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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
21 March 2002 (21.03.2002)

PCT

(10) International Publication Number
WO 02/23375 A2

(51) International Patent Classification[7]: G06F 17/00

(21) International Application Number: PCT/US01/17830

(22) International Filing Date: 1 June 2001 (01.06.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/661,332 13 September 2000 (13.09.2000) US

(71) Applicant: **SMARTSERV ONLINE, INC.** [US/US]; One Station Place, Stamford, CT 06902 (US).

(72) Inventors: **SANTOSSIO, Randy, L.**; 204 New Haven Avenue, Apt 2B, Derby, CT 06418 (US). **LA PORTE, Gerard**; 7 Hillcrest Drive, New Fairfield, CT 06812 (US).

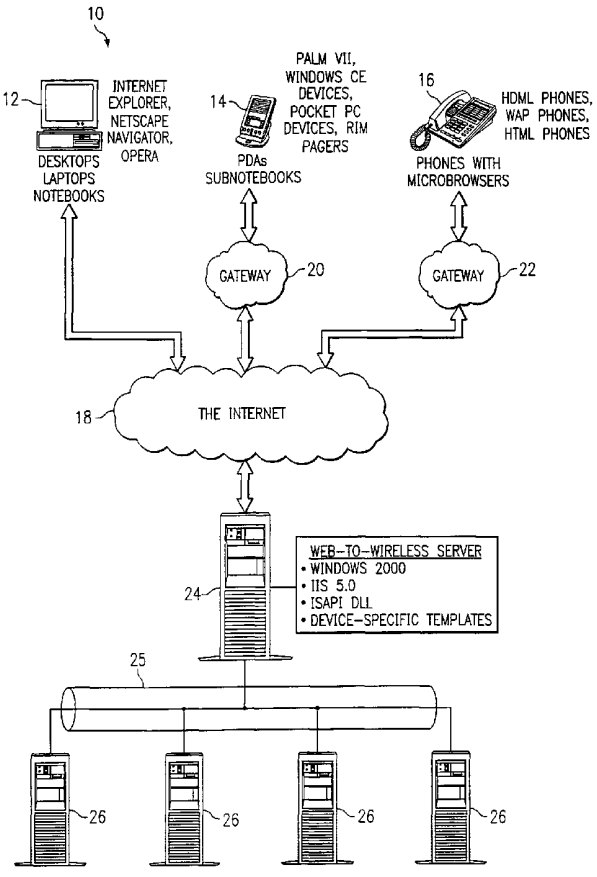
(74) Agents: **PERKINS, Jefferson** et al.; Piper Marbury Rudnick & Wolfe, P.O. Box 64807, Chicago, IL 60664-0807 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: METHOD AND APPARATUS FOR PROVIDING DEVICE-SPECIFIC FORMATTED DATA TO A PLURALITY OF BROWSER-ENABLED DEVICES



(57) Abstract: A method and system for facilitating communications between a content delivery server and a plurality of browser-enabled devices, including both wired and wireless devices. The method includes the step of initiating a request for data by a user using a browser-enabled device, where the request includes indicia of device and browser type. The request is transmitted from the browser-enabled device to the system's content delivery server across a communication network such as the Internet. The request is received by the content delivery server, which then retrieves the data requested by the user from networked data servers or from a third-party server. The data is formatted by the content delivery server as a function of the indicia of device and browser type. The formatted data is finally transmitted from the content delivery server to the browser-enabled device across the communication network.

WO 02/23375 A2

Published:

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

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

METHOD AND APPARATUS FOR PROVIDING DEVICE-SPECIFIC FORMATTED DATA TO A PLURALITY OF BROWSER-ENABLED DEVICES

FIELD OF THE INVENTION

The present invention relates to the retrieval and display of content and, in particular, to the display of requested Internet information on a plurality of browser-enabled devices, including both full capability browser-enabled devices and limited capability browser-enabled devices.

BACKGROUND OF THE INVENTION

It is estimated that the largest growth in Internet usage in the next few years will come from other than personal computer ("PC") users. The majority of that growth will come in the form of wireless Internet devices, including wireless personal data assistants ("PDA"), wireless phones and wireless interactive pagers. Information and material that is made available via the World Wide Web ("Web"), however, generally is designed to be accessed by Full Capability browser-enabled devices (hereinafter "FCB Devices"), predominantly PCs. Limited Capability browser-enabled devices (hereinafter "LCB Devices") display content in ways vastly different from PCs and from each other. In order for these different devices to be able to access the Web as efficiently as PCs do now, providers of Internet content may need to provide their material in more than one, and possibly several different formats, each of which is suited for a different type of browser-enabled device.

This reality is a particular problem for E-commerce providers. In the fast paced Internet business environment, it is important to adapt to customers' needs and desires as quickly as possible, or the competition will fill the space. Not only is it desirable to bring E-

commerce to the LCB Devices in a quick fashion, it is also desirable to do it cost effectively. If an E-commerce provider was required to replicate its current Internet offering across a variety of different servers for display on different devices, the cost could be prohibitively expensive.

An example of an E-commerce provider seeking to provide service by means of LCB Devices may be a brokerage that already provides an Internet trading service. The brokerage may desire to permit its existing customers to access its trading service via a portable wireless device to provide them with greater flexibility and service. The brokerage also may desire to capture potential new customers who may wish to access brokerage trading service functions via such a device. Thus, the brokerage would be faced with adapting its existing on-line trading service functionality for display on, and interaction with, a multitude of LCB Devices. Due to the vastly different programming requirements for displaying Web pages on different devices, this would not be a simple task. It may even entail, as stated above, the need to recreate the existing on-line brokerage service on multiple different web servers for each device potential customers may use.

Other examples of the above dilemma are not hard to imagine. Even the basic Web content provider who wishes to provide access to its content through a multitude of different browser-enabled devices is faced with the prospect of creating multiple different server systems for each individual device. There is a great need, then, for a system which will retrieve Internet information in response to a request from a user, recognize the specific type of device making the inquiry and supply the requested information in the proper format for that particular browser-enabled device. It is desirable that such a system is able to determine the type of browser -enabled device and browser requesting such information, so that the

system can retrieve the requested data, insert the data into a document in the required format for the specific device, and send the formatted data to the requesting device.

In addition to the multiple display requirements required for the multitude of Web devices, another problem encountered in adapting existing Internet E-Commerce sites for use via LCB Devices involves the use of cookies. Cookies provide for the storing, on a browser-enabled device, of information that can be accessed by a Web server to identify user or session information, and potentially to customize the resulting information for the user's needs. Currently, the ability to store cookies on LCB Devices is non-existent or very limited. Since many E-commerce sites require the use of cookies to function properly, it will be necessary for an E-commerce provider to resolve this problem. In other words, there is a need to address the use of cookies by E-commerce Internet sites that seek to be accessed by both FCB Devices, which have the ability to store cookies, and LCB Devices, many of which do not have the ability to store cookies.

SUMMARY OF THE INVENTION

In general, the networked content delivery system of the present invention involves a process in which Internet content providers may provide for the delivery and display of their content on a plurality of known browser-enabled devices, including PCs, wireless notebooks, PDAs, interactive pagers and wireless phones.

The basic operation of the invention is as follows. A customer, using a browser-enabled device, makes a request for data. The request can be as simple as requesting the display of information from a Web site by entering the address, or Uniform Resource Locator ("URL"), of the site into the device's browser, micro-browser or clipping service. The invention includes an intermediary system that determines the type of device that is the source

of the request, as well as the markup language that the device's browser can read. The system retrieves the data from a third-party Web server or from a data server on a local area network, and places it into a template for transmission to the requesting device. The template is a partially completed page with some dynamically replaced data variable markers, or tags, and is specific to the type of device making the request, the markup language being used and the type of request that was initially made. The system requires the request to have been initiated by a browser that is capable of reading at least one of a pre-defined set of markup languages, such as HTML, Handheld Device Markup Language ("HDML"), Extensible HyperText Markup Language ("XHTML") or Wireless Markup Language ("WML"). The system further requires the type of request itself to be one of a pre-defined group of requests.

An example of the system in operation, although the system is not limited to such example, involves the facilitation of an online brokerage system. A requesting browser-enabled device may be a PC, a laptop, a PDA, an interactive pager or a wireless phone, with the request being made by a browser that is capable of displaying the resulting document (a "Markup Language Document" or "MLD") formatted in one or more of the known markup languages: HTML, HDML, XHTML or WML. The type of request may be a stock quote, stock price history, stock research, stock transactions, stock news, transaction history, account information or customer service. Templates are created for each combination of the listed variables and are used for each request received.

Another aspect of the invention is its ability to recognize and reformat "cookies" for devices that may be unable to store them. Many existing Web sites, and virtually all E-commerce sites, require that the browser that is attempting to access the site be capable of accepting and storing a "cookie" file on the browser-enabled device. Cookies are files that contain a small amount of identifying information relating to the device, user, session or a

combination of these and possibly other pieces of information. Some wireless phones, PDAs and two-way pagers, however, currently are unable to accept and store cookie files from Web sites that attempt to place them. Under one embodiment of the present invention, when requested data is retrieved from a third-party Web site, the system of the invention will intercept a cookie that the third-party Web site is attempting to set, convert the cookie data into a text field, place the converted text into the response template and send it to the requesting device as part of the final Markup Language Document (in a manner that it will not be displayed to the user). If a subsequent request is sent back to the system from that results document, the cookie data is passed back to the system with the request, reconverted to its original format, and passed to the third-party Web server as a normal cookie. The process repeats itself with the response back to the system.

Thus, the invention provides for the ability of one Web server and one set of data sources to communicate with a plurality of browser-enabled devices, including PCs, laptops, PDAs, wireless phones, and two-way pagers. The invention eliminates the duplicity that would have been required under the prior art. The invention also allows for the rapid integration of wireless Internet access to existing Internet sites by being able to retrieve information from third-party Web servers and format the data for the specific LCB Devices accessing the site. An E-commerce provider with a current Web server for PCs can provide access to its Web site through a plurality of wireless LCB Devices without the costs or time associated with completely duplicating the server for each device. The invention's ability to translate the cookie information facilitates the ability to use existing Web site designs for access to new devices.

TERMINOLOGY

Definitions of certain terms used to describe the present invention, as defined by the inventors, are given below:

- Browser-enabled Device: A computer device, including, but not limited to, personal computers, notebook computers, personal data assistants, laptop computers, handheld computers, wireless two-way pagers, and wireless phones, containing browser software which allows the device to retrieve information from a remote source.
- Content delivery server: A computer server connected to browser-enabled devices across a network, programmed to retrieve data requested by the devices and format the retrieved data for display on the device consistent with the capabilities of the device browser.
- Full Capability Browser-enabled Devices ("FCB Devices"): Devices containing browser software capable of completely displaying documents formatted in the HyperText Markup Language ("HTML").
- Limited Capability Browser-enabled Devices ("LCB Devices"): Devices containing browser software capable of displaying documents formatted in certain markup languages, including other than HTML, but which may or may not be capable of completely displaying documents formatted in HTML.
- Networked content delivery system: A system for delivering content across a computer network where the network can be a local area network, the Internet, an intranet, or any other type of wired or wireless connection between one or more browser-enabled devices and a content delivery server.
- Markup Language Document (MLD): A document formatted in one of a known set of markup languages for display on a browser-enabled device.

- Markup Language Document (MLD) Template: A template formatted in one of a known set of markup languages used to create final MLDs for the display of requested data on browser-enabled devices.
- Specific Application Function (SAF): A programmed function residing within an ISAPI DLL which performs the specific commands requested by a user of a browser-enabled device on a networked content delivery system.
- Specific Application Function (SAF) Template: A template file formatted in a markup language, used to create datasets of requested data to be inserted into a larger, final MLD template used to display requested results to a user.

BRIEF DESCRIPTION OF THE DRAWINGS

Further aspects of the invention and their technical advantages will be discovered by reading the following detailed description, when read in conjunction with the drawings, in which like characters identify like parts and in which:

Figure 1 is a high-level schematic diagram of the networked content delivery system of the present invention in one embodiment;

Figure 2 is a high-level schematic diagram of the networked content delivery system of a present invention in a second embodiment;

Figure 3 is a functional block diagram of a content delivery server of the present invention as used in the embodiment of FIG. 1.

Figure 4 is a functional block diagram of a content delivery server of the present invention in the embodiment of FIG. 2.

Figure 5 shows a HyperText Transfer Protocol ("HTTP") header, received by the content delivery server of Figures 3 or 4, as part of a request from a device using Microsoft® Internet Explorer, Version 5.5.

Figure 6 shows an HTTP header, received by the content delivery server of Figures 3 or 4, as part of a request from a device using Palm, Inc.'s Web browser for Palm handheld devices.

Figure 7 shows an HTTP header, received by the content delivery server of Figures 3 or 4, as part of a request from a device using Phone.com's WAP/HDML browser.

Figure 8 shows a flow chart for an initial device/browser recognition module of the present invention.

Figure 9 shows a flow chart for a log-in function in one exemplary application of the networked content delivery system of the present invention.

Figure 10 is a first illustration example of the display of an HTML Web page on a two-way wireless pager.

Figures 11a, 11b, 11c and 11d show a flow chart for a specific application function in one exemplary application of the system of the present invention.

Figure 12 is a second illustration example of the display of an HTML Web page on a two-way wireless pager.

Figure 13 is a third illustration example of the display of an HTML Web page on a two-way wireless pager.

Figure 14 is the HTML code for a template for the display of the results of the specific application function illustrated in Figures 11a, 11b, 11c and 11d.

DETAILED DESCRIPTION OF THE INVENTION

Figure 1 shows the preferred embodiment of the networked content delivery system of the present invention. The system, indicated generally at 10, is capable of receiving requests from a multitude of browser-enabled devices, which can include PCs 12, PDAs 14 and wireless phones 16. Other browser-enabled devices such as two-way pagers (not shown) are capable of employing the invention; in general, the invention has application to any device which has a display and which has some form of browser software that permits a user to retrieve information from a remote source. These devices communicate with the Internet 18 either directly, as in the case with PCs or laptops, or through gateways 20, 22, as required by PDAs and browser-enabled wireless phones. These gateways 20, 22 can be a component of a commercial wireless service provider, similar to the more common Internet Service Providers ("ISP") that are used by PCs and laptops. Examples of services that implement such gateways 20, for PDAs are Palm.net from Palm, Inc. of Santa Clara, California, and Go.net from GoAmerica Communications Corp. of Hackensack, New Jersey. Examples of services that implement such gateways 22 for wireless phones include AT&T Pocketnet from AT&T Wireless Inc., of New York, New York, and Sprint PCS Web, from Sprint PCS Inc., of Kansas City, Missouri. Once a request from any of these devices 12, 14 or 16 is made, and is communicated through the Internet 18, it is directed to the content delivery server 24. The content delivery server 24 then communicates with and retrieves data from a back-end data server or servers 26. As shown in FIG. 1, the back-end data servers are connected to the content delivery server 24 through a network such as a local area network 25.

An alternative embodiment of the networked content delivery system 10 is depicted in FIG. 2. Here, a third-party Web server 28 replaces the back-end data servers 26 and is connected to the content delivery server 24 via a communication network 19. The

communication network 19 may be a local area network, a wide area network, a virtual private network or the Internet.

The operation of the networked content delivery system 10 will now be described. To facilitate the understanding of the system, each component of the system depicted in FIG. 3 will be generally described and then an example of the system's operation shall follow. Specifically, each component's functions, while executing an exemplary application entitled Snap Quote, from SmartServ Online, Inc. of Stamford, Connecticut, will be explained. The explanation of this particular application, however, should not be interpreted to limit this invention to this exemplary application.

In the illustrated embodiment, the core of the networked content delivery system 10 is a content delivery server 24 running an Internet Web server application such as Internet Information Services ("IIS"), by Microsoft Corporation, Redmond, Washington, running on operating system software such as Microsoft® Windows® 2000 Server, also by Microsoft Corporation. The specific application functions performed by the content delivery server 24 make use of a collection of Internet Server Application Programming Interface ("ISAPI") Dynamic Link Library ("DLL") files. ISAPI is a framework for creating DLLs to provide Internet server-side functionality. The ISAPI DLL files perform the specific functions requested by the software program and IIS facilitates this operation.

The IIS software on the content delivery server 24 is the first interface to receive requests from browser-enabled devices. For each received request, IIS calls a specific application function, or SAF, that resides within an ISAPI DLL to perform the request specified by the user. If at that point the ISAPI DLL is not yet loaded into memory on the content delivery server 24, for example because this is the first call to an SAF resident in that ISAPI DLL, then the ISAPI DLL will be loaded into the memory of the content delivery

server 24 and an instance of the SAF will be called by IIS. Along with the SAF call, IIS will pass a set of parameters to the SAF. These parameters are dependent on the type of request being made and the specific SAF being called.

Once a request is made via a call to the relevant SAF, it is handled generally as depicted in FIG. 3, which shows a block diagram of the basic process flow of the networked content delivery system 10 of the present invention. Each SAF has its own server request manager 30 as part of its structure. The task of the server request manager 30 is to listen for and receive requests from IIS 29 once the ISAPI DLL in which the SAF resides has been loaded into memory. When the server request manager 30 receives a request for information, it classifies and processes that request. The server request manager 30 then informs the thread pool manager 32 that a request has been received.

Each ISAPI DLL of the networked content delivery system 10 includes its own thread pool manager 32 that is responsible for managing the ISAPI DLL's own thread pool 34. The functionality of the thread pool manager 32 and thread pool 34 are the same in all of the system's ISAPI DLLs; there may, however, be a different number of threads allocated within the thread pool 34 for each ISAPI DLL. As an ISAPI DLL is loaded into memory, its associated thread pool manager 32 is initialized. Upon initialization of an ISAPI DLL's thread pool manager 32, it creates a certain number of threads, the amount of which is preset for each ISAPI DLL. A thread is the basic entity to which the operating system allocates time access to the computer's central processing unit. A thread can execute any part of an application's instruction code, including parts that currently are being executed by another thread. The thread pool manager 32 then corrals the created threads in the thread pool 34 and sets them to the "ready" state. Each thread receives its instructions from the thread pool manager 32.

The thread pool manager 32 manages the thread pool 34 to permit the content delivery server 24 to handle multiple requests at the same time. The individual threads in the thread pool 34 make all of the data access calls to the data servers 26 and collect the raw data necessary to fulfill requests from users. The thread pool manager 32 provides instructions to each thread, based on requests of the server request manager 30 that called the thread pool manager 32. When the thread pool manager 32 hands off the instructions that it received from the server request manager 30 to a thread, it marks that thread as "busy." While one thread is performing its assigned task, the thread pool manager 32 continues to accept calls from the server request manager 30 of this SAF (as well as from other server request managers 30 of other SAFs resident within the same ISAPI DLL) and to assign tasks to other threads within the thread pool for this ISAPI DLL. In the event that all of the threads in the thread pool for this ISAPI DLL are busy, the server request managers 30 begin to wait in line until a thread is "ready," at which point the thread pool manager 32 then accepts the instructions from the next waiting server request manager 30. When a thread has completed its assigned task, it notifies the thread pool manager 32, which then marks it as "ready."

The specific function of the request handler 36 is different for each SAF. It is controlled by the thread, which was assigned by the thread pool manager 32 when the SAF's server request manager 30 made a request. The request handler's 36 sole purpose is to request and receive data from the data servers 26. Its function differs for each SAF, each of which makes requests for different types of data. In almost every case, the thread's request handler 36 will create a socket (or sockets) of its own for data communication with a data server (or servers) 26. A socket is a Transmission Control Protocol/Internet Protocol ("TCP/IP") connection to another TCP/IP server. Using the parameters that were initially aggregated by the SAF's server request manager 30, the thread's request handler 36 builds a

command that is sent via the socket connection(s) to the data server(s) 26. The response from the data server(s) 26 is received by the request handler 36, which in turn passes the resulting data to the response handler 40.

The response handler 40 then begins to massage the data so that it can be delivered to browser-enabled devices 12, 14 or 16. The response handler 40 will perform any validation of the resulting data that may be required. The data is checked against a set of rules to make sure that it is valid before it is sent off to the data integrator 44.

FIG. 4 depicts a block diagram corresponding to the alternative embodiment of FIG. 2. All components are the same with the exception of the third-party Web server 28 and the inclusion of a request cookie handler 38 and a response cookie handler 39. In this alternative embodiment, the third-party Web server 28 may make use of state objects, or cookies, which are small files that provide for the storing, on a browser-enabled device, of information that is accessed by a Web server to identify user or session information, and potentially to customize the resulting information for the user's needs. Since many of the Web browsers on LCB Devices are not capable of accepting and storing cookies, the data within the cookie (which is returned from a third-party Web server 28) is converted to a format that can be stored within the final Markup Language Document that is to be returned to the device. Thus, when the request handler 36 receives a cookie as part of the response from a third-party Web server 28, the response handler 40 will call the response cookie handler 39 to convert the cookie's value into a text string. The response cookie handler 39 returns the converted text string to the response handler 40, which will build it into the final MLD. On a subsequent data request to a third-party Web server 28 known to expect the cookie as part of that request, the request handler 36 will call the request cookie handler 38 to reformat the cookie data in the template

into a true Web-based cookie. The request handler 36 then can send the resulting cookie to the third-party Web server 28 with the request.

As shown in FIGs. 3 and 4, the response handler 40 directly follows the request handler 36 in the process chain. The response handler 40 starts as soon as the response is received from a data server 26. The responses from the data servers 26 as in FIG. 3, or from the third-party Web server 28 as in FIG. 4, can be in varied formats. Thus, the response handler 40 is tasked with parsing the response to obtain the requested data. The response handler 40 will collect all of the data that was made available on the socket created by the request handler 36, close the socket, and begin to massage the response data as needed. If needed, the response handler 40 also performs data validation. If an empty response or error message is returned, that state is sent to the data integrator 44 for transmission to the user.

Once the response data is received and massaged, the networked content delivery system 10 begins to build a resulting Markup Language Document by placing the data results into templates formatted for final display on the browser-enabled devices 12, 14 or 16. The networked content delivery system 10 employs two types of templates. The data integrator 44 uses the first type of template to build datasets of data that will be inserted into the completed MLD. The second type of template is used to form MLDs by collecting and inserting individual data elements, or by inserting datasets of data.

The first type of template, called a Specific Application Function template (or "SAF templates"), is used only to build datasets of data. SAF templates are used to describe the data content and formatting of, for example, individual lines of a table or the items for a list box. The data integrator 44 is aware of the appropriate SAF template to be used to match the markup language of the final MLD. For example, a template to display a single stock quote

within a multiple stock watch list, which is a sub-template that may display multiple times on a single MLD, will have a version for each of the supported markup languages.

The second type of templates, Markup Language Document templates (or "MLD templates"), is used to create the final MLD, including the look and feel of the Web sites. Page navigation, layout, and other visual components are described here, and tags are used to hold the places for the maintenance and insertion of data. Among the tags included in almost every MLD template are tags for the maintenance of the user identification number and the session identification value. This inclusion, in a location that is not displayed to the user, allows the system to track users and requests across multiple pages, servers and sites.

These MLD templates have different names depending on the initial data request, and different extensions for different markup languages. In other words, there exist multiple versions of a particular MLD template with each version having the same name but a different filename extension that corresponds to a resulting markup language. The correct template to be used always is passed in from the server request manager 30 as one of the SAF parameters.

In the event that the response handler 40 returns an error, the data integrator 44 still is used to build a MLD for display to the user. Each SAF has an associated set of static error pages to be displayed in the event of an error of some kind. Some of these documents allow for data to be inserted to provide further specificity concerning the type of error.

After the data integrator 44 has completed its data substitution, it sends the completed MLD to the server response manager 48, which is the last stop before it goes back through the Internet Web server application 29. Prior to sending the page to the browser-enabled device 12, 14 or 16, the server response manager 48 sets the completed Markup Language Document's Multipurpose Internet Mail Extensions ("MIME") media type and subtype

identifiers. This is required because a Web browser will attempt to read and interpret a MLD based upon its specified MIME type. Thus, the server response manager 48 must examine the filename extension of the outgoing template and designate the proper MIME type(s) within the MLD. The following table shows the MIME type for each template file extension:

Template File Extension	MIME type	Markup Language
.htm	text/html	HTML
.hdrn	text/x-hdml	HDML
.wml	text/vnd.wap.wml	WML
.xtm	text/html	XHTML

Once the MIME type information is set and the MLD is complete, the document is handed off to the Internet Web server application 29 "processor" that returns an HTTP response to the requesting browser for display to the user. The thread, now having completed its task, notifies the thread pool manager 32, which marks it as "ready."

In operation, a session is started between a browser-enabled device 12, 14 or 16 and a content delivery server 24 by the request for a generic URL of the server. As an example, a user would type the URL into the address bar on a Web browser or select the URL from a list of stored addresses. The networked content delivery system 10 receives the request and will read other information included in the header of the HTTP request to determine which specific file to send to the device for appropriate display. This additional information is contained in the basic Internet protocol HTTP-based request.

FIGS. 5, 6 and 7 are examples of the header information contained with URL page requests generated from three different Web browsers. An HTTP header contains various

fields, and associated values for each field. The HTTP header information contains the same fields regardless of the type of browser being used. The HTTP_ACCEPT 50 request-header field specifies certain MIME media types that are acceptable for the response given the browser's display capabilities. The variable HTTP_USER_AGENT 52 provides specific information regarding the "brand name" of the browser.

In FIGS. 5 and 7, the value "*/*" 54 contained in the variable HTTP_ACCEPT 50 indicates that the browser can read all MIME media types. However, because several browsers erroneously send "*/*", the system 10 uses other means to determine the appropriate response format. In FIG. 7, for example, the variable HTTP_ACCEPT 50 also includes the value "text/x-hdml; version=3.1", which expressly indicates that the browser can read files published in the HDML format. The absence of this value (and any other HDML related value) in FIG. 5 indicates that this browser cannot read HDML files – notwithstanding the presence of the "*/*" 54 in FIG. 5, which is known by the system 10 to be an erroneous indication of that browser's capabilities. The networked content delivery system 10 then uses the information gathered from the initial request-header to determine the appropriate markup language to be used when formatting response MLDs. For browsers that can read only one of the supported markup languages, the networked content delivery system 10 sends all further MLDs in the appropriate markup language.

If a browser can read more than one markup language, the networked content delivery system 10 either can prompt the user to select an appropriate markup language for the current online session with the content delivery server 24, or it can obtain further information from the browser by reading the HTTP_USER_AGENT 52 header field. For example, a number of browser-enabled devices use HTML with vastly different display characteristics. Thus, it is not possible to deliver the same HTML document even to all of the potential HTML-

compatible browser-enabled devices. When the system 10 has determined that HTML is the markup language for response to the browser, the HTTP_USER_AGENT 52 field is used to distinguish among known HTML-compatible browsers. Examples of values that will distinguish the requesting browser from others can be seen in FIG. 5, where the returned value includes "MSIE5.5" 58, and in FIG. 6, where the returned value includes "Elaine/1.0" 60. These browser-specific values in the HTTP_USER_AGENT 52 field can be used to determine the correct format of the final MLD. As an alternative, the networked content delivery system 10 provides a simple menu for selecting the type of device and browser when confronted with an HTML-compatible browser. While all HTML templates have the same extension (for HTML), the system 10 manages a set of HTML templates for delivery to different identifiable HTML-compatible browsers.

Decision Flow

An example of the decision flow that the networked content delivery system 10 makes on an initial request to the content delivery server 24 is depicted in the flowchart on FIG. 8. In this case, the session is initiated by the request from a Web browser in step 62. The HTTP request-header field HTTP_ACCEPT is checked first for compatibility with WML, in step 64. If it can display a WML document, it is then also checked for HDML compatibility in step 66. If it is not HDML compatible, then the first page in WML format is sent to the Web browser in step 68. If it is HDML compatible, then, in this example, the networked content delivery system 10 sends a simple menu in HDML format (step 70) to the Web browser allowing the user to choose between various WML and HDML options for display of the first page.

If the browser is not WML compatible, it is then screened for HDML compatibility in step 72. If it is HDML compatible, then the networked content delivery system 10 sends the

first page of the session in HDML format to the Web browser in step 74. If it is also not HDML compatible, then the networked content delivery system 10 in this example sends a simple menu in HTML format (step 76) to the Web browser allowing the user to choose between various HTML options. Other alternative algorithms can be used for determining which markup language and device specific templates should be used.

Once the appropriate browser language and device type, if required, has been determined, the networked content delivery system 10 maintains this determination throughout the concurrent session. The first page displayed by the system, essentially a device-specific default page, will include non-displayed tags that determine the appropriate MLD filenames. Each subsequent page will contain at least the filename of the next results template or templates in the command parameters. Thus, throughout a session, the user will consistently receive appropriately displayed pages.

In certain applications, it may be desirable for the first page that is displayed to be a login page in order to identify the user and for security purposes. FIG. 9 depicts a flowchart of such a function. A display of a representative menu of options sent to a user after login has been completed may be as depicted in FIG. 10.

Example

An example of a data retrieval application of the networked content delivery system 10 is a stock quote retrieval request 77, here entitled "Snap Quote," which is one of the commands in the menu displayed in FIG. 10. A flowchart for this command is depicted in FIGS. 11a – 11d. The user starts at step 78, which is the command menu displayed in FIG. 10. In step 79, the user selects the Snap Quote command. The Web page based command, transmitted when the Snap Quote command 77 was clicked, is:

"w2w.smartserv.com/cmd/SmartServ.dll?MfcISAPICommand=Page&u=000000&s=1

1111111&t=\c\tSnQu.htm"

There are three parameters sent with the HTTP request that will be passed to the "Page" Specific Application Function: U = 000000, where U is the current user's identification number, S = 11111111, where S is the current session identification value, T = \c\tSnQu.htm, which is the MLD template to be used for this command.

This command instructs IIS 29, which receives the command in step 80, to call the "Page" SAF which resides in the SmartServ.DLL file 82 – if necessary, causing the SmartServ.DLL file to be loaded into memory on the content delivery server 24, e.g. because this is the first call to any SAF resident in the SmartServ.DLL file. The server request manager 30 of the "Page" SAF is handed this request in step 84. The SAF is then executed, causing the server request manager 30 to request a "ready" thread from the thread pool manager 32, in step 86, to process this request.

The thread, now marked "busy" by the thread pool manager 32, first verifies, in step 88, the login status of the user. If the user is not logged in, or if the login is invalid, then, in step 90, an "invalid user" command is sent to the response handler 40, which passes the information off to the Data Integrator 44 in step 91. The data integrator 44 will then return, in step 92, a MLD with the appropriate error information for display to the user. In this case, the thread, now having completed, will return to the "ready" state, as in step 94.

If the user is successfully logged in, the "Enter Symbol" MLD is sent to and displayed on the user's browser, in step 96, and the "Page" SAF thread completes and is returned to the "ready" state in step 97. An example of this completed MLD is displayed in FIG. 12. In step 98, the user enters a stock symbol, for example "MSFT" and requests the quote by

clicking the appropriate button. When the user directs the browser to get the quote, a Web page based command is transmitted, as follows:

"w2w.smartserv.com/cmd/RealTime.dll?MfcISAPICommand=Stock&st=MSFT&u=0

00000&s=llllllll&t=\c\tSnQuRe.htm&mut=\c\tsnQuMuRe.htm&net=\c\tCoHe.htm"

There are five parameters sent with the HTTP request that will be passed to the "Stock" SmartServ Command Function: U = 000000, where U is the current user's identification number, S = 11111111, where S is the current session identification value, T = \c\tSnQuRe.htm, which is the MLD template to be used if one stock was successfully requested; MUT = \c\tSnQuMuRe.htm, which is the MLD template to be used if more than one stock was successfully requested; and NET = \c\tCoHe.htm, which is the MLD template used to display any company news headlines that are available for the returned symbol(s). In some circumstances, the server request manager 30 may need to manipulate the passed parameters before proceeding. For example, if there is an unsupported character in any of the parameters that the SAF cannot handle, the server request manager 30 will substitute that character before proceeding.

This command instructs IIS 29, which receives the command in step 100, to call, in step 102, the "Stock" SAF which resides in the RealTime.DLL – if necessary, causing the RealTime.DLL file to be loaded into memory on the content delivery server 24, e.g., because this is the first call to any SAF resident in the RealTime.DLL file. The server request manager 30 of the "Stock" SAF is handed this request in step 104. The SAF is then executed, causing the server request manager 30 to request a "ready" thread from the thread pool manager 32, in step 106, to process this request.

The thread now "busy" executing the "Stock" SAF instructions, first verifies, in step 108, the log-in status of the user. If the user is not logged in or is invalid, then an "invalid

user" command is sent to the response handler 40, in step 110, which passes off the error to the data integrator 44 in step 111. The data integrator 44 then, in step 112, will return a MLD with the appropriate error information for display to the user. In this case, the thread, now having completed, will return to the "ready" state, as in step 114. If the user is successfully logged in, then the request handler 36 in the thread processes the user's request for a stock quote(s). First, in step 116, a socket connection is established to a server that will supply the requested data. Then, in step 118, the request handler 36 builds a properly formatted request that includes the symbol(s) in the user's request. Finally, in step 120, the entire request is sent to the data server 26.

The request handler 36 receives the information from the data server(s) 26, in step 122, and passes the resulting data to the response handler 40 in step 124. The response handler 40 then checks the resulting data for its return codes. If the response handler 40 receives an invalid symbol response, at step 126, or if the user is not authorized to receive this data, at step 128, the appropriate error code is sent to the data integrator 44 in steps 129 or 131. The data integrator 44 then transmits the appropriate MDL to the user, in steps 130 or 132, respectively. The thread, having completed its task, notifies the thread pool manager 32, which marks that thread as "ready" in step 134.

If the symbol(s) is/are valid and the user is authorized to receive the requested data, the data is passed on to the data integrator 44 in step 136, which then checks, in step 138, if more than one symbol was entered in step 98. If only one symbol was requested, then the data integrator 44 loads the MLD template file tSnQuRe.htm in step 140, or, if more than one symbol was requested, then the data integrator 44 loads the MLD template file tSnQuMuRe.htm in step 142. Then, in step 144, the data integrator 44 examines the returned data to determine if the data server 26 has indicated that there is current news associated with

any of the symbol(s). If so, the data integrator 44 will build a link or links, using SAF templates, to the SAF titled "News." These links will be inserted into the final MLD for display and possibly execution by the user to retrieve the associated news items.

After the data integrator 44 builds any necessary news link(s), the stock data is then used to replace the data tags in the MLD template, in step 148. Notwithstanding the number of data elements received from the data server 26 for each stock symbol, the data integrator 44 only inserts those data elements into the MLD for which the MLD template included data tags. Thus, the data integrator 44 and the MLD template control which data will be included in the final MLD.

In step 150, the completed MLD is then sent to and displayed on the user's browser. The thread, having completed its task, notifies the thread pool manager 32, which marks that thread as "ready" in step 152.

FIG. 13 shows an example of the final results template as it would be displayed on a browser-enabled device. FIG. 14 is the source code for the version of the final MLD template, tSnQuRe.htm, that is written in HTML.

The invention has been described with reference to specific exemplary embodiments thereof and various modifications and changes may be made thereto without departing from the broad spirit and scope of the invention. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense; the invention is limited only by the following claims.

CLAIMS

What is claimed is:

1. A method for facilitating communications between a browser enabled device and a content delivery server, the method comprising the steps of:

initiating a request for data by a user of a browser-enabled device, the request including indicia of device and browser type;

transmitting said request from said browser-enabled device to the content delivery server across a first communication network;

receiving said request by said content delivery server;

retrieving, by said content delivery server, said data requested by said user;

formatting said data by said content delivery server as a function of the transmitted indicia of device and browser type to obtain formatted data; and

transmitting the formatted data from said content delivery server to said browser-enabled device across the first communication network.

2. The method of claim 1, further comprising the step of displaying the formatted data on a display of the browser-enabled device.

3. The method of claim 1, wherein said first communications network is the Internet.

4. The method of step 1, wherein said formatting step comprises:

reading said indicia of device and browser type;

selecting a template corresponding to the device and browser type from a plurality of stored templates; and

inserting the requested data into the selected template.

5. The method of claim 4, further comprising the step of storing a group of pre-formatted templates for the display of data on each of a known set of browser-enabled devices.

6. The method of claim 5, wherein said request for data is selected from one of a group of predefined data requests.

7. The method of claim 6, wherein said templates are further formatted for each of the predefined data requests.

8. The method of claim 1, wherein the retrieving step comprises the steps of:

sending a request for said data from said content delivery server to a data server across a second communication network; and

receiving a response to said request for said data by said content delivery server from said data server across said second communication network.

9. The method of claim 8, wherein said second communication network is a local area network.

10. The method of claim 8, wherein said second communication network is the Internet.

11. The method of claim 8, further comprising the steps of:

receiving a state object from said data server by said content delivery server as part of said response to said request for data;

converting said state object to a format compatible with said browser-enabled device;

transmitting said converted state object from said content delivery server to said browser-enabled device as part of said formatted data.

12. The method of claim 11, further comprising the steps of:

receiving said converted state object from said browser-enabled device by said content delivery server as part of a subsequent request for data;

converting said converted state object to its original format;

transmitting said state object in its original format to said data server as part of said subsequent request for data.

13. A system for facilitating communications between a browser-enabled device and a content delivery server, the system comprising:

means for initiating a request for data by a user using a browser-enabled device, the request including indicia of device and browser type;

means for transmitting said request from said browser-enabled device to the content delivery server across a first communication network;

means for receiving said request by said content delivery server;

means for retrieving, by said content delivery server, said data requested by said user;

means for formatting said data by said content delivery server as a function of the transmitted indicia of device and browser type to obtain formatted data; and

means for transmitting the formatted data from said content delivery server to said browser-enabled device across the first communication network.

14. The system of claim 13, further comprising means for displaying the formatted data on a display of the browser-enabled device.

15. The system of claim 13, wherein said first communication network is the Internet.

16. The system of claim 13, wherein said means for formatting comprises:

means for reading said indicia of device and browser type;

means for selecting a template corresponding to the device and browser type from a plurality of stored templates; and

means for inserting the requested data into the selected template.

17. The system of claim 16, further comprising means for storing a group of pre-formatted templates for the display of data on each of a known set of browser-enabled devices.

18. The system of claim 17, wherein said request for data is selected from one of a group of predefined data requests.

19. The system of claim 18, wherein said templates are further formatted for each of the predefined data requests.

20. The system of claim 13, wherein the means for retrieving further comprises:

means for sending a request for said data from said content delivery server to a data server across a second communication network; and

means for receiving said data by said content delivery server from said data server across said second communication network.

21. The system of claim 20, wherein said second communication network is a local area network.

22. The system of claim 20, wherein said second communication network is the Internet.

23. The system of claim 13, further comprising:

means for receiving a state object in an original format from said data server by said content delivery server as part of said response to said request for data;

means for converting said state object to a format compatible with said browser-enabled device;

means for transmitting said converted state object from said content delivery server to said browser-enabled device as part of said formatted data.

24. The method of claim 23, further comprising:

means for receiving said converted state object from said browser-enabled device by said content delivery server as part of a subsequent request for data;

means for converting said converted state object to its original format;

means for transmitting said state object in its original format to said data server as part of said subsequent request for data.

25. A system for supplying data requested by a user over a communication network, comprising:

a remote device having a type selected from a plurality of predetermined device types, the remote device having a user data interface, a device memory, a device communications port for transmitting signals over the communications network and a processor coupled to the device memory, the user data interface and the communications port, the processor executing a browser of predetermined type, the device operable by the user to generate a request for data, the request including indicia of the device type and the browser type;

a content delivery server having a first communications port for transmitting and receiving data over the communications network, a link to a database, a memory, and a content delivery server processor coupled to the first communications port, the link and the memory, the content delivery server processor comprising:

data recognition means for recognizing a request for data from the remote device;

data retrieval means for retrieving the requested data from the memory over the link;

formatting means for formatting the retrieved data as a function of the indicia included in the request for data; and

transmission means for transmitting the formatted, retrieved data from the communications port over the first communications network to the remote device.

26. A method of transferring state information between a content delivery server and a browser-enabled device, the method comprising the steps of:

initiating a request for data by a user of said browser-enabled device;

receiving said request for data by said content delivery server;

transmitting said request for data to a data server;

receiving a response to said request for data from said data server by said content delivery server;

receiving a state object from said data server by said content delivery server;

converting said state object to a format compatible with said browser-enabled device;

transmitting said response to said request for data from said content delivery server to said browser-enabled device; and

transmitting said converted state object from said content delivery server to said browser-enabled device.

27. The method of claim 26, further comprising the steps of:

receiving said converted state object from said browser-enabled device by said content delivery server as part of a subsequent request for data;

converting said converted state object to its original format;

transmitting said subsequent request for data to said data server; and

transmitting said state object in its original format to said data server.

28. A server for facilitating communication between any of a plurality of predetermined recognizable browser-enabled devices including wireless devices and at least one database server, comprising:

a receiver coupled to a communication network for receiving a request for data from one of the client devices, the request for data including indicia of device and browser type;

a retriever for retrieving from said at least one database server the requested data;

a memory for storing a plurality of templates each corresponding to respective combinations of browsers and client device types;

a formatter for retrieving one of the stored templates which corresponds to said transmitted indicia and which formats the requested data into the retrieved template; and

a transmitter for transmitting the formatted data over the communications network to the requesting device.

FIG. 1

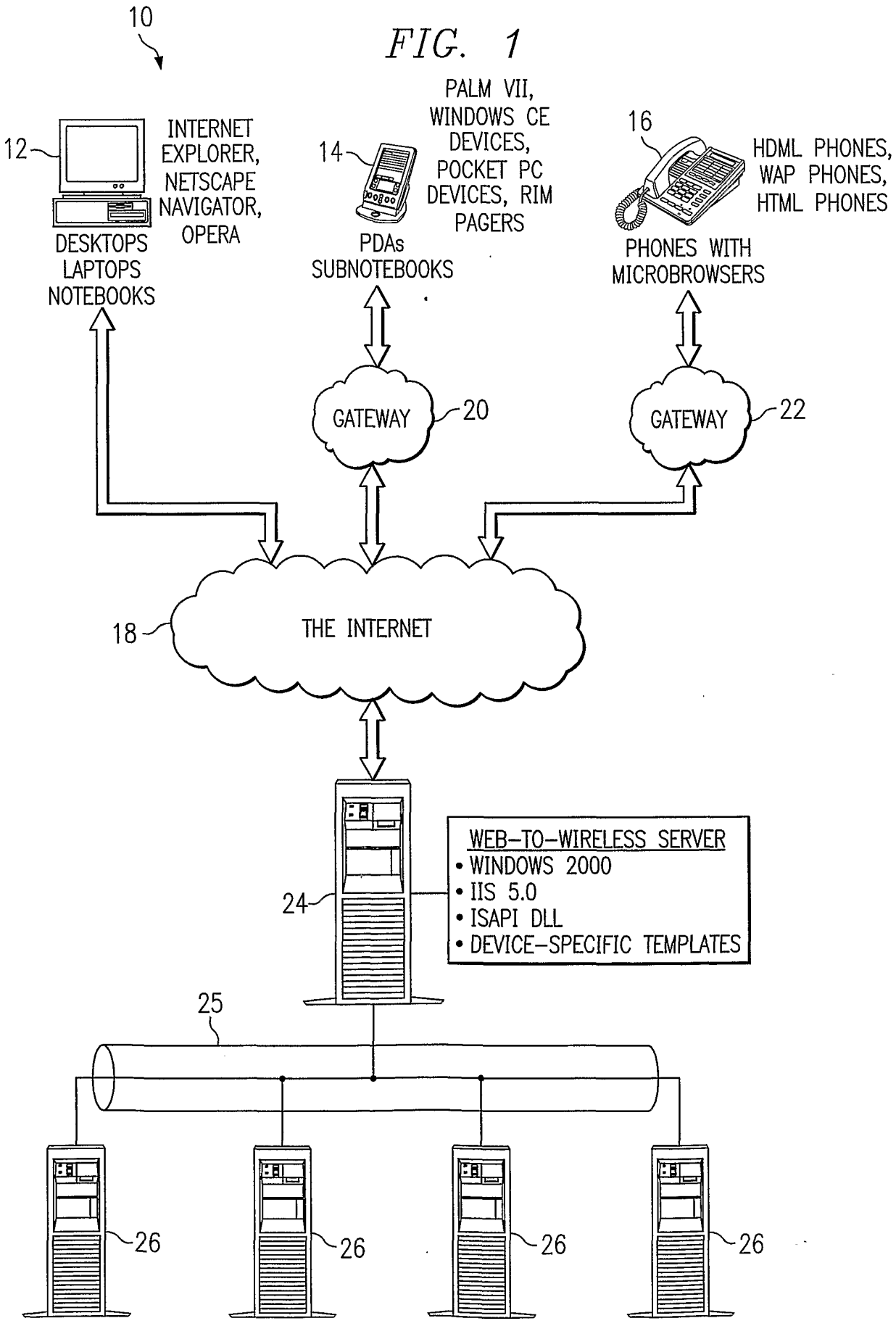
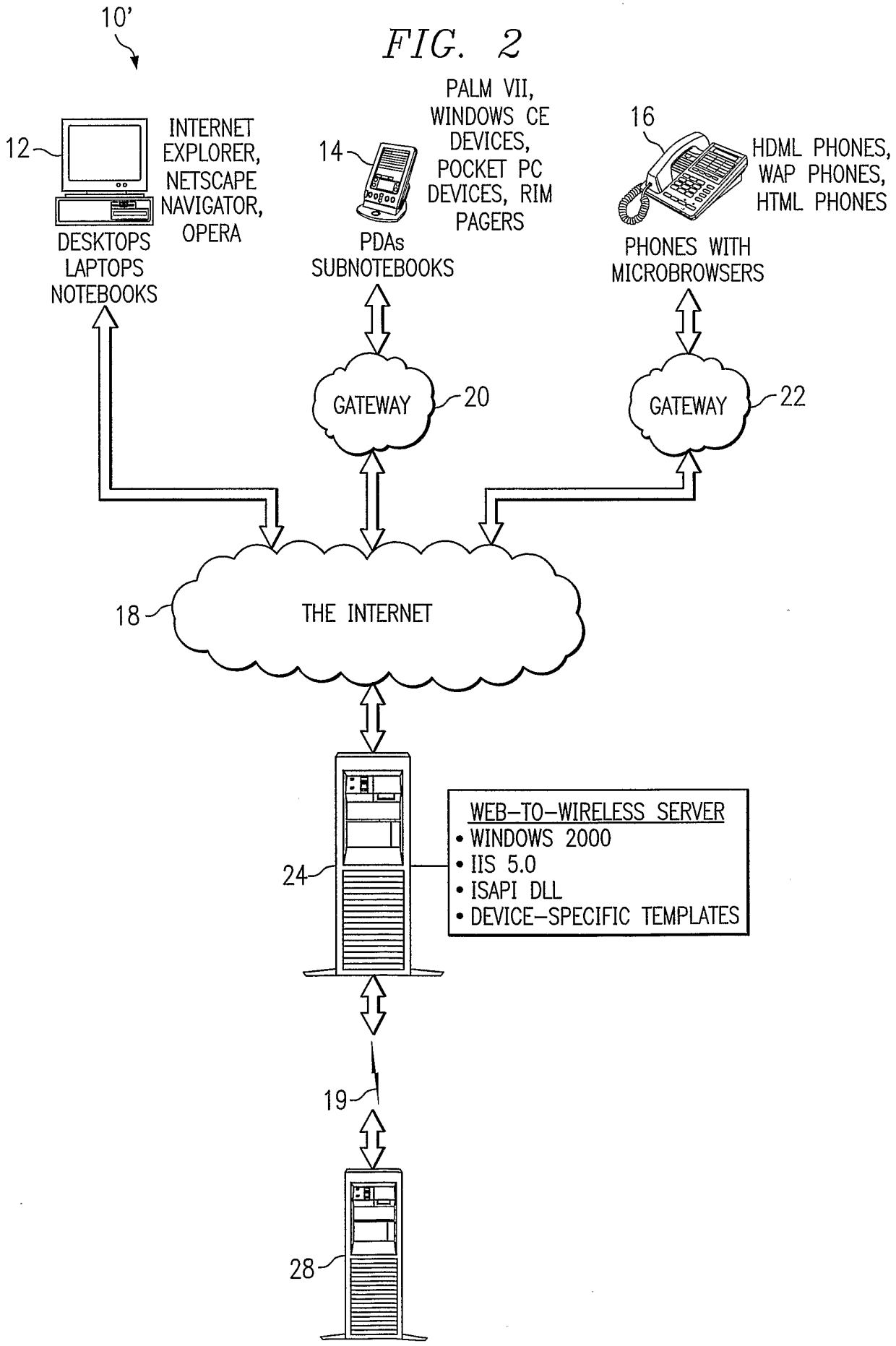

10
12
INTERNET EXPLORER, NETSCAPE NAVIGATOR, OPERA
DESKTOPS LAPTOPS NOTEBOOKS
14
PALM VII, WINDOWS CE DEVICES, POCKET PC DEVICES, RIM PAGERS
PDAs SUBNOTEBOOKS
16
HDML PHONES, WAP PHONES, HTML PHONES
PHONES WITH MICROBROWSERS
GATEWAY
20
GATEWAY
22
18
THE INTERNET
24
WEB-TO-WIRELESS SERVER
• WINDOWS 2000
• IIS 5.0
• ISAPI DLL
• DEVICE-SPECIFIC TEMPLATES
25
26
26
26
26

FIG. 2


10'
12
INTERNET EXPLORER, NETSCAPE NAVIGATOR, OPERA
DESKTOPS
LAPTOPS
NOTEBOOKS
14
PALM VII, WINDOWS CE DEVICES, POCKET PC DEVICES, RIM PAGERS
PDAs
SUBNOTEBOOKS
16
HDML PHONES, WAP PHONES, HTML PHONES
PHONES WITH MICROBROWSERS
GATEWAY
20
GATEWAY
22
18
THE INTERNET
24
WEB-TO-WIRELESS SERVER
• WINDOWS 2000
• IIS 5.0
• ISAPI DLL
• DEVICE-SPECIFIC TEMPLATES
19
28

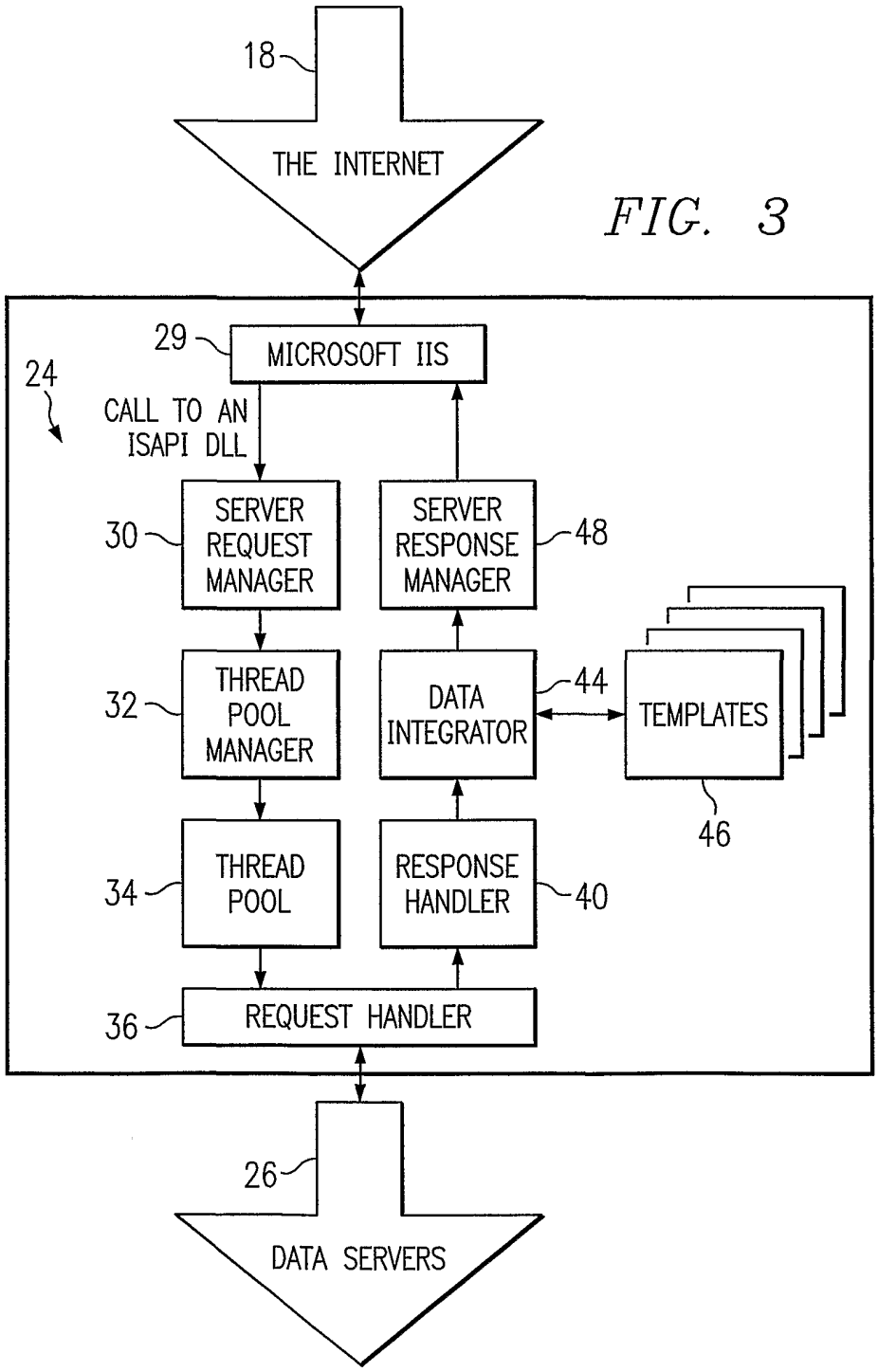

18
THE INTERNET
29
MICROSOFT IIS
24
CALL TO AN ISAPI DLL
30
SERVER REQUEST MANAGER
SERVER RESPONSE MANAGER
48
32
THREAD POOL MANAGER
DATA INTEGRATOR
44
TEMPLATES
46
34
THREAD POOL
RESPONSE HANDLER
40
36
REQUEST HANDLER
26
DATA SERVERS

FIG. 3

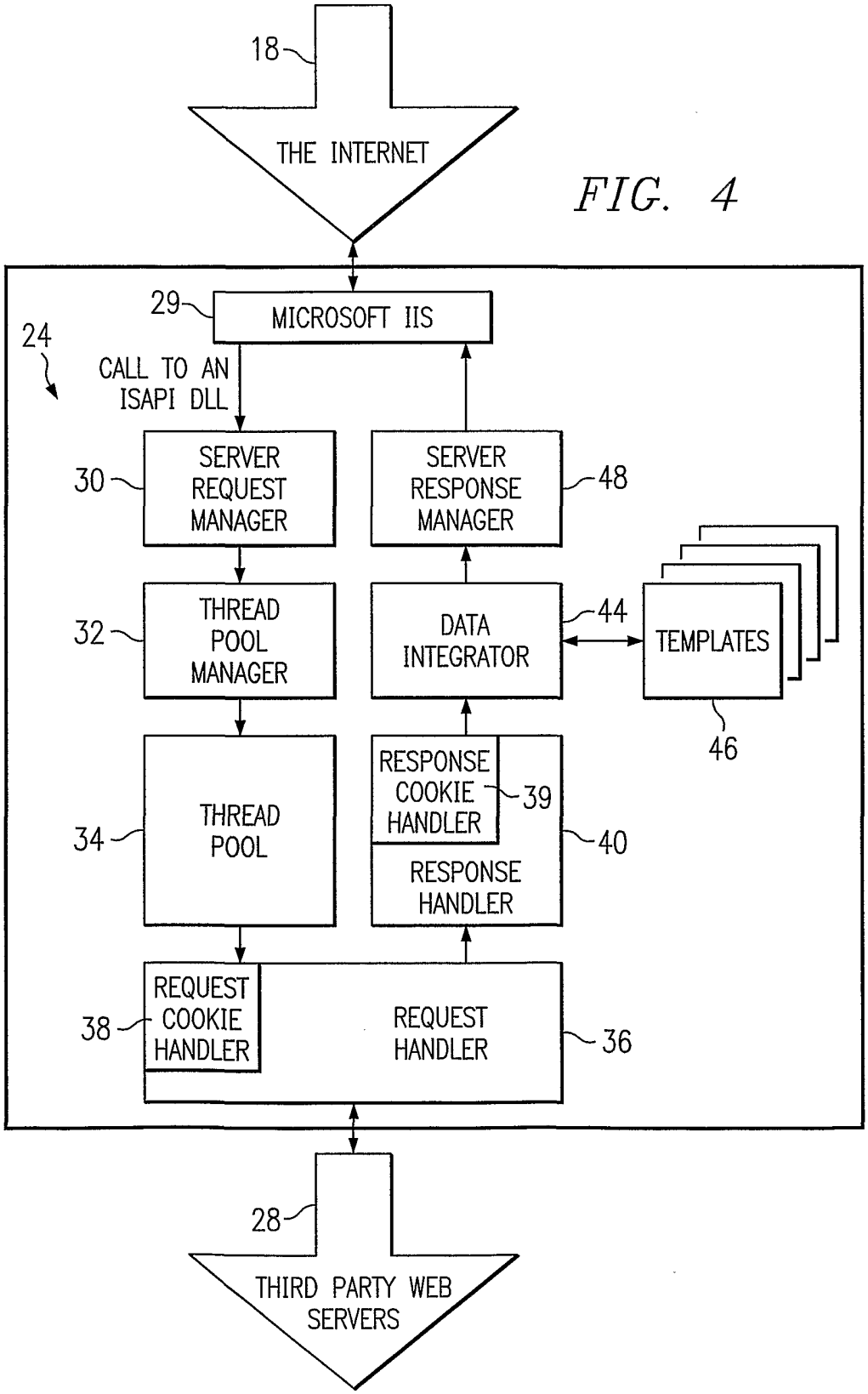

18
THE INTERNET
29
MICROSOFT IIS
24
CALL TO AN ISAPI DLL
30
SERVER REQUEST MANAGER
48
SERVER RESPONSE MANAGER
32
THREAD POOL MANAGER
44
DATA INTEGRATOR
TEMPLATES
46
34
THREAD POOL
RESPONSE COOKIE HANDLER
39
40
RESPONSE HANDLER
38
REQUEST COOKIE HANDLER
REQUEST HANDLER
36
28
THIRD PARTY WEB SERVERS

FIG. 4

Microsoft's Internet Explorer 5.5:

HTTP_ACCEPT:image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/msword, application/vnd.ms-powerpoint, application/vnd.ms-excel, */*
HTTP_ACCEPT_LANGUAGE:en-us.
HTTP_CONNECTION:Keep-Alive
HTTP_HOST:pikachu.smartserv.com
HTTP_USER_AGENT:Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)
HTTP_COOKIE:UserId=114137; Session=6615121
HTTP_ACCEPT_ENCODING:gzip, deflate

50, 54, 58, 52

FIG. 5

Palm VII:

HTTP_ACCEPT:text/html, image/jpeg, image/gif
HTTP_CONNECTION:Keep-Alive
HTTP_HOST:pikachu.smartserv.com
HTTP_USER_AGENT:Mozilla/2.0 (compatible; Elaine/1.0)
HTTP_VIA:1.1 WebCache (NetCache 4.0R4D11)
HTTP_X_FORWARDED_FOR:192.168.166.5

50, 52, 60

FIG. 6

FIG. 7

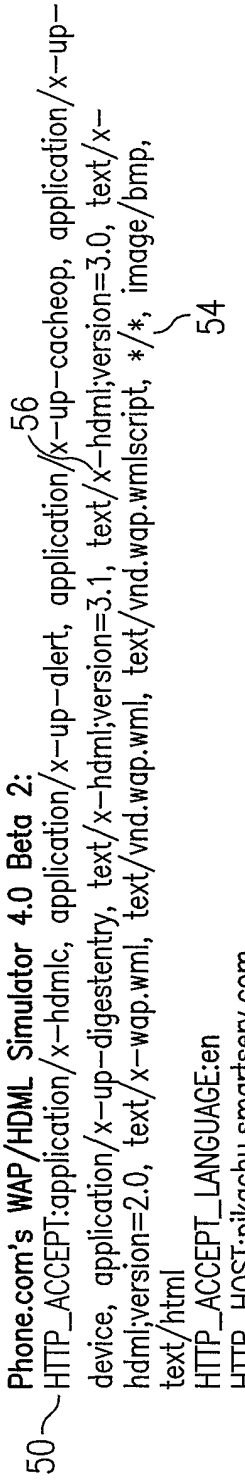
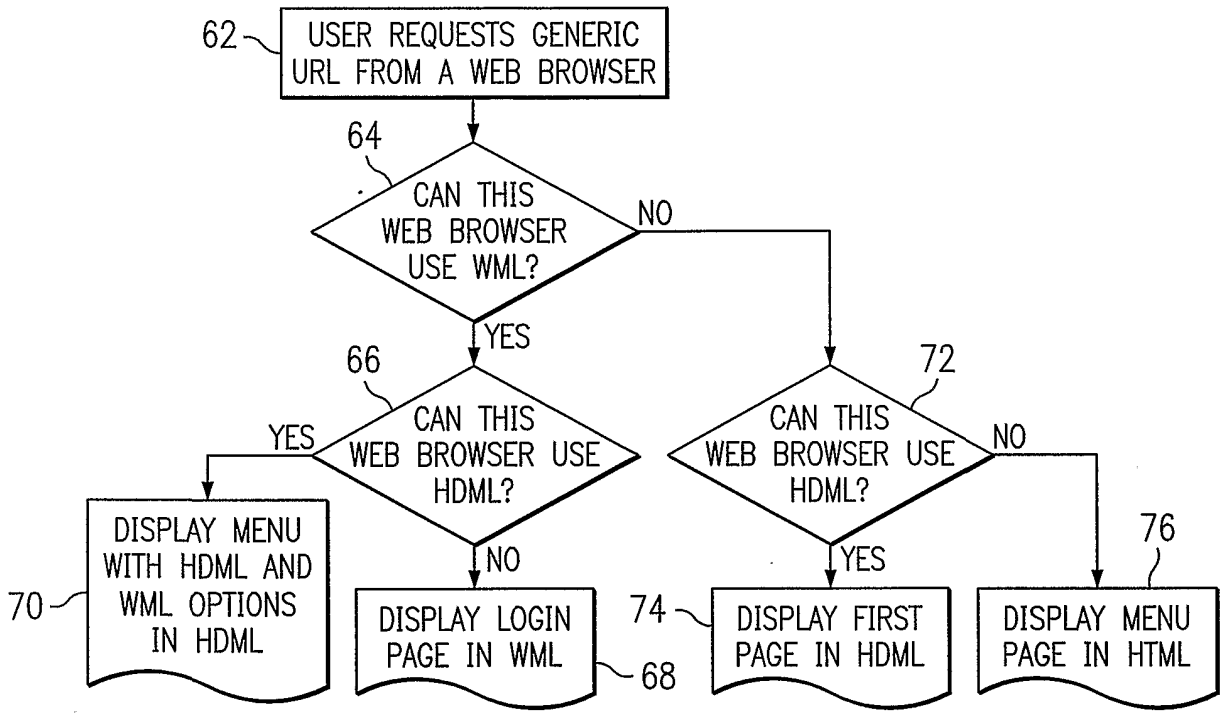

Phone.com's WAP/HDML Simulator 4.0 Beta 2:
50
HTTP_ACCEPT:application/x-hdmlc, application/x-up-alert, application/x-up-cacheop, application/x-up-device, application/x-up-digestentry, text/x-hdml;version=3.1, text/x-hdml;version=3.0, text/x-hdml;version=2.0, text/x-wap.wml, text/vnd.wap.wml, text/vnd.wap.wmlscript, */*, image/bmp, text/html
56
54
HTTP_ACCEPT_LANGUAGE:en
HTTP_HOST:pikachu.smartserv.com
52
HTTP_USER_AGENT:UPG1 UP/4.0.7
HTTP_CONTENT_TYPE:application/x-www-form-urlencoded
HTTP_ACCEPT_CHARSET:windows-1252
HTTP_X_UP_SUBNO:randy_pikachu.smartserv.com
HTTP_X_UP_UPFAX_ACCEPTS:none
HTTP_X_UP_UPLINK:none
HTTP_X_UP_DEVCAP_SMARTDIALING:1
HTTP_X_UP_DEVCAP_SCREEN_DEPTH:1
HTTP_X_UP_DEVCAP_HAS_COLOR:0
HTTP_X_UP_DEVCAP_IMMED_ALERT:1
HTTP_X_UP_DEVCAP_NUM_SOFTKEYS:2
HTTP_X_UP_DEVCAP_SCREEN_PIXELS:171,108
HTTP_X_UP_DEVCAP_EM_SIZE:8,18

FIG. 8


62
USER REQUESTS GENERIC URL FROM A WEB BROWSER
64
CAN THIS WEB BROWSER USE WML?
NO
YES
66
CAN THIS WEB BROWSER USE HDML?
YES
NO
70
DISPLAY MENU WITH HDML AND WML OPTIONS IN HDML
DISPLAY LOGIN PAGE IN WML
68
72
CAN THIS WEB BROWSER USE HDML?
NO
YES
74
DISPLAY FIRST PAGE IN HDML
76
DISPLAY MENU PAGE IN HTML

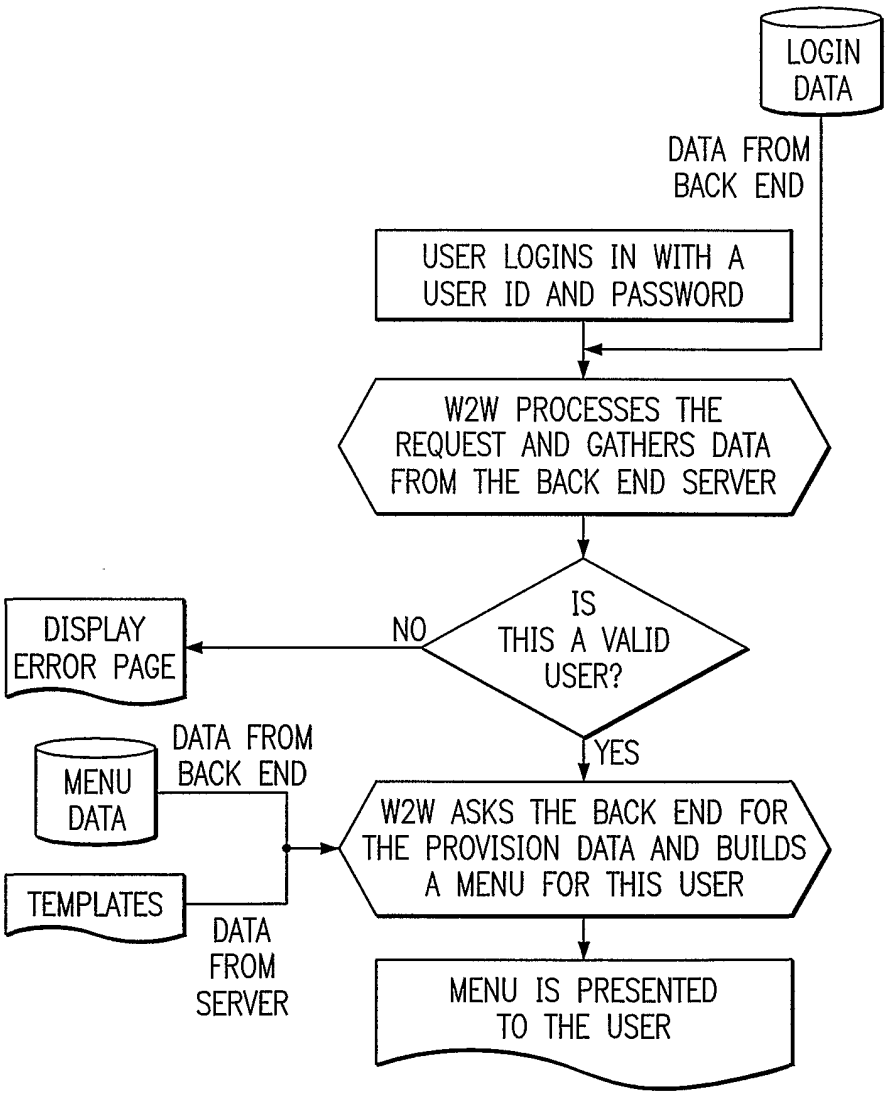

LOGIN DATA
DATA FROM BACK END
USER LOGINS IN WITH A USER ID AND PASSWORD
W2W PROCESSES THE REQUEST AND GATHERS DATA FROM THE BACK END SERVER
IS THIS A VALID USER?
NO
DISPLAY ERROR PAGE
YES
MENU DATA
DATA FROM BACK END
TEMPLATES
DATA FROM SERVER
W2W ASKS THE BACK END FOR THE PROVISION DATA AND BUILDS A MENU FOR THIS USER
MENU IS PRESENTED TO THE USER

FIG. 9

FIG. 10

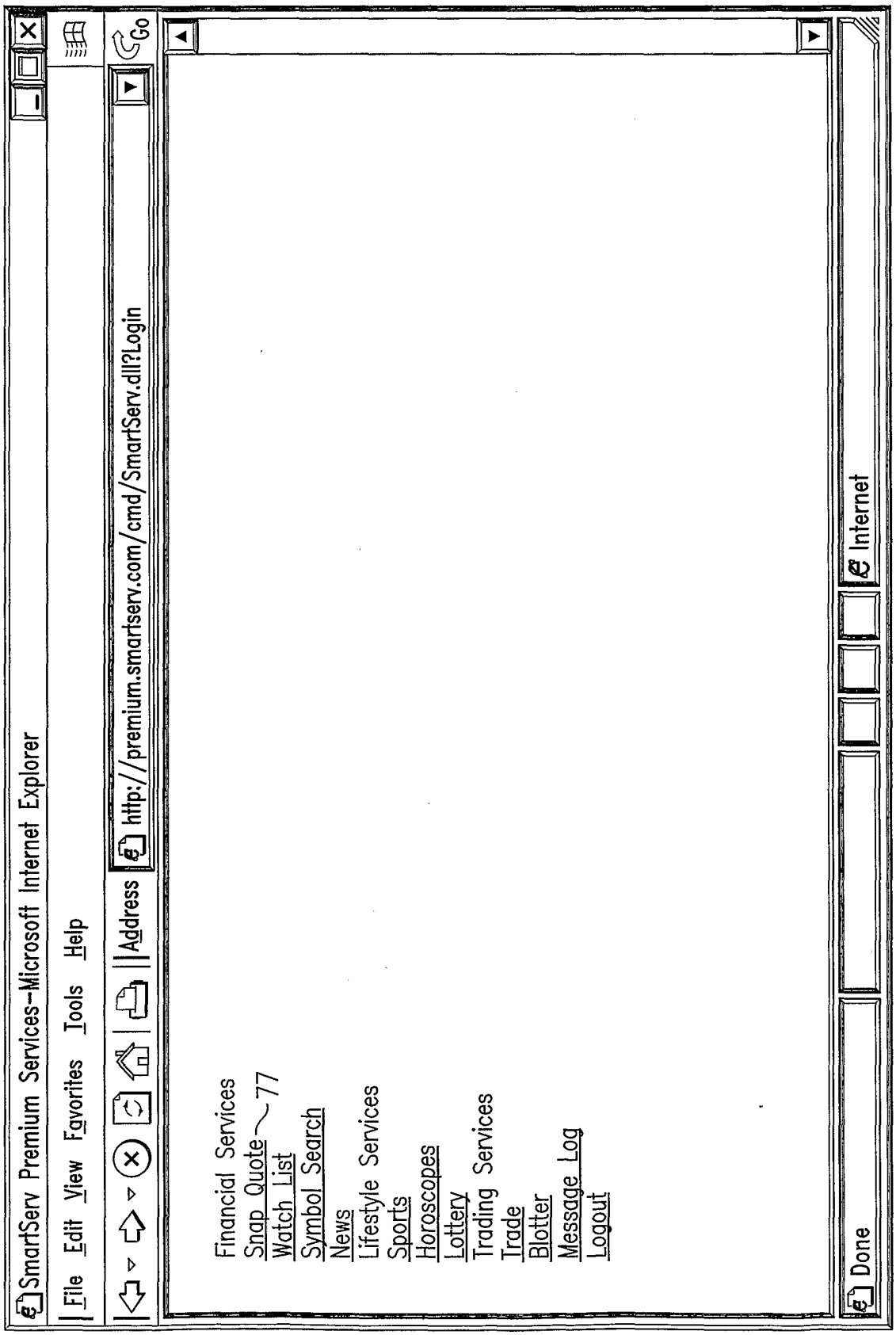
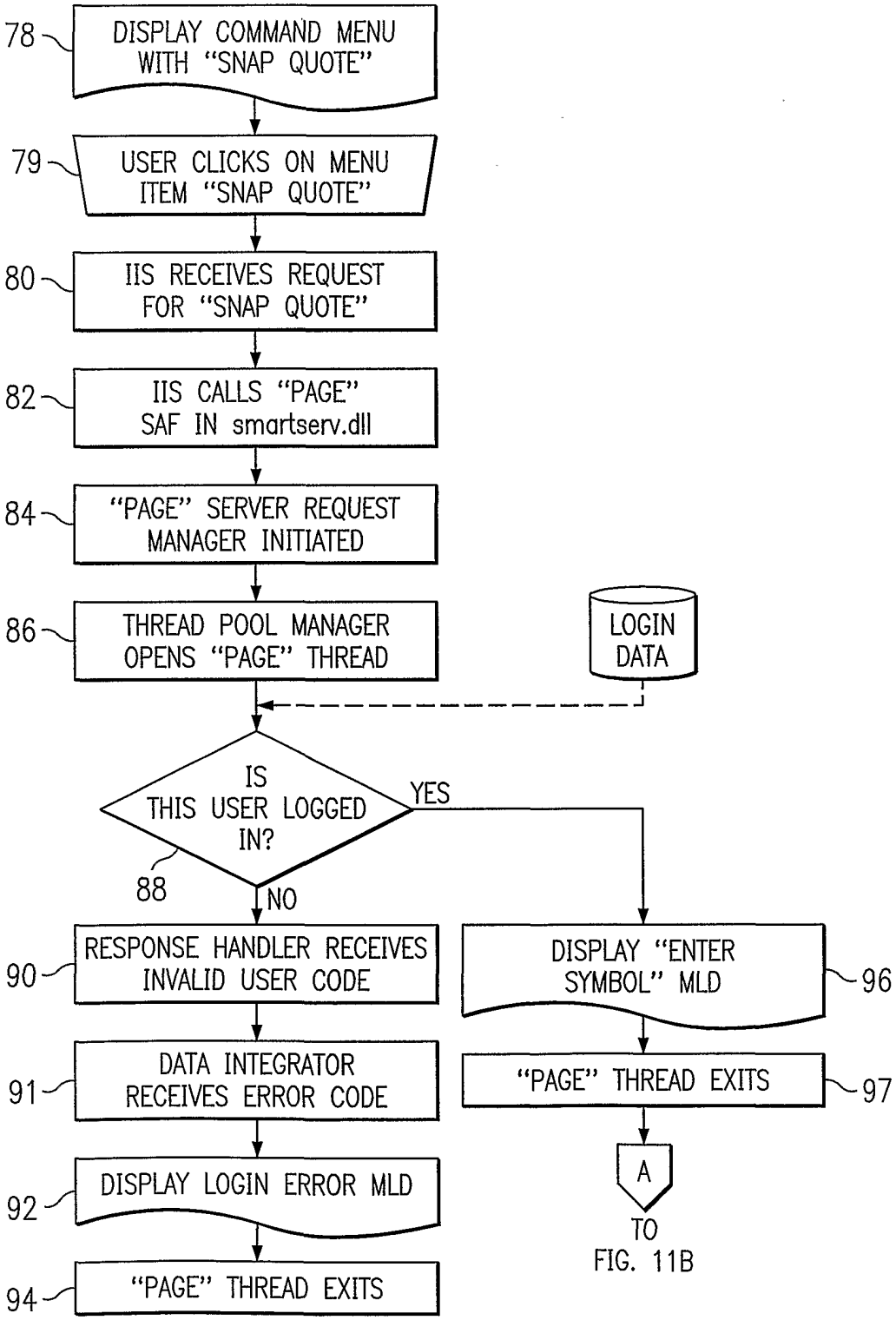

SmartServ Premium Services-Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://premium.smartserv.com/cmd/SmartServ.dll?Login
Go
Financial Services
Snap Quote
77
Watch List
Symbol Search
News
Lifestyle Services
Sports
Horoscopes
Lottery
Trading Services
Trade
Blotter
Message Log
Logout
Done
Internet

FIG. 11A


78
DISPLAY COMMAND MENU WITH "SNAP QUOTE"
79
USER CLICKS ON MENU ITEM "SNAP QUOTE"
80
IIS RECEIVES REQUEST FOR "SNAP QUOTE"
82
IIS CALLS "PAGE" SAF IN smartserv.dll
84
"PAGE" SERVER REQUEST MANAGER INITIATED
86
THREAD POOL MANAGER OPENS "PAGE" THREAD
LOGIN DATA
IS THIS USER LOGGED IN?
88
YES
NO
90
RESPONSE HANDLER RECEIVES INVALID USER CODE
91
DATA INTEGRATOR RECEIVES ERROR CODE
92
DISPLAY LOGIN ERROR MLD
94
"PAGE" THREAD EXITS
DISPLAY "ENTER SYMBOL" MLD
96
"PAGE" THREAD EXITS
97
A
TO FIG. 11B

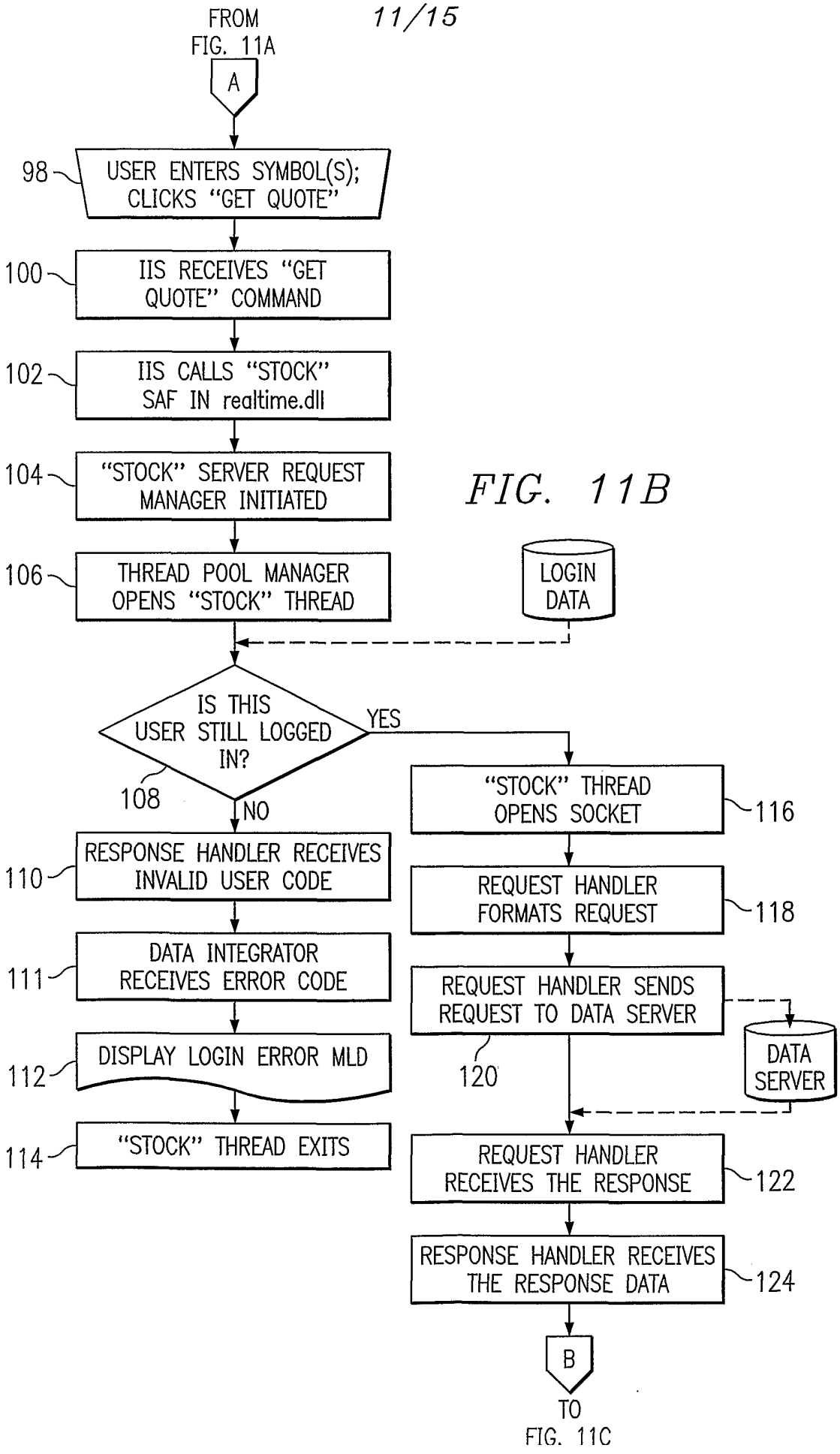

FROM FIG. 11A
A
98
USER ENTERS SYMBOL(S); CLICKS "GET QUOTE"
100
IIS RECEIVES "GET QUOTE" COMMAND
102
IIS CALLS "STOCK" SAF IN realtime.dll
104
"STOCK" SERVER REQUEST MANAGER INITIATED
106
THREAD POOL MANAGER OPENS "STOCK" THREAD
LOGIN DATA
IS THIS USER STILL LOGGED IN?
108
YES
NO
"STOCK" THREAD OPENS SOCKET
116
REQUEST HANDLER FORMATS REQUEST
118
REQUEST HANDLER SENDS REQUEST TO DATA SERVER
120
DATA SERVER
REQUEST HANDLER RECEIVES THE RESPONSE
122
RESPONSE HANDLER RECEIVES THE RESPONSE DATA
124
B
TO FIG. 11C
110
RESPONSE HANDLER RECEIVES INVALID USER CODE
111
DATA INTEGRATOR RECEIVES ERROR CODE
112
DISPLAY LOGIN ERROR MLD
114
"STOCK" THREAD EXITS

FIG. 11B

FIG. 11C

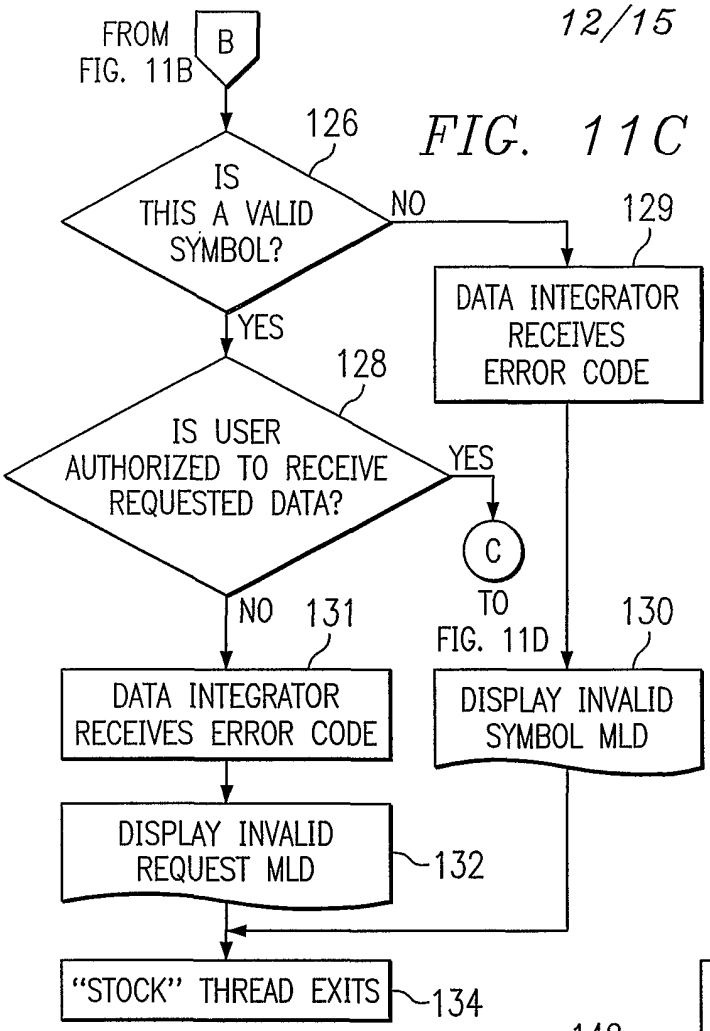


FIG. 11D

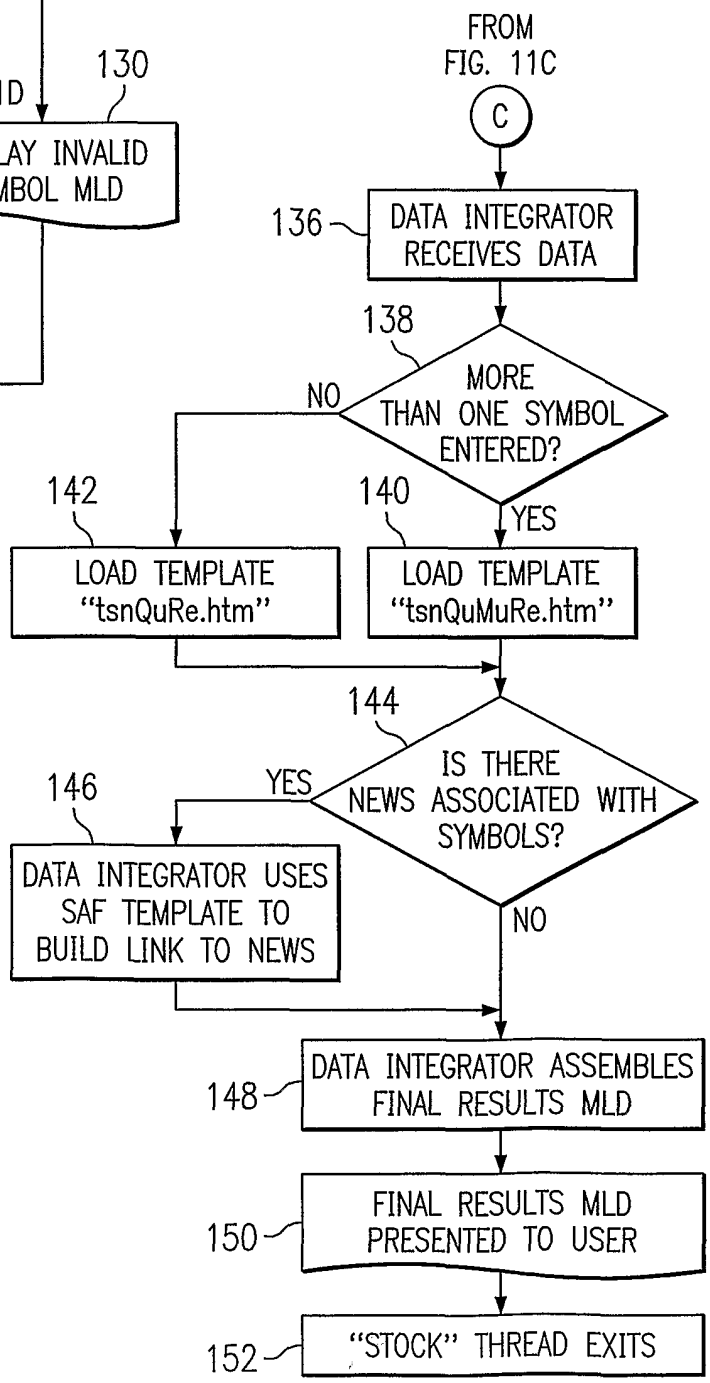


FIG. 12

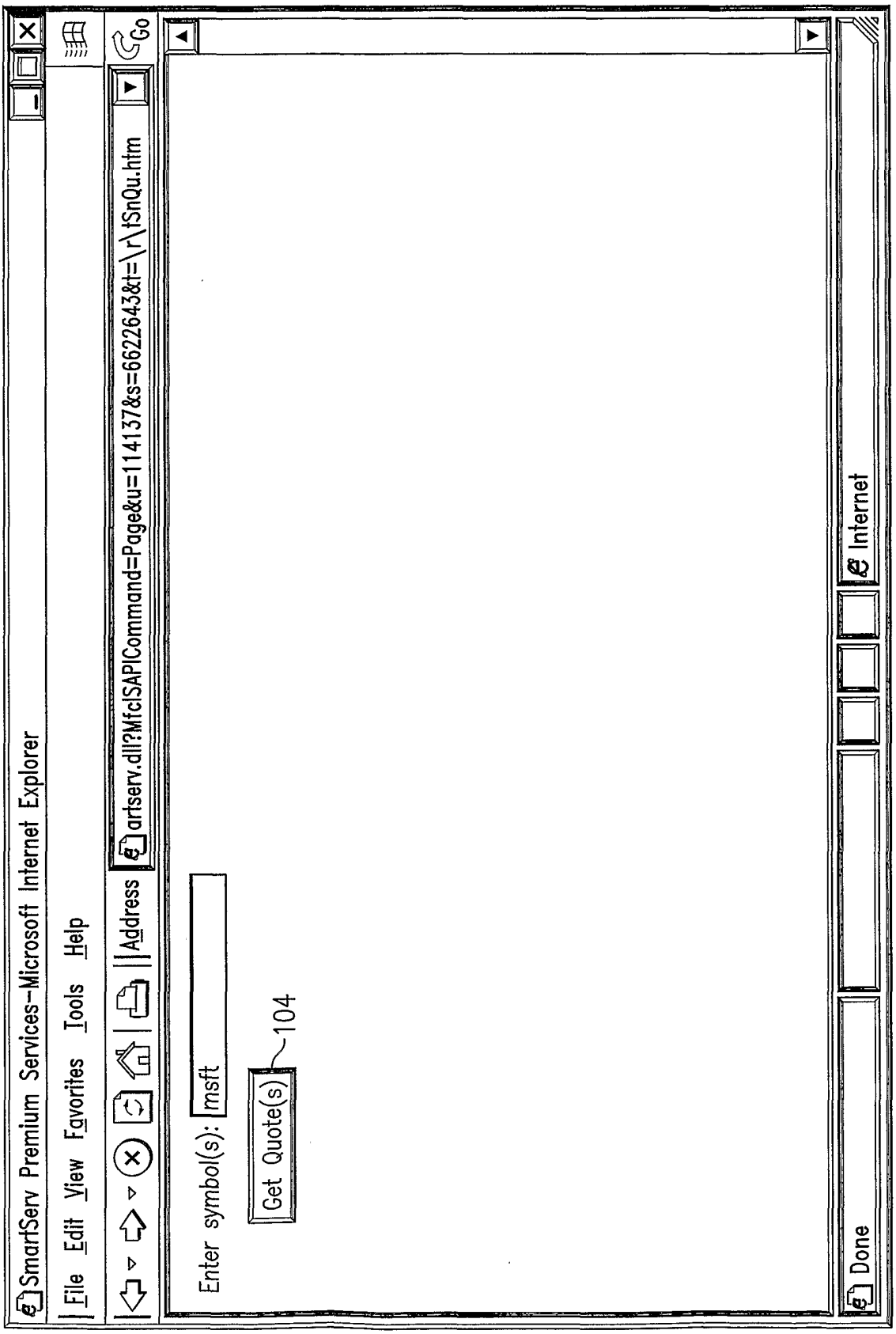

SmartServ Premium Services-Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address
artserv.dll?MfcISAPICommand=Page&u=114137&s=6622643&t=\r\tSnQu.htm
Go
Enter symbol(s): msft
Get Quote(s)
104
Done
Internet

FIG. 13

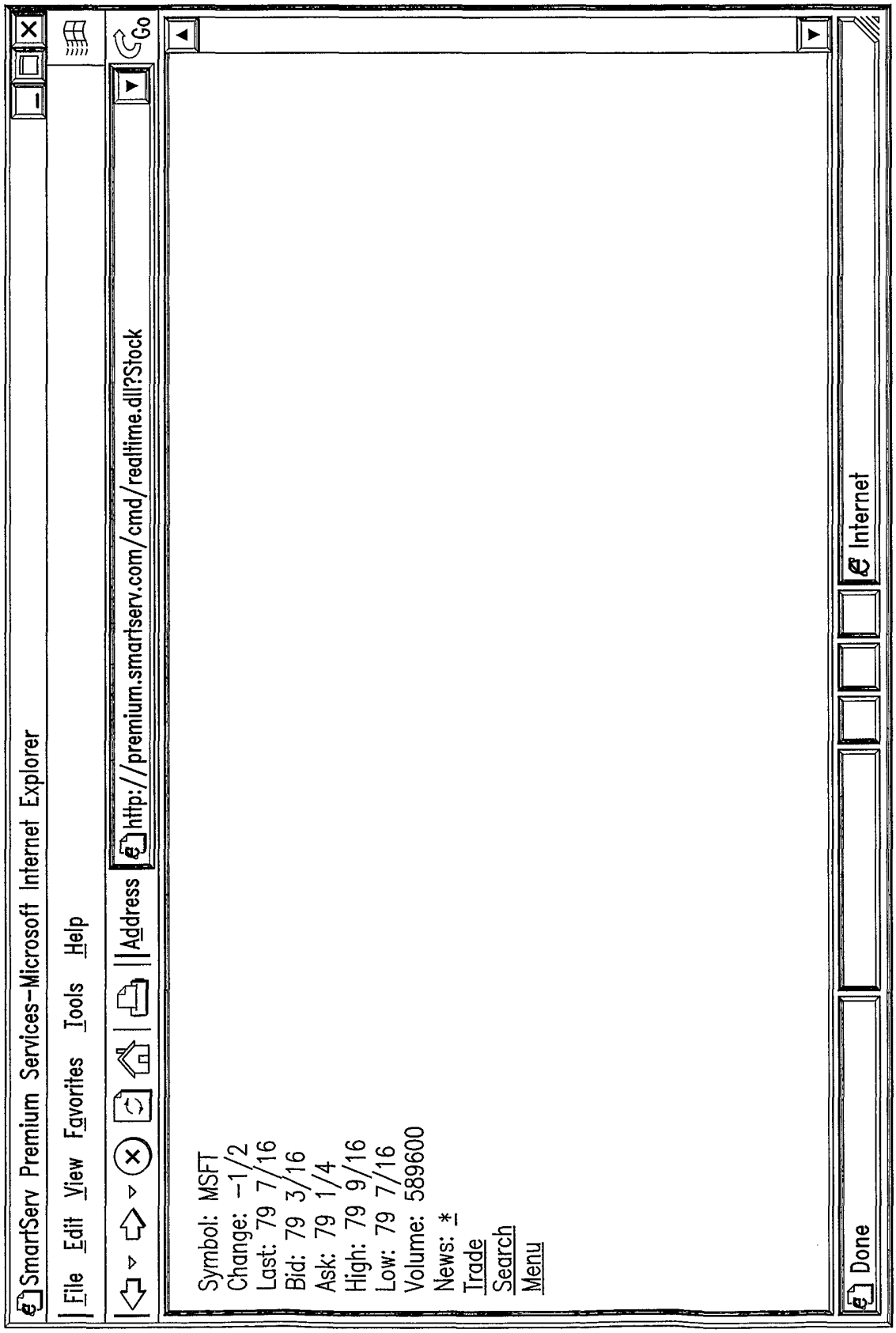

SmartServ Premium Services-Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address
http://premium.smartserv.com/cmd/realtime.dll?Stock
Go
Symbol: MSFT
Change: -1/2
Last: 79 7/16
Bid: 79 3/16
Ask: 79 1/4
High: 79 9/16
Low: 79 7/16
Volume: 589600
News: *
Trade
Search
Menu
Done
Internet

FIG. 14

tSnQuRe.htm

```
<HTML>
<HEAD>
<TITLE>SmartServ Online w2w2000</TITLE>
<META HTTP-EQUIV="Cache-Control" content="no-cache">
</HEAD>
<BODY>
Symbol: [$SYMBOL]<BR>
Change: [$CHANGE]<BR>
Last: [$LAST]<BR>
Bid: [$BID]<BR>
Ask: [$ASK]<BR>
High: [$HIGH]<BR>
Low: [$LOW]<BR>
Volume: [$VOLUME]<BR>
News: [$NEWS_IND]<BR>
<A
HREF="/cmd/peckweb.dll?MfcISAPICommand=Trade&u=[$USERID]&s=[$SESSIONID]&r=I&O=\r\tOr
.htm&C=\r\tOrCo.htm&sy=[$SYMBOL]">
[$TRADE_TEXT]</A><BR>
<A
HREF="/cmd/smartserv.dll?MfcISAPICommand=Page&u=[$USERID]&s=[$SESSIONID]&t=\r\tse.ht
m">
Search</A><BR>
<A
HREF="/cmd/SmartServ.dll?MfcISAPICommand=Menu&u=[$USERID]&s=[$SESSIONID]&t=\r\tMe.ht
m">Menu</A>
</BODY>
</HTML>
```