text
text
text
Delete
Update this User
User Status:
HomeDot User
Dot Developer
Dot Administrator
Partner
Active/Inactive:
Active
Inactive
First Name:
User 5
Last Name:
User 5
Login*:
ehow
Password*:
****
Retype Password*:
****
E-Mail*:
User5@ehow.com
Address 1:
Address 2:
City:
State/Province:
-Select-
Postal Code:
Country:
-Select-
Age:
-Select-
Gender:
-Select-
Occupation:
None
Please contact me from time to time about specials and new products

FIG. 31B

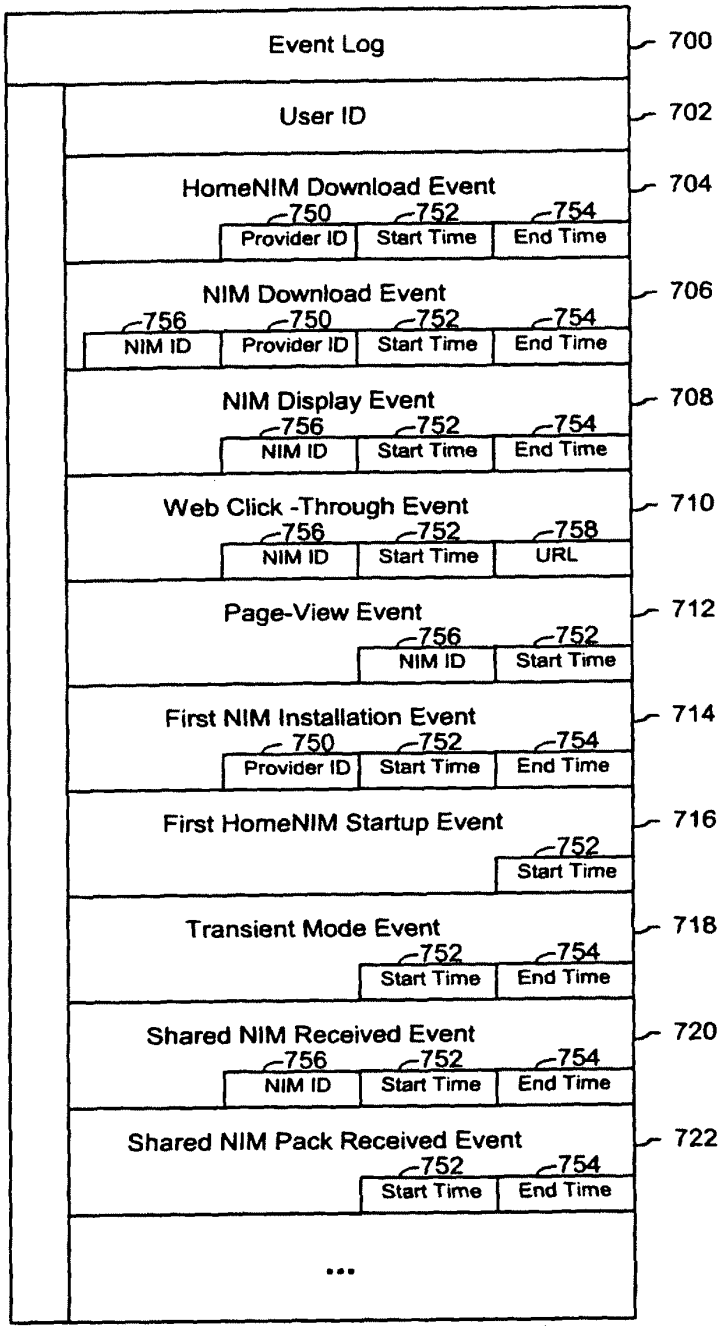

Event Log
700
User ID
702
HomeNIM Download Event
704
750
752
754
Provider ID
Start Time
End Time
NIM Download Event
706
756
750
752
754
NIM ID
Provider ID
Start Time
End Time
NIM Display Event
708
756
752
754
NIM ID
Start Time
End Time
Web Click -Through Event
710
756
752
758
NIM ID
Start Time
URL
Page-View Event
712
756
752
NIM ID
Start Time
First NIM Installation Event
714
750
752
754
Provider ID
Start Time
End Time
First HomeNIM Startup Event
716
752
Start Time
Transient Mode Event
718
752
754
Start Time
End Time
Shared NIM Received Event
720
756
752
754
NIM ID
Start Time
End Time
Shared NIM Pack Received Event
722
752
754
Start Time
End Time
...

FIG. 32

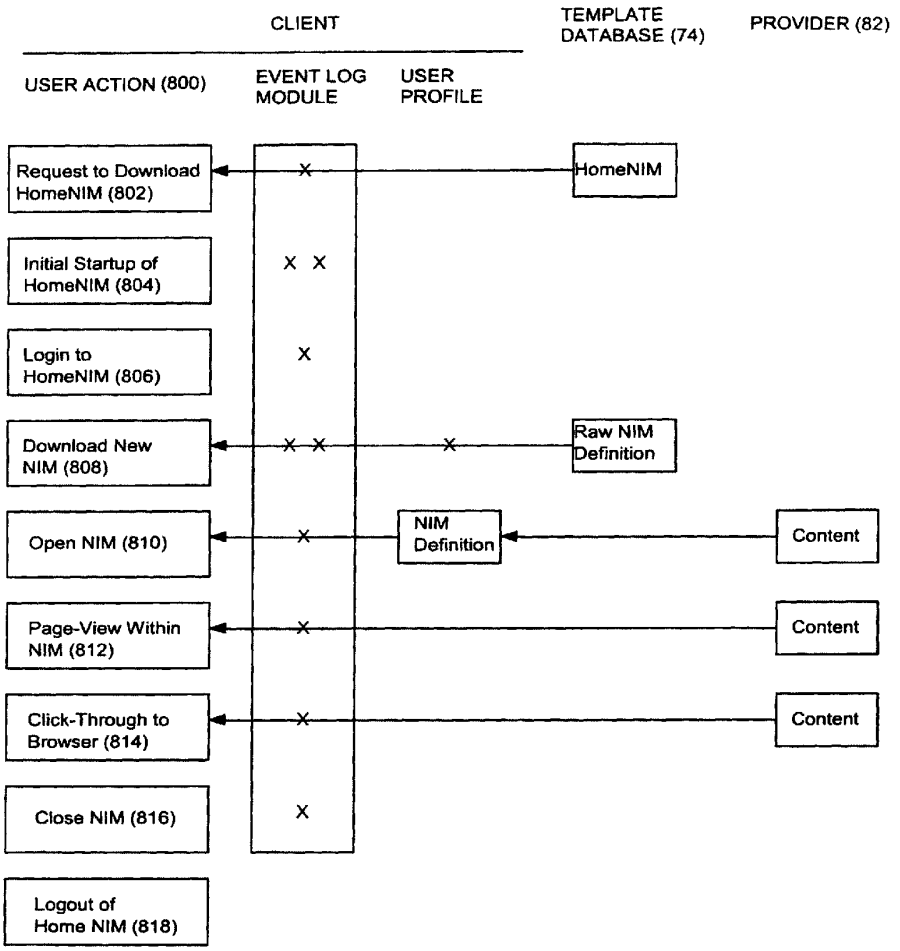

CLIENT
TEMPLATE DATABASE (74)
PROVIDER (82)
USER ACTION (800)
EVENT LOG MODULE
USER PROFILE
Request to Download HomeNIM (802)
HomeNIM
Initial Startup of HomeNIM (804)
Login to HomeNIM (806)
Download New NIM (808)
Raw NIM Definition
Open NIM (810)
NIM Definition
Content
Page-View Within NIM (812)
Content
Click-Through to Browser (814)
Content
Close NIM (816)
Logout of Home NIM (818)

FIG. 33

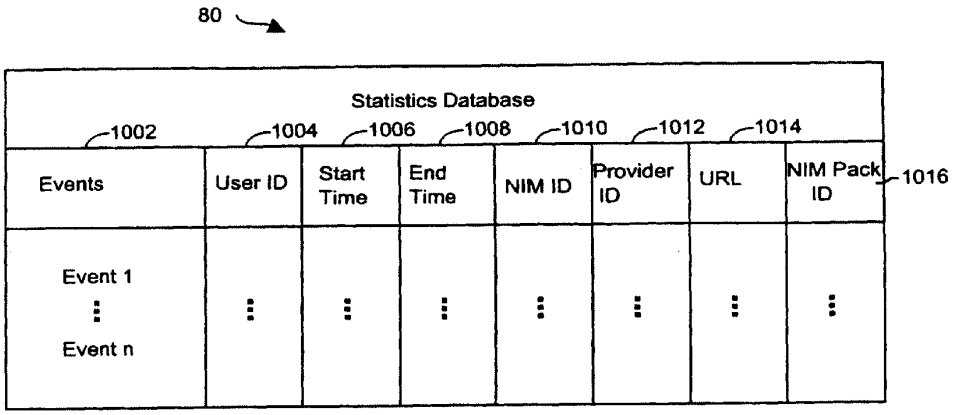

80
Statistics Database
1002
1004
1006
1008
1010
1012
1014
1016
Events
User ID
Start Time
End Time
NIM ID
Provider ID
URL
NIM Pack ID
Event 1
Event n

FIG. 34

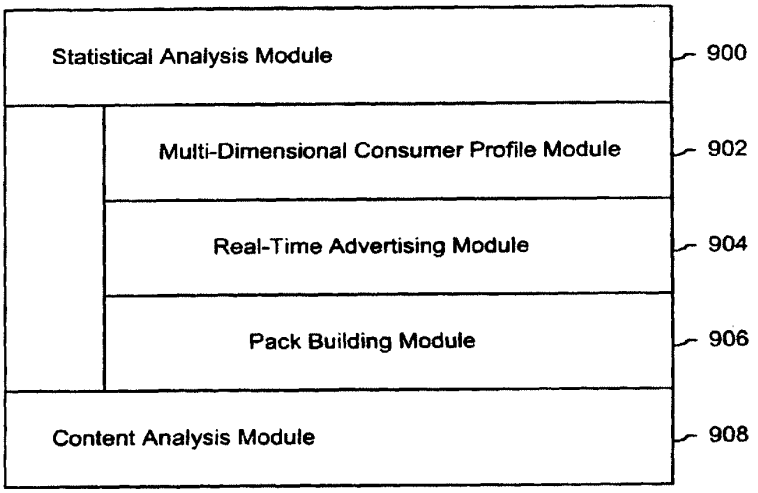

Statistical Analysis Module
900
Multi-Dimensional Consumer Profile Module
902
Real-Time Advertising Module
904
Pack Building Module
906
Content Analysis Module
908

FIG. 35

1050

Content Database

1052 Content Descriptor	1054 NIM ID	1056 Start Time	End Time 1058
Content Descriptor 1			
⋮	⋮	⋮	⋮
Content Descriptor n			

FIG. 36

1100

User Account Database	
User ID (1102)	User Information Provided at Login (1104)
User ID 1 ⋮ User ID n	⋮

FIG. 37

DISPLAYING TIME-VARYING INTERNET BASED DATA USING APPLICATION MEDIA PACKAGES

CROSS-REFERENCE TO RELATED APPLICATIONS

The present application is a continuation of and incorporates by reference U.S. Non-Provisional patent application Ser. No. 09/558,925, filed Apr. 26, 2000, now U.S. Pat. No. 7,660,868, which claims priority from and incorporates by reference U.S. Provisional Application Ser. Nos. 60/131,083, filed Apr. 26, 1999, 60/131,114, filed Apr. 26, 1999, 60/131,115, filed Apr. 26, 1999, 60/176,687, filed Jan. 18, 2000, and 60/176,699, filed Jan. 18, 2000. The present application claims priority to U.S. Non-Provisional patent application Ser. No. 09/558,925, filed Apr. 26, 2000, now U.S. Pat. No. 7,660,868, and each of the aforementioned applications to which it claims priority.

The present application is also related to and incorporates by reference the following U.S. patent applications: Non-Provisional application Ser. No. 09/558,922, filed Apr. 26, 2000, now U.S. Pat. No. 7,756,967; Non-Provisional application Ser. No. 09/558,923, filed Apr. 26, 2000; Non-Provisional application Ser. No. 09/558,924, filed Apr. 26, 2000, now U.S. Pat. No. 7,356,569; Non-Provisional application Ser. No. 11/932,286, filed Oct. 31, 2007 titled "Component For Accessing And Displaying Internet Content"; Non-Provisional application Ser. No. 11/932,286, filed Oct. 31, 2007, titled "Server Including Components For Accessing And Displaying Internet Content And For Providing Same To A Client"; Non-Provisional Application Ser. No. 11/932,392, filed Oct. 31, 2007, titled "Method For Accessing And Displaying Internet Content"; Non-Provisional application Ser. No. 11/932,427, filed Oct. 31, 2007, titled "Component For Coordinating The Accessing And Rendering Of An Application Media Package"; Non-Provisional application Ser. No. 11/932,456, filed Oct. 31, 2007, titled "Tracking and Tracing User Activity with Application Media Packages"; Non-Provisional application Ser. No. 11/932,585, filed Oct. 31, 2007, titled "System and Methods for Creating and Authoring Internet Content using Application Media Packages"; Non-Provisional application Ser. No. 11/932,630, filed Oct. 31, 2007, titled "Methods of Obtaining Application Media Packages"; Non-Provisional application Ser. No. 11/932,663, filed Oct. 31, 2007, titled "Indexing, Sorting, and Categorizing Dots"; Non-Provisional application Ser. No. 11/932,692, filed Oct. 31, 2007, titled "System and Methods of Messaging between Application Media Packages; and, Non-Provisional application Ser. No. 11/932,763, filed Oct. 31, 2007, titled "Component For Accessing And Displaying Internet Content In Association With a Web Browser Application".

BACKGROUND OF THE INVENTION

A user operating a client computer typically accesses the Internet by using a viewer application, such as a browser to view web content provided at a destination address, typically a web page. In this context, web content and web applications are designed to fill the entire web page. It is known to divide the web content into different regions of a single web page. For example, personalized web pages can be specified, such that a user views a variety of content sources in a single page, such as stock information, weather information, and sports information, which is aggregated at the server that delivers the web page to the user, who then views the aggregated content in a single web page. Observe that even when disparate content is aggregated, in this manner, it is reassembled into a full web page and is served through a full-screen browser. Web content and application developers therefore have limited control over the user experience: content is typically trapped within the frame of the browser. A developer's only alternative to engaging a user page-by-page in a browser is to develop, distribute, and support custom client software. In the Web browser scenario, it is the content provider, not the user that aggregates the information that is viewed by the user. Thus, the user is not in a position to separately aggregate the content at a client computer, instead the user is constrained to view the content that has been delivered in the manner provided by the server computer hosting the web page. There is a growing desire for individual users to fully control the aggregation and presentation of content and web applications that appears on a client computer.

A user who wishes to view multiple web pages or applications can open multiple instances of a browser. However, the user will not be able to view each "full-screen" page at the same time. Instead, the user must adjust the windows corresponding to each browser instance and view only part of each page. The information appearing in each browser is not designed for viewing in this manner. Thus, the user cannot create an optimized display of content from multiple sources.

Currently, content providers and end users have limited tools to alter the browser in which content appears. That is, the controls associated with a browser are not fully configurable. Thus, the vendor of a browser is in a position to brand the browser and regulate the controls associated with the browser. There is a growing desire for content providers to not only fill a browser with their content, but to also fully brand and control the frame in which the content appears. Further, in some instances, content providers desire to limit the controls associated with a browser or viewer, so that a user is more inclined to view a single set of content, for example, by having limited access to previously viewed content.

At the present time, it is relatively difficult to trace the content viewing activity of a client computer. In other words, it is difficult to identify the type of content that a particular user of a client computer favors. Consequently, there are limited tools 25 available to provide a user with tailored information that would be of particular interest to the user.

In view of the foregoing, there is a need in the art to provide a technique for accessing multiple instances of distributable computer readable media in their entirety simultaneously, where these instances are typically smaller than the full pages used in 30 current web pages and web applications. There is a further need for providing the user with flexibility in selecting, collecting, relating and viewing such computer readable media, and for giving the media provider flexibility in directing media to a specific user and controlling the framework in which media is presented. Finally, there is a need to gather more accurate information regarding the type of content that a user enjoys, so that the user can be automatically provided with this content.

SUMMARY OF THE INVENTION

The invention includes a method of presenting distributable computer readable media to a user in response to a user request. The method comprises the steps of identifying a definition of a Networked Information Monitor (NIM). A NIM frame is defined for the NIM using the definition. Content is then retrieved for the NIM. Then, the content is placed in a NIM viewer defined by the frame.

The invention also includes a method of altering a Networked Information Monitor (NIM). The method includes

the step of receiving a message at a NIM. The message specifies a configurable feature of the NUM. The NIM is altered in accordance with the configurable feature of the message.

The apparatus of the invention includes a computer readable memory to direct a computer to function in a specific manner. The computer readable memory includes a first executable module to identify a definition of a Networked Information Monitor (NIM). A second executable module defines a NIM frame for the NIM using the definition. A third executable module retrieves content for the NIM. A fourth executable module places the content in a NUM viewer defined by the frame.

The apparatus of the invention further includes a computer readable memory with a first executable module to receive a Networked Information Monitor (NIM) message. The NIM message specifies a configurable feature of a NIM. A second executable module alters the NIM in accordance with the configurable feature of the NIM message.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** illustrates a system for processing distributable computer readable media in accordance with one embodiment of the present invention;

FIG. **2** shows exemplary processing steps used to initiate an instance of a home networked information monitor (NIM) using the apparatus of FIG. **1**;

FIG. **3**A illustrates a screen logo in accordance with one embodiment of the present invention;

FIG. **3**B illustrates a login construct in accordance with one embodiment of the present invention;

FIG. **4** illustrates a set of visual manifestations corresponding to a collection of NIMs, including a home NIM;

FIG. **5** illustrates a visual manifestation of a home NIM as well as a visual manifestation of a reference NIM that references additional NIMs;

FIG. **6** illustrates how a NIM, referenced by the reference NIM of FIG. **5**, is added to a processed user profile in response to a designated keyboard entry sequence or mouse click;

FIG. **7** illustrates how a NIM is shared with other users in accordance with an embodiment of the present invention;

FIGS. **8**A and **8**B illustrate screen panels that facilitate the collection of the description of a set of designated NIMs into a pack;

FIG. **9**A illustrates how the relative position of visual manifestations that correspond to NIMs remains fixed when the visual manifestations are within a predetermined distance of each other;

FIG. **9**B illustrates a representative pack in accordance with the present 20 invention;

FIGS. **10**A, **10**B and **10**C illustrate how a set of visual manifestations corresponding to a collection of NIMs is aligned against a boundary when a user selects the visual manifestations and pushes them against the boundary;

FIG. **11** is a flow chart of the steps taken by a client to give a user access to a NIM where the user "collects" the NIM, in accordance with one embodiment of the invention;

FIG. **12** is a flow chart of the steps taken by a client to present a NIM to a user, where the NIM has been "collected" previously by the user, in accordance with one embodiment of the invention;

FIG. **13** illustrates a data structure for a NIM definition, stored in the NIM application server's template database or user profile database;

FIG. **14** illustrates NMA message routing between NIMs and the message interface in the client parser application;

FIG. **15** is a diagrammatic illustration of an embodiment of a NIM Management Module utilized in accordance with an embodiment of the invention;

FIG. **16** is a diagrammatic illustration of an embodiment of the NIM Templates database utilized in accordance with an embodiment of the invention;

FIG. **17** is an illustration of a main NIMIndex Web page used in accordance with an embodiment of the invention;

FIG. **18** is an illustration of a single NIMIndex category used in accordance with an embodiment of the invention;

FIG. **19** is an illustration of a full description of NIM content provided in accordance with an embodiment of the invention;

FIG. **20** is an illustration of a Web page displayed to the user once the user has clicked to collect the NIM;

FIG. **21** is an illustration of the main home NIM graphical user interface used in accordance with an embodiment of the invention;

FIG. **22** is an illustration of a "Get New NIM" graphical user interface that may be used in accordance with an embodiment of the invention;

FIG. **23** is an illustration of a "More NIMs" graphical user interface representative of an embodiment of the invention;

FIG. **24** is a diagrammatic illustration of the ShareLink database used in accordance with an embodiment of the invention;

FIG. **25** is an illustration of a Share NIM's graphical user interface according to an embodiment of the invention;

FIG. **26** is an illustration of the main DevZone Web page utilized in accordance with an embodiment of the invention;

FIG. **27** is a partial view of a NIM modification web page utilized in accordance with an embodiment of the invention;

FIGS. **28**A to **28**D are graphical user interfaces of development NIMs (DevNIMs) utilized in accordance with an embodiment of the invention;

FIGS. **29**A and **29**B are illustrations of Administrative Zone (AdminZone) Web pages utilized in accordance with an embodiment of the invention;

FIGS. **30**A and **30**B are also illustrations of Administrative Zone (AdminZone) Web pages utilized in accordance with an embodiment of the invention;

FIGS. **31**A and **31**B are further illustrations of Administrative Zone (AdminZone) Web pages utilized in accordance with an embodiment of the invention;

FIG. **32** illustrates an embodiment of an event log that may be used in accordance with an embodiment of the invention;

FIG. **33** illustrates the tracking of events in an event log module in accordance with an embodiment of the invention;

FIG. **34** illustrates a statistics database that may be used in accordance with an embodiment of the invention;

FIG. **35** illustrates a statistical analysis module and a content analysis module that may be used in accordance with an embodiment of the invention;

FIG. **36** illustrates a content database that may be used in accordance with an embodiment of the invention; and

FIG. **37** illustrates a user account database that may be used in accordance with an embodiment of the invention.

DETAILED DESCRIPTION OF THE INVENTION

The present invention discloses a technology that is capable of processing distributable computer readable media. Distributable computer readable media includes, but is not limited to, standard web content, such as HTML, dHTML, images, imbedded ActiveX and Java applications, JavaScript, CSS, Pen scripts, Streaming Media, and/or Flash. The present invention is advantageous relative to prior art systems and

methods because it provides improved mechanisms for simultaneously interacting with several independent sources of distributable computer readable media, collecting references to such media, and sharing such references with other users. The ac disclosed technology is further advantageous because it provides improved systems and methods for on screen management of distributable computer readable media.

In the system and method of the present invention, a user logs into a server by providing a login identifier to a login construct. The login identifier is used by the server to obtain an unprocessed user profile that corresponds to the user. The unprocessed user profile is processed by the server to generate a processed user profile. Advantageously, this processing step allows for up-to-date refinement of the user profile. Up-to-date refinements include, for example, the addition of advertisements directed to the user based on one or more characteristics in the user profile. The processed user profile is delivered to the client computer associated with the user. The processed user profile includes references to the networked information monitors (NIMs). As used herein, the term networked information monitor or NIM refers to a fully configurable frame with one or more controls; the frame through which content is optionally presented. The fully configurable frame utilized in accordance with the invention stands in contrast to present web browsers, which are branded by the browser vendor and which have limited means by which to alter the controls associated with the browser.

Attention is initially directed toward the home NIM of the invention, which coordinates the activities of all other NIMs that are accessed by a user. The home NIM facilitates much of the technology of the present invention, including the ability to simultaneously review multiple sources of distributable computer readable media as well as to package and distribute such media.

FIG. **1** is a general illustration of a system in accordance with one embodiment of the present invention. In FIG. **1**, a network **10** is operated in accordance with the present invention. Network **10** includes at least one user or client computer **20**, at least one server computer of class **50**, and optionally one or more server computers of class **82**. User computer **20** as well as server computers of class **20 50** and **82** are each connected by transmission channel **44**, which is any wired or wireless transmission channel.

User computer **20** is any device that includes a Central Processing Unit (CPU) **24** connected to a random access memory **30**, a network connection **28**, and one or more user input/output ("i/o") devices **40** including output means **42**. Output means **42** is any device capable of communicating with a user and includes, for example, a video monitor, a liquid crystal display, voice user interfaces, and/or integrated graphic means such as mini-displays present in web-phones. Typically, user computer **20** includes a main non-volatile storage unit **22**, preferably a hard disk drive, for storing software and data. Further, user computer **20** includes one or more internal buses **26** for interconnecting the aforementioned elements. In a typical embodiment, memory **30** includes an operating system **32** for managing files and programs associated with user computer **20**. In some embodiments, operating system **32** includes a registry **34** that has one or more references to specified locations in system **10**. The exemplary memory **30** of FIG. **1** further includes a web browser **36** for viewing web content and a client parser application **38** for facilitating low level functionality, such as login and logout procedures, in accordance with the present invention. In some embodiments, client parser application **38** uses the one or more references in registry **34** to obtain a login construct from server **50**. In various embodiments, in accordance with the present invention, client parser application **38** runs in conjunction with one or more software modules, such as an event log module **98**, which tracks user activity, a message interface module **106**, which serves as a communication interface between the client parser application **38** and web server **58** and/or external web servers, a home NIM **108**, which references one or more NIMs **110**, and a visual management system **114** which regulates the characteristics of visual manifestations of NIMs **108** and **110** when displayed on output device **42**. Furthermore, in some embodiments, client parser application **38** runs in conjunction with instances of web browser **36** as well as web server **58** as detailed below.

Server computer **50** includes standard server components, including a network connection device **46**, a CPU **52**, a main non-volatile storage unit **54**, and a random access memory **56**. Further, server computer **50** includes one or more internal buses **48** for interconnecting the aforementioned elements. Memory **56** stores a set of computer programs, modules and data to implement the processing associated with the present invention.

The embodiment of memory **56** illustrated in FIG. **1** includes a web server **58** for processing requests received from client computer **20**. Web server **58** has many components, including a variety of modules and data structures to assist users that want to log into system **10**. Namely, login module **60** handles an entry request from a client computer **20** and accepts a login identifier that corresponds to a user from client computer **20**. Login constructor **62** generates a login construct in response to a call for a login construct and transfers the login construct to client **20**. Login constructor **62** dynamically generates login constructs using updated login NIM content **64**. Login validation module **66** works in conjunction with user profile database manager **100** to identify an unprocessed user profile, which is associated with a user provided login, in user profile database **76**. If an unprocessed user profile corresponding to the user provided login does not exist in user profile database **76**, login validation module **66** associates a new unprocessed user profile with the login identifier.

When an unprocessed user profile is identified by login validation module **66**, it is processed by user profile processor module **68** to produce a processed profile. It will be appreciated that the services of user profile processor module **68** are highly advantageous because they allow for last minute user profile revisions. Such revisions include, for example, the addition or customization of NIMs referenced by the user profile, and/or server redirect information that is a function of current server load in system **10**. Importantly, the processed user profile includes a reference to a home NIM. The home NIM is capable of accessing each of the NIMs that are represented in the processed user profile. Memory **56** further includes delivery module **70**, which coordinates the delivery of portions of the home NIM to the client based on parameters specified in the processed user profile.

Once a user has successfully logged into system **10**, request server module **72** handles requests for specified NIMs from client **20**. When such a request is received, request server module **72** routes the request to an address that corresponds to the specified NIM and transmits the specified NIM to client **20**. One class of specified networked information handled by request server module **72** is requests for NIMs. When such a request is received, request server module **72** searches NIM templates database **74** for the specified NIM. NIM templates database **74** includes a large number of NIM templates. Each NIM template defines the characteristics of a specific NIM,

including fully configurable frame characteristics, viewer and control characteristics, and NIM content references.

The web server **58** illustrated in FIG. **1** further includes additional modules **102** to handle specialized features of the present invention. For example, one embodiment of the present invention provides a mechanism that allows users to distribute NIMs to each other. In such embodiments, a special server module **102** provides instructions for storing the NIMs, which are to be distributed, in sharelink database **78**. Advantageously, NIMs that are distributed to other users are customizable. A user can, for example, resize and position a particular NIM prior to sharing it with another user. Indeed, it is possible, in such embodiments, for a user to arrange a series of NIMs in a unique arrangement and then distribute the collection of NIMs in the designated NIMs in the designed arrangement. As an illustration, a user arranges a first NIM that represents a scrolling stock ticker at the bottom of an output means, such as a computer screen, a second NIM that tracks the NASDAQ top ten most heavily traded stocks in the upper left corner of the output means, and a third NIM that tracks headline news on the upper right hand corner of the output means. Then, the user distributes the three NIMs in this customized arrangement to other users. Observe that in this example a user of a client computer is aggregating separate sets of information in different NIMs. This stands in contrast to prior art approaches where a web server running on a server computer aggregates information in a single page.

System **10** is highly scalable and thus supports a large number of users. This scalability stems from the fact that the server **50** is delivering the definition associated with a NIM. The content displayed in the NIM may be located on a separate computer.

Memory **56** provides a statistical analysis module **104** for tracking key events associated with users. This information is stored in statistics database **80**. The information collected by statistical analysis module **104** is used for a wide variety of purposes, including server load optimization and directed advertising, as discussed below. As described below, the statistical information gathered in accordance with the invention includes fully traced events defining the type of content and the duration over which all content is viewed by a user. This type of comprehensive information is not available using present techniques.

Much of the distributable computer readable media that is available for processing is stored as content elements **94** on server **82**. Server **82** is a standard web server that includes components such as a network connection device **88**, a CPU **86**, a main non-volatile storage unit **84**, a random access memory (RAM) **92**, and one or more internal buses **90** for interconnecting the aforementioned elements. RAM **92** includes some of the content elements **94** stored by server **82**. Other content elements **94** are stored in storage unit **84**. In some embodiments, a single web server **58** is capable of directly accessing content elements **94** located on one or more servers **82**. In other embodiments, each server **82** has a resident web server module that works in conjunction with server **50** to identify, optionally dynamically generate, and serve content elements **94** upon demand.

Now that general architecture of a system in ac dance with the present invention has been disclosed, attention turns to FIG. **2**, which discloses a method for logging into system **10** (FIG. **1**). In the first processing step shown in FIG. **2** (**202**), a user initiates a session on system **10** by requesting the global login script "session_config." It will be appreciated that the term "session_config" merely provides an illustrative name for the global login script and that the technology of the present invention is by no means limited to this name or the script described.

The request for "session_config" originates on client **20** and is sent to server **50** where it is processed by login module **60** of web server **58**. Upon receiving request **202**, login module **60** creates a "session_config" global login script (**204**). Processing step **204** is advantageous relative to systems that have static global login scripts because it allows for the incorporation of highly variable information. This highly variable information includes, for example, system settings such as up-to-date server redirect information, server content address changes, directed advertisements, and messages. An exemplary "session_config" is found in Example 1 below. Each line of data has an associated numeral. The remaining text in the example describes the purpose of select data.

EXAMPLE 1

Version tag that identifies the latest home NIM version
(1) </SESSION_CONFIG VERSION="alpha:3"
Upgrade event that is sourced when home NIM version is outdated
(2) <UPGRADE=http://www.NIM.com/QuickOpen.exe>
LOCATION OF SERVER-SUPPORTED FUNCTIONALITY Default base address for server supported functionality
(3) <METHODS BASEURL=http://neo.NIM.com/servlet/NIMServer/>
(4) <ADD_USER URL="addUser/>
(5) <GET_USER URL="getUser"/>
(6) <SET_USER URL="setUser/>
(7) <GET_SES_CONFIG URL=http://www.NIM.com/home_NIM/s_cnfg.xml"/>
Server-based functionality for setting password
(8) <SET_PASSWORD URL="setPassword"/>
List of all publically available NIMs
(9) <GET_MASTER_NIM_LIST URL= "getMasterNIMList"/>
(10) <GET_ALL_CONFIG URL="getAllConfig"/>
(11) <SET_ALL_CONFIG URL="setAllConfig/>
(12) <SET_ALL_STATS URL="setAllStats"/>
(13) <GET_NIM_TEMPLATE URL= "get_NIM_Template"/>
Location of server-side NIM and pack sharing functionality
(14) <ADD_SHARE URL="addShare"/>
(15) <GET_SHARE URL="getShare"/>
(16) <AUTH_TEST URL="DOeCHO?AUTH=TRUE"/>
Redirect information
(17) <DO_REDIR URL="doRedir"/>
(18) </METHODS>
Flexible content layer that defines default NIM frame appearance, including the default appearance of the frame of a home NIM
(19) <FRAMES>
Default NIM frame appearance
(20) <NIM>
(21) <IMAGES BASEURL=http://www.NIM.com/home_NIM/NIM_FrameImages/>
(22) </NIM>
Default home NIM frame appearance
(23) <HOME_NIM>
(24) <IMAGES BASEURL=http://www.NIM.com/home_NIMImages/>
(25) </HOME_NIM>
(26) </FRAMES>

Location of system NIM templates

(27) <NIMs>

(28) <ADD TEMPLATE="http://www.NIM.com/ . . . / add_NIM_XML.xml"/>

(29) <HELP TEMPLATE="http://www.NIM.com/ . . . /help_NIM_XML.xml"

(30) <LOGIN TEMPLATE="http://www.NIM.com/ . . . / login2.xml"/

(31) </NIMs>

(32) </SESSION_CONFIG>

Line 1 of the exemplary "session_config" of Example 1 provides the version tag for the expected version of the home dot system that corresponds to the "session_config" script. In one embodiment, client parser application **38** determines whether it is up-to-date using the information in line 1. If client parser application **38** determines that it is outdated, an upgrade request is made in accordance with the instructions provided by the UPGRADE flag of line 2. In one embodiment, the UPGRADE flag in line 2 of Example 1 describes the location of an executable program, one of skill in the art will appreciate that this flag can in fact reference any form of instruction, including a flat file, a web page, a script, a symbol, or an address.

Lines 3 through 18 in Example 1 define the functionality that is provided by a server, such as server **50**. For example, line 8 of Example 1 provides the location of a set of instructions that are called when a user requests a password change. Furthermore, line 9 of Example 1 provides the location of master list of NIMs that are publically available. Lines 14 and 15 of Example 1 provide the location of specialized server-side functionality that allows users to share data such as NIM definitions.

Lines 19 through 26 of Example 1 define where the default appearance of a NIM and a home NIM are found within system **10**. Lines 27 through 31 define a collection of system NIMs. A system NIM is any type of NIM that is to be distributed to each user of system **10**. In some embodiments, system NIMs are used to provide a core functionality. In Example 1, line 28 defines a NIM that provides users with a convenient mechanism for collecting additional NIMs. Line 29 defines the location of a NIM that is invoked when the user presses a help button associated with a home NIM. Finally, line 30 defines the location of a NIM that is used to log into system **10**.

Returning to FIG. **2**, once login module **60** has created "session_config," it is sent back to requesting client **20** (**206**). When a "session_config" is received by client **20**, client parser application **38** parses the global login script in order to identify a reference to a login constructor **62** (**208**). Login constructor **62** is a server-based module that generates a construct that allows a user to log into system **10**. When client parser application **38** locates the reference to login constructor **62** in "session_config," a request for a login construct is directed to the identified reference (**210**). In Example 1 above, the reference to the login construct is provided in line 30. On line 30, the global variable "LOGIN TEMPLATE" is assigned the URL address "http://www.NIM.com . . . / login2.xml." Client parser application uses the URL assigned to the global variable "LOGIN TEMPLATE" to make a request for a login constructor **62** that is directed to this URL. When login constructor **62** receives a request for a login construct, it generates a login construct (**212**).

Login construct **148** (FIG. **3**B) illustrates a type of login construct that is generated in one embodiment of the present invention during processing step **212**. Before the login construct is executed on client **20**, a schematic such as logo **146** (FIG. **3**A) is displayed on output means **42**. As illustrated in FIG. **1**, login constructor **62** is a component of web server **58**. However, there is no requirement that login constructor **62** be a component of web server **58**. In fact, login constructor **62** is a standalone software program in some embodiments of the present invention whereas in other embodiments, login constructor **62** is merely a script, such as a PERL script, that is processed by an interpreter program native to server **50**. In still other embodiments, login constructor **62** is merely a simple flat file that includes a set of instructions that are interpretable by client parser application **38**. In such embodiments, login constructor **62** is the login construct. In embodiments in which a login construct is dynamically generated, it is possible to introduce last minute changes in the login construct. Thus, an advantage of the exemplary login process shown in FIG. **2** is that there are multiple stages in which updated information is used to customize the login process based on the environmental variables.

Once a login construct has been prepared by login constructor **62**, it is transferred back to client **20** (**214**) (FIG. **2**) and executed in conjunction with client parser application **38** (**216**). The login constructs of the present invention are a form of NIM. Therefore, one function of processing step **216** is to obtain the login NIM content **64** (FIG. **1**) specified by the login construct from server **64**. In login construct **148**, for example, the login NIM content includes the shape and functionality of "Exit button **160**," message **150**, the shape and functionality of "New user" button **152**, the functionality of "Forgot it?" button **154**, and login panel **156**. When processing step **216** is completed, the user uses the login construct to provide a login identifier (**218**).

In FIG. **3**B, a user has provided the login identifier "Galliani." The definition of login identifier as used in the present invention is to be broadly construed. In some embodiments, login identifiers include a unique name and a corresponding password. In other embodiments, a login identifier does not have a password. This is particularly the case when the user is a guest or a new user and there is no user profile associated with the user.

Working in conjunction with client parser application **38**, the login construct accepts the user login and sends it to server **50** for validation (**218**). As illustrated in the exemplary system of FIG. **1**, web server **58** includes a login validation module **66** to verify the login identifier provided by user (**220**). Typically, processing step **220** involves a look-up operation in which the login identifier is used to query user profile database **76** for an unprocessed or raw user profile that matches the login identifier. In embodiments that include a password, validation step **220** includes a password verification step. Successful completion of processing step requires entry of a valid login identifier sequence in processing step **218**. When processing step **220** has been successfully completed, the raw or unprocessed user profile corresponding to the login identifier is obtained from user profile database **76** (FIG. **1**) (**222**) and is processed by user profile process module **68** to produce a processed or finalized user profile that is delivered to client **26** (**226**). In sonic embodiments, a user profile **76** includes user contact information, such as the name, address, telephone number and email address of a user. Additionally, some embodiments of system **10** provide different types of access privileges. For example one embodiment of the present invention includes developer access privileges, administration access privilege, and general user access privileges. In such embodiments, the access privileges that have been granted to a user are stored in the user profile **76** associated with the user.

The processed user profile includes a reference to each NIM in system **10** that is associated with the login identifier provided in processing step **216**. One of the NIMs referenced

by the processed user profile is the home NIM that corresponds to the login identifier provided in processing step **216**. When executed in conjunction with client parser application **38** in processing step **226**, the home NIM provides a mechanism for accessing each of the NIMs referenced by the processed user profile Like the login construct, the home NIM includes several components, including pull down menus and screen manipulation functionality. The reference to the home NIM in the processed user profile includes the system **10** address of each of these components. Therefore, in one embodiment, construction of the home NIM in processing step **226** involves one or more requests to server **50** and/or server **82** for content (**228**) that is then rendered (**230**) in accordance with the home NIM description provided in the processed user profile. In some embodiments, the home NIM is distinct from other NIMs in the sense that a large proportion of the home NIM in such embodiments is pre-compiled. Such embodiments are advantageous because some of the functionality provided by the home NIM requires substantial client **30** processing resources. Therefore, to minimize such processing resource requirements, many aspects of the home NIM are pre-compiled in some embodiments. In other embodiments, however, the home NIM has a structure that is substantially the same as a regular NIM. In such embodiments, simple script commands are used to identify the NIM as a home NIM.

Upon completion of processing step **230**, the user is granted access to all of the technologies of the present invention, including the ability to view multiple NIMs simultaneously, collect new NIMs, customize NIMs, and share customized NIMs with other users. An exemplary processed user profile is provided in Example 2. Once again, each line of data is identified with a numeral, while the remaining text in the example describes select data. In some embodiments, the user is granted specific privileges and the extent to which the user is granted access to system **10** is regulated by the types of privileges that have been granted to the user.

EXAMPLE 2

(1) SAMPLE PROCESSED USER PROFILE
(2) <ALL CONFIG>
NIMs AND PACKS THAT CORRESPOND TO THE USER
(3) USER
NIM definition 1
(4) <NIM DOMAIN="ZDNet" GLOBALID="1" KND="news"
(5) <FRAME BACKGROUNDCOLOR=#FFFF00" COLLAPSED="FALSE"
(6) FIXHEIGHT="TRUE" FIXWIDTH="TRUE" NAME="ZDNet Breaking News"
(7) PIXELHEIGHT="275" PIXELWIDTH="235" X="RIGHT" Y="TOP">
(8) <TITLE COLOR=#000000" JUSTIFY="RIGHT" TEXT=" "/>
(9) <TITLEBARIMAGE DOWN=http://www.NIM.com/ . . . /feed/titlebar.gif
(10) HOVER=URL address to a first GIF file <param 1> . . . <param N>
(11) INACTIVE=URL address to a second GIF file <param 1> . . . <param 2>
(12) NORMAL=URL address to a third GIF file <param 1> . . . <param 2>
(13) <BOTTOMBARIMAGE DOWN=http://www.NIM.com/ . . . /feed/bottombar.gif
(15) HOVER=URL address to a fourth GIF file <param 1> . . . <param 2>
(16) INACTIVE=URL address to a fifth GIF file <param 1> . . . <param 2>
(17) NORMAL=URL address to a sixth GIF file <param 1> . . . <param 2>
(18) </FRAME>
(19) <MENU/>
(20) <CONTROL_LAYOUT HEIGHT="1" HEIGHTSCALES="TRUE" WIDTH="1"
(21) WIDTHSCALES="TRUE"><CONTROL CLASS="Browser" HEIGHT="1"
(22) ID="1" KIND="A" LEFT="0" TOP="0"
(23) URL=http://www.mandala.com/cgl/zdnet/zdfeedl.cgi WIDTH="1"/>
(24) </CONTROL_LAYOUT>
(25) <CATEGORIES/>
(26) <EVENTS/>
(27) </NIM>
NIM definition 2
(28) <NIM DOMAIN=NIM DOMAIN 2 GLOBALID='2'
(29) </NIM>
(92) NIM definition N
(30) <NIM DOMAIN=NIM DOMAIN 2 GLOBALID='N'
(31) </NIM>
Pack definition 1
(32) <PRESET TITLE="New DotPack">
(33) <NIM GLOBALID="1" X="RIGHT" Y="TOP"/>
(34) <NIM GLOBALID="2" X=RIGHT Y="320"/>
(35) </PRESET>
(36) </SHARE>
Last state of the home NIM
(37) <LASTSTATE>
(38) <PRESET TITLE=" ">
(39) <NIM GLOBALID="1" X="RIGHT" Y="TOP"/>
(40) <NIM GLOBALID="2" X=RIGHT Y="280"/>
(41) </PRESET>
(42) <HOMENIM COLLAPSED="FALSE" HEIGHT="134" X=616" Y="109"/>
(43) </LASTSTATE>
(44) </ALL_CONFIG>

Example 2 describes a representative processed user profile in accordance with the present invention. In general, a processed user profile includes three major components: (i) a definition of each NIM associated with the user, (ii) a description of each pack associated with the user and, (iii) the last state of each home NIM associated with a user. In Example 2, the definition of each NIM associated with the user is found on lines 4 through 31. Specifically, lines 4 through 31 describe NIM definitions I through N. In Example 2 there is only one pack associated with the user. This pack, entitled "New DotPack," is found on lines 32 through 35 of Example 2. The final major component of the processed user profile found in Example 2 is the last state of the home NIM, which is defined on lines 37 through 43. This code stores the last state of the home NIM. Such last state information includes whether the home NIM was collapsed, and the position of the home NIM on the screen.

When the user wishes to log out of system **10**, the processed user profile is transferred from client **20** to server **50**. When web server **58** receives the processed user profile, it passes the processed user profile to user profile database manager **100**. User profile database manager **100** stores the processed user profile as the unprocessed user profile **76** corresponding to the user. In some embodiments, such a storage operation involves a conversion process. For

example, advertisements or specific system NIM definitions are stripped from the processed user profile in order to convert the processed user profile to the unprocessed user profile **76** that corresponds to the user. In some embodiments, the processed user profile is periodically transferred, in its entirety or incrementally, from client **20** to server **50** and saved in the manner described in the log out procedure above. Such timed periodic or event based backup procedures are possible because NIM definitions are efficiently described, thus the absolute size of a processed user profile remains relatively small. Accordingly, timed backups of a processed user profile to user profile database **76** are possible without extensive use of system **10** bandwidth or server **50** resources.

At this stage, a system (FIG. **1**) and a login procedure (FIG. **2**) in accordance with the present invention has been disclosed. Although the system and login procedure was discussed using an example where only one home NIM was associated with a user, it will be appreciated that, in some embodiments, any number of distinctly different home NIMs are associated with a user. Furthermore, a user can simultaneously execute multiple instances of a particular home NIM on client **20** or, indeed, any number of different home NIMs. In one embodiment, a developer or merchant provides a user with a highly customized home NIM that provides specialized functionality. In such embodiments, the user collects the home NIMs and, therefore, a processed user profile includes a description of more than one home NIM.

Attention now turns to some of the advantages and features of the present invention. In FIG. **4**, a visual manifestation of the home NIM **162** is illustrated. One advantage of the home NIM, which is an advantage that is common to NIMs in general, is that the content of the NIM is not trapped in a third party viewer. In fact, the home NIM definition regulates the actual appearance of the home NIM. The home NIM definition is formed by general parameters and commands found in "session_config" as well as customized parameters and commands in the processed user profile. The division of the home NIM definition between a system level file and a user level file represents a balance in the tension between the need for a system **10** host to insure a consistent level of quality, through the proper implementation of general parameters and commands, and the desire of each user to create highly customized home NIMs. Lines 20 through 22 of Example 1 provide an example of general parameters that are defined in "session_config." Lines 23 through 25 define the source location of home NIM frame images. In home NIM **162** (FIG. **4**), such home NIM frame images include the image used to represent buttons **164** through **174**, and menu tabs **1** d **178**. Furthermore, lines 20 through 22 of Example 1 define the location of other images that are used to construct default NIMs. An example of user initiated home NIM customization is found in lines 33 through 39 of Example 2, which define a "LAST-STATE" definition for the home NIM, including the dimensions of the visual manifestation corresponding to the home NIM on line 38 (HEIGHT="134" X="616" Y="109") and indicates that the home NM is not collapsed upon startup (COLLAPSED="FALSE").

The visual manifestation of home NIM **162** illustrates additional benefits and features of a home NIM in accordance with the present invention. When a user selects tab **176**, a list of the NIMs that are present in the processed user profile associated with the user is displayed in viewer **180**. As disclosed in more detail below, a user has the option to associate a collection of NIMs into an object termed a "pack". The pack references some subset of the NIMs associated with a user as well as associated state information. This arrangement includes, for example, whether a visual manifestation corresponding to each MM is displayed on output means **42** or not, the dimensions of each visual manifestation, and the position of each visual manifestation. The name of each pack is stored in the processed user profile. A user reviews packs associated with the user by selecting tab **178** (FIG. **4**). In FIG. **4**, the user only has one pack, "Customized DotPack" **182**. When the user selects pack **182**, each NIM in the pack is restored in accordance with the state information stored in the pack definition.

In total, FIG. **4** represents a typical visual experience provided by one embodiment of the present invention. In addition to home NIM **162**, visual manifestations **184** and **186**, corresponding to two additional NIMs in the processed user profile, are displayed. Visual manifestation **184** provides functionality that allows a user to manage an address book, schedule appointments, or create groups and plan activities. Visual manifestation **186** represents a NIM that provides time and date information.

FIG. **5** shows the visual manifestation of home NIM **162** of FIG. **4** with tab **176** selected. Accordingly, each of the NIMs in the processed user profile associated with the user is listed in list **188**. The user can activate any of the listed NIMs by clicking on the NIM name. In addition to the NIMs in list **188**, home NIM **162** includes core NIMs that are defined in "session_config." In the "session_config" of Example 1, cores are found on lines 28 and 29. Specifically, line 28 provides the address of an XML-based definition for the add template functionality associated with button **172** in FIGS. **4** and **5**, and line 29 provides the address of an XML-based definition for the help template functionality associated with button **174** in FIGS. **4** and **5**.

Importantly, the user can categorize NIMs using filter **190**. Categories include such topics as sports, personal, weather, etc. Furthermore, the user can add NIMs to the processed user profile associated with the user as well as delete NIMs. There are a variety of mechanisms that enable a user to add a NIM to the processed user profile. One mechanism is to receive links to NIMs from other users of system **10** (FIG. **1**), as disclosed below. Another mechanism is to toggle button **172** in order to activate a visual manifestation associated with NIM **192** (FIG. **5**).

NIM **192** provides a system that enables users to add select NIMs to their user profile with a single click or keystroke sequence. NIM **192** includes tab **194** that allows the user to select premiere NIMs and a general tab **196** that allows the user to review a general catalog of NIMs that is present in NIM templates database **74** (FIG. **1**). In one embodiment, when a user selects a NIM in list **198** (FIG. **5**), the NIM is added to list **188** and is incorporated into the processed user profile associated with the user. In this way, the user can collect NIMs of interest to the user using a single mouse click. By illustration, consider the case in which a user selects the NIM "AnyDay Calender" in list **198**. In response to this selection, a definition of the NIM "AnyDay Calender" is obtained from NIM templates database **74** and is copied directly into the processed user profile associated with the user. Furthermore, the title of the selected NIM, "AnyDay Calender" is added to list **188**. Finally, a visual manifestation that corresponds to the NIM "AnyDay Calender" is displayed on output means **42** (FIG. **1**). As a result, the display illustrated in FIG. **5** adopts the appearance illustrated in FIG. **6**.

In FIG. **6**, the NIM "AnyDay Calendar" appears at the top of list **188**. Furthermore, a control **101** associated with the NIM "AnyDay Calendar" in list **188** is filled, indicating that the NIM is currently active. Additionally, as illustrated in FIG. **6**, a visual manifestation **103** corresponding to NIM "AnyDay Calendar" appears on the output means. The user

has the ability to toggle this NIM between an inactive and active state by selecting control **101**. (**105**) In one embodiment, the user is provided with the option of (i) incorporating a NIM selected in list **198** into the processed user profile or (ii) transiently executing the NIM on client **20**. Furthermore, when the user receives NIMs from other users, the user has the option to transiently operate the received NIMs on client **20**. If the user decides to keep the transient NIMs at a later date, the user has the option to add the transient NIMs to the processed user profile at that time. Thus, in such embodiments, the user effectively has the option to "preview" NIMs before adding them to the processed user profile. This is advantageous because it reduces the chances of filling the user profile with undesirable NIMs. Such a feature is particularly advantageous in the case of novice or inexperienced users of system **10**. Furthermore, one of skill in the art will appreciate that the concept of transient NIM execution raises the possibility of executing NIMs on a client **20** during a period of time in which the user is not logged into system **10**. For example, consider a NIM that is executed on a client **20** after a user initiated response to a web page advertisement presented in web browser **36**. Although the user is not logged into web server **58** and therefore does not have a processed user profile resident on client **20**, the user can execute the NIM on client **20** on a transient basis. Furthermore, if the user wishes to add the transiently executed NIM to the user profile **76** associated with the user, the user can log into web server **58** and then add the NIM to the processed user profile that is delivered to client **20** as a function of the log in process.

Another important feature of the present invention is the ability for users to share NIMs with each other. For example, if a user wishes to share the NIM "AnyDay Calendar" that was added to list **188** in FIG. **6**, the user clicks "share" button **170** (FIG. **6**). In response, panel **105** is displayed (FIG. **7**). Because "share" button **170** is pressed while tab **176** is active in the illustration provided by FIGS. **6** and **7**, panel **105** lists each of the NIMs associated with the user. If, however, "share" button **170** is pressed while tab **178** is active rather than tab **176**, panel **105** will list each of the packs associated with the user instead of each of the NIMs. Returning to the situation illustrated in FIG. **7**, the user shares a NIM with other users by selecting the NIM to be shared from list **107** and then toggling button **109** "Share via email." In one embodiment, the user has the option to select multiple NIMs from list **107** using predefined keystroke operations. For example, in one embodiment, the user selects multiple NIMs by clicking on several of the NIMs in list **107** with a mouse button while depressing the "shift" button on the keyboard. When a user decides not to share a NIM and panel **105** is displayed, the user presses cancel button **111** and panel **105** is dismissed.

When a user toggles "share via email" button **109** at a time when one or more NIMs in list **107** have been selected, the definition of each selected NIM is copied from the processed user profile associated with the user into a container and the container is sent to server **50** (FIG. **1**). In the embodiment shown in FIG. **1**, the container is received by web server **58**. Web server **58** includes instructions for routing the container to sharelink database **78** where the container is stored. When the container is stored, a unique identifier is assigned to the container. Although a large number of different mechanisms for generating a unique identifier are practiced in accordance with this aspect of the invention, in one embodiment, the unique identifier assigned to the container upon storage in sharelink database **78** can be subsequently processed to form a URL address that specifically references the container within the context of system **10**. In one embodiment, after a unique identifier has been assigned to the container, an e-mail program is launched on client **20** and the user is requested to designate the recipients of the designated NIMs. Then, each recipient is provided with the unique identifier associated with the container in an e-mail message. When the recipient clicks on the unique identifier, a call is made for a copy of the associated container from sharelink database **78** and the container is delivered to the client **20** associated with the recipient.

As is readily apparent upon review of FIG. **7**, the user has the option to size and position the visual manifestation that corresponds to each NIM. Furthermore, by toggling controls, such as toggle button **101** (FIG. **7**), the visual manifestation of a NIM is toggled between an on state and an off state. Such functionality is highly advantageous. First, by using this functionality, the user has the option to create unique arrangements. Second, NIM developers have the ability to control the default position and size of NIMs as well, and can therefore produce an arrangement of NIMs to further specialized purposes. Finally, because the NIMs of the present invention are not trapped in third party applications that have a set of undesirable features such banner ads, the utility and overall appearance of an arrangement of NIMs is enhanced and adopts an independent value. Using the technology disclosed in the present invention, the user collects an assortment of NIMs and arranges them in a customized fashion. The user has the option to "capture" favored arrangements into constructs known as packs, which have been briefly discussed previously.

FIGS. **8** and **9**A illustrate the formation of a pack using the arrangement of NIMs illustrated in FIG. **4**. The process begins when the user toggles button **164** "Make Pack" in FIG. **4**. In the embodiment illustrated by FIGS. **8** and **9**A, panel **113** (FIG. **8**) is displayed when the user toggles button **164** (FIG. **4**). Panel **113** advises the user to open and arrange each of the NIMs that are to be included in a pack. In the case of FIG. **4**, for example, such an arrangement could include the arrangement of NIMs **184** and **186**. The user indicates that specified NIMs are in a desired arrangement by selecting button **115** "Next" (FIG. **8**). When button **115** is toggled, prompt **113** is terminated and prompt **117** is displayed to prompt the user for a name to associate with the designated pack. The user indicates that a name **119** has been provided for the pack by selecting "Done" button **121**.

In the embodiment shown in FIG. **8**, the user further has the option to return to panel **113** and rearrange the specified NIMs before committing to pack creation by selecting the "Back" button **123**. In the situation illustrated in FIG. **8**, the user has provided the name "New DotPack." FIG. **9**A illustrates the state of the visual manifestation corresponding to home NIM **162** after the user has selected "Done" button **121** (FIG. **8**). Specifically, the name "New DotPack" is added to pack list **125** and tab **178** is activated to display the user pack list rather than the user NIM list that is displayed when tab **176** is activated. Furthermore, in response to the user selection of "Done" button **121** in FIG. **8**, a reference to each NIM specified by the user is collected into a pack, along with some state information, and the pack is stored in the processed user profile associated with the user. Representative state information for each NIM stored in a pack includes whether the NIM was collapsed and the position of the NIM. In some embodiments, the state information includes the dimensions of the last visual manifestation corresponding to the NIM to have been displayed on output means **42**.

FIG. **9** illustrates pack **139**, which is delineated with a dashed box. Pack **139** includes five NIMs **133**. Each NIM **133** includes two primary components, a viewer **135** for viewing

content and a frame **137** for providing user functionality. Each viewer **135** provides a platform for reviewing machine readable information, such as HTML, dHTML, images, imbedded ActiveX and Java applications, JavaScript, CSS, Streaming Media, and/or Flash. Furthermore, in some embodiments, viewer **135** provides functionality for processing digitally recorded sound. Observe then that each NIM of the invention utilizes machine readable information that is easily retrieved from a specified address. If the content provider of this information desires to change the machine readable information, changes can be made and then delivered the next time that the machine readable information is addressed. This stands in contrast to prior art techniques in which updates to executable code can be relatively difficult to distribute.

Each frame **137** provides functions and controls for manipulating the visual manifestation of the NIM **133** corresponding to the frame. For example, some frames **137** include a dismiss button **141**. When the user clicks on dismiss button **141**, the corresponding NIM **133** is dismissed. Additionally, some frames **137** include a menu button **143**. When the user clicks on button **143**, a menu appears. In one embodiment, the menu is a pull down menu and the menu items are defined in the corresponding NIM definition. In an abstract example, the NIM definition provides a menu definition of the format:

Menu 143-(I)-"Get more NIMs" URL

where (I) refers to the N.sup.th element of the menu that is activated when menu button **143** is pressed, "Get more NIMs" is the menu element name, and URL is the event or web address that is called when the user selects the N.sup.th element of the menu associated with button **143**.

The developer has full control over all aspects of the appearance and functionality of NIM frame **137**. Thus, a NIM developer has the ability to control, for example, the functionality located at any corner of frame **137**, functionality placed along the top or bottom of the frame, or on the sides of the frame. As an illustration, frame **137-3** includes a control button **145** that allows the user to resize the visual manifestation of NIM **133-3**. One of skill in the art will appreciate that the bottom row of NIM frame **137** could have any number of buttons, including a forward control, a backward control, and similar controls, each of which provides the user with distinct functionality.

An advantage of the present invention is that it is possible to embed commands that affect frames **137** in the content of the computer readable media delivered to frame viewer **135**. The use of embedded commands provides NIM developers with powerful tools and additional flexibility. For example, a developer can use embedded commands, such as the menu command above, to design a NIM that has a context sensitive help menu. Each time a particular event occurs, the developer changes the content of the help menu using embedded commands. For example, when a sound file is delivered to a NIM, the sound file may be attached to a set of embedded commands that provide sound manipulation functionality in the form of a pull down menu. Elements of such a pull down menu include a command for saving the sound file to disk, commands for fast forward, stop, and play, and commands for sound enhancement. After the sound file has been played, embedded commands are used in this example by the developer to reset the menu associated with button **143** to some default state. One of skill in the art will appreciate the benefits and advantages of a frame **137** that is capable of being modified based on commands embedded in the machine readable media delivered to the corresponding viewer **135**. The developer can use presentation tools, such as adding transient help buttons, resizing the visual manifestation of the NIM, changing the frame border color, changing the title of the frame, and changing the frame border patterns, to create a more effective application.

One of the advantages of the disclosed pack system is that it provides a convenient mechanism for rapidly assembling NIMs that track diverse sources of information. Furthermore, when a useful set of NIMs is collected into a pack, the user can share the pack with other users using the same procedure previously identified for sharing one or more NIMs. It will be appreciated that in some embodiments, NIMs are commercial applications and that appropriate use of packs provides an additional dimension for application development. Therefore, in some embodiments, pack recipients are subscribers to a service provided by a pack developer. In other embodiments, pack recipients are purchasers or licensees of packs. In still other embodiments, pack recipients receive packs developed by friends, family members, or business associates.

The technology of the present invention further provides a set of NIM management tools to help a user manage displayed NIMs. In some embodiments of the present invention, these management tools are provided by visual management module **114** (FIG. **1**). Two such NIM management tools are, in fact, properties that are associated with NIMs, namely magnetism and snapping. The property of magnetism is exhibited when a visual manifestation corresponding to a first NIM is dragged or moved near a visual manifestation corresponding to a second NIM. When this occurs, the two NIMs exhibit a magnetism that causes the first NIM to accelerate toward the second NIM. However, when the visual manifestations are within a predetermined distance of each other, the NIMs snap together. In one embodiment, the predetermined distance that triggers the two visual manifestations to snap together is a gap of about five pixels. While the above discussion describes the principles of snapping and magnetism based on a pair of NIMs, there is in fact no limitation on the number of NIMs that can be snapped together and furthermore, the principle of magnetism is not dependent on whether a NIM is in fact snapped to another NIM or not.

An additional management tool, illustrated in FIG. **9**A, provides a mechanism for selecting multiple NIMs and for moving the NIMs in a coordinated fashion. In FIG. **9**A, visual manifestations **162**, **184** and **186** corresponding to respective NIMs are locked together. In response, halo **127** is drawn around the selected locked visual manifestations to graphically notify the user which NIMs are locked together. As mentioned previously, the user has the option to position NIMs as a coordinated group. For example, in one embodiment, when halo **127** is displayed and the user moves mouse arrow **129** after selecting one of the visual manifestations corresponding to a locked NIM, a target manifestation **131** is displayed to indicate to the user the position that the selected NIMs will be relocated to if the user clicks a mouse key. In some embodiments, target manifestation is a shadow image of the NIMs within halo **127** rather than the box depicted in FIG. **9**.

The present technology further provides additional methods for controlling visual manifestations of NIMs. For example, in one embodiment, the user has the option to select multiple NIMs by pressing a predefined key such as the keyboard "Ctrl" key, before selecting a specified NIM. While continuing to depress the control key, the user has the option to select additional NIMs and add the corresponding NIMs to a group. The user then has the option to move each of the NIMs in a single coordinated fashion as a group. Furthermore, by entering a designated keyboard or mouse sequence, the user has the option to move a single NIM even in situations where the NIM is in a group. In one embodiment in accor-

dance with this aspect of the invention, the user clicks the visual manifestation corresponding to a locked NIM that the user wishes to move in an independent manner and the user does not click the visual manifestation when the user wishes to move the NIM in a manner that is coordinated with the other NIMs. Additional features of the present invention include the option to select rows or columns of NIMs using specialized control sequences. For example, in one embodiment of the present invention, the user selects a column of NIMs by clicking on a NIM while holding down the alphanumeric character "c" on the keyboard.

It will be appreciated that one advantage of the present invention is that it is possible to display multiple NIMs and that each NIM provides a specialized visual experience. Therefore, NIM alignment tools are advantageous because they allow users to quickly make room on output means **42** for additional NIMs and/or to produce highly styled NIM arrangements. Accordingly, the present technology provides a specialized feature to rapidly align NIMs. This technology is illustrated in FIG. **10**. The technology works in conjunction with the tools for selecting multiple NIMs. In FIG. **10**A, the user selects the visual manifestations **147**, **149** and **151** corresponding to respective NIMs using, for example, the column select feature disclosed above. Then, the user pushes the selected NIMs against boundary **153**. In one embodiment, boundary **153** is the horizontal or vertical edge of output means **42**. In another embodiment, all visual manifestations corresponding to NIMs are displayed in a single viewport such as a window. In such embodiments, the horizontal and vertical edges of the window each represent a boundary. FIG. **10**B illustrates how visual manifestations **147**, **149**, and **151** are automatically aligned when they are pushed against a boundary, such as boundary **153**. In some embodiments, the relative alignment between the visual manifestations is preserved even after the manifestations are moved in subsequent action by the user. The present technology further allows for the rearrangement of NIMs along a particular axis. For example, if NIMs are substantially oriented along a vertical axis as shown in FIG. **10**B and the user wishes to realign the NIMs on the horizontal axis, all the user has to do is push the collection of NIMs against a horizontal border. For example, when the user pushes visual manifestations **147**, **149** and **151** against border **155**, the visual manifestations realign to conform to border **155** thus resulting in the view depicted in FIG. **10**C.

The features of the home NIM of the invention have been fully described. Attention presently turns to the architecture and operation of individual NIMs utilized in accordance with the invention.

In one embodiment of the invention, after a user has logged into the system, as discussed above in connection with FIGS. **1** to **10**, the user interface **40** displays the home NIM **162** as shown in FIG. **5**. The home NIM typically includes a list of NIMs **188**, referred to in FIG. **5** as "MyDots." These are NIMs which have been "collected" by the user. The list of collected NIMs, along with their associated definitions, is stored on the server in the user profile database **76**, and downloaded from the application server **50** in response to a request from the client parser application **38**. The local copy of the processed user profile is then further processed when the user collects or uses NIMs.

Collected NIMs may be opened or closed by clicking on the control button next to the NIMs name or on the NIM's name itself, in list **188**, and all NIMs may be closed by clicking on the "all off button' **166**. The user may place NIMs into categories in a list of categories **190**, which can be edited by clicking on the-Edit" button **168**. New NIMs may be added to the user's collection of NIMs by clicking on the "Get" button **172**.

FIG. **5** also shows a NIM **192** with a list of NIMs **198**, which may be previewed and/or collected by the client **20**. The user may preview or collect a NIM by clicking on the associated name of the NIM, e.g., "eHOW". The steps taken to provide the NIM to a user are shown in FIG. **11**. After logging in (step **240**) the user clicks on the name of a NIM, and the client parser application **38** sends a request including the NIM_ID of the NIM definition, to the applications server **50** via the transmission channel **44** (step **241**). Alternatively, the user may click on a NIM link before logging in, for example if the link has been e-mailed to the user, and then, after clicking on the link, log in. In another aspect, the user could view, but not collect, a transient NIM without ever logging in.

After the user is logged in and has clicked on the NIM, the applications server **50** retrieves the NIM definition from the NIM template database **74** using the NIM ID, in step **242**, and provides it to the client **20** in step **243**. The client **20** receives the NIM definition from the applications server **50** in step **244**, and the client parser application **38** creates a frame in the display of the user interface **42** in step **245**. In step **246**, the client **20** requests the necessary content elements **94** stored at the URLs identified in the NIM definition from the corresponding content server **82**. The content server **82** transmits the content **94** in step **247**, and in step **248** the client parser application **38** places the content in the viewer, which is enclosed by the frame, allowing the user to preview the NIM. Alternatively, the client parser application **38** may simply collect the NIM, adding it to the user's processed user profile.

The user may then view the NIM on the user interface display **42**, and may interact with the NIM much in the same way as a user may interact with Internet content or web applications. This may change the NIM from its present, "raw" state to a used state reflecting alteration or use of the NIM by the user. For example, the user may direct the NIM to different content within the NIM if the NIM content enables the user to do so. Or, the user may provide information to the content server **82** which allows the NIM to be personalized. The user may additionally be given the option of changing the size of the frame.

If the user collects the NIM, the NIM will be added to the user's list of collected NIMs such as the list **188** shown in FIG. **5**. Additionally, the client parser application will add the NIMs definition to the processed user profile, and, on logout, send the processed user profile to the application server **50**. Thus, the NIM's "state" will be preserved. Alternatively, the client parser application may collect the NIM automatically, without waiting for a user command, by adding the NIM definition directly to the processed user profile.

If the NIM's state has been altered by the user or by the content—if for example, the user has directed the NIM to Internet content other than the initially-displayed content, provided personalizing information, or changed the properties of the frame, or if the content itself has caused an alteration in the NIM—this alteration will be reflected in the NIM definition stored in the user profile database **76**. Information which personalizes the resulting content, instead of being stored in a "cookie" on the client's hard drive, can be stored as part of the NIM definition. This advantageously permits personalization of content, such as web content that is associated with the NIM content and the user, without storing a cookie on the client **20**.

A user may also access a NIM which has been previously collected, and possibly altered by use as explained above. As

previously described, the user profile **76** includes NIM definitions for NIMs which have been viewed and collected by each user. A screen shot showing NIMs **188**, which have been previously collected by a user is shown in FIG. **5**. The steps taken to provide the user with NIMs which have been previously collected are shown in FIG. **12**.

As discussed above, on login (step **250**) the user's profile is retrieved by the client parser application **38** in the client **20** (step **251** and **252**). The user's profile, stored in the user profile database **76**, includes the NIM definition for each of the NIMs previously collected, and possibly altered, by each user. The NIM definitions, as discussed above, includes the NIM frame definition and the definition of the controls for filling the viewer within the frame with content. After log in, a local copy of the processed user profile is stored on the client **20**, and this copy is further processed as the user collects new NIMs, or uses new or collected NIMs such that the NIMs are altered.

When the user clicks on the name of a collected NIM (step **253**) the client parser application **38** creates a frame in the display **42** of the user interface **40** in step **254**. At step **255**, the client **20** requests the necessary content elements **94** stored at the URLs identified in the NIM definition from the corresponding content servers **82**, which provide the content **94** in step **256**. It will be appreciated that these URLs need not be the same as the initialization URLs in the "raw" NIM definition stored in the NIM template database **74** on applications server **50**, and in fact the content servers need not be the same content servers corresponding to the initialization URLs. In step **257**, the NIM parser application **38** places the content in the NIM frame, and the NIM is then fully opened.

FIG. **13** illustrates a data structure for a NIM definition. As discussed above, a NIM is defined as a frame that contains a collection of controls, or functional units, such as a web rendering control or a GIF rendering control. The NIM frame surrounds a viewer, which displays the addressed content. The MM has a defined layout or arrangement of the controls, and defined initialization input data, e.g. data and URLS, for each control or element, in the NIM. NIM definitions are available to the client parser application via NIM links. The NIM links "point" to NIM definitions, which include all the information needed to build a NIM frame and fill the NIM with NIM content. Thus, NIMs links are easily collected, associated into packs, and shared by users.

In one embodiment, the NIM definitions are defined using Extensible Markup Language (XML), so that the NIM as a whole—the frame and the content within the viewer—is advantageously as flexible as standard web content. NIMs are extremely flexible, because the definition of the NIM is content, rather than compiled code. The NIM definition defines the structure of the NIM, and everything that is visible in a NIM is based on standard Internet content, such as HTML, dHTML, or GIFs, and is referenced or pointed to by the NIM definition. An "application"-type NIM, such as a web calendar or web mail, may be changed by the user, by the content provider, or by other content, while advantageously avoiding the need to distribute and support a hard-coded compiled application. The definition of a NIM thus includes everything that is needed for the NIM to be rendered and filled with Internet content.

As shown in the exemplary embodiment of FIG. **13**, the definition of a NIM includes tags that identify the NIM **270**, define and configure the NIM frame **271**, specify and layout the controls **273** in the NIM viewer, and specify parameters to initialize all the NIM's components with content or data.

In one embodiment, a NIM is identified by three ID strings **270**: GlobalID, Domain and Kind. A GlobaliD is used when the MM definition is within a share. It is unique with respect to other NIM tags in the share. A NIM's domain is a unique label for the owning company or developer of the NIM, such as "dodots.com." Finally, a NIM's kind, which is specified by the NIM's developer, is a helpful identifier for finding the NIM, but need not be unique. Examples of possible NIM kinds include "mp3", "scriblets," and "calculator." As discussed above, a NIM definition will typically be written in a format which facilitates sharing of data over the Internet, such as XML. An XML specification for the NIM identification strings, for one embodiment of the invention follows. The bold text identifies NIM definition data, while the remaining text describes the data.

GLOBALID="string" Used only within <SHARE> tags. This GLOBALID must be unique with respect to other <NIM> tags in this <SHARE>.

DOMAIN="string"

Unique label for the owning company of this NIM. In theory, NIMs may be limited to communicating with NIMs only from their own domain.

KIND="string"

Helpful identifier for finding such a NIM from another NIM. Does not have to be unique.

The NIM definition also includes the definition of a frame **271**, which specifies the frame size and shape, and optionally the frame orientation and/or location on the user's screen. The space within the frame is the control space or viewer; visible controls are distributed within the control space or viewer.

The NIM definition may optionally include controls for: a titlebar; a NIM menu with flexible menu entries; an exit button; and a bottombar. A typical layout for these components is: titlebar at the top of the control space, with menu on the left and exit button on the right, and the bottombar at the bottom.

The titlebar component gives the user a place to grab and drag the NIM in a windowed environment. In one embodiment, it is implemented as a GIF rendering control that can be targeted to a local or remote titlebar image. The titlebar will preferably have a fixed height and width that is a function of the NIM's width. The titlebar is preferably capable of being located at any position on the periphery of the NIM. Overlay text can also be specified to layer on top of the titlebar image. The bottombar may be implemented in a similar fashion, but typically will not include text overlay. The titlebar and bottombar may be filled in with initialization data from a fixed data file, or alternatively with Internet content from, example, an initialization URL.

In one embodiment, a menu definition **271** is also included in the NIM definition. The menu includes items and actions of the NIM provider's choosing. For example, menu items may include the title "browse" associated with the action of targeting a full-screen browser or another NIM, and retrieving content for that browser or NIM from a specified address such as a URL. Logging off, or directing the NIM to another address or URL, are also possible menu action items. Menu action items that require communication of messages between the NIM and another NIM may also be provided—for example, opening another NIM, or changing the content of another NIM that is already open. Communication of messages between different parts of the system is discussed below.

An XML specification for a frame, titlebar, bottombar and menu, for one embodiment of the invention follows:

<FRAME>

<TITLE>

TEXT="string"

JUSTIFY=–"LEFT"|"CENTER"|"RIGHT"

COLOR="#XXXXXX"
PIXELWIDTH="integer"
Width in pixel units. Overrides WIDTH attribute.
PIXELHEIGHT="integer"
Height in pixel units. Overrides HEIGHT attribute.
WIDTH="integer"
Width in NIM units. Default value is 1.
HEIGHT="integer"
Height in NIM units. Default value is 1.
X="integer"|"LEFT"|"CENTER"|"RIGHT"
Initial X position in screen coordinates. Default is center.
Y="integer"|"TOP"|"CENTER"|"BOTTOM"
Initial Y position in screen coordinates. Default is center.
FIXWIDTH="TRUE"|"FALSE"
Default is false.
FIXHEIGHT="TRUE"1"FALSE"
Default is false.
BACKGROUND COLOR="#XXXXXX"
Default is white.
<TITLEBARIMAGE>
JUSTIFY="LEFT"|"CENTER"|"RIGHT"
TILELEFT="integer"
TILERIGHT="integer"
NORMAL="URL"
DOWN="URL"
HOVER="URL"
INACTIVE="URL"
<BOTTOMBARIMAGE>
JUSTIFY="LEFT"|"CENTER"|"RIGHT"
TILELEFT="integer"
TILERIGHT="integer"
NORMAL="URL"
DOWN="URL"
HOVER="URL"
INACTIVE="URL"
<MENU>
Contains zero or more <ITEM> tags.
<ITEM>
TITLE="string"
TOOLTIP="string"
ICON="URL"
ID="string"
Must be unique.
<ACTION>RECIPIENT="address" MESSAGE="string"

As shown in FIG. **13**, the NIM definition also includes layout and definition of the controls **273**. A control may be visible and render some sort of visual or text display, either static or dynamic. A control may be hidden, for example a functional element that is not necessarily visual such as a Java control. The control definition **273** includes identification of the types of controls, the layout of the controls, and initialization information. In one embodiment, NIM controls are specified and identified by class, kind and ID. Class defines the type of NIM control and is not unique. Kind is a useful identifier selected by the developer, and again is not unique. The NIM ID is unique within a user's processed profile.

Different classes of controls may be used. For example, a control may be a web rendering object, which can render web content such as HTML, dHTML, images, imbedded ActiveX and Java applications, JavaScript, CSS, Streaming Media, and/or Flash. Alternatively, a control may be any object capable of rendering any kind of computer readable media, such as a GIF rendering object or an custom-designed object to display a particular kind of information. Alternatively, a control may be an object capable of processing any kind of application logic, such as a Java module. For example, an on-line brokerage firm could implement a custom stock-chart-rendering control, and define a NIM to use this control.

As discussed above, the control space is located within the frame, and one or more controls can be flexibly positioned within the control space, and these controls may include a titlebar and a bottombar, as well as other controls. The location of controls is specified by the layout in the definition of the controls **273** within the NIM definition. In one embodiment, the controls are laid out or positioned within the NIM frame according to a flexible grid. In this embodiment, the NIM definition allows the control space to be subdivided into equal vertical and horizontal units, and then for the controls to be positioned and sized within the control space.

A control definition will typically include initialization data. For example, where a control is a web rendering object, the definition will include initial URLs. When the NIM is opened, the control will navigate to the initial URLs to obtain content and render the NIM. If the control is a GIF, the control could retrieve the GIF file from a content server or from the application server. A NIM definition may optionally include additional tags identifying initialization parameters for different platforms: for example, a URL is suitable for a PC, but a "P-URL" may be provided as well, pointing to content suitable for users viewing NIMs through a personal digital assistant (PDA) or similar device.

Controls are typically installed on the applications server, and may be updated after installation by the applications server. The home NIM code, downloaded when the client becomes "NIM-enabled," includes the then-existing controls. Controls are updated as new controls are installed on the server or when a user requests a NIM that requires a new control. The server may then may download such updates to the client parser application, for example on log-in. The NIM framework allows any control to be positioned and initialized in a control space in a NIM, as discussed above.

An XML specification for control definition and layout, in accordance with one embodiment of the invention, follows:
<CONTROL_LAYOUT> Contains zero or more <CONTROL> tags.
WIDTH="integer" Divides control space into this many evenly spaced columns. Default is 1.
HEIGHT="integer" Divides control space into this many evenly spaced rows. Default is 1.
WIDTHSCALES="TRUE"|"FALSE" Default is true.
HEIGHTSCALES="TRUE"|"FALSE" Divides control space into this many evenly spaced columns. Default is 1.
<CONTROL>
CLASS="string" Class may be "Browser," "GIF reader," or other object for rendering computer readable media.
KIND="string"
ID="string" Must be unique with other controls in this NIM.
LEFT="integer" X position of the control in container unites. Default is 0.
TOP="integer" Y position of the control in container units. Default is 0.
WIDTH="integer" Width in container units. Default is 1.
HEIGHT="integer" Height in container units. Default is 1.
URL="URL" This is read if and only if this control is of class "Browser". This is the URL to which this control navigates. Otherwise, may include address for other control content, e.g. GIF address in applications server.
<CATEGORIES> Contains zero or more <CATEGORY> tags.
<CATEGORY> Adds the NIM to this category. This is the only way categories are specified; i.e. there is no master category list.
NAME="string" This is the name of the category.

A NIM definition may also optionally include home NIM categories **274**. A home NIM category used by home NIM **204** is a convenient way for a user to keep track of collected NIMS. When a user adds a NIM to a category **204**, the category is added, as a string element, to the categories element **274** of the NIM definition in the user profile. For example, a user may categorize a particular NIM as "entertainment," or "news," or "reference." This category will then be added to the categories element **274** of the MM definition.

A NIM definition may also optionally include an events element **275**, which defines actions to certain NIM events. For example, the OnClose event, when a NIM is closed, may be assigned a specific and targeted action, similar to a menu item. An XML specification for the event element in a NIM definition, in accordance with one embodiment of the invention, follows:

```
<EVENTS>
<ONCLOSE>
  Executes this action list on close.
<ACTION_LIST>
  Contains zero or more <ACTION> tags.
<ACTION>
RECIPIENT="address"
MESSAGE="string"
```

A sample NIM definition, in an XML file format in accordance with the above specification, follows:

```
<NIM DOMAIN="calculator" KIND="basic">
<FRAME CLASS="Standard"
BACKGROUNDCOLOR="#FFFF00"
WIDTH="6" HEIGHT="4" FIXWIDTH="TRUE"
FIXHEIGHT="TRUE">
<TITLE TEXT="Basic Calculator" COLOR="#0000FF"
JUSTIFY="LEFT"/>
<TITLEBARIMAGE JUSTIFY="LEFT" TILELEFT="1"
TILERIGHT="1"
NORMAL=" "INACTIVE=" "HOVER=" "DOWN=" "/>
<BOTTOMBARIMAGE JUSTIFY="LEFT"
TILELEFT="1"
TILERIGHT="1" NORMAL=" "INACTIVE=" "
HOVER=" " DOWN=" "/>
</FRAME>
<MENU/>
<CONTROL LAYOUT WIDTH="1" HEIGHT="1">
<CONTROL CLASS="Browser" KIND="A" ID="1"
TOP="0" LEFT="O" WIDTH="1"HEIGHT="1"
URL="http://www.dodots.com/dots/Calc/
CALCULATOR2.htm"/>
</CONTROL LAYOUT>
</NIM>
```

The first line of this definition establishes the identification of the NIM definition, as discussed above: it is in the domain "calculator," and the kind of display is "basic."

In one embodiment, the domain will be the domain name associated with the content provider. The domain name is a unique label for the provider or developer of the NIM. The NIM's "kind" is a helpful identifier for locating the NIM, and need not be unique. A NIM may also be identified using a GlobalID, when the NIM is being shared. Since the NIM defined by this XML file is not being shared, it does not have a GlobalID.

The second line of the example XML NIM definition establishes the size and appearance of the NIM frame, defining a NIM viewer in which the NIM content will be placed. The third line ensures that the height and width of the frame are fixed—that is, the size of this frame cannot be adjusted by the user. The fourth and fifth lines establish the title of the NIM—"Basic calculator"—and its location. The next four lines establish the location and placing of the titlebar and bottombar, and relevant images, e.g. mouse-over. Thus, the first part of the example NIM definition defines the NIM frame. The definition of a frame, titlebar, menu and other aspects distinguish a NIM from a browser—the content provider has control over the frame size and every aspect of the NIM's appearance, whereas when a browser is used, the content provider has to adapt to the browser display size, and browser titlebar, menu, logo and other aspects cannot be controlled by the content provider.

The rest of the NIM definition identifies, positions, and initializes the NIM's controls, which, in this case, are contained by the NIM frame. In this example, the next few lines establish that a single control will start in the upper left corner of the NIM viewer, that the control is of the type "browser," or web-rendering, and that the initialization URL for the control is www.dodots.com/dots/Calc/CALCULATOR2.htm. This URL is typically referred to as the "initialization URL," because it is where the NIM looks for NIM content when it is opened. Where the control is of type "browser," the content will typically be HTML content. However, any standard Internet content—HTML, dHTML, flash, streaming media, or Java, for example—may be used. As discussed earlier, a control, could include types other than a browser. The final two lines of the XML file close the definition file.

It will be appreciated that the NIM is designed such that content consumes the entire frame. In one embodiment, the content for the corners of the frame—the menu and the exit button—and the frame sizing images are served by an application server and referenced when the user logs in. Everything else is developed and served by a separate NIM developer. This differs fundamentally from the current approach to providing web content, in which there is a strong distinction between the viewer application—the browser—and the web page or web content. Using present browser-type technology, the content is trapped within the viewer. To obtain a cohesive application feel and access to application features, the current alternative is to develop custom client applications. NIMs allow a developer to provide an application feel without developing custom client applications.

NIMs and the client parser application have a messaging architecture—the NIM messaging architecture, or NMA—that enables NIMs, controls, and the client parser application to communicate. Messaging, in combination with the NIM definition, gives the content of a NIM access to the application/rendering program—the client parser application—and to other NIMs, allowing true application behavior. The NIM definition, discussed above, is accessible, flexible, and may be changed by a NIM or a user while the NIM is in use, even after it's been rendered.

The content provider, the user, or other NIMs can change a NIM. For example, the content—which includes the titlebar and menu elements—may be changed by the NIM provider by simply enhanced NIM content, using 20 messaging, so that NIMs can exhibit true application behavior. For example, an online brokerage firm can go beyond providing a NIM that renders stock tracking charts, and allow users to trade on-line via a second NIM that can interact with other NIMs such as the first NIM to help facilitate the activity.

A NIM can be changed by its content, or by another NIM, using messaging. This enables a NIM to, for example, notify a user of events, such as a change in content. A NIM could, for example, remain open, but in a collapsed mode, until a particular event occurs, and could then either expand to normal size, or open another related NIM. For example, if a particular stock hits a predetermined price, the stock tracking chart NIM discussed earlier can notify the user by expanding, by pop-

ping up a message, or by opening another NIM (such as a stock trading NIM). Alternatively, the NIM could notify the user of a particular event by coming into focus or changing size or content. These changes could also be made by sending messages from a content or applications server to the NIM.

As illustrated in FIG. **14**, all elements of the system can send and receive NMA messages. Message routing between NIMs, from a NIM to itself (that is, between e.g. the frame elements and a control, or one control and another), and from NIMs to the system, are handled by a message interface module **106**, which is part of the client parser application **38** in the client **20**. In one embodiment, the message interface module **106** resolves addressing queries, executes system-level commands from the NIMs such as "close all NIMs," and passes messages between NIMs. The message interface **106** may also communicate messages to controls such as the browser class controls, for example "navigate the addressed NIM's browser control to the argument URL." Alternatively, the message interface module **106** may route a message to an application or content server (not shown in FIG. **14**) for expanded functionality. In one embodiment, the message interface module **106** uses an HTTP request to access an application program interface (API) call, with data optionally being sent and received in XML format. For example, the message interface **106** could send a message providing user input, such as credit card information to a credit processing module on a web server **82**.

In one embodiment, an NMA message has two components: a recipient, or address, and the message body. Both are represented as strings. The address may specify an exact NIM, a kind of NIM or control, a domain, or the system, meaning the overall home NIM display. For example, in one embodiment, the address may be in the form:

#<NTM specifier>:<control specifier>

if the communication is between NIMs in the same domain, or

#<domain>:<NIM specifier>:<control specifier>

or

#<domain>:<NIM specifier>:<NIM id>:<control kind>:<control id>

if the communication is between NIMs in different domains, or

#system

if the communication is to the system.

In one embodiment, if the address is not properly specified it defaults to #system. The message interface **106** in the client **20** can flexibly allow, restrict MM addressing or sending messages—for example, the message interface can ensure that only certain NIMs can send message to a particular NIM. This allows NIM developers to develop coordinated NIMs that can interact, by sending messages to e.g. change content or open one another, without allowing other NIMs to interact with their coordinated NIMs.

In one embodiment, a specifier in n address may be the unique identification of the NIM, control or domain in question: <specifier>:=<ID>. Alternatively, where the kind of NIM or control receiving the message is important but the specific NIM or control is not, the specifier may address a message to the closest matching recipient using a search criteria: <specifier>:=<kind>#<search criteria>. The kind should be a type of NIM or control that is installed in the system. Search criteria may be, for example, "any," "open," or "closed." Finally, where the control is being specified, the specifier may be a symbol, such as ".", indicating that the message is addressed to the sending NIM. For example, in one embodiment, the address #system sends the message to the system. The address #. sends the message to the NIM which sent the message. The address #7 sends the a message to the NIM with the identifier 7. The address #B#open:5 sends a message to the control with the identifier 5, in the first open dot of kind B found in the system.

In one embodiment, certain NIMs will have privileges to send particular messages to certain NIMs, and not to other NIMs. For example, NIMs in a domain may be permitted to send control messages to other NIMs in the same domain, but not to NIMs in other domains. Thus, a NIM provider may have control over messaging between NIMs in his domain, and prevent NIMs in other domains from changing NIMs in his domain. Alternatively, NIM providers may coordinate with NIM providers in other domains, permitting certain messaging privileges between some of the NIMs in their respective domains. The HomeNIM and other system NIMs, such as the login NIM, which are in the system domain, may have certain messaging privileges that no other NIMs have, such as logging the user out or closing all the currently-open NIMs.

In one embodiment of the invention, the second part of the message, the body, is represented as a string of characters. Messages may be sent to the client parser application, to the frame of a NIM, or to a control. Messages may be specific, defined messages, as shown in the following examples, or may be any javascript, which may be sent in or out of NIM content. Examples of defined messages to the client parser application, in one embodiment, are:

Application Message<arg>	Function
Refresh	Refreshes the user's profile.
#have-NIM <NIM-address>	Check if the user has the specified NIM as part of the user's profile.
#delete-NIM <NIM-address>	Remove the specified NIM from the user's profile.
#get-screen-width	Returns the width of the screen.
#get-screen-height	Returns the height of the screen.
#close-all-NIMs	Closes all open NIMs.
#get-NIM-ids <NIM-address>	Returns the NIM ID of the specified NIM.

Messages can also be sent from a NIM to itself, or to another NIM, and the identified actions or functions, specified in the body, are performed on the receiving NIM. The following are examples of messages to a NIM for one embodiment of the invention:

Defined NIM Message<arg>	Function
#set-size<width><height>	Sets the size of the NIM.
#set-width<width>	Sets the width of the NIM.
#set-height<height>	Sets the height of the NIM.
#set-position<x-pos><y-pos>	Sets the position, of screen, of the NIM.
#set-title<title>	Sets the title of the NIM.
#collapse	Collapses, but does not close, the NIM.
#uncollapse	Uncollapses the NIM.
#set-user-sizable<width true/ false> <height true/false> sizeable	Establishes whether the NIM is by the user
#set-background-color<color>	Sets background color of the NIM.
#set-title-text-justify<justify-keyword>	

The above examples of NIM messages may, in one embodiment of the invention, be sent to a NIM by another NIM. For example, a user may provide input to a NIM, for example a stock tracking chart NIM, indicating an interest in another NIM, such as a stock trading NIM. The current NIM may then send an "open" message to the second NIM to open

it, if it wasn't already open. The current NEM may then send a navigate message (see below) that may include an argument, such as a URL or other content pointer, so that the second NIM could be opened to a specific URL indicated by the first NIM.

Finally, messages may be sent to a control of a NIM, either by the NIM itself, another control, the HomeNim system, or another NIM. As examples, the following control messages are provided in one embodiment of the invention:

Defined Control Message	Function
##<any javascript>	Run any javascript in the control, e.g. javascriptin a browser-type control.
#show	Set visibility control within a NIM.
#hide	Set invisibility of a control within a NIM.
#get-size	Get size of control.
#get-address	Query for unique ID of a control by kind.
#navigate	Navigate the control.

Control messages may be used by a NIM, addressed to its own control, or to the control of another NIM. Control messages may also be sent by the HomeNIM, or by the applications server or content server.

An example of a message to a NIM is window.external-.PostMessage("#.:","#collapse)" this is a message from a NIM, to itself, collapsing the NIM. Another example of a message from a NIM to another NIM is "window.external-.PostMessage("#mp3#any:","#open")", which is addressed to a NIM with the kind "mp3," but only if it is in the same domain as the sending NIM. The body of the message instructs the message of type mp3 to open. An example of a javascript message is "window.external.PostMessage (Vmp3#any:1", "33play( ))", which sends a message to the mp3 NIM control that calls the javascript function play( ).

The operation of the home NIM and individual NIMs has been described. Attention presently turns to different techniques used in accordance with the invention **25** to host NIMs.

As shown and described in relation to FIG. **1**, the application server **50** includes a NIM Management Modulel **12**, a NIM Templates Database **74**, a ShareLink Database **78**, and a user profile database **76**.

FIG. **15** is a diagrammatic illustration of an embodiment of the NIM Management Modulel **12**. NIM Management Modulel **12** may contain the NIM Templates Database **74** and the ShareLink Database **78**, discussed in further detail below. NIM Management Module **112** may also contain content **402** for filling in a NIM frame or for rendering Internet pages. Alternatively, content **402** may be stored elsewhere, such as on a Web server similar to the server **82** shown in FIG. **1**. As discussed above, the content preferably contains Internet content such as HTML (Hypertext Markup Language), dHTML, and images.

In addition, Management Modulel **12** preferably contains executable procedures **403** for controlling and managing the NIM system. These procedures **403** may include: a Collection procedure **404** for obtaining new NIMs; a Sharing procedure **405** for sharing NIMs with others; Development procedures **406**, such as a DevZone procedure **407** and a DevNIM procedure **408**, for creating, modifying or deleting NIMs; Publishing procedures **409**, such as a PubZone procedure **410** and a PubNIM procedure **411**, for publishing NIMs so that they are publically accessible; and Administration procedures **412**, such as an AdminZone procedure **413** and an AdminNIM procedure **414**, for administering the system. It should be noted that the NIM sharing procedure, discussed in further detail below, may be processed by either the Server module (**102** of FIG. **1**), or the Sharing procedure (**405** of FIG. **15**). Control and management of the server and the NIM Management Modulel **12** components will now be discussed in further detail.

FIG. **16** is a diagrammatic illustration of an embodiment of the NIM Templates database **74**. In this embodiment, NIM Templates database **74** primarily stores XML NIM definitions in their initial unmodified state as initially designed by a partner and which have not been altered by a user in any way. The unmodified NIMs are also referred to as "Raw NIMs". The NIM Templatesdatabase **74** is used as the starting point for the development of NIMs which may later be customized or modified by a user, developer, or system administrator, as discussed below.

For ease of explanation, the individual or organization that controls the server (**50** of FIG. **1**) will hereinafter be referred to as the system provider and the individual or organization who supplies the content will hereinafter be referred to as a partner. It should be understood that a provider, partner, user, developer, and administrator of the system may be distinct entities, the same entity, or a combination of both. Furthermore, as discussed above, each of the above entities is assigned access rights or privileges which permit or forbid that entity from performing different actions on the system.

FIG. **16** illustrates an embodiment of a NIM Template database **74**. A NIM identification number (NIM_ID) **416** is stored in a NIM Template table 41.5 in the NIM Templates database **74**. A Raw NW, identified by its includes a plethora of RAW NIM characteristics, including, but not limited to, a Raw NIM creation date **417**, which indicates when the NIM was created and is useful when searching for NIMs created during a specific time, a Raw N.I.M definition module **418**, and the NIMIndex categories **422** in which the NIM has been categorized. Each NIM when created in typically classified into one or more NIM Index categories by the developer, such as "Applications", "Business", "Entertainment" and "News".

Each NIM is fully configurable and definable. The NIM definition module **418** contains details defining the NIM, such as the look-and-feel **419**, of the Raw NIM, initialization URLs (Uniform Resource Locators) **420**, and a location **421** of where the developer would like the NIM to open on a users's computer screen. The look and feel of the NIM is the appearance and function of the NIM interface. The look and feel may characterize the frame or skeleton layout, the graphics used to represent certain functions, such as opening and closing the NIM, whether the frame is sizable, and the appearance and operation of menus in the frame.

The definition module **418** may also contain Initialization URLs **420** which reference resources containing content. The content resources may be HTML (Hypertext Markup Language), dHTML, images, programs such as Java applets, or any other file supported by HTTP. The Initialization URLs **420** contains the name of the protocol required to access the resource, a domain name that identifies a specific computer on a network, such as the Internet, and a hierarchical description of a file location on that specific computer. These files or resources are then used by the home NIM to fill in the frame and controls with content. In addition, the definition module **418** may contain other details such as the location **421** on a user's computer screen where the NIM should initially open.

The NIMIndex may be used to search for, learn about, and collect NIMs. The

NIMIndex is typically accessed from either a Web browser, such as Internet Explorer® or Netscape Navigator® or from

the home NIM, **108** of FIG. **1**. A user may search for NIMs by, or according to, any field of the NIM Templates table **415** via the NIMIndex.

A user accessing the NIMIndex from a Web Brow typically navigates to a main NIMIndex web page such as that shown in FIG. **17**. FIG. **17** is an illustration of the main NIMIndex Web page **423**. A user typically navigates to main NIMIndex Web page **423** from a NIM home-page (not shown), or while anywhere within the NIM Web site by clicking on the "Collect the DOtS™" link **424** in a menu **425**. A user may search the NIMIndex by entering a search term in a form **426** and clicking on the "Search" button **428** which implements the Collection Procedure (**404** of FIG. **15**) to search the NIM Templates database (**74** of FIG. **15**) for NIMs that match the query. In one embodiment, the NLMIndex may be searched by NIM title, NIM description, or partner, as shown in the pull down menu **430**. The user may also browse the NIMIndex by clicking on a link **432** to a NIMIndex category **434** which will navigate the user to a NIMIndex category Web page **440**, as shown in FIG. **18**.

FIG. **18** is an illustration of a single NIMIndex category, the "Applications" category **435**, shown in FIG. **17**. A list of sub-categories (not shown), as well as a list of NIMs **442** and their short descriptions are shown in FIG. **18**. A user may click on the "more detail" link **444** to be taken to a page displaying a full description (discussed later in relation to FIG. **29**B) of the NIM, shown in FIG. **19**.

Navigation of the Internet generally occurs through the use of URLs (Uniform Resource Locators), which are the addresses of files or resources accessible on the Internet. The type of resource depends on the Internet application protocol. Using the World Wide Web's protocol, the Hypertext Transfer Protocol (HTTP), the resource can be an HTML (Hypertext Markup Language) page, an image file, a program such as a Java applet, or any other file supported by HTTP. The URL contains the name of the protocol required to access the resource, a domain name that identifies a specific computer on the Internet, and a hierarchical description of a file location on the computer and usually takes the form: "URL=protocol://machine.name[:port]/directory/document. name'?[&argu- ments] " The "protocol" is the Internet protocol used to reach the document or resource. On the Web, the "protocol" is typically HTTP, but it can take any number of forms, such as ftp (file transfer protocol), file (a local file), gopher (gopher protocol), mailto (electronic mail address), news (Usenet news), telnet and tn3270 (interactive sessions), wais (wide area information servers), or the like.

The "machine.name" is the name of the host come where the document resides (such as www.NIM.com). The ":port" portion of the address is optional and is only necessary when a resource is accessible through a non-standard TCP port number. Although the standard port number for HTTP is 80, there are numerous Web servers on the Internet that use non-standard ports, such as port 8000.

The NIM system, however, may also utilize a proprietary NIM protocol. An example of a URL using the proprietary NIM protocol is: "NIMS:?NIMTemplate=<N/M_ID>"

The NIM protocol URL is used to collect, distribute, and share NIMs. When collecting NIMs the NIM protocol URL is referred to as a NIMLink. When distributing or sharing NIMs the NIM protocol URL is referred to as a ShareLink.

The "NIMs:" term defines the NIM protocol or scheme and is always followed by a colon. The "?NIMTemplate=<NIM_ID" is an argument, where a dollar sign (5) and a question mark (?) are used to denote path and/or search elements. It should be noted that no path is supplied (i.e.: "//path/to/something"). The argument instructs the client parser application (**38** of FIG. **1**) how to handle a user's selection of a NIM protocol URL and what the NIM protocol URL must do. For example, to obtain a NIM, the argument might read "NIMTemplate=123', to obtain a Share (discussed below) the argument might read "Share=123", to obtain a Pack (discussed below) the argument might read "Pack=123", etc. The argument can be used to cause the client parser application to do anything within it's system of functionality by specifying new argument sets to build new types of special client parser application links.

In one embodiment, the address for where the client parser application (**38** of FIG. **1**) searches the system (**10** of FIG. **1**) for the NIM Template or ShareLink database (**74** and **78** of FIG. **1**) is specified within the processed login script or session_config, although it could alternatively be specified within the NIM protocol URL. When a user clicks on a NIM protocol URL (from any where you can place and click on a link, for example in a browser, in a NIM, in email, in a document, etc.), the client parser application processes the NIM protocol URL in the same manner as a browser processes HTTP links and an email program processes mailto links.

When a protocol URL takes the form of a NIMLink, the client parser application responds by obtaining the NIM definition from the NIM Template database, optionally adds the NIM to the user's processed user profile (unless the NIM has been specified to be opened in transient mode, which may be specified in the argument), and optionally opens the NIM on the user's display screen.

In one embodiment, by default, unless specified otherwise, a NIM will be added to a users collection (transient—false) and will be opened (open—true). A NIMLink with arguments may look as follows: "NIMS:? NIMTemplate&transient=true" or "NIMS:? NIMTemplate&open=false". More than one additional argument could be added by appending another argument to the URL which may read as "&argument=value".

A ShareLink (discussed below) is similar to a NIMLink and may read "NIMS:?share=123", where 123 is the SHARE ID referencing the share module within the ShareLink Database. Pack Links (discussed below) typically read as "NIMS:? pack-123", where 123 is the PACK_ID referencing a pack module within the NIM Template Database.

FIG. **19** is an illustration of a full description of NIM content **446**. A graphic of the opened NIM may also be displayed **448**.

Once the user decides that he would like to add a NIM to his home NIM, the user clicks on the "get it now" or "Get This Dots™ NIMLink **450** (FIGS. **18** and **19**) which either runs the Collection procedure (**404** of FIG. **15**) which obtains that NIM's NIM definition module (**418** of FIG. **16**) from the NIM Template table (**415** of FIG. **16**), or opens another Web page as shown in FIG. **20**.

FIG. **20** is an illustration of a Web page **452** which might be displayed to the user once the user has clicked on the NIM-Link **450**. The user is presented with an option of either collecting the NIM **456**, or if the user does not have the home NIM application, the user may first download the home NIM by clicking on "Get the homeDotIm" **454**. Once the user clicks on the download the NIM button **456**, the Collection procedure (**404** of FIG. **15**) obtains that NIM's NIM definition module (**418** of FIG. **16**) from the NIM Template table (**415** of FIG. **16**).

The Collection procedure (**404** of FIG. **15**) transmits the NIM definition to the user's home NIM, which optionally opens the MM and saves the NIM definition module (**418** of FIG. **16**) on the user's local processed user profile. All NIM

definition modules (**418** of FIG. **16**) on the user's computer may subsequently be saved to the user profile database, as discussed earlier in this writing. Alternatively, a "preview" button may be provided which transiently displays the NIM on the user's computer screen without adding the NIM to the user's local processed user profile. The user may also search the NIMIndex from their home NIM.

FIG. **21** is an illustration of the main home NIM graphical user interface (GUI) **464**, similar to that shown in FIG. **5**. The home NIM displays a list of all NIMs **466** that the user has collected. Furthermore, any NIMs that the user has collected in groups or packs, can be accessed by clicking on the "My Dotpacks'" tab **468**. One way to obtain new NIMs is to click on the "Get" button **470**, which opens the NIM shown in FIG. **22**.

FIG. **22** is an illustration of a get new NIM GUI **474**. A list of all NIMs **476** (or a featured subset) that may be collected by the user are displayed. Clicking on the "More Dots" tab **478** (shown in FIG. **23**) displays further NIMs which may be collected.

When a user selects or clicks on any of the NIMLinks **480**, NIMLink **480** references the NIM_ID (**416** of FIG. **16**) for that NIM in the NIM Templates database (**74** of FIG. **16**). The collection procedure (**404** of FIG. **15**) receives the (**416** of FIG. **16**) from the user, locates the NIM definition module (**418** of FIG. **16**) corresponding to that NIM_ID in the NIM templates database, and transmits the NIM definition module to the user's computer. That NIM may automatically be opened on the user's computer screen. The NIM is saved to the user's list of NIMs on their home NIM (**466** of FIG. **21**), and the NIM definition module is saved in the user's local processed user profile. Alternatively a "preview" button may be provided which transiently displays the NIM on the user's computer screen without adding the NIM to the user's local processed user profile, as discussed above in relation to the NIMLink. All the NIM definitions that the user has listed on their home NIM are saved to the user profile database either periodically, at a set time, by event, or when the user closes their home NIM. The technique of the invention facilitates a viral distribution architecture. In other words, the technique of the invention facilitates rampant distribution of generated NIMs, as described below.

Users (or developers) may share NIMs they have collected, and perhaps even modified, with other users (or developers) in accordance with this viral distribution architecture. Because the NIM definition contains basic reference information, such as data to instantiate the NIM and URLs and other references to where the NIM content is located, a NIM is easily and quickly distributed, collected, and shared. By-packaging Internet content and applications as NIMs and referencing the NIMs by NIMLinks, the system advantageously gives Internet content viral characteristics as the NIMs can easily be distributed or shared between users.

Each NIM definition contains just enough information to define and initialize the NIM's components (NIM frame, controls, etc.). For example, this information may contain data to configure the skeleton or frame that is filled in by NIM content from a developer's server. The NIM definition is therefore fairly small in size (~2K), and is therefore easily distributable as an XML file or Blob (binary large object), which is communicated using the same mechanisms (HTTP/HTTPS requests) as regular Web pages.

This is especially useful where a user has collected a NIM or a group of NIMs (Packs) that he would like to send to another user. For example, a user may have an online trading NIM, calculator NIM, and stock research MM all set up in various positions on his screen, and would like to share the entire Pack with a friend who is remotely connected to the Internet.

To share NIMs with others, the system utilizes the Sharelink database **78** of **25** FIG. **15** and the Sharing procedure **405** of FIG. **15**. FIG. **24** is a diagrammatic illustration of the Sharelink database **78**. NIM Sharelink database **78** stores a list of all NIMs shared by users, developers, or administrators, in a share table **484**. Each NIM or group of NIMs shared is assigned a Share ID **486** which points to a Share module **488**. Each Share Module **488** may also include a creation date **490**, multiple **30** individual MM definition modules **492**, or multiple packs of NIMs that have been shared (Sharepack module **494**) containing multiple NIM definition modules **496** and **498**.

FIG. **25** is an illustration of a Share NIM's GUI **500**. All dots collected by the user (**466** of FIG. **21**) can be shared with other users by clicking on the "Share" button **502** shown in FIG. **21**. Once the user has clicked on the "Share" button **502**, the GUI **500** shown in FIG. **25** is launched. The user may then highlight any of the NIMs or packs of NIMs **504** he has collected or created and thereafter share the NIMs or packs of NIMs **504** by clicking on the "Share via email" button **506**. It should be noted that other means of distributing the NIMs may be used together with, or instead of, email.

When users share NIMs or NIM packs, their home NIM application generates a **10** share module, which may for example be an XML Blob containing the N1M definition or Sharepack modules shared. The shared NIM XML is then sent to, and saved in, the Sharelink database (**78** of FIGS. **1** and **24**). The Sharing procedure **405** of FIG. **15** then automatically generates a shared link (ShareLink) that references or points to the address of the shared XML on the Sharelink database. This ShareLink is then sent or **15** distributed (via email or posted on a Web site) to other users.

If a user receives shared NIM(s) or pack(s) and has a home NIM installed on his client computer, then clicking on the Sharelink adds the NIM(s) to the user's home NIM and opens the shared NIM(s) on the user's screen. If a recipient of a shared NIM does not have the home NIM installed on his computer, then the home NIM is downloaded and installed (with the user's cooperation), the shared MM is added to his local processed user profile, and the NIM is opened.

The NIM management module (**112** of FIG. **15**) may also be responsible for controlling and managing the development of new NIMs via the DevZone and the DevNIM discussed below.

Because NIM content is based on existing Internet content standards (HTML, DHTML, GIFs, etc.) developers can create MM content using their existing Internet content development tools and methodologies. Therefore, no special hardware or software is required to develop or serve NIM content.

Furthermore, as the application server (**50** of FIG. **1**) hosts and delivers NIM definitions from the NIM Templates database (**74** of FIG. **1**) developers merely define and package the NIM content without directly authoring, hosting, or serving the XML NIM definitions. Therefore, no special hardware or software is required on the developer-side to host and serve the NIM content, other than required for their regular Internet content.

Two means are provided for creating NIMs. First, a Developer Zone Web site (DevZone) and second, a set of developing NIMs (DevNIMs). Both means enable NIM developers to create, define, and modify NIM definitions, and to support the NIM development process which results in XML NIM definitions being added to the NIM Templates database and NIMLinks generated.

The DevZone is a Web site where NIM developers can view a list of NIMs they have defined and/or published, add new NIMs, and categorize, view, modify, or delete their existing NIMs. The DevZone is preferably rendered in a Web browser, is hosted on the Web server (**82** of FIG. **1**), and is implemented with a DevZone procedure (**406** of FIG. **15**). To access the DevZone, the developer may typically pass through a secure portal, such as by supplying a login identity and password.

FIG. **26** is an illustration of the main DevZone Web page **510**. All NIMs created by the NIM developer appear in a customized NIM list **514** that may only be accessed by that NIM developer. All NIMs created by a developer appear on the NIM list **514**, unless they have been deleted by the developer or by a system administrator. The NIM list may contain the NIM name **518**, the date the NIM was created 520, and an indication **516** of whether the NIM is in development or accessible by the public in the NIMIndex (i.e. "in-development" or "published").

To access the NIM definition (for modification or review) the developer clicks on a "modify" or "preview" link **524** as transient (e.g. to add the NIM to their home NIM for previewing and testing). By clicking on the "modify" link, the developer is taken to the NIM modification web page, as shown in FIG. **27**. Alternatively, by clicking on the "Create a Dot" button **522**, the developer is taken to a web page similar to the NIM modification web page shown in FIG. **27**, where the developer may create a new NIM.

FIG. **27** is a partial view of a NIM modification web page **530**. To modify an existing NIM, or create a new NIM definition, a developer preferably utilizes Web forms, such as **534** to **542**, or any area that contains objects that capture user input, such as text entry spaces, check boxes, and selection buttons. Developers typically fill in forms with information which defines the NIM, where the details might include the NIM's name **534**, the URL for any image associated with NIM **540** (as shown in FIG. **18**), the URL for a detailed image **542** (**448** of FIG. **19**), and such details as NIM frame (e.g., size of NIM, sizeable), layout of the controls (e.g., WebConduit control), and to specify the initial MM content (e.g., the initial target URLs for the WebConduit control, TitleBar, BottomBar), and any categories in which the developer would like the NIM to be listed in the NIMIndex. Once the developer has completed or modified the forms, he may either save or delete the NIM **532**. If the developer selects either the development check box **536** or the public check box **538**, and then saves the NIM, the DevZone procedure (**407** of FIG. **15**) generates a XML NIM definition, stores the XML NIM definition in the NIM Templates database (**74** of FIGS. **1** and **15**) and returns a NIMLink pointing to that NIM which is listed on the NIM list (**514** of FIG. **26**) on the developer's home NIM. The only difference being that once the developer selects the public check box **538** and saves the NIM, the NIM definition is published utilizing the PubZone publishing procedure (**410** of FIG. **15**) to a publically accessible portion of the NIM Template Table (**415** of FIG. **16**), from where users can access, download, and collect the NIM. If the developer selects the development check box **536**, the NIM can only be viewed and or modified by the developer and system administrator. It should be noted that the DevZone only allows control of certain characteristics of each NIM. Other characteristics may be set to default while still other characteristics can only be altered by an administrator. In an alternative embodiment, the DevZone may be rendered in a NIM or group of NIMs just as it was rendered in a Web browser. In either embodiment, NIM developers fill out one or more forms specifying NIM definition parameters, an XML NIM definition gets created and stored in the NIM Templates. Database, and a NIMLink gets generated that points to the new NIM. The Developer can then view or debug this NIM by clicking on the NIMLink to add it to his home NIM, or preview as transient, and thereafter render it on his screen. NIM definitions may also be developed using NIMs and NMA messages. A **30** developer may create Raw NIMs from empty NIM Templates using a development NIM (the DevNIM) on the developer's home NIM.

FIGS. **28**A to **28**D are GUIs of a development NIM (the Dev NIM). A developer may obtain a DevNIM by either collecting the DevNIM in the usual manner, as discussed above, or the system, via the system administrator, may share the NIM with the developer, also as discussed above. The DevNim contains a DevNIM procedure (**408** of FIG. **15**) which is transmitted to the developer's home NIM, as discussed above.

To create a new NIM, the developer launches the DevNIM and enters a NIM name **550** into the DevNIM. The server then obtains an empty NIM (a NIM with default or no initialization data and with only basic characteristics) from the NIM Template Database using the procedure for collecting NIMs described above, and saves the empty NIM under the supplied new NIM name **550** locally in the developer's processed user profile. The developer may then modify the empty NIM to the required form using the DevNIM. In the preferred embodiment a pull down menu **552** is provided where the developer can select which feature to modify, such as the frame characteristics (FIG. **28**B), the titlebar (FIG. **28**C), or initialization URLs for different frame or control elements (FIG. **28**D).

Each time the developer modifies a setting, the DevNIM, using the DevNIM procedure, sends NMA messages to the newly saved NIM to modify its definition parameters. For example, modifying the NIMs name, size, TitleBars, BottomBars, or WebConduits (as shown in FIGS. **28**A to **28**C).

Unlike the DevZone, a new XML NIM definition and NIMLink is not generated every time a modification is made. All modifications (during the development cycle) are made locally to the NIM definition and are stored in the developer's processed user profile. The DevNIM embodiment, therefore, requires a separate publishing step that promotes the newly created NIM definition from the developer's user profile, to the NIM templates database on the application server.

To publish the NIM, the NIM developer categorizes the NIM and the NIM definition is copied from the developer's processed user profile to the NIM Templates database.

A publishing NIM (PubNIM), implemented with Publishing procedures **409**, is provided to handle these functions. The PubNIM may therefore be shared or transmitted to the developer along with the DevNIM. The PubNIM contains a PubNIM procedure (**411** of FIG. **15**) which controls the publication of the NIMs to the NIM template database, as discussed above. The PubNIM procedure sends a NIM definition module to the application server which receives the NIM definition module, extracts the NIM definition from the share module, stores it in the NIM Templates database, and associates the NIM with the developer so that the NIMLink shows up on the developer's NIM list (in their DevZone account).

Alternatively, the new NIM may be published directly from the DevNIM. Once a user is satisfied with the NIM, he may select an option which publishes that NIM definition to the NIM Templates database. It should be noted that a developer may modify his NIMs at any time from the DevNIM.

As mentioned earlier, when a developer is first authorized to create and/or modify NIMs, or at any time thereafter, information about that developer is saved in that developer's user profile (**76** of FIG. **1**) on the application server.

A developer may also create application programs using NIMs, which a user may access from his client computer. Just as client-side application characteristics (sizing, position, menus) are accessible to content via NMA, the system may offer server-side application functionality, or toolkits, which are accessible through the NMA.

A developer can build a NIM application without implementing, hosting, or supporting complex server or client applications. By using the server toolkits, a developer can develop N1Ms that exhibit server-application behavior by focusing on implementing NIM content (just like standard Internet content).

For example, a NIM's content (an HTML page) may send a message to the system (or server) to request a credit card to be processed. Other toolkit examples may include credit card billing, user profiling, targeted advertising, email, chat rooms, Internet telephony applications, or calendars.

Any server-side application can be made accessible through the NMA, as a toolkit, just as client-side application behaviors are made accessible. In the current implementation, server-side application functions could be offered by a NIM (exposed via javascript functions on a page in a hidden frame). Other NIMs could access this functionality by sending NMA messages to this "Toolkit NIM" calling the functions. A NIM developer may therefore focus on Internet content development while accessing the features, behaviors, and functionality of an application just as if he had developed custom client and server side applications. The NIM management module (**112** of FIG. **15**) may also be responsible for controlling and managing the administration of the system via the AdminZone and the AdminNIM discussed below.

A system administrator has the power to create, modify or delete users, developers, NIMs, other administrators, or NIMIndex categories, depending on that administrators access privileges. In a similar manner to the DevZone and DevNIM, system administrators may utilize either a Web browser administration zone (AdminZone), or an administration NIM (AdminNIM) which both make use of Administration procedures (**412** of FIG. **15**).

To access the AdminZone, an administrator typically passes through a secure portal, such as by supplying a login identity and password. Once within the AdminZone, the administrator may search for a NIM by NIM name or title **552**, category, developer, developer contact name, or status, as shown in FIG. **29**A. The administrator may also selectively search for NIM's in development or publically accessible NIMs **554**.

Utilizing an AdminZone procedure (**413** of FIG. **15**), once the required NIM **20** is located the administrator may modify or delete the NIM in a similar manner to a developer as shown in FIG. **29**B, and described above.

Also utilizing the AdminZone procedure, the administrator may manage NIMIndex categories by creating new categories, modifying or deleting existing categories, and/or adjusting the layout of the NIMs within those categories as shown in FIGS. **30**A and **30**B. For example, an administrator may change a category's name **558**, designate the category active or inactive **556**, or create sub-categories **560**. The system administrator may also select a category or categories for the NIM to appear in, where each NIM may be registered in more than one category.

Finally, utilizing the AdminZone procedure (**413** of FIG. **15**), an administrator may search for users, providers, or developers and adjust their details, as shown in FIGS. **31**A and **31** B. The system administrator may, for example, change a users contact details. In addition to adding, modifying or deleting NIMs, system administrator may have the task of reviewing NIM submissions from developers and promoting NIMs to the public. A submissions list of newly submitted NIMs may be displayed to an administrator, who may promote the NIM to the public or view the NIM. Once promoted, changes are made to the NIM Templates database and the NIM is automatically removed from the submissions list (again by utilizing the AdminZone procedure (**413** of FIG. **15**)).

The foregoing discussion has explored the inherent nature of NIMs. Attention now turns to different techniques that may be used to exploit information that is associated with the use of NIMs. In particular, the following discussion is directed toward the accumulation of statistical information that is only available in view of the architecture of the present invention.

Currently, the predominant method of tracking and collecting user online behavior is severely limited for a number of reasons. First, most Internet use or visitor statistics are single-dimensional (linear, sequential) because Internet content is presented to users one full-screen page at a time. Second, users visit and leave sites so rapidly their visits are barely meaningful. Third, user's browsing habits are often discontinuous (browsers give users navigational bypass controls—back, forward, home, refresh, stop, etc.). Fourth, user behavior tracking is limited from a single site's server point of view. Current use statistics are plagued with the challenge of tracking continuous user behavior (especially from a cross-company perspective), with more than a single dimension of use context. Finally, because a computer may have multiple users, or a single user may use multiple computers, tracking continuous user on-line behavior is extremely difficult.

One of the advantages of the NIM system as illustrated in FIG. **1**, is that the Server **50** is able to track continuous, long-term NIM use information about each user. This is because the NIM server, through communication with the home NIM, can track each NIM event performed by each user. Therefore, it is possible to track each individual user's entire NIM use activity from the moment the user downloads the home NIM.

Referring to FIG. **32**, in one embodiment of the invention the following events may be tracked by the Event Log Module **98** (within the client computer **20** of FIG. **1**): home NIM Download Event **704** NIM Download Event **706** NIM Display Event **708** Web Click-Through Event **710** Page-View Event **712** First NIM Installation Event **714** First home NIM Startup Event **716** Transient Mode Event **718** Share NIM Received Event **720** NIM Pack Received Event **722**.

A home NIM Download Event **704** is logged when the user clicks on a link to request the home NIM user application. Preferably, the start time **752**, and the end time **754** are recorded for this event. Also recorded is the provider ID **750** which is a parameter (generally, an integer) that represents the content provider partner who provided the link to the user.

NIM Download Event **706** is logged when the home NIM acquires a NIM via a NIMLink. The start time **752**, the end time **754**, and the provider ID **750** are recorded for this event. Also recorded is the NIM ID **756** which is a parameter (generally, an integer) that represents the NIM that was just downloaded.

NIM Display Event **708** is logged when a user activates a NIM. The NIM ID **756**, the start time **752**, and the end time **754** are recorded for this event. Web Click-Through Event **710** is logged whenever a user links from a NIM to a full-screen browser. This can occur when a user clicks on a link in the NIM, or it can occur automatically through the NIM messaging, or directly through the content provider. The NEM ID **756**, the start time **752**, and the Internet address **758** of the link are recorded. Page-View Event **712** is logged

whenever a user views a page of content within a NIM. The NIM ID **756**, and the start time **752** are recorded for this event.

First NIM Installation Event **714** is logged the first, a NIM or NIM Pack is installed from a web site. This event is logged only once for each user account. The NIM ID **756**, start time **752**, and end time **754** are recorded for this event.

First home NIM Startup Event **716** is logged when the home NIM runs for the first time. This event is logged only once for each user account. The start time **752** is recorded for this event.

Transient Mode Event **718** is logged when the home NIM runs in transient mode. Transient mode occurs when the home NIM runs before the user has logged in. The start time **752**, and the end time **754** are recorded for this event.

Shared NIM Received Event **720** is logged for each NIM a user receives as part of a share. If a NIM Pack is shared, this event will be recorded for each NIM in the shared pack. The NIM ID **756**, the start time **752**, and the end time **754** are recorded for this event.

Shared NIM Pack Received Event **722** is logged for each NIM Pack a user receives as part of a share. Thus, when a NIM Pack is shared, an Event **720** will be logged for each NIM in the NIM Pack, while an Event **722** will be logged once for the NIM Pack itself. The start time **752**, and the end time **754** are recorded for this event.

The events listed above are tracked in one particular embodiment. Other embodiments may track more or perhaps fewer events. This comprehensive event tracking is possible because each user event can be identified by the NIM Server through communication with the home NIM. Additional events may include tracking when a user sends a share or tracking when a user sends a NIM or a NIM Pack.

FIG. **33** shows a typical series of user actions **800** as they are tracked by the Event Log Module **98**. First, a user may request to download the home NIM application (step **802**) from either a partner's web site or the NIM Server **50**. The Event Log Module **98** records a home NIM Download Event, as shown with field **704** of FIG. **32**. The start time **752**, and the end time **754** are preferably recorded. Also, the provider ID **750** of the site from where the home NIM download request was received is recorded.

Returning to FIG. **33**, the user subsequently activates the home NIM for the first time (step **804**). The Event Log Module **98** records a First home NIM Startup Event **716**, as shown in FIG. **32**. The start time **752** is preferably recorded. In addition, the home NIM is activated and the user lot yet logged in, a Transient Mode Event **718** is logged and the start time **752** is recorded.

As shown in step **806** of FIG. **33**, the user logs into the home NIM. When this occurs, the end time **754** may be recorded for the Transient Mode Event **718**.

A user download of a new NIM (step **808**) may be from a partner's web site or the NIM Server. When this occurs, the raw NIM definition is copied into the user's User Profile **76**. The event log **98** records two events. First, because this is the first NIM the user has installed, a First NIM Installation Event **714** is recorded. The start time **752**, the end time **754**, and the provider ID **750** of the download site are preferably recorded. The second event recorded is a NIM Download Event **706**. The Event Log Module **98** preferably tracks the NIM ID **756**, the provider ID **750**, the start time **752**, and the end time **754** for this event. The next thing a user may do is open the NIM (step **810**). This consists of retrieving the NIM definition from the user's User Profile and getting NIM content from the provider **82**, as discussed above. The NIM is displayed for the user and the Event Log Module **98** records a NIM Display Event **708**. However, at this point, the Event Log Module **98** can only record the start time **752**, and the NIM ID **756** for this event. The end time **754** is recorded when the NIM is closed.

For every page of content a user views within a NIM **812**, a Page-View Event **712** is recorded. Some page views may require content from the provider **82**. The NIM ID **756**, and the start time **752** are recorded for this event.

The NIM may also enable the user to click on a link that results in navigating to a full screen web browser (step **814**). When a user does this, a Web Click-Through Event **710** is recorded. The Event Log Module **98** records the NIM ID **756**, the start time **752**, and the URL of the web site that is passed from the NIM content to the browser **758**.

When the NIM closes (step **816**), the end time **754** for the NIM Display Event **708** is recorded. When the user logs out of the home NIM (step **818**), the event log is uploaded to the Server **50** (of FIG. **1**).

In one embodiment of the invention, the previously described Event Log Module **98** (within the client computer **20** of FIG. **1**) tracks user events in the home NIM user application and uploads the information to the Statistics Database **80** (of the server computer **50** of FIG. **1**) at predetermined intervals alternate embodiments, the Event Log **700** (in FIG. **32**) may be processed by the NIM Server before it is stored in the Statistics Database **80**. For example, the NIM Server may process NIM use status information for each user that is currently logged in.

The Statistics Database **80**, illustrated in FIG. **34**, preferably lists every event **1002** by every user of home NIMs along with the corresponding fields associated with each event. For-example, if a NIM Display Event is recorded, the User ID **1004** of the user that performed the event is listed, the start time **1006** is listed, the end time **1008** is listed, and the NIM ID **1010** is listed. If a Web Click-Through Event is recorded, the User ID **1004** is listed, the NIM ID **1010** is listed, the start time **1006** is listed, and the URL of the web site **1014** is listed. The Statistics Database **80** therefore allows the list of events to be easily referenced and searched by each event or by each of the fields associated with the events.

Referring to FIG. **35**, the Statistical Analysis Module **900** uses the Statistics Database **80** in order to provide various services for the content provider partners **82**. Preferably, the Statistical Analysis Module **900** includes a Multi-Dimensional Consumer Profile Module **902**, a Real-Time Advertising Module **904**, and a Pack Building Module **906**, as discussed below.

A primary advantage of the present invention is that, because NIMs are used in groups and are used more often and for longer periods of time than web pages or web sites, real-time multi-dimensional NIM use data (that's a function of which NIMs are activated simultaneously) can be accumulated. In accordance with an embodiment of the invention, this accumulated data is used to generate a multi-dimensional consumer profiling database. The Multi-Dimensional Consumer Profile Module **902** uses information from the Statistics Database **80** to examine, for each user, the start time, and the end time of each NIM Display Event It then determines the NIMs (using the NIM IDs) that are opened simultaneously for each user. The Module **902** determines, for every selected NIM, the other NIMs that a given user may use in conjunction with the selected NIM. The Module **902** also determines how often these other NIMs are used simultaneously with the selected NIM. For example, Company X provides a NIM for selling its books. The Multi-Dimensional Consumer Profile Module **902** determines for Company X that a particular user has a NIM related to finance activated 30% of the time the user has the book-selling NIM acted, a NIM related to computers 20% of the time the user has the

book-selling NIM activated, and a NIM related to wedding gifts 5% of the time the user has the book-selling NIM activated. This will provide Company X with a more complete profile of the user's interests.

The Real-Time Advertising Module **904** determines the NIMs that each user has displayed at any given moment. This information is used by a content provider partner or by the NIM Server to target advertising information. For example, if a user has a NIM related to sports displayed simultaneously with Company X's book-selling NIM, Company X uses this information to stream an advertisement for a sports book. In one embodiment, this is accomplished by associating each NIM with a context keyword. This is done by incorporating the context keyword into the NIM definition or, alternatively, by maintaining a table of NIMs and their corresponding context keywords. For example, the NIM related to sports is associated with the context keyword "sports." Moreover, the Real-Time Advertising Module **904** may combine the real-time user information with the historical user information from the Statistics Database **80** to provide advertisers with a complete picture of a user's interests.

The Pack Building Module **906** uses the Statistics Database **80** to determine which NIMs are being used simultaneously. The Module **906** also determines which NIMs are being shared as NIM Packs. From this, the Module **906** provides information to content provider partners about which NIMs should be bundled together. In alternate embodiments, the Module **906** builds a NIM Pack based upon the information it processes. For example, if the Pack Building Module **906** determines that an airline NIM is being used with a hotel NIM and a car rental NIM, the Module **906** may build a NIM Pack with a restaurant NIM.

Additionally, in one embodiment of the present invention, the NIM Server **82** may track the content within a NIM in a Content Database **1050**, as illustrated in FIG. **36**. A content descriptor **1052** which may be a string describing the content that is shown within the NIM is recorded for content shown in the NIM. For example, if a NIM displayed an advertisement for an automobile followed by an advertisement for a restaurant, the two recorded content descriptors might say "automobile ad" and "restaurant ad." In addition, the NIM ID **1054**, the start time at which the content is displayed **1056**, and the end time **1058** are all preferably recorded for each content descriptor.

Referring to FIG. **35**, The Content Analysis Module **950** is able to correlate, at any moment, the content displayed to the user as recorded in the Content Database with the user's NIM activity recorded in the Statistics Database. For example, if one NIM displays to a user an advertisement for a travel book, the user may open a NIM related to Florida, a NIM owned by a specific airline, and a NIM owned by a car rental company. This pattern of user behavior will allow the company that provides the travel book advertisement to better understand the effect of the advertisement on the user. The company may use this information to make cross-promotions with other NIM providers, or, simply to provide more effective targeted advertisements.

In an alternative embodiment, each of the content providers may track its own content information. The content providers could then compare its content information with the user information provided by the Statistics Database of the NIM-Server.

Finally, referring to FIG. **37**, all of the user event information may be used in conjunction with user information provided at login. During the login process, the user may be required to enter demographic information such as age, marital status, etc. In one embodiment, this information is stored in a User Account Database **1100**. Each User ID **1102** is listed along with the corresponding user information **1104**. Therefore, it is possible to match the user events with personal information about the particular user to give advertisers or NIM content providers a more complete behavior profile of each user.

The foregoing description, for purposes of explanation, used specific nomenclature to provide a thorough understanding of the invention. However, it will be apparent to one skilled in the art that the specific details are not required in order to practice the invention. In other instances, well known circuits and devices are shown in block diagram form in order to avoid unnecessary distraction from the underlying invention. Thus, the foregoing descriptions of specific embodiments of the present invention are presented for purposes of illustration and description. They are not intended to be exhaustive or to limit the invention to the precise forms disclosed, obviously many modifications and variations are possible in view of the above teachings. The embodiments were chosen and described in order to best explain the principles of the invention and its practical applications, to thereby enable others skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the following claims and their equivalents.

What is claimed is:

1. A client computing device configured to access content over a network, the client computing device comprising:

electronic storage configured to store networked information monitor template associated with a networked information monitor, the networked information monitor template having therein a definition of a viewer graphical user interface having a frame within which time-varying content in a web browser-readable language may be presented on a display associated with the client computing device, wherein the frame of the viewer graphical user interface lacks controls for enabling a user to specify a network location at which content for the networked information monitor is available; and

one or more processors configured to execute one or more computer program modules, the one or more computer program modules being configured to access the networked information monitor defined by the networked information monitor template, wherein accessing the networked information monitor defined by the networked information monitor template results in:

transmission, over a network to a web server at a network location, of a content request for content to be displayed within the frame of the viewer graphical user interface defined by the networked information monitor template;

reception, over the network from the web server at the network location, of content transmitted from the web server in response to the content request, the content being time-varying;

presentation, on the display, of the viewer graphical user interface defined by the networked information monitor template outside of and separate from any graphical user interface of any other application; and

presentation, on the display within the frame of the viewer graphical user interface defined by the networked information monitor, of the time-varying content received from the web server.

2. The method of claim **1**, further comprising, responsive to reception of one or more elements included in the received time-varying content, modifying a feature of said viewer

graphical user interface defined by the networked information monitor template in accordance with a modification corresponding to the received one or more elements.

3. The client computing device of claim **2**, wherein said modification corresponding to the received one or more elements comprises a modification to an image defined by the networked information monitor template as forming a part of said viewer graphical user interface.

4. The client computing device of claim **2**, wherein the correspondence between the modification and the received one or more elements is defined by the networked information monitor template.

5. The client computing device of claim **2**, wherein the one or more computer program modules and the networked information monitor template are configured such that modifying the feature of the viewer graphical user interface comprises adjusting a size of the frame of the viewer graphical user interface.

6. The client computing device of claim **2**, wherein the one or more computer program modules and the networked information monitor template are configured such that modifying the feature of the viewer graphical user interface comprises changing a color of a frame border or background of the viewer graphical user interface.

7. The client computing device of claim **2**, wherein the one or more computer program modules and the networked information monitor template are configured such that modifying the feature of the viewer graphical user interface comprises modifying text of the viewer graphical user interface in a manner defined by the networked information monitor template.

8. The client computing device of claim **1**, wherein the networked information monitor template includes a markup language file.

9. The client computing device of claim **1**, wherein one or more computer program modules are configured such that the time-varying content is received from the web server over the network according to the TCP/IP protocol.

10. The client computing device of claim **1**, wherein the network location corresponds to a uniform resource locator included in the networked information monitor template.

11. The client computing device of claim **10**, wherein the one or more computer program modules are further configured such that accessing the networked information monitor defined by the networked information monitor template results in transmission of the content request to the uniform resource locator included in the networked information monitor template, and the content request being transmitted according to the TCP/IP protocol over the network.

12. The client computing device of claim **1**, wherein the one or more computer program modules are further configured:

to transmit, over the network to a networked information monitor server, a request for the networked information monitor template;

to receive, from the networked information monitor server over the network, the networked information monitor template; and

to store the networked information monitor template to the electronic storage.

13. A computer-implemented method of accessing content over a network on a client computing device, the client computing device having electronic storage and one or more processors configured to execute one or more computer program modules, the client method comprising:

storing, to the electronic storage, a networked information monitor template associated with a networked information monitor, the networked information monitor template having therein a definition of a viewer graphical user interface having a frame within which time-varying content in a web browser-readable language may be presented on a display associated with the client computing device, wherein the frame of the viewer graphical user interface lacks controls for enabling a user to specify a network location at which content for the networked information monitor is available;

accessing the networked information monitor defined by the networked information monitor template, wherein accessing the networked information monitor defined by the networked information monitor template results in:

transmission, over a network to a web server at a network location, of a content request for content to be displayed in the viewer graphical user interface defined by the networked information monitor template;

reception, over the network from the web server at the network location, of content transmitted from the web server in response to the content request, the content being time-varying;

presentation, on the display, of the viewer graphical user interface defined by the application media package template outside of and separate from any graphical user interface of any other application; and

presentation, on the display within the frame of the viewer graphical user interface defined by the networked information monitor, of the time-varying content received from the web server.

14. The method of claim **13**, responsive to reception of one or more elements included in the received time-varying content, modifying a feature of said viewer graphical user interface defined by the networked information monitor template in accordance with a modification corresponding to the received one or more elements.

15. The method of claim **14**, wherein said modification corresponding to the received one or more elements comprises a modification to an image defined by the networked information monitor template as forming a part of said viewer graphical user interface.

16. The method of claim **14**, wherein the correspondence between the modification and the received one or more elements is defined by the networked information monitor template.

17. The method of claim **14**, wherein modifying the feature of the viewer graphical user interface comprises adjusting a size of the frame of the viewer graphical user interface.

18. The method of claim **14**, wherein modifying the feature of the viewer graphical user interface comprises changing a color of a frame border or background of the viewer graphical user interface.

19. The method of claim **14**, wherein modifying the feature of the viewer graphical user interface comprises modifying text of the viewer graphical user interface in a manner defined by the networked information monitor template.

20. The method of claim **13**, wherein the networked information monitor template includes a markup language file, and wherein storing the networked information monitor template comprises storing the markup language file.

21. The method of claim **13**, wherein the time-varying content is received from the web server over the network according to the TCP/IP protocol.

22. The method of claim **13**, wherein the network location corresponds to a uniform resource locator included in the networked information monitor template.

23. The method of claim **22**, wherein accessing the networked information monitor defined by the networked information monitor template results in transmission of the content request to the uniform resource locator included in the networked information monitor template, and the content request being transmitted according to the TCP/IP protocol over the network.

24. The method of claim **13**, further comprising:

prior to storing the networked information monitor template to the electronic storage, transmitting, over the network to a networked information monitor server, a request for the networked information monitor template; and

receiving, from the networked information monitor server over the network, the networked information monitor template.

* * * * *