US 20020186249A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2002/0186249 A1

Lu et al. (43) Pub. Date: Dec. 12, 2002

(54) METHOD AND SYSTEM OF FACILITATING AUTOMATIC LOGIN TO A WEB SITE USING AN INTERNET BROWSER

(76) Inventors: **Qi Lu**, Saratoga, CA (US); **Ashish Baldua**, San Jose, CA (US)

Correspondence Address:
James J. DeCarlo
Stroock & Stroock & Lavan LLP
180 Maiden Lane
New York, NY 10038 (US)

(21) Appl. No.: **10/177,501**

(22) Filed: **Jun. 21, 2002**

Related U.S. Application Data

(63) Continuation-in-part of application No. 09/429,585, filed on Oct. 28, 1999. Continuation-in-part of application No. 10/015,816, filed on Nov. 1, 2001.

(60) Provisional application No. 60/331,565, filed on Nov. 16, 2001.

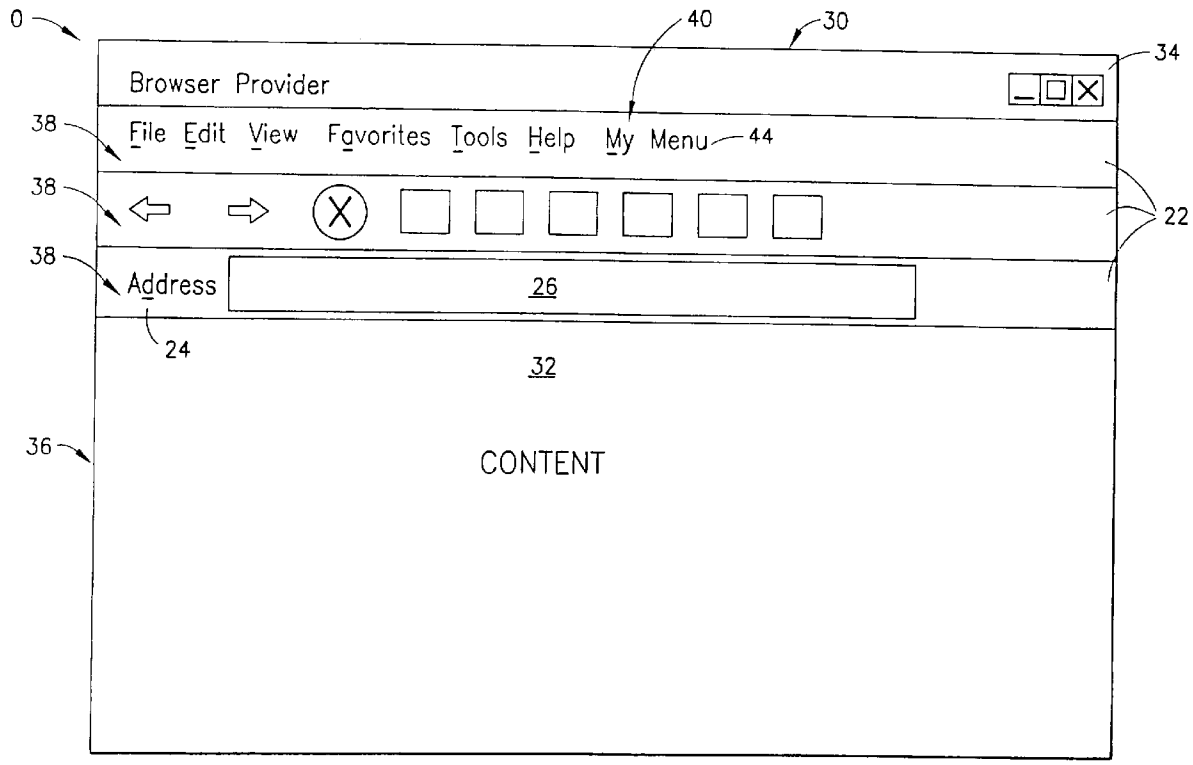
Publication Classification

(51) **Int. Cl.**[7] **G09G 5/00**

(52) **U.S. Cl.** **345/781**

(57) **ABSTRACT**

A method and system of adding functionality to an Internet browser interface. In one embodiment of the present invention, the added functionality may facilitate automatic login to a web site using an Internet browser. In another embodiment, the added functionality may enable the user to perform various tasks using the Internet browser such as, by way of non-limiting example, performing various tasks required to navigate one or more web pages, or to retrieve information desired by the user from one or more web pages or web sites.



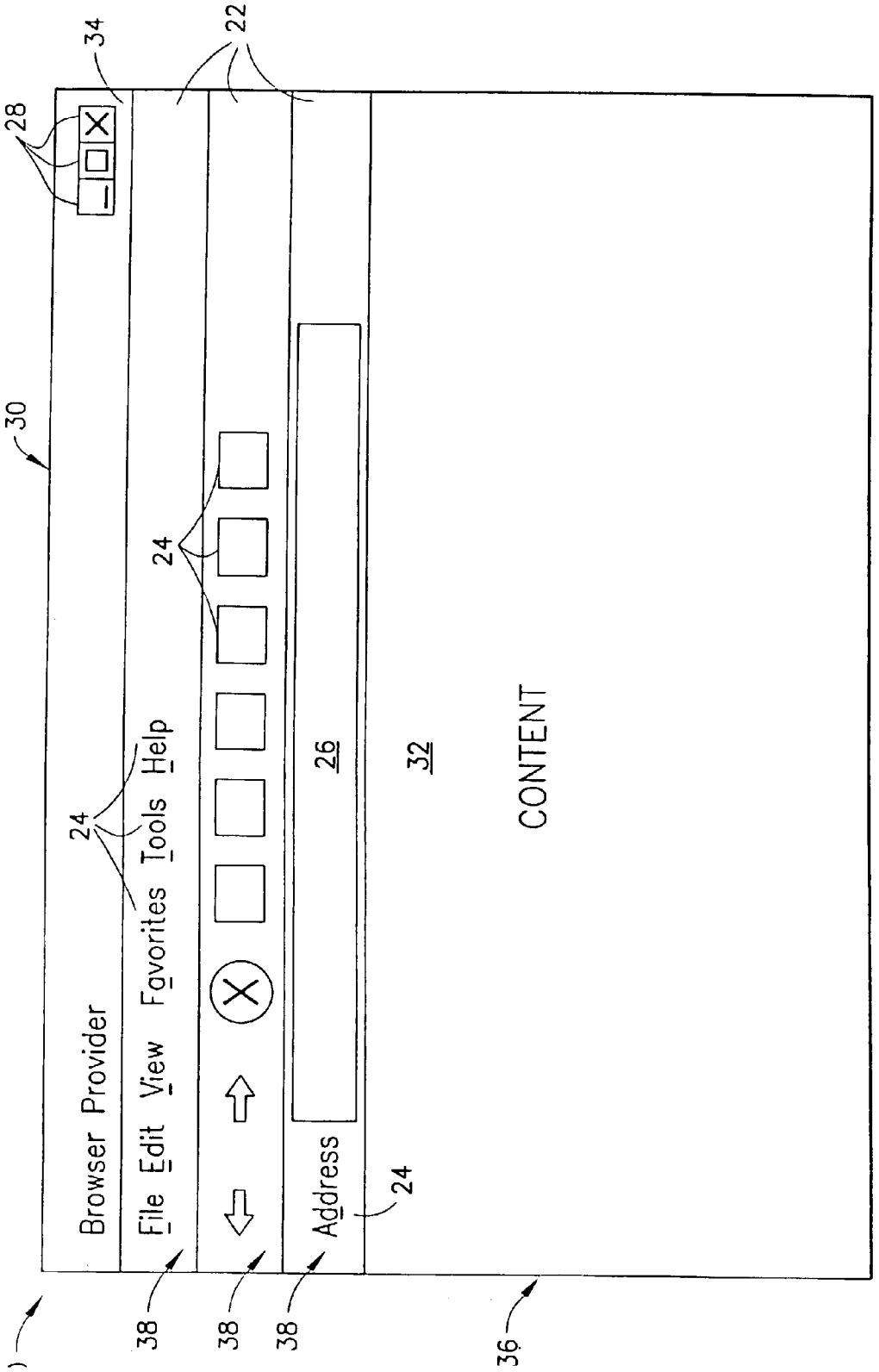


FIG. 1
PRIOR ART

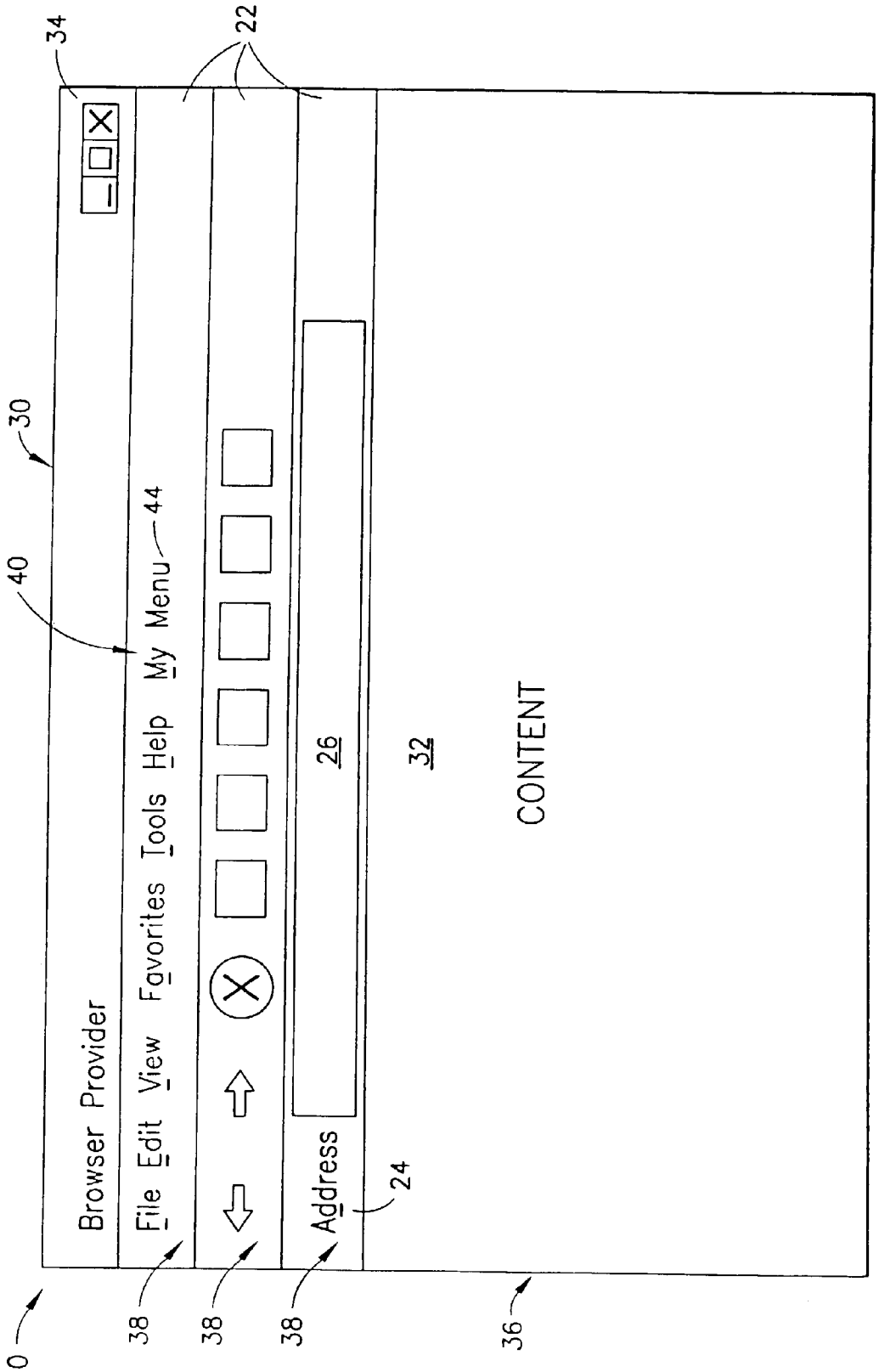

34
22
30
40
44
Browser Provider
File Edit View Favorites Tools Help My Menu
26
32
CONTENT
Address
24
0
38
38
38
36

FIG.2

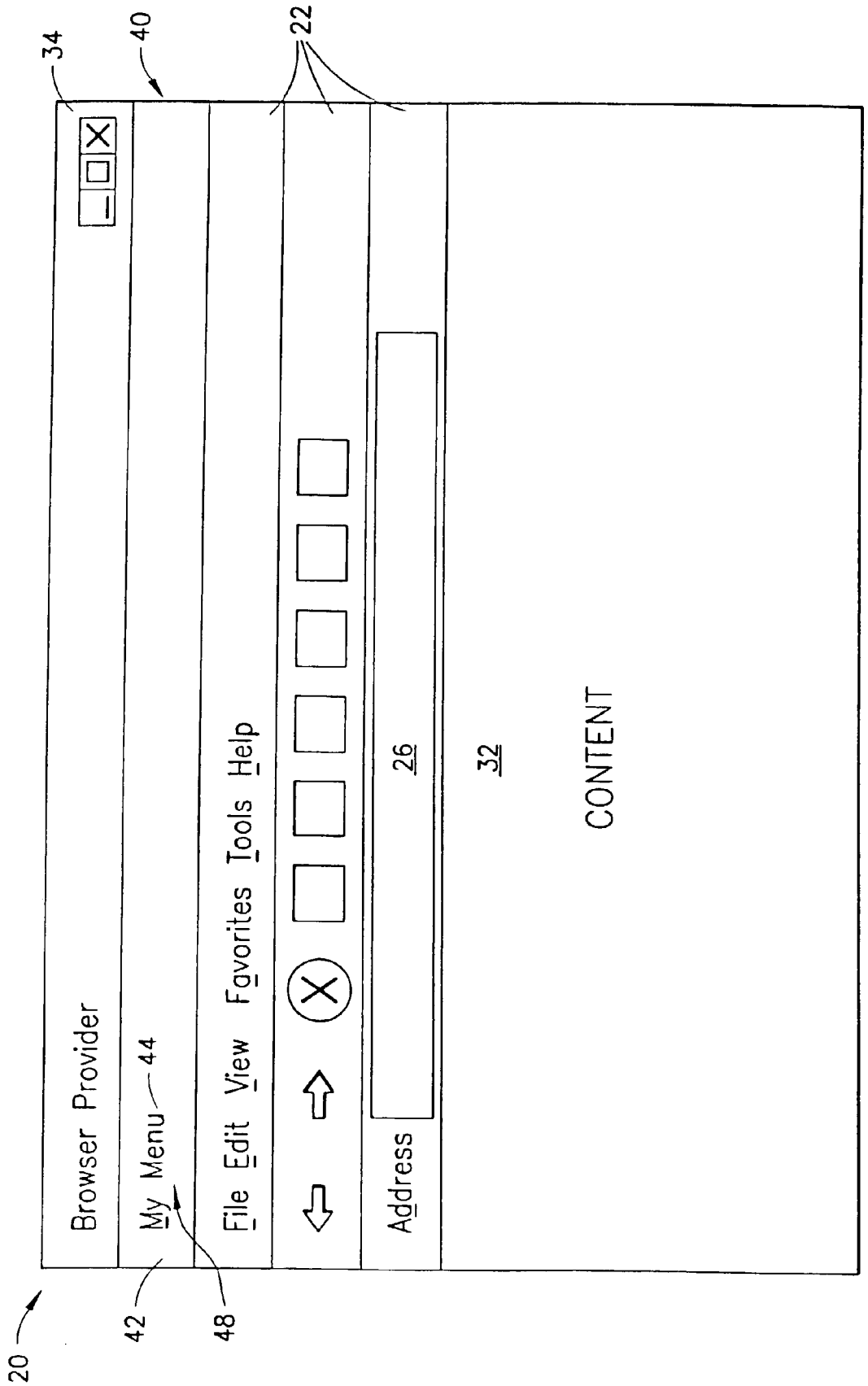

20
42
48
Browser Provider
My Menu
44
File Edit View Favorites Tools Help
Address
26
32
CONTENT
34
40
22

FIG.3

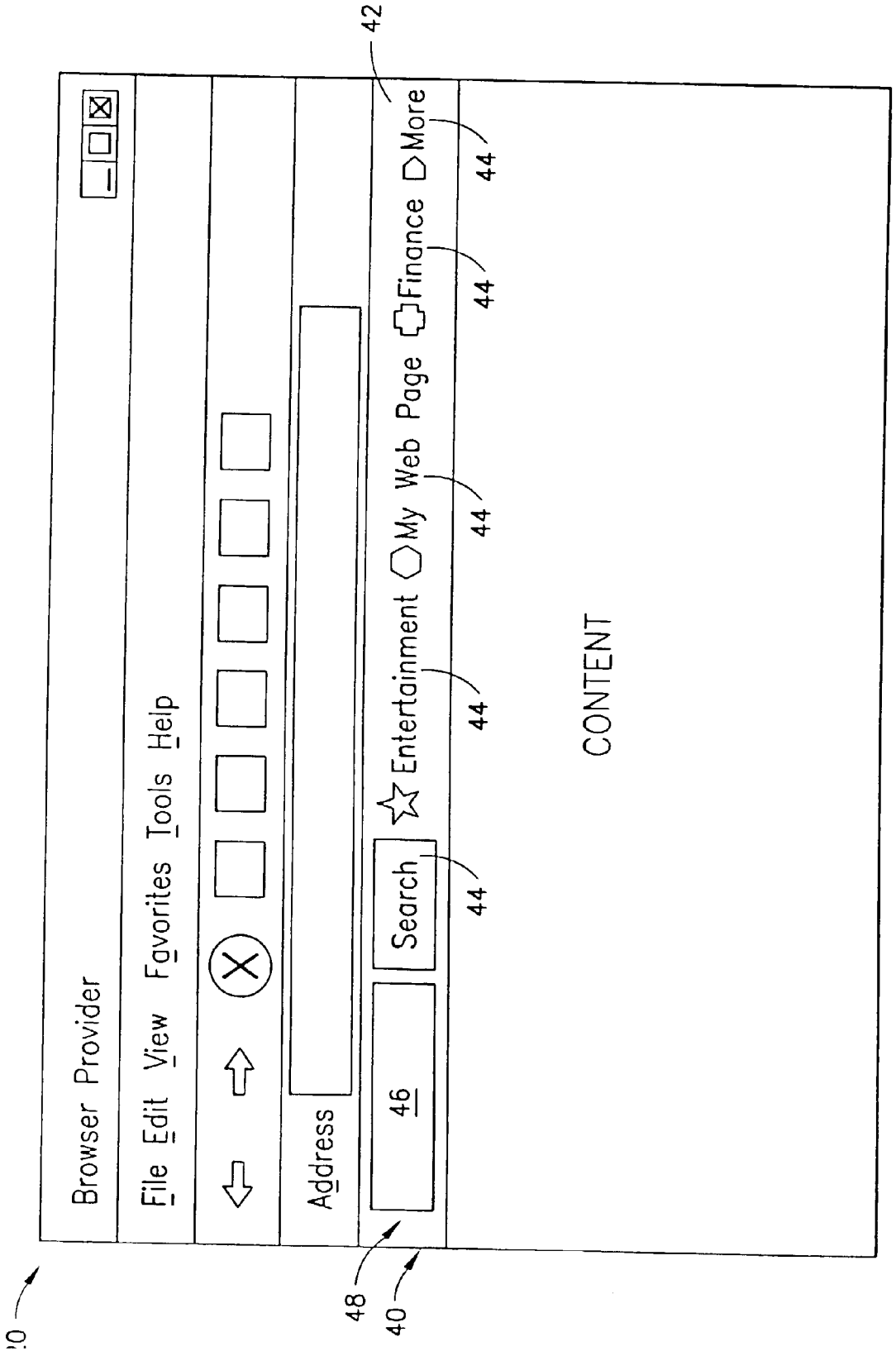

Browser Provider
File Edit View Favorites Tools Help
Address
46
Search
Entertainment
My Web Page
Finance
More
CONTENT
42
44
44
44
44
44
48
40
10

FIG.4

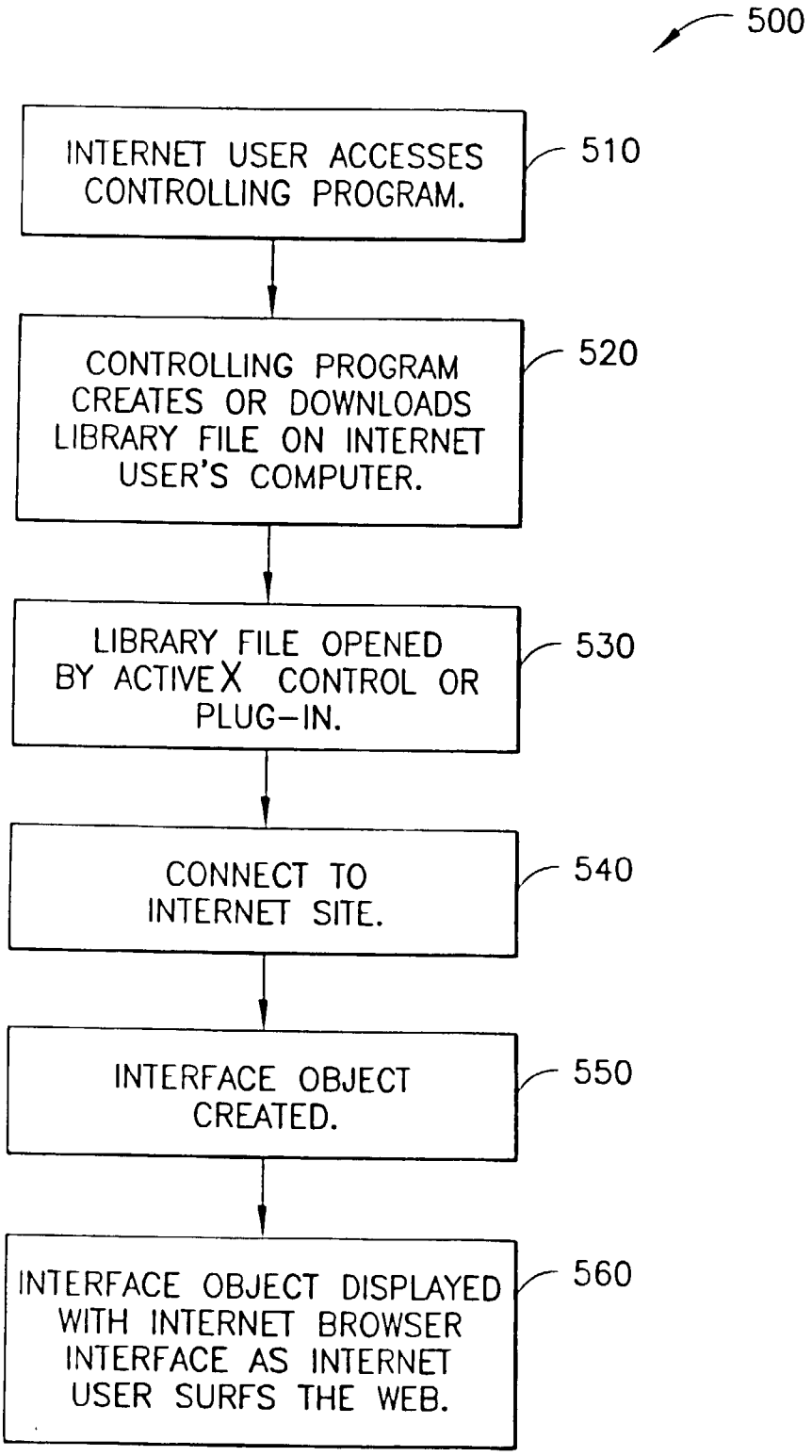

500
INTERNET USER ACCESSES CONTROLLING PROGRAM.
510
CONTROLLING PROGRAM CREATES OR DOWNLOADS LIBRARY FILE ON INTERNET USER'S COMPUTER.
520
LIBRARY FILE OPENED BY ACTIVEX CONTROL OR PLUG-IN.
530
CONNECT TO INTERNET SITE.
540
INTERFACE OBJECT CREATED.
550
INTERFACE OBJECT DISPLAYED WITH INTERNET BROWSER INTERFACE AS INTERNET USER SURFS THE WEB.
560

FIG.5

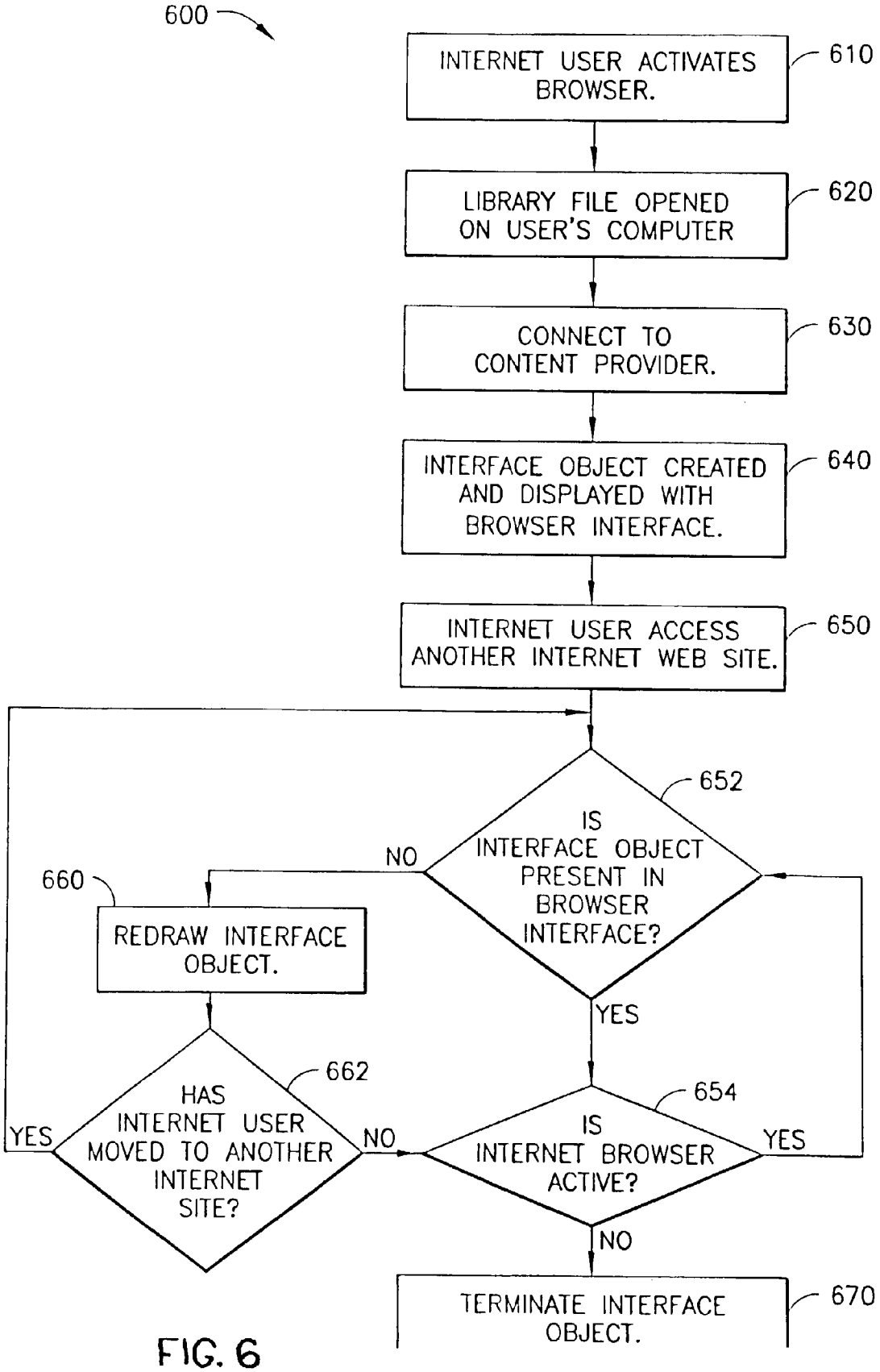

600
INTERNET USER ACTIVATES BROWSER.
610
LIBRARY FILE OPENED ON USER'S COMPUTER
620
CONNECT TO CONTENT PROVIDER.
630
INTERFACE OBJECT CREATED AND DISPLAYED WITH BROWSER INTERFACE.
640
INTERNET USER ACCESS ANOTHER INTERNET WEB SITE.
650
652
IS INTERFACE OBJECT PRESENT IN BROWSER INTERFACE?
NO
660
REDRAW INTERFACE OBJECT.
YES
662
HAS INTERNET USER MOVED TO ANOTHER INTERNET SITE?
YES
NO
654
IS INTERNET BROWSER ACTIVE?
YES
NO
TERMINATE INTERFACE OBJECT.
670

FIG. 6

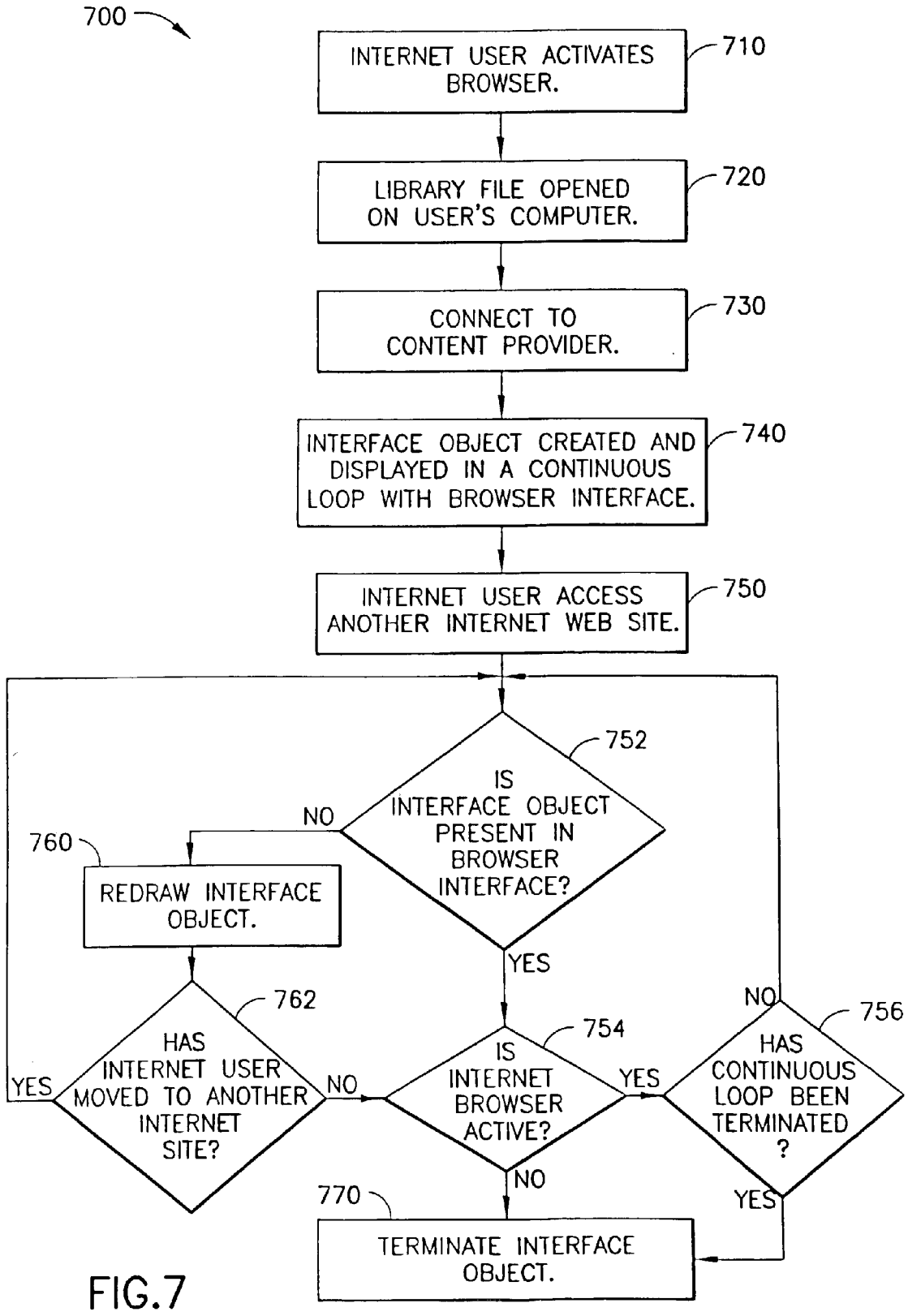

700
INTERNET USER ACTIVATES BROWSER.
710
LIBRARY FILE OPENED ON USER'S COMPUTER.
720
CONNECT TO CONTENT PROVIDER.
730
INTERFACE OBJECT CREATED AND DISPLAYED IN A CONTINUOUS LOOP WITH BROWSER INTERFACE.
740
INTERNET USER ACCESS ANOTHER INTERNET WEB SITE.
750
752
IS INTERFACE OBJECT PRESENT IN BROWSER INTERFACE?
NO
760
REDRAW INTERFACE OBJECT.
YES
762
HAS INTERNET USER MOVED TO ANOTHER INTERNET SITE?
YES
NO
754
IS INTERNET BROWSER ACTIVE?
YES
NO
756
HAS CONTINUOUS LOOP BEEN TERMINATED ?
NO
YES
770
TERMINATE INTERFACE OBJECT.

FIG.7

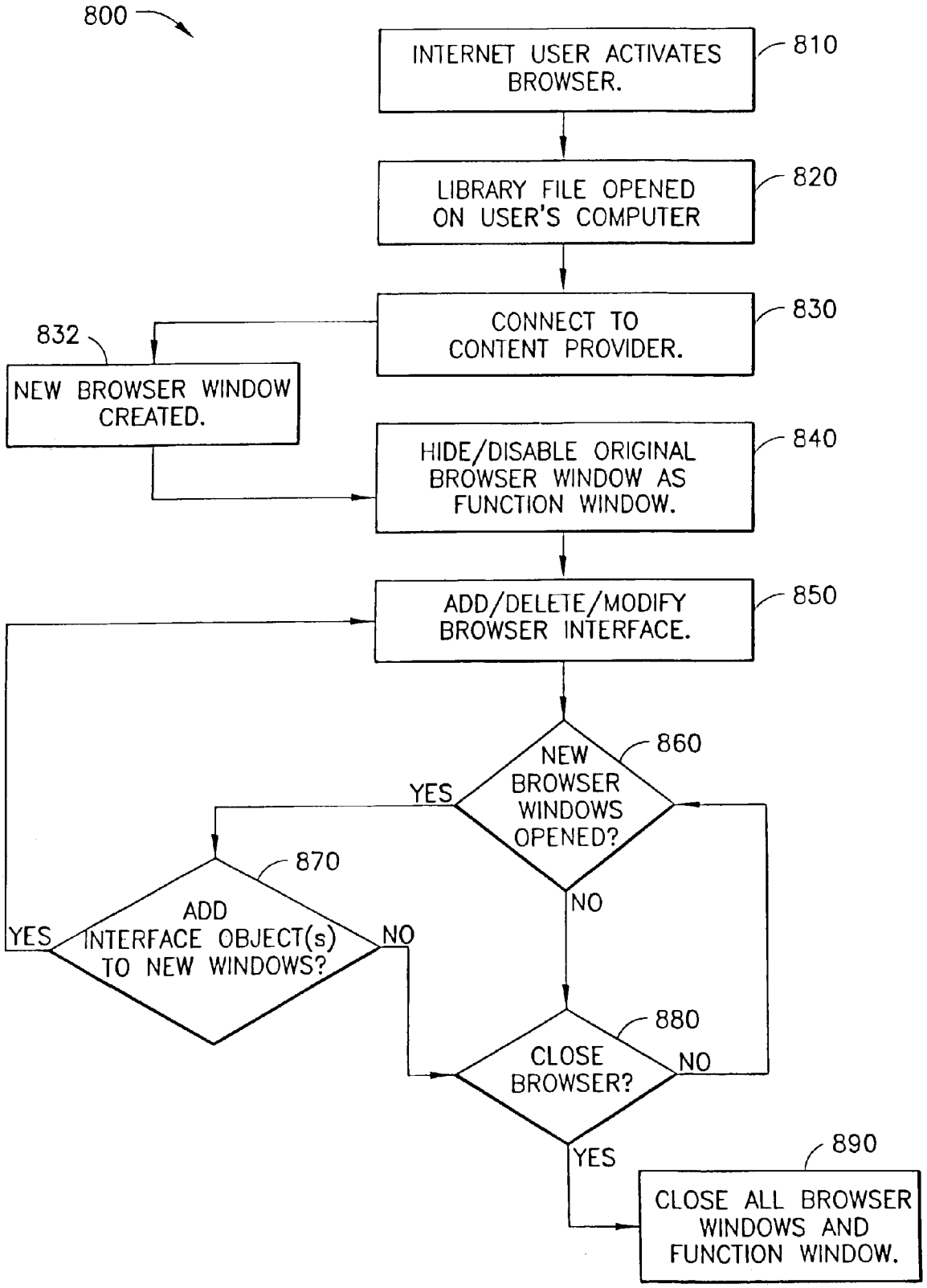

800
INTERNET USER ACTIVATES BROWSER.
810
LIBRARY FILE OPENED ON USER'S COMPUTER
820
CONNECT TO CONTENT PROVIDER.
830
832
NEW BROWSER WINDOW CREATED.
HIDE/DISABLE ORIGINAL BROWSER WINDOW AS FUNCTION WINDOW.
840
ADD/DELETE/MODIFY BROWSER INTERFACE.
850
NEW BROWSER WINDOWS OPENED?
860
YES
NO
870
ADD INTERFACE OBJECT(s) TO NEW WINDOWS?
YES
NO
880
CLOSE BROWSER?
NO
YES
890
CLOSE ALL BROWSER WINDOWS AND FUNCTION WINDOW.

FIG.8

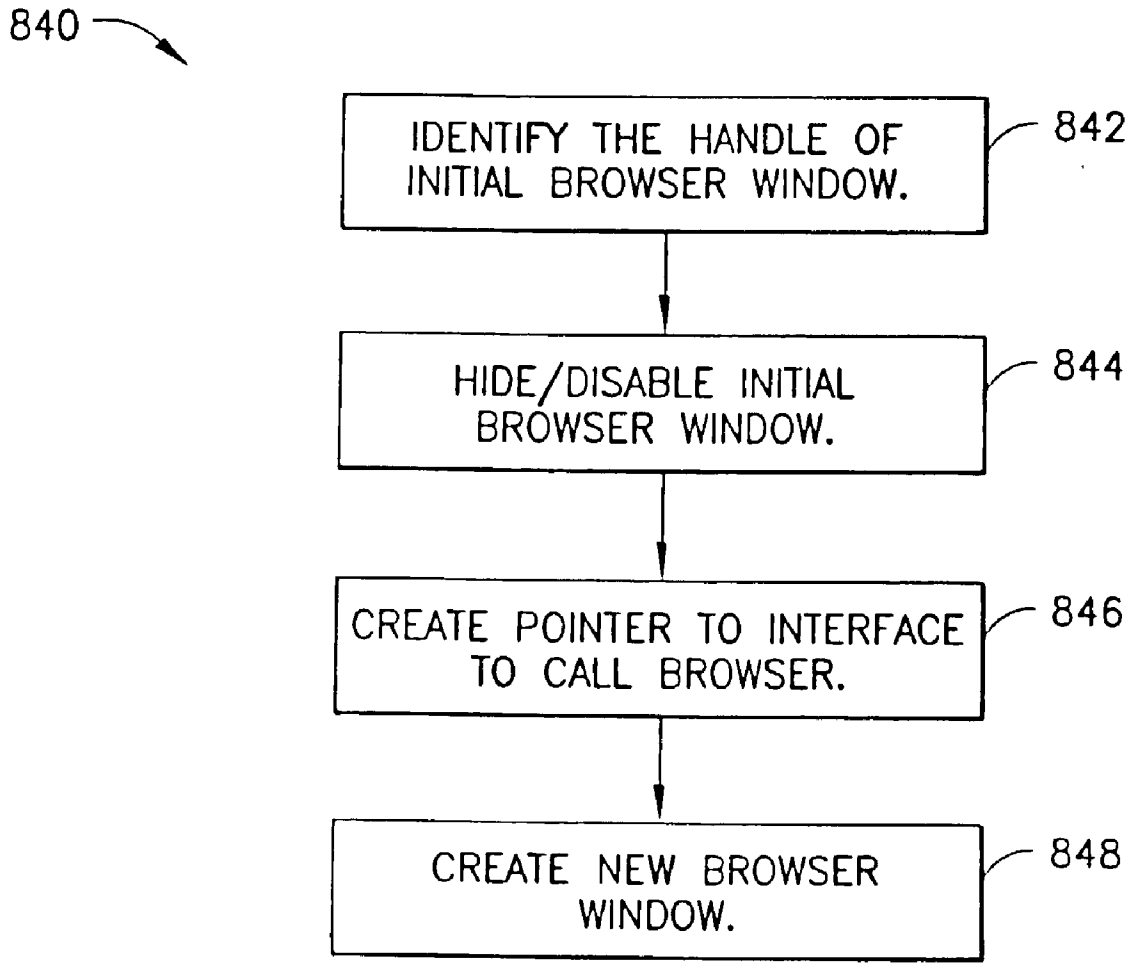

840
IDENTIFY THE HANDLE OF INITIAL BROWSER WINDOW.
842
HIDE/DISABLE INITIAL BROWSER WINDOW.
844
CREATE POINTER TO INTERFACE TO CALL BROWSER.
846
CREATE NEW BROWSER WINDOW.
848

FIG.8A

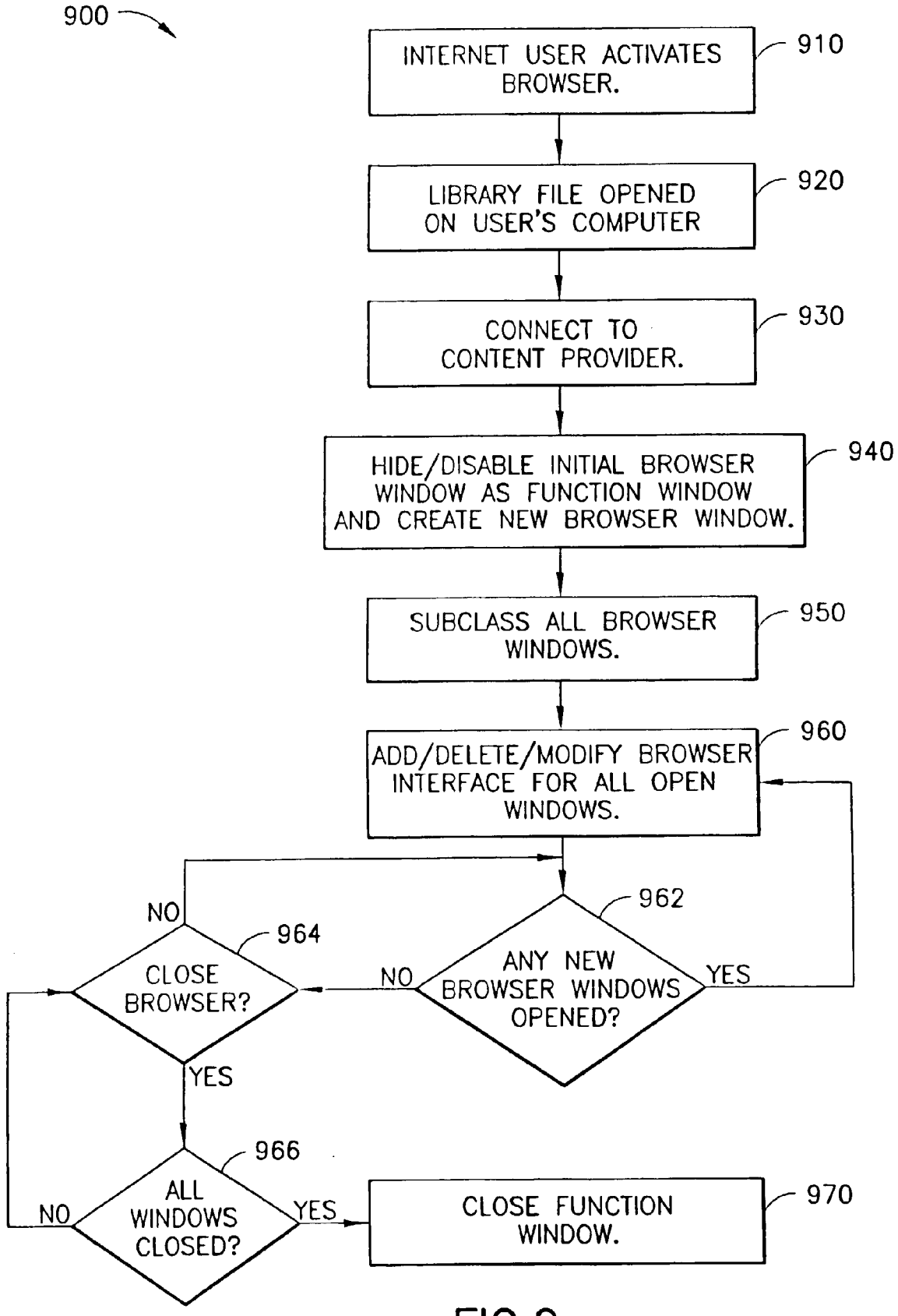

900
INTERNET USER ACTIVATES BROWSER.
910
LIBRARY FILE OPENED ON USER'S COMPUTER
920
CONNECT TO CONTENT PROVIDER.
930
HIDE/DISABLE INITIAL BROWSER WINDOW AS FUNCTION WINDOW AND CREATE NEW BROWSER WINDOW.
940
SUBCLASS ALL BROWSER WINDOWS.
950
ADD/DELETE/MODIFY BROWSER INTERFACE FOR ALL OPEN WINDOWS.
960
962
ANY NEW BROWSER WINDOWS OPENED?
YES
NO
964
CLOSE BROWSER?
NO
YES
966
ALL WINDOWS CLOSED?
NO
YES
CLOSE FUNCTION WINDOW.
970

FIG.9

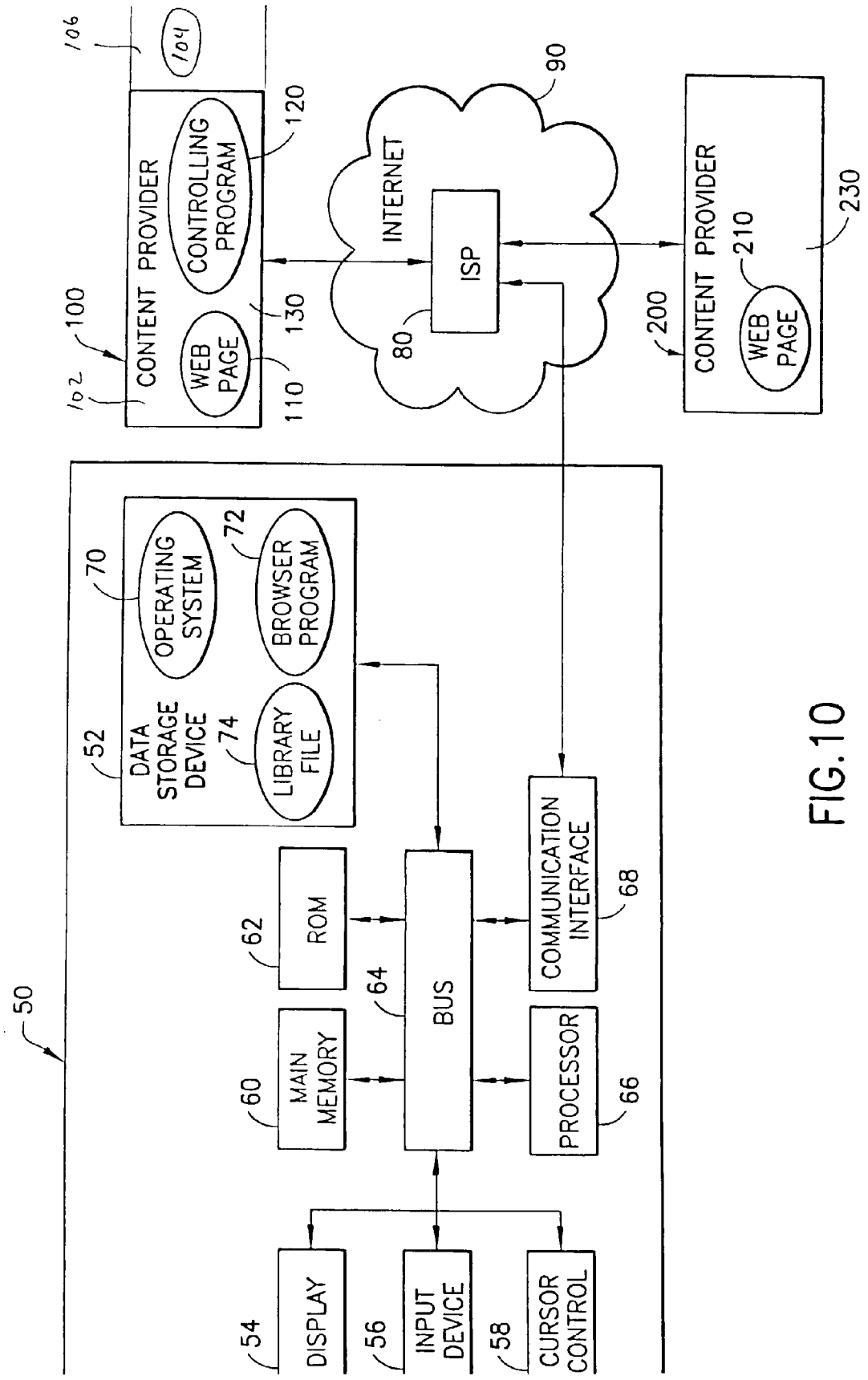

106
104
CONTENT PROVIDER
CONTROLLING PROGRAM
120
100
102
WEB PAGE
110
130
INTERNET
90
ISP
80
CONTENT PROVIDER
210
WEB PAGE
200
230
50
DATA STORAGE DEVICE
52
OPERATING SYSTEM
70
BROWSER PROGRAM
72
LIBRARY FILE
74
ROM
62
MAIN MEMORY
60
BUS
64
COMMUNICATION INTERFACE
68
PROCESSOR
66
DISPLAY
54
INPUT DEVICE
56
CURSOR CONTROL
58

FIG. 10

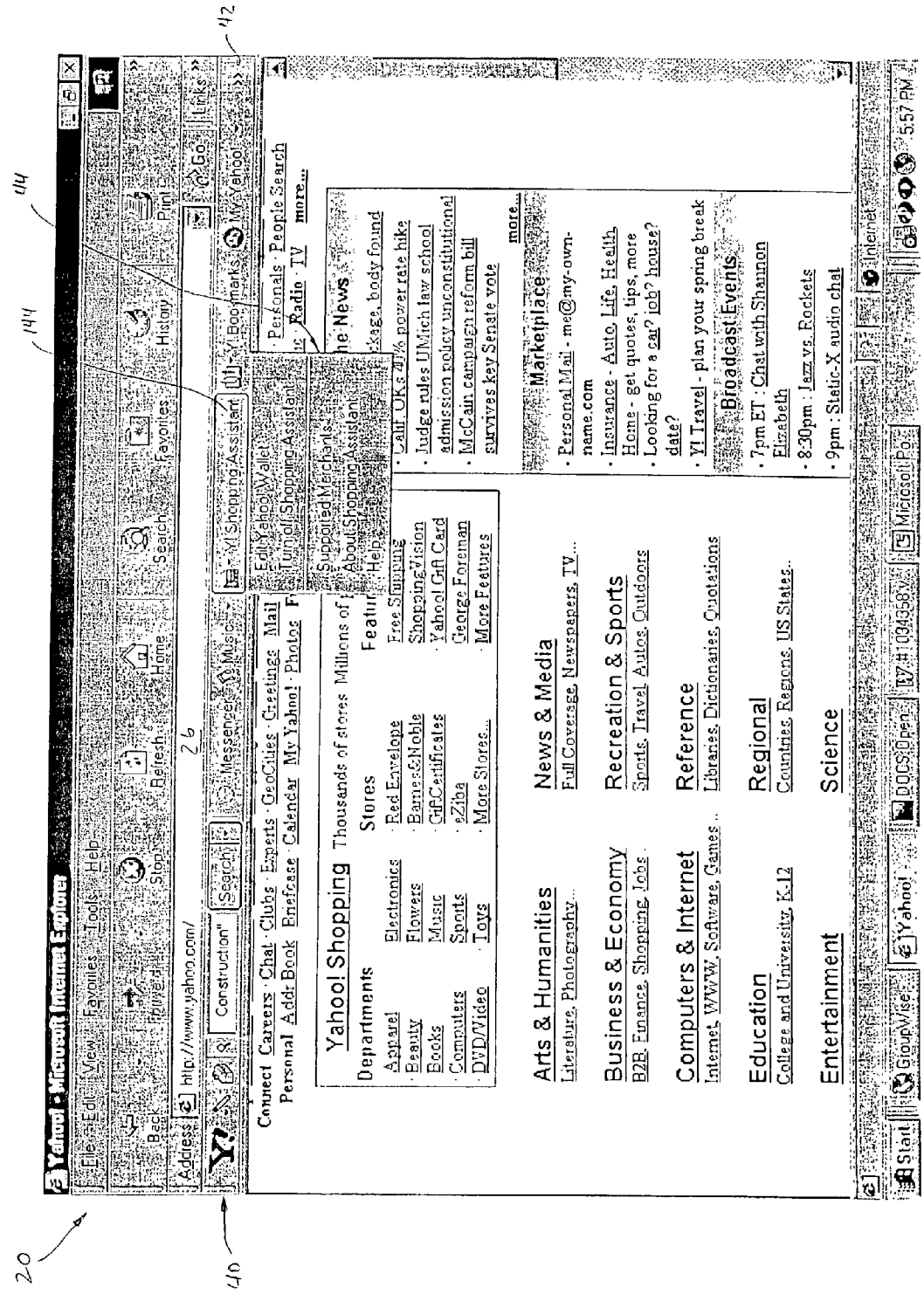

20
40
26
42
44
144
Yahoo! - Microsoft Internet Explorer
Yahoo! Shopping
Thousands of stores
Departments
Stores
Arts & Humanities
News & Media
Business & Economy
Recreation & Sports
Computers & Internet
Reference
Education
Regional
Entertainment
Science
Marketplace
Broadcast Events

FIG. 11

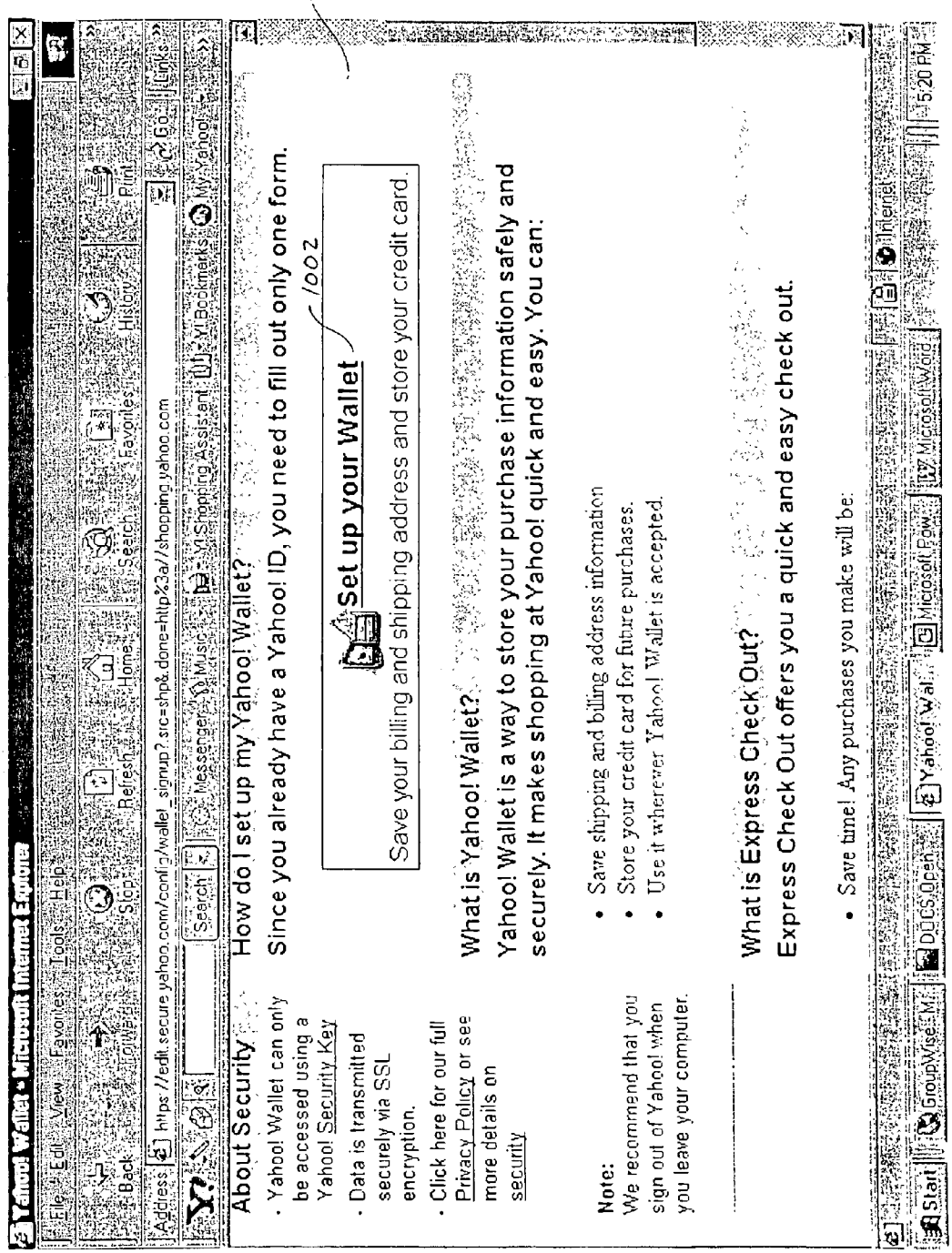

Yahoo! Wallet - Microsoft Internet Explorer
About Security
How do I set up my Yahoo! Wallet?
Since you already have a Yahoo! ID, you need to fill out only one form.
1002
Set up your Wallet
Save your billing and shipping address and store your credit card.
What is Yahoo! Wallet?
Yahoo! Wallet is a way to store your purchase information safely and securely. It makes shopping at Yahoo! quick and easy. You can:
Save shipping and billing address information
Store your credit card for future purchases.
Use it wherever Yahoo! Wallet is accepted.
What is Express Check Out?
Express Check Out offers you a quick and easy check out.
Save time! Any purchases you make will be:
Yahoo! Wallet can only be accessed using a Yahoo! Security Key
Data is transmitted securely via SSL encryption.
Click here for our full Privacy Policy or see more details on security
Note:
We recommend that you sign out of Yahoo! when you leave your computer.

FIG. 12

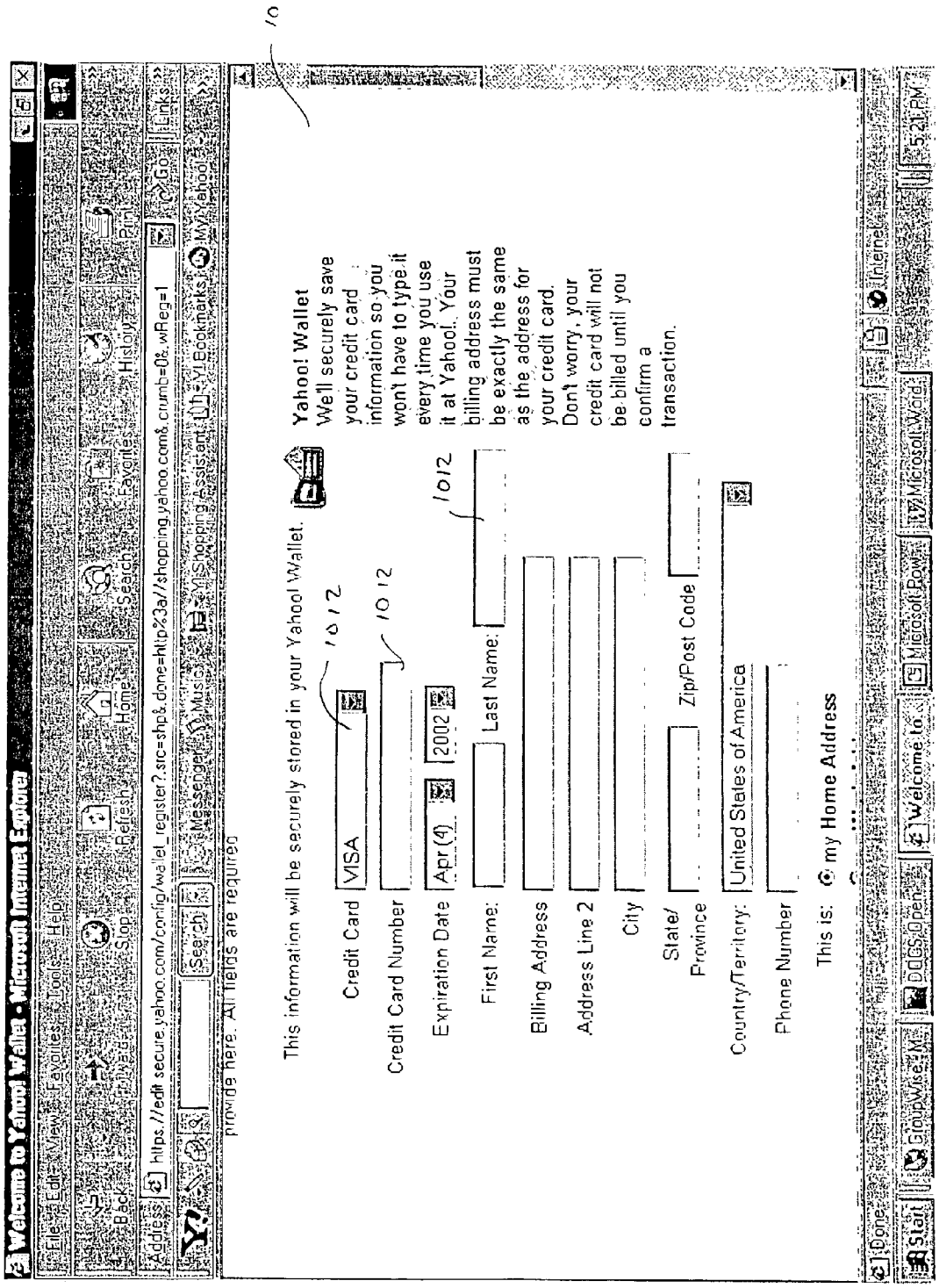

Welcome to Yahoo! Wallet - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Print
Address https://edit.secure.yahoo.com/config/wallet_register?.src=shp&.done=http%3a//shopping.yahoo.com&.crumb=0&.wReg=1
Go Links
Search Messenger Music Shopping Assistant Bookmarks My Yahoo!
provide here. All fields are required
This information will be securely stored in your Yahoo! Wallet.
Yahoo! Wallet
We'll securely save your credit card information so you won't have to type it every time you use it at Yahoo!. Your billing address must be exactly the same as the address for your credit card. Don't worry, your credit card will not be billed until you confirm a transaction.
Credit Card
VISA
Credit Card Number
Expiration Date
Apr (4)
2002
First Name:
Last Name:
Billing Address
Address Line 2
City
State/ Province
Zip/Post Code
Country/Territory:
United States of America
Phone Number
This is:
my Home Address
1012
1012
1012
10
Done
Internet
Start
GroupWise M
DOGS Open
Welcome to
Microsoft Pow
Microsoft Word
5:21 PM

FIG. 13

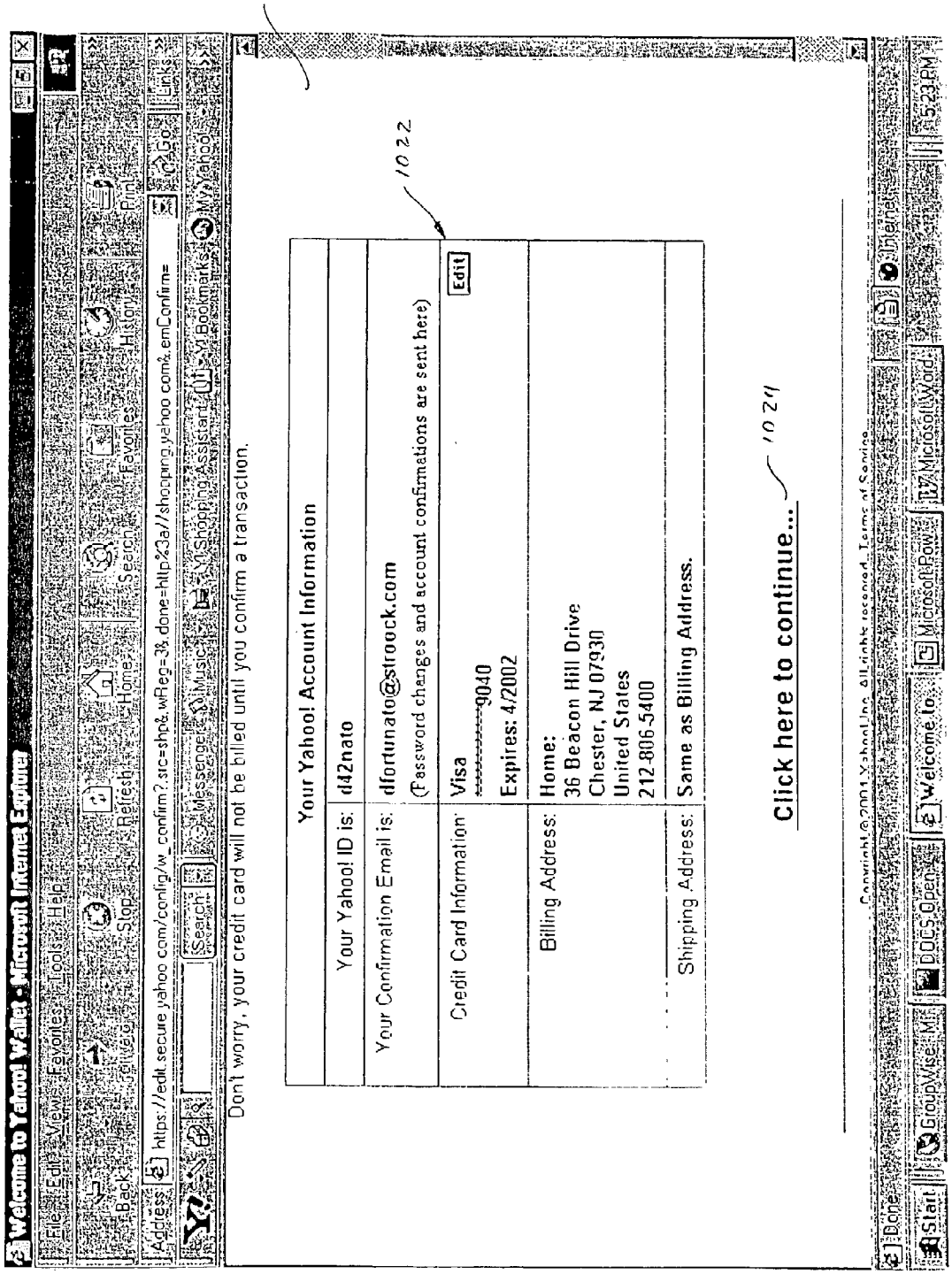

Welcome to Yahoo! Wallet - Microsoft Internet Explorer
Don't worry, your credit card will not be billed until you confirm a transaction.
Your Yahoo! Account Information
Your Yahoo! ID is:
d42nato
Your Confirmation Email is:
dfortunato@stroock.com
(Password changes and account confirmations are sent here)
Edit
Credit Card Information:
Visa
9040
Expires: 4/2002
Billing Address:
Home:
36 Beacon Hill Drive
Chester, NJ 07930
United States
212-806-5400
Shipping Address:
Same as Billing Address.
Click here to continue...
1022
1021

FIG. 14

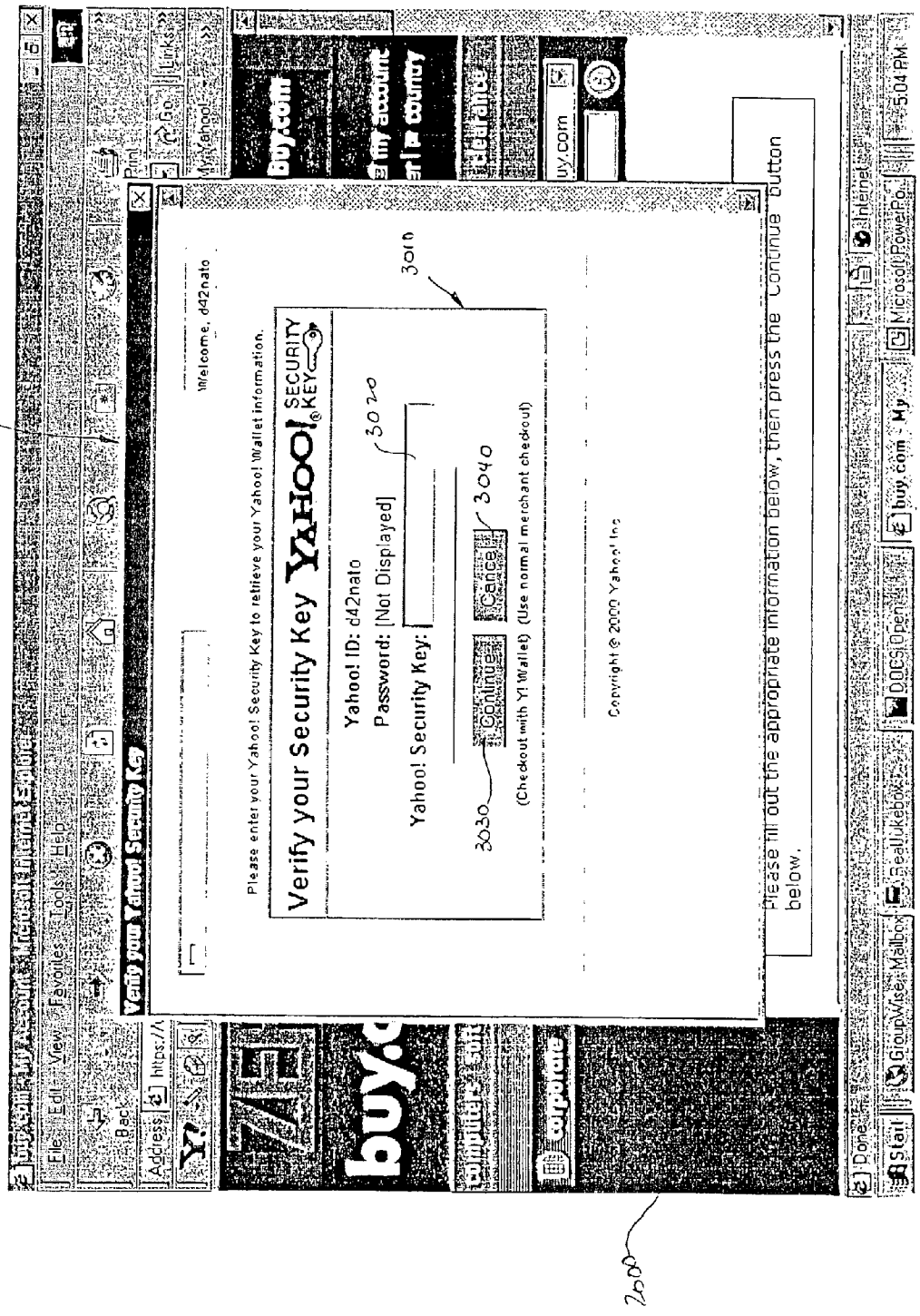

3000
Verify your Yahoo! Security Key
Welcome, d42nato
Please enter your Yahoo! Security Key to retrieve your Yahoo! Wallet information.
Verify your Security Key YAHOO! SECURITY KEY
Yahoo! ID: d42nato
Password: [Not Displayed]
Yahoo! Security Key:
3010
3020
3030
3040
Continue
Cancel
(Checkout with Y! Wallet) (Use normal merchant checkout)
Copyright © 2000 Yahoo! Inc.
Please fill out the appropriate information below, then press the "Continue" button below.
2000

FIG. 15

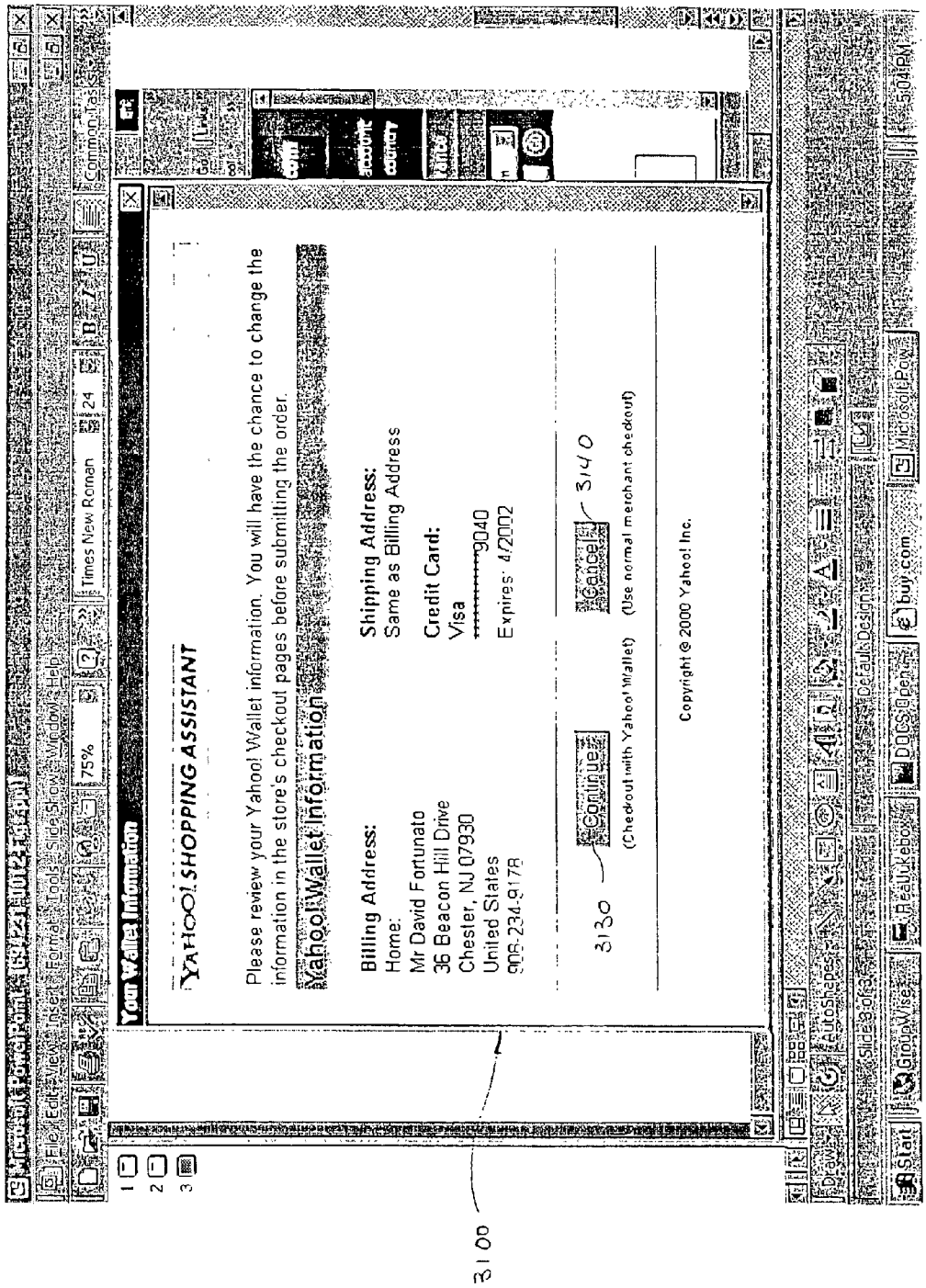

Your Wallet Information
YAHOO! SHOPPING ASSISTANT
Please review your Yahoo! Wallet information. You will have the chance to change the information in the store's checkout pages before submitting the order.
Yahoo! Wallet Information
Billing Address:
Home:
Mr David Fortunato
36 Beacon Hill Drive
Chester, NJ 07930
United States
908-234-9178
Shipping Address:
Same as Billing Address
Credit Card:
Visa
**********9040
Expires: 4/2002
Continue
(Checkout with Yahoo! Wallet)
Change
(Use normal merchant checkout)
Copyright © 2000 Yahoo! Inc.
3130
3140
3100

FIG. 16

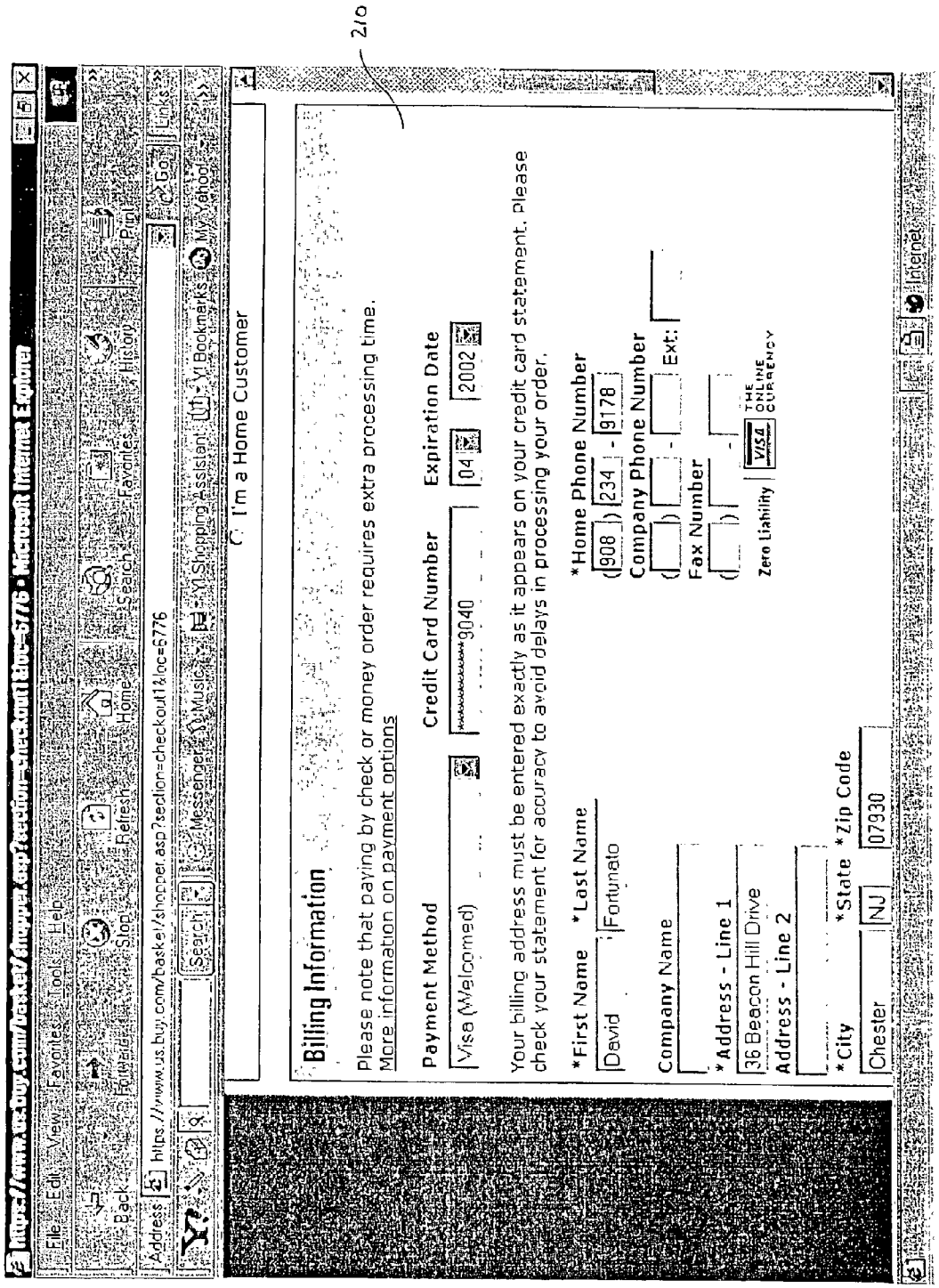

210
I'm a Home Customer
Billing Information
Please note that paying by check or money order requires extra processing time.
More information on payment options
Payment Method
Credit Card Number
Expiration Date
Visa (Welcomed)
04
2002
Your billing address must be entered exactly as it appears on your credit card statement. Please check your statement for accuracy to avoid delays in processing your order.
*First Name
*Last Name
David
Fortunato
*Home Phone Number
908) 234 - 9178
Company Name
Company Phone Number
Ext:
Fax Number
*Address - Line 1
36 Beacon Hill Drive
Address - Line 2
*City
*State
*Zip Code
Chester
NJ
07930
Zero Liability
THE ONLINE CURRENCY

FIG. 17

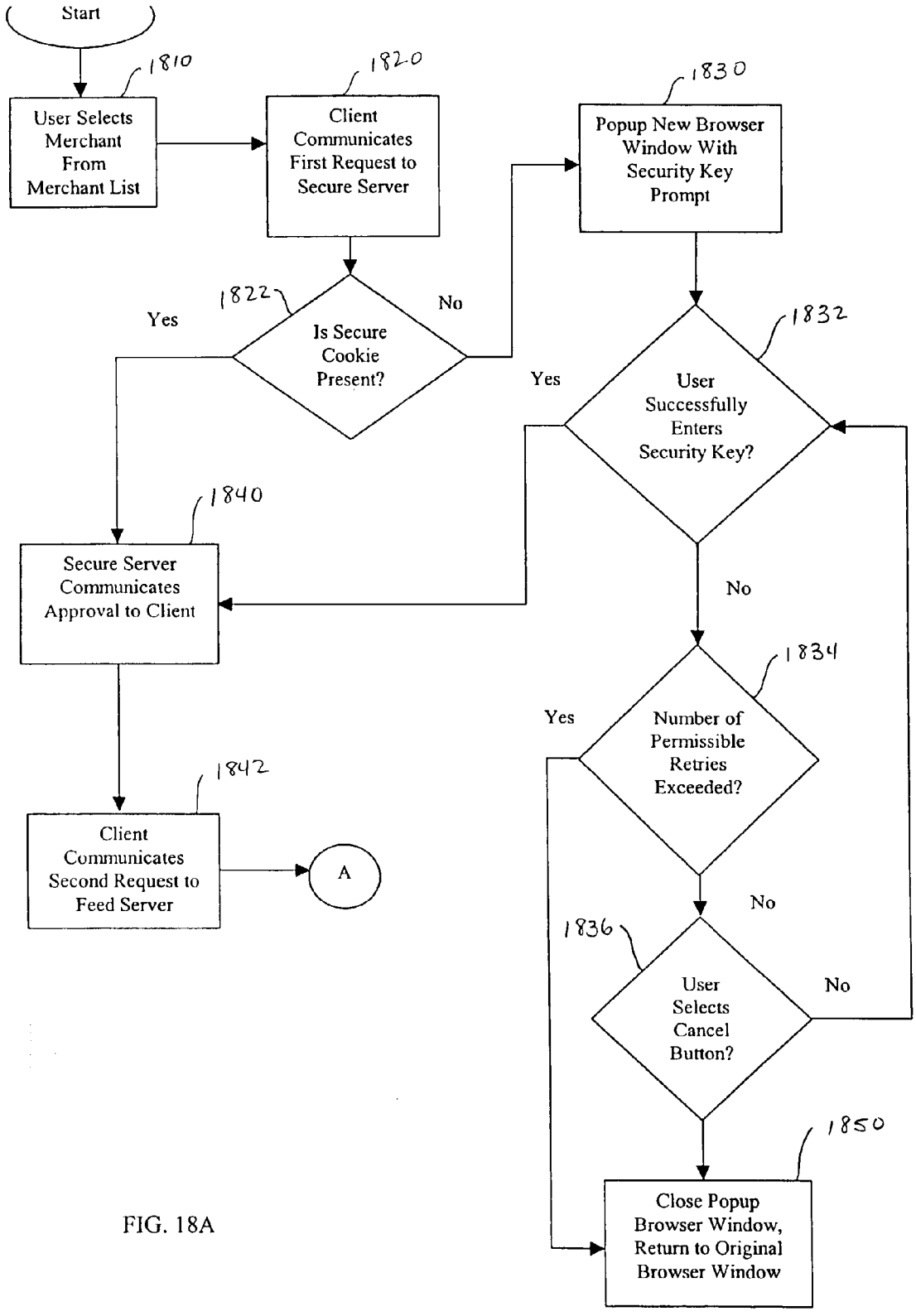

Start
1810
User Selects Merchant From Merchant List
1820
Client Communicates First Request to Secure Server
1822
Is Secure Cookie Present?
Yes
No
1830
Popup New Browser Window With Security Key Prompt
1832
User Successfully Enters Security Key?
Yes
No
1834
Number of Permissible Retries Exceeded?
Yes
No
1836
User Selects Cancel Button?
No
1840
Secure Server Communicates Approval to Client
1842
Client Communicates Second Request to Feed Server
A
1850
Close Popup Browser Window, Return to Original Browser Window

FIG. 18A

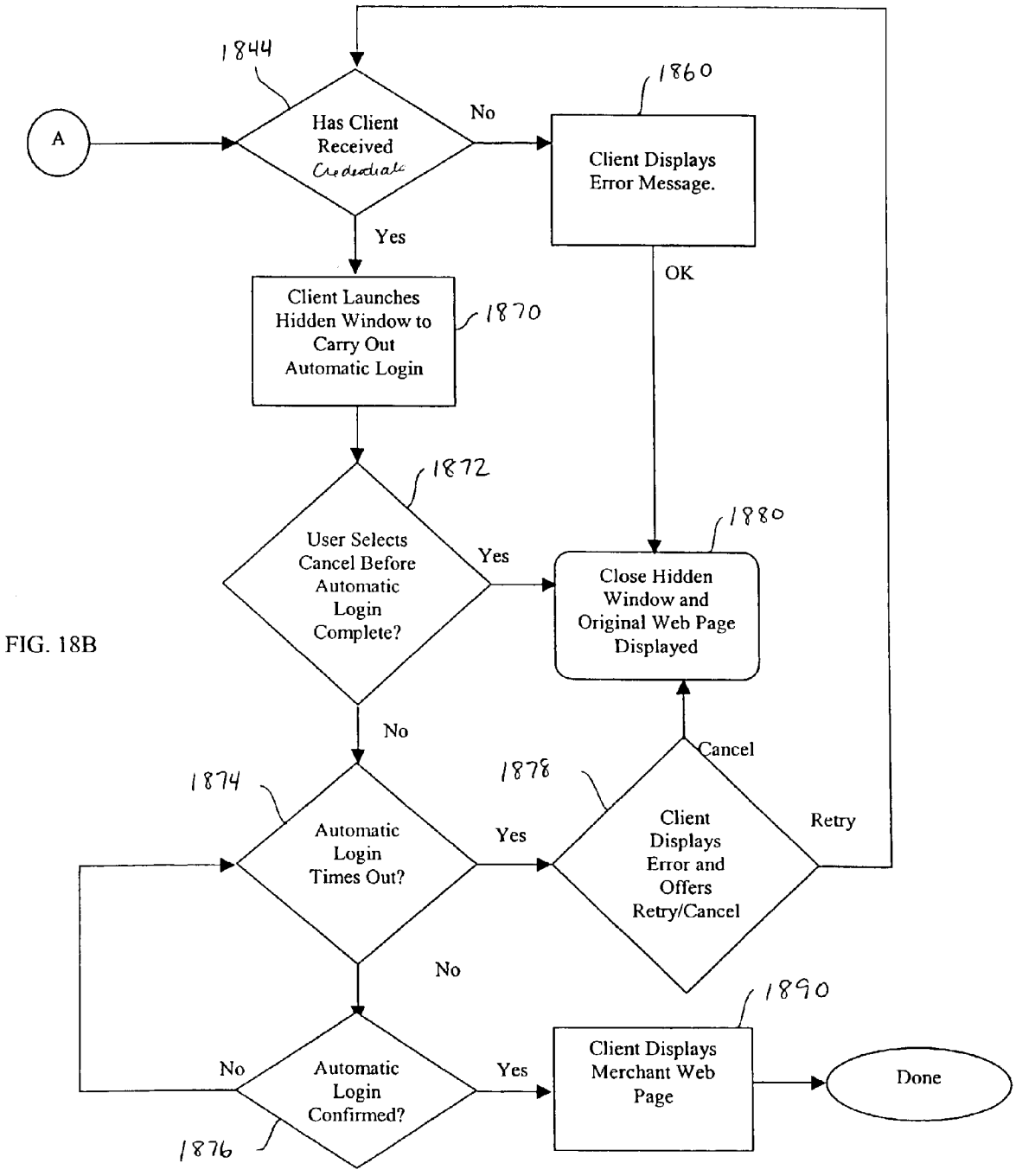

A
1844
Has Client Received Credentials
No
1860
Client Displays Error Message.
Yes
OK
Client Launches Hidden Window to Carry Out Automatic Login
1870
1872
User Selects Cancel Before Automatic Login Complete?
Yes
1880
Close Hidden Window and Original Web Page Displayed
No
Cancel
1874
Automatic Login Times Out?
Yes
1878
Client Displays Error and Offers Retry/Cancel
Retry
No
1890
No
Automatic Login Confirmed?
Yes
Client Displays Merchant Web Page
Done
1876

FIG. 18B

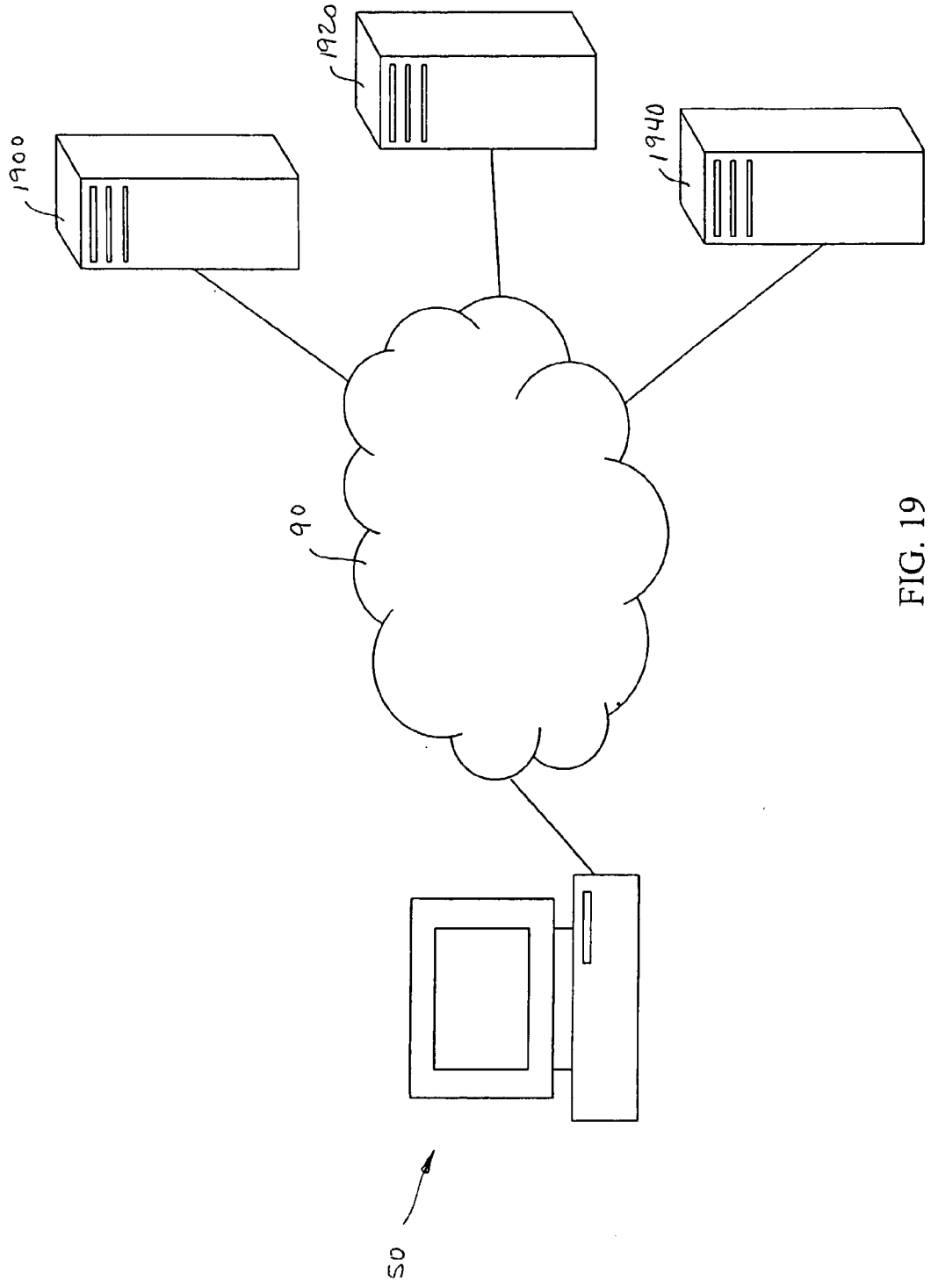


FIG. 19

METHOD AND SYSTEM OF FACILITATING AUTOMATIC LOGIN TO A WEB SITE USING AN INTERNET BROWSER

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application is a Continuation-In-Part of U.S. patent application Ser. No. 09/429,585, filed on Oct. 28, 1999, and a Continuation-In-Part of U.S. patent application Ser. No. 10/015,816, filed on Nov. 1, 2001, entitled Method And System Of Facilitating On-Line Shopping Using An Internet Browser, both currently pending. This application also claims priority to U.S. Provisional Patent Application Serial No. 60/331,565, filed on Nov. 16, 2001.

FIELD OF THE INVENTION

[0002] The present invention is directed to a method and system for adding functionality to an Internet browser interface.

BACKGROUND OF THE INVENTION

[0003] When accessing the Internet (i.e., the worldwide web, the web, etc.), an Internet user typically launches or activates, via a computer, a browser software program such as, for example, Netscape Navigator™ or Microsoft Internet Explorer™. The browser program (also referred to herein as a browser) establishes a link between the user's computer and the Internet (via a modem and an Internet Service Provider (ISP), for example) and also provides a textual and graphical user interface, i.e., a browser interface, having a predetermined look and functionality, neither of which can currently be significantly changed by the Internet user. Thus, the browser interface remains relatively static as the Internet user navigates the Internet, moving from web site to web site, application to application, or HTML (Hyper-text Mark-up Language) page to HTML page.

[0004] Limited control of the browser interface is currently available via an executable software program that may, for example, add functional buttons to the browser interface. However, the additional functionality is added to the browser interface when the browser is initially activated and remains static thereafter. Thus, it is not possible for a browser displaying a browser interface modified as just-described to dynamically download information from an Internet site and customize itself, either when the browser is initiated or as the users navigates the Internet. Such a modified browser interface also does not provide access to the various browser Application Programming Interfaces (APIs) for Plug-ins and interfaces. It is thus desirable to provide a method for modifying a browser interface, and to provide a browser interface, that overcomes the above-described shortcomings of the prior art.

[0005] When accessing certain web sites, it may be necessary for a user to enter a user identifier and password before being permitted to access data via that web site specific to that user. For example, most financial institutions, investment companies, and other service-providing entities, including, but not limited to on-line shopping web sites, permit a user (or client of the entity) to access his or her account(s) via the Internet. For obvious reasons, such access is predicated on the user entering certain user-specific information prior to obtaining access the user's account. For example, to set up an on-line account, a user may be required to provide certain user-specific information such as, for example, a user identifier and password. Once that user-specific information is provided and an on-line account is established, the user may only access his or her on-line account by providing the user-specific information as part of a login process.

[0006] It is not uncommon for a user to maintain a plurality of on-line accounts (with a plurality of financial institutions, investment companies, and other service-providing entities) that may be accessible via a plurality of web sites. While it is conceivable that a user may use the same login identifier and passwords for each of the plurality of web sites, such a practice is not recommended. In fact, a more desirable and recommended practice is to use different login identifiers and password for each on-line account. That will increase security of the user's on-line accounts and reduce the possibility of unauthorized access to those accounts if someone happens to obtain one of the user's login identifiers and passwords. However, using different login identifiers and passwords for each on-line account present its own problems; not the least of which is remembering each login identifier and password.

[0007] It is also desirable for a user to retrieve information from one or more web sites, without having to navigate to each web site. It is further desirable to obtain such information simply and quickly using an Internet browser.

SUMMARY OF THE INVENTION

[0008] The present invention is directed to a method and system of adding functionality to an Internet browser interface. In one embodiment of the present invention, the added functionality may facilitate automatic login to a web site using an Internet browser. In another embodiment, the added functionality may enable the user to perform various tasks using the Internet browser such as, by way of non-limiting example, performing various tasks required to navigate one or more web pages, or to retrieve information desired by the user from one or more web pages or web sites.

[0009] In accordance with embodiments of the present invention, computer code is communicated from a server to a user's computer that adds certain functionality to the user's Internet browser. The present invention thus provides dynamic control of an Internet browser interface by enabling a user to selectively add buttons and/or toolbars to the browser interface, and to add features and functionality to the browser. In a preferred embodiment, the computer code enables a user to modify the browser interface by selectively adding an interface object to the Internet browser interface. The interface object may comprise a button that may be added to the browser interface as part of an existing toolbar, or as part of a new toolbar added to the browser interface. The button enables a user to access a merchant list and also adds automatic login functionality to the browser. Access to the merchant list enables a user to adds merchants, edit merchants, and select a merchant for automatic login. Once the button is added to the browser interface, the user may click on the added button, generally referred to herein as an automatic login button, and gain access to a pull-down menu which displays one or more merchants as a merchant list. By selecting one of the merchants on the merchant list, the functionality added to the browser in accordance with the

present invention will cause the user to automatically login to the selected merchant and cause the user's browser to navigate to the merchant's web site. That process is essentially transparent to the user and initiated with the user's selection of a merchant from the merchant list.

[0010] The automatic login functionality provided in accordance with the present invention preferably utilizes a secure login process that requires the user to enter a security key before the automatic login process to the selected merchant will be completed. If a user has not previously entered the security key during a particular Internet session, or if the user entered a security key during a session, and a predetermined amount of time has passed since entering the security key, the user will be prompted to enter a security as part of the automatic login process. Only after entering a correct security key will the automatic login process proceed. The just-described secure login process may be carried out by a secure server and the user's computer.

[0011] If a user has securely logged in to a security server during an Internet session, a secure cookie may be communicated from the secure server and stored on the user's computer. If the secure cookie is present on the user's computer and valid (i.e., if it has not expired, for example), the secure cookie is communicated by the computer code to the secure server as part of a first request when the user selects a merchant from the merchant list. The secure server receives the request and determines if it contains a secure cookie, and if the secure cookie is valid. If a valid secure cookie is included in the first request, the secure server communicates an affirmation to the computer code which, in turn, communicates a second request to a feed server (which may be the same or different from the secure server, as a matter of design choice). The second request includes a user identifier and a merchant identifier. When the feed server receives the second request, it compares the user identifier and merchant identifier with the database entries to determine if there is a match. If a match is found, the feed server communicates a request for credentials to another server, which may be the secure server. That other server includes a secure database having a plurality of data for a plurality of users. Each user's data is identifiable by a user identifier, and may include a login identifier and password for each of a plurality of merchants, with each of the plurality of merchants having an associated merchant rules file. Each database entry consisting of a login identifier, password and merchant rules file may also be referred to herein as credentials. Once the credentials associated with the merchant selected by the user have been identified, that other server communicates the credentials to the user's computer. Upon receipt of the credentials, the computer code provided in accordance with the present invention causes the user's browser to navigate to a predetermined web page of the selected merchant, such as a login web page, for example, and automatically populates any fields in that web page necessary for a user to login and access that user's account. Again, that process is transparent to the user. If the automatic login is successful, a merchant web page that typically follows a successful login will be communicated by the merchant's server and displayed in the user's browser window.

[0012] Thus, by selecting a merchant from the merchant list, a user can automatically navigate and login to that merchant's web page, and access the user's account.

[0013] As used herein, the term controlling and controllable refer to, by way of non-limiting example, adding to, removing from, and modifying an Internet browser interface. An Internet browser interface, as referred to herein, means the visual or aural presentation presented to a browser user, and via which a user interacts with the browser. The term customize (and variations thereof) may also be used herein to describe the controllability provided in accordance with the present invention. As used herein, the term dynamically controlling (and variations thereof) refers to a method by which a part of an Internet browser interface (i.e., an interface object) may be selectively added, displayed and periodically changed or updated.

[0014] A browser, as used herein, is given its general, art recognized meaning, and generally refers to a software program that provides an interface to the World Wide Web or Internet. The browser enables an Internet user to navigate the Internet and establish a connection to various Internet sites (also referred to herein as web sites) and view the web page(s) provided at the various Internet sites by content providers. A browser is specifically capable of calling an ActiveX control or Plug-in via an Internet site. The browser also enables an Internet user to navigate between and among Internet sites, i.e., to surf the web. The browser provides a browser interface to the Internet user, the format of which is determined by the provider of the browser software program. The browser interface typically defines the layout (e.g., color, size, number and location) and functionality provided by the browser to the Internet user. The browser interface may comprise a first parent window that typically defines the general size, color, and layout of the browser interface and includes window control buttons (e.g., minimize, close, etc.). The browser interface may also comprise a second parent window (that may be a child to the first parent window), and one or more windows dependent from the second parent. The second parent and its dependent windows may provide, for example, various information (advertisements, coupons, news, HTML links, etc.) and functionality (toolbars, pull-down menus, Plug-ins, applications, etc.) to the Internet user.

[0015] An ActiveX control, as used herein, refers to a tool for linking desktop applications for the Internet and is based on art recognized, Microsoft-developed specifications. A Plug-in, as used herein, refers to a type of program developed for use with Netscape browsers and that integrates with a larger application (e.g., a browser software program) to add a specific functionality or capability to that larger program. An ActiveX control and a Plug-in as described above and as referred to herein may be used with any Internet browser.

[0016] As used herein, the term Internet site (or web site) refers to a location on the Internet defined by an Internet address or URL (uniform or universal resource locator). As used herein, the term Internet web page refers to a collection of hypertext markup language (HTML) commands provided at an Internet site that provides formatting information for the text, graphics, and functionality to create a hypertext document that may be displayed on an Internet user's computer display via a browser. For example, an Internet user enters a URL to establish a connection to an Internet site, and that Internet site provides HTML commands to the user's browser to enable display of that Internet site's web page on the user's computer display. The browser interprets

HTML commands embedded in a web page and uses HTML commands to format the text and graphics for that web page.

[0017] The present invention provides advantages to an Internet user, an Internet content provider, and to an Internet Service Provider (ISP). For an Internet user, the present invention provides a method of dynamically controlling or customizing that user's Internet browser interface. The Internet user may now customize the browser interface so that each time the user accesses the Internet using a browser, user-defined information and/or functionality (also collectively referred to herein as information) will be displayed with the browser interface. For example, the user may include bookmarks, address and phone books, personal financial information, personalized news, and various functionality such as is available via ActiveX control and Plug-ins.

[0018] In addition, if an Internet user has an account with a content provider, that user's specific account information may now be dynamically displayed with the browser interface by the browser. Currently, an Internet user can only access that user's specific account information while connected to the content provider's Internet site. The user must return to the content provider's site to receive updated account information. The account information is not displayed with the browser interface once user leaves that Internet site. The present invention provides a method of dynamically controlling and a dynamically controllable browser interface that enables an Internet user to display with the browser interface and continuously update information and/or functionality specific to that user.

[0019] For a content provider, the present invention ensures that an Internet user (via a browser) automatically establishes a connection to that content provider's Internet site every time that user accesses the Internet using a browser. Thus, as soon as an Internet user causes a browser to execute (by selecting a browser icon, for example), the browser automatically establishes a connection to the content provider's Internet site to load that user's customized browser interface information. The present invention may also periodically and automatically cause the user's browser to connect to the content provider's Internet site while the browser is active and while the user surfs the web. In one embodiment, the content provider may provide an Internet user with access to a program for controlling the browser interface. Once the Internet user has accessed that controlling program to customize that user's browser interface, a connection to that content provider may be automatically established by that user's browser every time that user accesses the Internet. Thus, and in contrast to currently available browsers which establish a connection to an Internet site only when the user enters a URL (or otherwise positively acts to cause a connection to be established such as, for example, by selecting a link or banner advertisement), the present invention automatically establishes a connection to the content provider when the browser is initiated, independent of any home web page selection pre-programmed into the browser, whether by the Internet user or browser vendor. Once that initial connection is established, the content provider may load user-specific information and/or functionality to the Internet user's computer for display with that user's browser interface.

[0020] In addition, the content provider can also periodically cause the browser to automatically reconnect to that content provider's Internet site to update, download new, or otherwise communicate information and/or functionality for the Internet user's browser interface. For example, if an Internet user subscribes to an email service of the content provider, email messages for that Internet user received by the content provide may be automatically communicated to the Internet user even though the user is "surfing" elsewhere. When the user's browser initially establishes a connection to the content provider's Internet site upon execution of that user's browser, the information communicated by the content provider to the Internet user includes instructions for the browser to periodically reconnect to the content provider's Internet site. Thus, regardless of the number of Internet sites the user accesses, and regardless of the particular Internet site currently accessed by a user, a connection back to the content provider's Internet site will be automatically established at intervals determined by the content provider; those reconnections being transparent to the Internet user except when the user receives a notification from the content provider (i.e., new mail has arrived). Thus, the browser interface may be dynamically controlled as the Internet user surfs the web. For an ISP, the benefits are at least as great as for a content provider.

[0021] Initially, the browser interface for an Internet user must be customized using a software program that may be provided by the content provider or ISP, or that may be available on the user's computer. The software program, referred to hereinafter as computer code, a controlling program or a program for controlling (and other variations thereof), changes the manner in which the user's browser functions. More specifically, the controlling program downloads or creates a library file on the Internet user's computer. The library file may be, for example, a Dynamic Link Library (DLL) (for a Windows operating system) that creates a shell (or plurality of shells) within the browser and within which various information and/or functionality may be loaded as an ActiveX control or Plug-in. The library file includes ActiveX control or Plug-in functionality that defines an interface object added to the browser interface in accordance with the present invention. When an Internet user launches or activates a browser, the library file is opened and the ActiveX control or Plug-in code contained within that file is made available to the browser and incorporated into the browser interface, thus causing the interface object to be displayed with the browser interface. The library file, and consequently, the shell (or shells), remain open as long as the browser is activated, generally as long as the user is surfing the web. Thus, the information and/or functionality for customizing the browser interface and loaded in the shell remain active even as the user moves from Internet site to Internet site. When used in this context herein, the terms information and functionality refer to any information, data, and/or software-driven functionality that can be contained in or part of the library file.

[0022] The library file also causes the browser to establish a connection to the content provider's Internet site when the browser is initially activated by the user. The content provider's Internet site will load information and/or functionality for the interface object to the user's computer for use in the browser and for display in the browser interface. The information and/or functionality loaded by the content provider may be specific to an Internet user if, for example, that user has an account with the content provider. Alternatively, the content provider may load general information and/or

functionality if, for example, the Internet user does not have an account with the content provider (i.e., is a guest).

[0023] The present invention uses an object linking and embedding (OLE) in-process server to control the information and/or functionality of a browser interface. Using an ActiveX control or a browser Plug-in (each being referred to herein as a browser interface overlay (BIO) Library) contained in a library file, virtually any information and/or functionality available with an ActiveX control or Plug-in may be added to a browser interface using the present invention. The library file (via the BIO Library) thus includes the code required to customize, i.e., add, remove and/or modify, the browser interface.

[0024] Once an Internet user has accessed the controlling program and customized the operation of that user's browser, the customized browser interface is displayed when the browser is activated. In contrast to prior art browser modification methods, the present invention provides a method and browser interface that may be dynamically controlled. Updated or changed information and/or functionality may be communicated to the browser and displayed in the browser interface as the Internet user surfs the web and while maintaining the customized information and/or functionality of the browser interface. Thus, an Internet user may automatically receive up-to-date information such as, for example, stock quotes, email, new headlines, at that user's browser interface, at any Internet site and as long as the user is surfing the web using the browser.

[0025] The present invention also provides a method and system of facilitating on-line shopping by modifying a browser interface to provide a shopping assistant button and functionality. The modified browser interface provided in accordance with this embodiment of the present invention simplifies on-line shopping by creating a wallet for each shopper or user. The wallet is stored in a database on a secure server and may be accessed, used, or modified only by the user and only in connection with a user-provided security key. The wallet contains user-supplied information that may be directly ported to a supported merchant check-out web page. Thus, when a user is shopping on-line at a supported merchant's web site, and desired to purchase certain merchandise and/or services from that merchant, the present invention provides the user with access to his/her wallet and facilitates porting of the data contained in the wallet to quickly and effortlessly fill out the merchant's order forms provided at the merchant's check-out web pages.

[0026] In yet another embodiment of the present invention, automatic login functionality may be added to the Internet browser. The steps for adding an automatic login interface object to the browser interface are the same as described above with regard to the various other embodiments of the present invention. Thus, those steps need not be repeated for this embodiment. An automatic login interface object adds various functionality to the browser such as, by way of non-limiting example, access to a merchant list and the ability to revise, add, and/or remove merchants from the list, and the ability to automatically login to a selected merchant web site. It should be noted that although a "merchant site" is referred to herein in connection with this embodiment, the present invention is applicable to any web site at which a user may establish an on-line account which may require a user to enter login, password, and/or other user data to access the user's account via that web site.

[0027] Other objects and features of the present invention will become apparent from the following detailed description, considered in conjunction with the accompanying drawing figures. It is to be understood, however, that the drawings are designed solely for the purpose of illustration and not as a definition of the limits of the invention, for which reference should be made to the appended claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0028] In the drawing figures, which are merely illustrative, and wherein like reference characters denote similar elements throughout the several views:

[0029] **FIG. 1** depicts a view of a prior art Internet browser interface;

[0030] **FIG. 2** depicts a view of an Internet browser interface including an interface object in the browser toolbar in accordance with an embodiment of the present invention;

[0031] **FIG. 3** depicts a view of an Internet browser interface having an interface toolbar including interface object in accordance with an embodiment of the present invention;

[0032] **FIG. 4** depicts a view of an Internet browser interface having an interface toolbar including a plurality of interface objects in accordance with an embodiment of the present invention;

[0033] **FIG. 5** is a flow diagram of a method of controlling an Internet browser interface in accordance with the present invention;

[0034] FIGS. **6-9** are a flow diagrams of a method of controlling and displaying an Internet browser interface in accordance with various embodiments of the present invention;

[0035] **FIG. 10** is a schematic block diagram of a computer connected to the Internet and upon which the present invention may be implemented;

[0036] **FIG. 11** is an exemplary screen shot of an Internet browser interface having a shopping assistant button in accordance with an embodiment of the present invention;

[0037] **FIG. 12** is an exemplary screen shot of a wallet set-up web page in accordance with an embodiment of the present invention;

[0038] **FIG. 13** is an exemplary screen shot of a wallet data entry web page in accordance with an embodiment of the present invention;

[0039] **FIG. 14** is an exemplary screen shot of a wallet summary web page in accordance with an embodiment of the present invention;

[0040] **FIG. 15** is an exemplary screen shot of a security key window in accordance with an embodiment of the present invention;

[0041] **FIG. 16** is an exemplary screen shot of a wallet information window in accordance with an embodiment of the present invention;

[0042] **FIG. 17** is an exemplary screen shot of supported merchant check-out web page in accordance with an embodiment of the present invention;

[0043] FIGS. **18A-18B** is a flow diagram of a method of automatically logging a user on to a merchant web site in accordance with an embodiment of the present invention; and

[0044] **FIG. 19** is a schematic diagram of a system for facilitating automatic login by a user to a merchant web site in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION OF THE PRESENTLY PREFERRED EMBODIMENTS

[0045] The present invention is directed to a method and system of facilitating automatic login to a web site using an Internet browser. In accordance with an embodiment of the present invention, computer code is communicated from a server to a user's computer that adds certain functionality to the user's Internet browser. The present invention thus provides dynamic control of an Internet browser interface by enabling a user to selectively add buttons and/or toolbars to the browser interface, and to add features and functionality to the browser. In a preferred embodiment, the computer code enables a user to modify the browser interface by selectively adding an interface object to the Internet browser interface. The interface object may comprise a button that may be added to the browser interface as part of an existing toolbar, or as part of a new toolbar added to the browser interface. The button enables a user to access a merchant list and also adds automatic login functionality to the browser. Access to the merchant list enables a user to adds merchants, edit merchants, and select a merchant for automatic login. Once the button is added to the browser interface, the user may click on the added button, generally referred to herein as an automatic login button, and gain access to a pull-down menu which displays one or more merchants as a merchant list. By selecting one of the merchants on the merchant list, the functionality added to the browser in accordance with the present invention will cause the user to automatically login to the selected merchant and cause the user's browser to navigate to the merchant's web site. That process is essentially transparent to the user and initiated with the user's selection of a merchant from the merchant list.

[0046] The present invention also provides a method and system of facilitating on-line shopping by modifying a browser interface to provide a shopping assistant button and functionality. The modified browser interface provided in accordance with this embodiment of the present invention simplifies on-line shopping by creating a wallet for each shopper or user. The wallet is stored in a database on a secure server and may be accessed, used, or modified only by the user and only in connection with a user-provided security key. The wallet contains user-supplied information that may be directly ported to a supported merchant check-out web page. Thus, when a user is shopping on-line at a supported merchant's web site, and desired to purchase certain merchandise and/or services from that merchant, the present invention provides the user with access to his/her wallet and facilitates porting of the data contained in the wallet to quickly and effortlessly fill out the merchant's order forms provided at the merchant's check-out web pages.

[0047] Referring now to the drawings in detail, **FIG. 10** is a block diagram of a computer **50** connected to the Internet **90** and upon which the present invention may be implemented. Computer **50** includes an internal bus **64** that facilitates communication of information between and among the various devices of the computer **50** and that also facilitates communication between the computer and external devices and systems via a communication interface **68**. A processor **66** coupled to the bus **64** process information within the computer **50**. The computer **50** also includes a main memory **60** such as, for example, Random Access Memory (RAM) or other equivalent dynamic memory storage device, coupled to bus **64** for receiving and storing instructions communicated from the processor **66**. Main memory **60** may also be used to temporarily store variable or other intermediate information while the processor **66** executes instructions. Read-Only-Memory (ROM) **62** is also coupled to the bus **64** for storing static data and instructions for use by the processor **66**. Various input and output devices are provided as part of the computer **50**, including, by way of non-limiting example, a display **54** (e.g., cathode ray tube (CRT), liquid crystal display (LCD), etc.), an input device **56** such as a keyboard, and a cursor control device **58** such as a mouse, or trackball, for example. A data storage device **52** such as, for example, a magnetic disk drive and magnetic disk, a CD-ROM drive and CD-ROM, or other equivalent devices and data storage mediums, is coupled to the bus **64** for communication with the processor **66**, main memory **60**, and communication interface **68**. The storage device **52** preferably has an operating system **70** and an Internet browser software program **72** stored thereon. As will be discussed in greater detail below, a library file **74** may also be stored on the data storage device **52**. As used herein, the term "computer" is intended to have its broadest interpretation, and not be limited to any physical size or construction, or to any architectural configuration. Rather, the term "computer", as used herein, is intended to include any device capable of transmitting and/or receiving digital data, storing and/or retrieving digital data, processing digital data, and that is connectable in any manner and by any means to a network such as, for example, the Internet. A computer may be a computer of any style, size, and configuration including, without limitation, a server, workstation, desktop, laptop, Internet appliance, notebook, personal digital assistant (PDA), Internet enabled cellular phone, or other now known or hereafter developed device. A computer may include some or all of the following components: a central processing unit (CPU or processor) operable in connection with software (e.g., operating system, application programs, etc.), a hard-disk unit (HDU), permanent memory (e.g., ROM), temporary memory (e.g., RAM), a removable data storage device (e.g., CD-ROM, floppy drive, etc.), an input device (e.g., keyboard, mouse, trackball, etc.), an output device (e.g., monitor or display), and an I/O device (e.g., modem, infra-red transmitter/receiver, radio (cellular) transmitter receiver, etc.). It is known to a person skilled in the art that a computer may comprise some or all of those components, in addition to components not listed.

[0048] The computer **50** may connect to the Internet **90** via the communication interface **68** over a transmission media including, but not limited to, coaxial cable, copper wires, and fiber optical cables. Communication between the computer **50** and the Internet **90** may also be via a wireless or cellular interface. The communication interface **68** facili-

tates two-way communication between the computer **50** and another electronic device or system, e.g., a server computer (not shown) provided by a content provider **100**, **200**.

[**0049**] An Internet user (not shown) using the computer **50** may gain access to the Internet **90** by causing the browser **72** to execute, thereby opening a communication link between the communication interface **68** of the computer **50** and an Internet site **130** of a content provider **100**, via an Internet Service Provider (ISP) **80**. The browser **72** provides, to the computer display **54**, a browser interface **20** (see, e.g., **FIG. 1**) having a layout (e.g., color, size, number, location) and functionality of windows **30** that is predetermined in the browser **72** by the browser vendor. Internet content is communicated by the content provider **100** to the computer **50** for display in a content window **32** of the browser interface **20**.

[**0050**] In accordance with an embodiment of the present invention, a first Internet content provider **100** may provide an Internet user with access to a program **120** for controlling the browser **72** and browser interface **20**. When executed by the user, the controlling program **120** downloads or creates a library file **74** such as, for example, a Dynamic Link Library (DLL), on the data storage device **52** of the Internet user's computer **50**. The library file **74** preferably includes ActiveX control or Plug-in functionality. Thereafter, when the Internet user accesses the Internet using the browser **72**, the browser **72** opens the library file **74** and preferably automatically establishes a connection to the content provider's Internet site **130**. The content provider, in response to the connection established by the browser **72**, loads information and/or functional data into a shell operating within the browser and created by the library file **74**. For example, if the user has an account with the content provider **100**, customized information and/or functionality may be loaded into the library file **74**. If the user does not have an account, more generalized information and/or functionality may be loaded.

[**0051**] In accordance with the various embodiments of the present invention, as described in more detail herein, data may be communicated between computers (e.g., between a server and a user or client computer). The present invention contemplates that such data communication (or transmission/reception, download/upload, etc.) may be carried out in any manner using any medium, using any now known or hereafter developed medium or method.

[**0052**] The library file **74** essentially opens a shell (or a plurality of shells) within the browser **72** that contains the ActiveX control or Plug-in code that may control the Internet browser **72** and the browser interface **20**. When loaded with the ActiveX control or Plug-in, the library file **74** preferably contains functions, objects, data, and other software, referred to generally herein as information, that may be used to control the browser **74** and browser interface **20**. The present invention ensures that the library file **74** (and shell) does not close when the Internet user moves from Internet site **130** to Internet site **230**. Thus, the information and/or functionality provided via the ActiveX control or Plug-in is not lost when the Internet user disconnects from the Internet site that loaded the ActiveX control or Plug-in, and connects to another Internet site.

[**0053**] Referring next to **FIG. 1, a** prior art Internet browser interface **20** having a plurality of windows, each providing various functionality to the Internet user, is there depicted. The browser interface **20** may comprise a first parent window **30** that typically defines the general size, color, and layout of the browser interface and includes window control buttons **28** (e.g., minimize, close, etc.) for that window **30**. The browser interface **20** may also comprise a second parent window **36** (a child to the first parent window) within the first parent window **30**, and one or more child windows **38** dependent from the second parent. The second parent window **36** and child windows **38** typically define information and/or functionality that will assist an Internet user when accessing and navigating the Internet. For example, the second parent **36** and its dependent windows **38** may provide toolbars, pull-down menus, Plug-ins, applications, etc.

[**0054**] For example, three windows **38** provided at the top (in the drawing) of the interface **20** define three toolbars **22**, which may include a variety of interface controls **24** such as, for example, pull-down menus, functional buttons (e.g., stop, back, forward, home, etc.), and a combination of functional buttons and windows (e.g., a search button and window). The uppermost toolbar **22** provides a plurality of pull-down menus **24**; the middle toolbar **22** provides a plurality of functional buttons **24**; and the lowermost toolbar **22** provides a pull-down menu and a window **26** (a URL address window). A content window **32** is also provided as part of the interface **20** within which content from an Internet content provider **100** (see, e.g., **FIG. 10**) may be displayed. The Internet user may toggle any of the lower three (in the drawing) toolbars **22** on and off using a View toolbar object **24** (pull-down menu) provided in the second toolbar **22**. However, the Internet user currently may not add, remove, or otherwise modify the browser interface **20**.

[**0055**] An Internet browser **20** configured in accordance with various embodiments of the present invention is depicted in FIGS. **2-4**. In **FIG. 2**, the browser interface **20** includes an interface object **40** that is defined by the ActiveX control or Plug-in loaded in the library file **74** by the content provider **100**. The interface object **40** comprises a pull-down menu **44** and is displayed in the browser toolbar **22** with the interface controls (i.e., browser toolbar objects) **24** provided by the browser **72**. In **FIG. 3**, an interface object **40** comprises an interface object toolbar **42** and a pull-down menu **44** displayed as a separate window **48** within the browser interface **20**. In **FIG. 4**, an interface object **40** comprises an interface toolbar **42** including a plurality of pull-down menus **44** and a search window **46** displayed within a separate **48** within the browser interface **20**. An interface object **40**, in accordance with the various embodiments of the present invention, may comprise virtually any type of information and/or functionality available via a browser. Thus, by way of non-limiting example, an interface object **40** may comprise a pull-down menu, a toolbar and a pull-down menu, textual information (e.g., advertisements, coupons, etc.), textual and/or aural information (e.g., a textual advertisement with accompanying sound), textual, aural, and/or graphical (animated or not) information, video, video and audio, audio, etc.

[**0056**] The various embodiments of the inventive Internet browser interface **20** depicted by FIGS. **2-4** are merely illustrative, non-limiting examples of the present invention. Variations to the depicted browser interfaces **20** may be possible in accordance with the teachings provided herein.

[0057] Referring next to **FIG. 5, a** method of controlling an Internet browser interface **20** is there depicted in accordance with an embodiment of the present invention, and designated generally as **500**. At step **510**, an Internet user accesses a controlling program **120** via an Internet web site **130**, an ISP **80**, or the user's computer **50**. The controlling program **120**, when executed by an Internet user, provides that user with the ability to thereafter control that user's Internet browser interface **20**, as discussed in more detail below. The controlling program **120**, at step **520**, downloads or creates a library file **74** on the Internet user's computer **50** that includes ActiveX control or Plug-in code that define an interface object. Each time the Internet user launches or activates the browser **72**, the library file **74** is opened and a connection is automatically established to a predetermined Internet web site **110**, one that is preferably configured to communicate to the Internet user's computer **50**, ActiveX control or Plug-in code for the interface object, as indicated at steps **530** and **540**. The open library file **74** essentially provides a shell within the browser **70** within which the functionality provided by the ActiveX control or Plug-in may be added to the browser interface **20**. Neither the library file **74** nor the shell close until the browser **72** is closed. At step **550**, an interface object **40** is created that will be displayed in the browser interface **20**; the information and/or functionality of the interface object **40** is defined by the ActiveX control or Plug-in. In accordance with the various embodiments of the present invention, the interface object **40** remains displayed by the browser **72** with the browser interface **20**, as indicated at step **560**, for as long as the user continues to surf the web, or as long as the browser **72** is activated. Thus, the functionality added to the browser **72** and browser interface **20** in accordance with the present invention is not lost as the Internet user surfs the web.

[0058] The present invention provides various embodiments for controlling the browser **72** and browser interface **20**, described generally above with reference to **FIG. 5**, each of which will now be described in detail.

[0059] Referring next to **FIG. 6** and with continued reference to **FIG. 10, a** description of an embodiment of a method of controlling and displaying a browser **72** and browser interface **20** in accordance with the present invention, designated generally as **600**, will now be provided. For purposes of **FIG. 6** and for the following discussion directed thereto, a library file **74** has already been downloaded or created on the Internet user's computer **50**, as discussed herein. At step **610**, an Internet user launches or activates a browser **72** to initiate access to the Internet **90**. At step **620**, the library file **74** is opened and a connection automatically established to a predetermined content provider, as indicated at step **630**. The functionality defined by the ActiveX control or Plug-in code of the BIO Library is communicated by the content provides to the user's computer **50** to create an interface object **40** which may be added to the browser interface **20**. The interface object **40** is displayed with the Internet browser interface **20**, as indicated in step **640**. The functionality of the interface object **40**, as defined by the ActiveX control or Plug-in code, remains with the Internet browser interface **20** as the Internet user traverses the Internet **90**, regardless of the number or type of Internet sites the user visits, and as long as user is accessing the Internet **90** using the browser software program **72**. When the Internet user moves from one Internet site to another, as indicated at step **650**, the present invention determines whether the interface object **40** has survived that move; whether it is still displayed by the browser **72** in the browser interface **20**, as indicated at step **652**. If the interface object **40** is not displayed in the browser interface **20** (i.e., if it has been removed from the browser interface **20** or otherwise terminated), the interface object **40** is redrawn, as indicated in step **660**. If the interface object **40** has survived a user move from one Internet site to another and remains displayed in the browser interface **20**, the present invention also determines, at step **654**, if the browser **72** is active; since the interface object **40** is only displayed in the browser interface **20** when the browser **72** is operational. If the browser **72** is not activated, the interface object **40** is terminated and the library file **74** is closed, as indicated at step **670**. If the browser **72** remains active, the present invention continues, at step **652**, to ensure that the interface object **40** is displayed with the browser interface **20**. As the Internet user moves between and among Internet sites, the present invention monitors the status of the interface object **40** and ensures that it is displayed by the browser **72** with the browser interface **20** as long as the user is traversing the Internet **90**.

[0060] According to the embodiment depicted in **FIG. 6**, the present invention ensures that the interface object **40** is displayed by the browser **72** with the browser interface **20** when the Internet user leaves the Internet web site **130** at which the BIO Library initially was called and loaded to the user's computer **50**. For example, when the user launches or activates the browser **72**, a connection is automatically established to a first, predetermined Internet site **130** that is maintained by a first content provider **100** and the BIO Library is loaded onto the user's computer **50**. The functionality provided by the BIO Library is then available to the browser **72** via a shell created by the library file **74**. When the Internet user connects to a second Internet site **230** maintained by a second content provider **200**, the functionality for the interface object **40** will continue to be present in the browser interface **20**. In this embodiment, the present invention prevents the browser **72** or operating system **70** of the computer **50** from disabling the functionality of the BIO Library by unloading the library file **74** when the link to the first Internet site **130** is terminated.

[0061] For example, when the browser **72** initially connects to the first Internet site **130**, that site **130** communicates functional information in the form of ActiveX control software code to the Internet user's computer **50** as a BIO Library, which is loaded into the library file **74**. If the library file **74** is not explicitly instructed by the operating system **70** or the browser **72** to close or unload when the connection to the first Internet site **130** is terminated, the library file **74** will remain loaded, providing the desired functionality for the interface object **40** in the browser interface **20**, even after the connection to the first Internet site **130** is closed. Keeping the library file **74** loaded while the Internet user moves between and among various Internet sites enables the loading of data, functions and objects outside of the ActiveX control (which is only communicated to the Internet user **50** by the first Internet site **130**) but inside of the library file **74**. As long as the library file **74** remains loaded, any data or objects created in the library file **74** and outside of the ActiveX control will stay loaded and continue to function in the browser interface **20**.

[0062] To keep the library file **74** open during surfing, after the browser **72** has connected to the first Internet site **130**,

and before that connection is terminated, a global object, object A, is created in the program heap of the Internet user's computer **50**, not the calling function heap. Thus, the global object survives the completion of the calling function. The global object may be created using the C++ new operator, or by declaring a global object in the global declarations. In either case, the global object will survive termination of the connection between the browser **72** and the first Internet site **130**.

[**0063**] A global object thus defined remains functional after the ActiveX control provided by the first Internet site **130** closes, i.e. after the initial connection to the first Internet site **130** is terminated. Once the global object has been created, an interface is created using the global object. That interface will serve to, for example, remove, replace and/or add functionality to the browser **72** and browser interface **20**. The interface may be created as part of the global object, or by the global object allocating a new interface object **40**. For example, the interface object **40** may be created by creating, for example, an interface object window **48** within a browser window **38** (see, e.g., **FIGS. 3 and 4**), and adding it to the browser interface **20** as a child window Alternatively, the browser interface **20** may be directly modified such as, for example, by adding or modifying a browser toolbar **22** or a browser toolbar object **24** in the browser interface **20**. Yet another alternative is to create an object interface toolbar **42** that is separate from the browser interface **20**, as depicted in **FIG. 4**.

[**0064**] Additionally, a pointer is required that is used to control the browser **72** and to instruct the browser **72** to establish a connection to a predetermined Internet site **130**, for example. That pointer is preferably stored globally so that it is accessible by any function or object in the library file **74** that sends commands to the browser **72**. In Microsoft Internet Explorer, for example, the IWebBrowser, IWebBrowser2, or IWebBrowserApp object linking and embedding (OLE) interface commands may be used to create the pointer. Using Microsoft Foundation Class (MFC), for example, the pointer may be created using the GetClientSite member of the COleControl class to retrieve a pointer from the first Internet site **130**, i.e., the Internet site which loaded the BIO Library. The GetClientSite serves as the entry point for the browser **72** to communicate with the BIO Library. A GetContainer member of the IOleClientSite class returned by the previous step may be used to get a pointer to a container for the BIO Library. The BIO Library's container is a container within which an ActiveX control is loaded. An Internet browser interface **20** generally consists of several parts, including the browser toolbars **22** and the content window **32**. A document object is created by the browser **72** for every web page an Internet user accesses and contains all of the data that appears in a particular web page. The document object is also the container for the BIO Library. Thus, a document object may also be referred to as a BIO Library's container.

[**0065**] A QueryInterface member of the IOleContainer class returned by the previous step may be used to get a pointer to the IServiceProvider interface, which may be used to locate any of the other interfaces that are presented by the browser **72**. Finally, a QueryService member of the IServiceProvider class returned by the previous step may be used to get a pointer to the IWebBrowserApp, IWebBrowser, or IWebBrowser2 interface, depending on the specific interfaces provided by the browser **72** that called the BIO Library.

[**0066**] In an alternative embodiment, the present invention may be used to modify the entire browser window **30**. If the entire browser window **30** is modified, as opposed to integrating an interface object **40** into an existing browser window **38**, the entire class for the modified browser window **30** may be subclassed or, alternatively, the specific browser window **38** may be subclassed. As used herein, the term subclassing a window, also referred to as hooking a window, refers to the replacement of an original browser window message handling procedure for handling all messages sent to a window, with a user-defined window message handling procedure. For example, a window may be subclassed using the Microsoft Foundation Class (MFC) CWnd:SubclassWindow( ) function. Alternatively, a window may be subclassed using the call the SetWindowLong function (a Microsoft Windows function), with the GWL_WNDPROC argument (a Microsoft Windows constant). The pointer returned by the SetWindowLong function call may be stored to the original browser window message handling procedure for the subclassed window. This enables the BIO Library to intercept all messages passed to a window **30** or **38**, and the BIO Library may interpret commands from interface controls **24** including buttons, menus, etc., provided by the browser **72** or from interface objects **40** that have been added by the BIO Library in accordance with the present invention.

[**0067**] The user-defined window message handling procedure that the BIO Library provides and that replaces the original browser window message handling procedure is referred to herein as the BIO Procedure. Using the BIO Procedure for the browser **72**, messages (e.g., commands) intended for the browser **72** may be intercepted and modified, or replacement or new messages (e.g., message handlers for the interface object **40**) may be communicated to the browser **72** by the BIO Procedure.

[**0068**] The present invention also ensures that the interface object **40** has not been removed from the browser interface **20**. For example, some Internet browsers **72** redraw the entire browser interface **20** when an Internet user accesses a new web site. While the global object may still be functional following such Internet movement by the user, the interface object **40** will be removed from the browser interface **20** and thus, will not be displayed with the browser interface **20**. To prevent this from occurring, messages from the browser **72** to repaint the browser interface **20** may be intercepted by the BIO Procedure and the interface object **40** may be redrawn immediately after the browser interface **20** is redrawn. Alternatively, the presence of the interface object **40** in the browser interface **20** may be periodically tested and if not present in the browser interface **20**, the interface object **40** may be redrawn. Such periodic testing should preferably occur at intervals of less than approximately one second.

[**0069**] As an example of the above-described embodiment (depicted in **FIG. 6**), an ActiveX control is loaded as a BIO Library and adds menu items (and functionality) to the browser interface **20**. The present invention creates an ActiveX control that dynamically creates a new global object, object A, which creates a new menu object (which may be an interface object **40**) with a desired functionality

to be added to the browser interface **20**. The menu object **40** may be added to the browser interface **20** using, for example, the instructions: AfxGetMainWnd( )→GetMenu( )→AppendMenu( ), where the appended menu of the browser interface **20** would include a popup menu that points to the menu object **40**. The browser interface window would then be subclassed and handle the messages for the menu object **40** handled by the BIO Procedure and the messages for the browser interface **20** being passed to the message handler for the browser **72**. When the Internet user disconnects from the first Internet site **130** (e.g., leaves the web page containing the ActiveX control), the ActiveX control will close, but the global object will remain in the program heap and continue to provide the desired functionality to the browser interface **20**.

[**0070**] Instead of subclassing the browser window within which the interface object **40** is added, object A, or one of its descendants, may retain ownership of the interface object **40**. Then, a message handler for the interface object **40** may be created. For example, an interface object **40** may be added to a browser toolbar **22** in accordance with the above-described embodiment of the present invention, except that ownership of the interface object **40** is set using the Microsoft Foundation Class SetOwner function to be object A or one of it's descendants.

[**0071**] With reference next to **FIG. 7** and continued reference to **FIG. 10, a** description of an alternate embodiment of a method of controlling and displaying a browser interface **20** in accordance with the present invention, designated generally as **700**, will now be provided. For purposes of **FIG. 7** and for the following discussion directed thereto, a library file **74** has already been created on the Internet user's computer **50**, as described herein. At step **710**, an Internet user launches or activates a browser **72** to access the Internet **90**. At step **720**, the library file **74** is opened on the user's computer **50**, and a connection is automatically established to a predetermined Internet site **130**, as indicated at step **730**. At step **740**, the functionality defined by the ActiveX control or Plug-in code of the BIO Library is communicated by the content provider to the user's computer **50** (i.e., to the library file **74**) to create an interface object **40** which may be displayed in the browser interface **20** using a continuous loop to control the display of the interface object **40**. The interface object **40** may only be removed (i.e., its functionality terminated) upon termination of the continuous loop. The functionality of the interface object **40**, as defined by the ActiveX control or Plug-in code, remains with the Internet browser interface **20** as the Internet user traverses the Internet **90**, regardless of the number or type of Internet web sites the user visits, and as long as the browser **72** remains activated and as long as the Internet user is accessing the Internet using that browser software program. When the Internet user moves from one web site to another, as indicated at step **750**, the present invention determines whether the interface object **40** has survived that move, i.e., whether it is still displayed in the browser interface **20**, as indicated at step **752**. If the interface object **40** is not displayed by the browser **72** in the browser interface **20**, the interface object **40** is redrawn, as indicated in step **760**. If the interface object **40** has survived a user move from one web site to another and remains displayed in the browser interface **20**, the present invention also determines, at step **754**, if the browser **72** is active; since the interface object **40** is only displayed by the browser **72** in the browser interface **20** when the browser **72** is operational. If the browser **72** is not activated, the interface object **40** is terminated and the library file **74** is closed, as indicated at step **770**. If the browser **72** remains active, the present invention determined, at step **756**, whether the continuous loop has been terminated. If the loop has been terminated, the interface object **40** is also terminated, as indicated at step **770**. If the loop is still executing, the present invention determines whether the interface object **40** is still displayed in the browser interface **20**, as indicated at step **752**.

[**0072**] The embodiment of the present invention depicted in **FIG. 7** prevents an ActiveX control from unloading by freezing the operation of the library file **74** (within which the ActiveX control code is loaded), even if the operating system **70** or browser **72** generate an instruction to unload or terminate the library file **74**. Because the library file **74** is frozen and never finishes unloading, all of the data, functions and objects created inside of the library file **74** by the ActiveX control will continue to exist and function after the library file **74** has been instructed to unload.

[**0073**] In some operating systems **70** and/or browsers **72**, the BIO Library will be closed when the ActiveX control is no longer present at an Internet site. This can also occur when the Internet user moves from an Internet site having the ActiveX control, e.g., Internet site **130** in **FIG. 10**, to another Internet site that does not, e.g., Internet site **230** in **FIG. 10**. To enable the interface object **40** to continue to operate in the absence of the ActiveX control, the BIO Library (and the library file **74**) must be prevented from closing.

[**0074**] To accomplish this, a pointer is created that is used to control the browser **72**. That pointer is preferably stored globally so that it is accessible by any function or object in the library file **74** that may need to send commands to the browser **72**. In Microsoft Internet Explorer, for example, the IWebBrowser, IWebBrowser2, or IWebBrowserApp OLE Interface commands may be used to create the pointer. To do this using Microsoft Foundation Class, the GetClientSite member of the COleControl class (which may be used to communicate with the BIO Library) may be used to retrieve a pointer to the BIO Library's Internet site, i.e. that Internet site that provides the ActiveX control. A GetContainer member of the IOleClientSite class returned by the previous step may be used to get a pointer to the BIO Library's container. A QueryInterface member of the IOleContainer class returned by the previous step may be used to get a pointer to the IServiceProvider interface. The IServiceProvider interface is used to easily find any of the other interfaces that are presented by the browser **72**. A QueryService member of the IServiceProvider class returned by the previous step may be used to get a pointer to the IWebBrowserApp, IWebBrowser, or IWebBrowser2 interface depending on the interfaces presented by the version of the browser **72** that called the BIO Library.

[**0075**] To prevent the library file **74** from closing, its operation is halted before it is able to terminate. To freeze or halt the operation of the library file **74**, a continuous program loop may be created and executed that terminates only when the BIO Library is to be unloaded, at which time, the program loop also pumps the message queue The program loop is referred to herein as a message pump, and may be created using for example, the PeekMessage, GetMessage,

TranslateMessage and DispatchMessage commands in a loop. Exemplary C++ code to carry out the message pump is provided below:

```
while (m_Continue)
    {
        if (PeekMessage (&msg, NULL, 0, 0, PM_NOREMOVE) != 0)
        {
            GetMessage( &msg, NULL, 0, 0 );
            TranslateMessage( &msg );
            DispatchMessage( &msg );
        }
    }
```

[0076] where m_Continue is a Boolean variable that is used to instruct the loop when to stop, and PeekMessage, GetMessage, TranslateMessage, and DispatchMessage are all Windows functions. If m_Continue equals false, the loop will end, thus ending the message pump. The msg argument is a reference to a Windows MSG structure.

[0077] The way the message pump preferably operates is that it checks to see if there are any messages waiting in the message queue using the PeekMessage function. If there is a message, the message pump grabs the message from the message queue using the GetMessage function and translates it from a virtual-key message into a character message using the TranslateMessage function. Finally, the message pump sends the message to the original window message handling procedure that is due to receive the message using the DispatchMessage function.

[0078] The built in capabilities of an operating system **70** may be also used to construct a message pump to pump the message queue. For example, a modal dialog box or message box, using a command such as the MFC command CWnd::MessageBox("my modal", MB_OK), serves this purpose well and may be used to provided the desired freezing or halting of the operation of the library file **74**.

[0079] As long as the message pump is executing a continuous loop, the ActiveX control will not terminate, even when the Internet user accesses other Internet sites.

[0080] The embodiment of the present invention depicted by **FIG. 7** performs in the same manner (e.g., subclassing, message handling, etc.) as the above-described embodiment of **FIG. 6**.

[0081] For both of the above-described embodiments (of **FIGS. 6 and 7**), it is preferable, although not necessary, to provide an exit function for the interface object **40** so that object A and all of its descendants will be closed. Possible exemplary scenarios for calling an exit function include intercepting the message to close the browser window in the BIO Procedure, or periodically looking for the browser window and, if it is not found, terminating the ActiveX control.

[0082] Referring next to **FIG. 8**, yet another embodiment of the method of the present invention is depicted and designated generally as **800**. For purposes of **FIG. 8** and for the following discussion directed thereto, a library file **74** has already been created on the Internet user's computer **50**, as described herein. At step **810**, an Internet user launches or activates a browser **72** to access the Internet **90**. At step **820**, the library file **74** is opened on the user's computer **50** and a connection is automatically established to a predetermined Internet site **130** (see, e.g., **FIG. 10**), as indicated at step **830**. At step **832**, a new browser interface is created that is a duplicate of the initial browser interface provided by the browser **72**. At step **840**, a Function Window is created that represents the original browser interface within which the functionality of the Plug-in was initially loaded.

[0083] The steps for creating a Function Window in accordance with this embodiment of the present invention are depicted in **FIG. 8A** and designated generally as **840**. At step **842**, the handle for the initial browser interface window is identified. The initial browser interface window, now the Function Window, is hidden and/or disabled at step **844** so that it cannot be closed, which would cause the BIO Library to crash. At step **846**, a pointer is created that is used to control the browser **72**. Finally, a new browser interface window is created at step **848** that may be used by the Internet user to continue traversing the Internet.

[0084] With reference again to **FIG. 8**, and beginning at step **850**, the browser interface may now be controlled by first subclassing any browser windows, or any windows used by the browser, that are to be controlled, and then adding, deleting, and/or modifying the window(s) as described in more detail below. The original browser window message handling procedure is replaced with a BIO Procedure (as defined above). At step **860**, the present invention determines whether any new browser windows have been opened, i.e., windows that may not already be subclassed or that may not have been controlled in accordance with the present invention. If new windows have been opened, the present invention determines, at step **870**, whether the interface object **40** is to be added to those new windows. If not, the invention determines whether the Internet user desires to close the browser **72**, as indicated at step **880**. If so, the invention closes, at step **890**, all browser windows, including the Function Window. If the user does not desire to close the browser **72**, as determined at step **880**, the invention returns to step **860** and again determines if new browser windows have been opened.

[0085] With continued reference to **FIGS. 8, 8A** and **10**, the above-described embodiment of the present invention will now be discussed in more detail. When an ActiveX control is loaded by a content provider **100** via an Internet site **130**, typically in response to a browser **72** establishing a connection to that web site **130** and calling an ActiveX control, a library file **74** located on the user's computer is caused to open creating a shell within the browser **72** within which the code for the ActiveX control may be loaded. If the library file **74** that contains the ActiveX control is explicitly instructed, by the operating system **70** or the browser **72**, to unload or close when the ActiveX control is closed (when the user terminates the connection to the Internet site **130**), any data, functions or objects that have been created outside of the ActiveX control but in the library file **74** will be destroyed when the library file is **74** unloaded. To prevent the library file **74** from unloading, the browser **72** is prevented from closing the ActiveX control until instructed. If the ActiveX control is never instructed to close, the library file **74** is never unloaded.

[0086] A preferred method of accomplishing this is to hide and/or disable the initial browser interface window that

loaded the ActiveX control and to create a new copy of that same window within which the Internet user may continue to work and traverse the Internet. Because the initial browser interface is preferably hidden and/or disabled, the ActiveX control cannot be closed until the library file **74** shows, enables or closes the initial browser interface window, i.e. the window that loaded the library file **74**.

[**0087**] The above-described method of **FIG. 8** preferably loads the BIO Library as a standard ActiveX control in a browser **74**, using, for example, the <Object> tag typically contained in a web page **130** or as a Band Object, and as described in the Microsoft Internet Explorer 4.x Software Development Kit. This instructs the browser **72** to initialize and run the library file **74** that contains the code for BIO Library.

[**0088**] The first time the BIO Library is initialized and called, a Function Window is created that keeps the BIO Library open by keeping a session with the ActiveX control itself open while the Internet user visits other Internet web sites **210**, i.e., other web pages. The Function Window also makes it possible for browser windows that do not have a copy of the BIO Library open to access the OLE interfaces to the browser **72**.

[**0089**] To create the Function Window, the initial browser interface window (i.e., that window which loaded the BIO Library) is preferably hidden and/or disabled. This may be accomplished by identifying the handle of the initial browser interface window, beginning with the handle of the BIO Library. To retrieve the initial browser interface window handle from the handle of the BIO Library, the GetParent function (a Windows Function) is continuously called until the present value for that function call represents one level below the desktop window. For example, a statement such as "m_Handle=GetParent(m_Handle)" executed in a loop may provide the desired functionality and result, where the value of m_Handle is initially equal to the value for the handle of the BIO Library, and will eventually return the handle to the initial browser interface window.

[**0090**] The next step is to hide and/or disable the initial browser interface window, now referred to as the Function Window, so that the Internet user cannot close the Function Window (by closing the browser **72**, for example), thereby causing the BIO Library to unload and removing the functionality provided by the BIO Library from the browser interface **20**. To hide the initial browser interface window from the user and/or disable that browser window from user-driven events, WM_SHOWWINDOW and/or WM_ENABLE messages (both Windows constants) may be sent to the initial browser interface window with values to hide and/or disable the browser window. For example, the PostMessage or SendMessage function (existing Windows functions) may be used to send a message to the initial browser interface window with the browser window handle. Alternatively, the ShowWindow and EnableWindow functions (existing Windows functions) may be used to achieve the same result.

[**0091**] A pointer is created to control the browser **72**. This pointer is preferably stored globally so that it is accessible by any function or object in the library file **74** that sends commands to the browser **72**. In Microsoft Internet Explorer for example, the IWebBrowser, IWebBrowser2, or IWebBrowserApp OLE interface may be used to create the pointer. To do this using Microsoft Foundation Class for example, the GetClientSite member of the COlecontrol class that serves as the entry point for the browser **72** may be used to communicate with the BIO Library, and to retrieve a pointer to the BIO Library's Internet site, i.e., that Internet site **130** that loaded the ActiveX control. A GetContainer member of the IOleClientSite class returned by the previous step may be used to get a pointer to the BIO Library's container. A QueryInterface member of the IOleContainer class returned by the previous step may be used to get a pointer to the IServiceProvider interface; with the IServiceProvider interface preferably being used to find any of the other interfaces that are presented by the browser **72**. A QueryService member of the IServiceProvider class by in the previous step may be used to get a pointer to the IWebBrowserApp, IWebBrowser, or IWebBrowser2 interface depending on the interfaces presented by the version of the browser **72** that called the BIO Library.

[**0092**] Finally, a new browser window is created that the Internet user may use to continue surfing the web and continue accessing various different Internet sites, since the browser window previously used to create the Function Window has been hidden and/or disabled. Preferably, any of the IWebBrowser, IWebBrowser2, or IWebBrowserApp OLE interface is used to create a new browser window, for example, using the Navigate or Navigate2 members of that OLE interface. Alternatively, a WM_COMMAND message that corresponds to any command the browser **72** might use to open a new browser window such as a New Window command or Open In New Window command, etc., may be sent to the browser **72**. A new window may also be opened using the Dynamic Data Exchange (DDE) support provided by the browser **72**.

[**0093**] The BIO Library must now control various browser interface features and functionality. The first step is to subclass any of the browser windows or any of the windows the browser uses (i.e., children) that are to be controlled in accordance with the present invention. A BIO window message handling procedure is used to replace the original window message handling procedure, and is hereinafter referred to as the BIO Procedure.

[**0094**] Once the browser window, or any of it's children, have been subclassed, it is possible to add menus to a subclass by retrieving a pointer to the browser window menu using the GetMenu function. Once the pointer to the menu's handle is obtained, the menu functions such as ModifyMenu, AppendMenu, InsertMenu, etc., may be used to add any desired menus to the browser window. Any commands assigned to a menu must be handled by the BIO Procedure used to subclass the BIO window. The same command identifier must not be used in creating a menu as any that are included in the browser.

[**0095**] An interface object toolbar **42** may be added to the browser interface **20** by retrieving the handle of the window to which the toolbar **42** is to be added to (hereinafter referred to as the Frame Handle) using standard Windows functions. Typically, the window will be the BIO window or a frame window that is a child of the BIO window. A window is then created using the Frame Handle as its parent. For example, to add a dialog bar (which is a form of a toolbar) as an interface object **40**, an object derived from or of a type CDialogBar (a Microsoft Foundation Class) is created and

it's Create method called using Frame Handle. If resources such as, for example, images, toolbars, dialogs, etc., are being used and the browser **72** does not share the same resources as the BIO Library, the browser's resources are temporarily replaced with the BIO Library resources before any data may be loaded from the BIO Library resource. The BIO Library resources may then be replaced with the browser's original resources.

[**0096**] As new browser windows are opened, it may be desirable to add interface object(s) **40** to those new windows. A timer may be created using SetTimer Windows function that would call a user-defined function and that function would use the FindWindowEx function (a Windows function) to check every child of the desktop window for a window with the same class name as the Function Window. For those browser windows that have not already been modified, i.e., that do not have the interface object **40**, the necessary handles may be retrieved and the same changes made as were made for the original BIO Window.

[**0097**] Finally, when the Internet user desires to close the browser **72**, it must be determined if all of the browser windows are closed, except the Function Window, and the Function Window must also be closed if all other browser windows are closed. This may be accomplished by listening for a WM_CLOSE message (a Windows constant) in the BIO message handling procedure or by setting a timer that periodically determines how many browser windows are open. To close the original Function Window, a WM_CLOSE message may be sent to that window.

[**0098**] Referring next to **FIG. 9**, another alternative embodiment of a method of controlling and displaying an Internet browser interface **20** in accordance with the present invention is depicted and generally designated as **900**.

[**0099**] Steps **910**, **920**, and **930** are essentially the same as described above for the embodiments of FIGS. **6-8**. At step **940**, a new browser interface window is created and the initial browser interface window is hidden and/or disabled, and referred to as a Function Window. The Plug-in identifies the handle for the initial browser interface window, hides and/or disables that window, and creates a new browser interface window that may be used by the Internet user. At step **950**, all the browser windows are subclassed, and then the browser interface may be controlled, as indicated at step **960**, for all open windows. At step **962**, the present invention determines if any new browser windows have been opened, in which case the invention returns to step **960**. In no new browser interface windows have been opened, step **962** proceeds to step **964** to determine if the Internet user desires to close the browser. All windows must be closed prior to closing the Function Window, and that is determined at step **966**. If all windows are closed, the Function Window is closed, as indicated at step **970**. Otherwise, step **966** returns to step **964**.

[**0100**] In yet another alternative embodiment of the present invention, the present invention provides a method of controlling an Internet browser interface using a browser Plug-in to control the functionality of the calling browser **72** and to retain the Plug-in functionality after the user leaves the Internet site **130** that loaded the Plug-in.

[**0101**] When a browser Plug-in is loaded to an Internet user's computer **50** in response to a browser **72** establishing a connection to an Internet site **130** and calling the Plug-in, a library file **74** establishes a shell within the browser **72** and within which the code for the Plug-in may be loaded. If the library file **74** is explicitly instructed by the operating system **70** or the browser **72** to unload when the Plug-in is closed, any data, functions or objects that have been created outside of the Plug-in but in the library file **74**, will be destroyed when the library file **74** is unloaded. To prevent the library file **74** from unloading, the browser **72** is prevented from closing the Plug-in until the browser **72** receives an instruction to close the Plug-in. If the Plug-in is never instructed to close, the library file **74** also is never instructed to unload. This may be accomplished by hiding and/or disabling the initial browser window that loaded the Plug-in and by creating a new copy of that same window for the Internet user to continue to use to access and traverse the Internet. Because the initial browser window is preferably hidden and/or disabled, the Plug-in can not be closed until the library file **74** chooses to show, enable or close the initial browser window that loaded the Plug-in.

[**0102**] For example, a browser Plug-in is loaded as a standard Plug-in in a browser **72**, preferably by using the <Embed> tag in a web page **110** (see e.g., **FIG. 10**), which instructs the browser **72** to initialize and load the library file **74** that contains the code for the Plug-in (i.e., the BIO Library).

[**0103**] The first time the BIO Library is initialized and called, a Function Window is created by hiding and/or disabling the original browser window, thus preventing the BIO Library from unloading by keeping a session with the Plug-in itself open. The Function Window also makes it possible for browser windows that do not have a copy of the BIO Library open to access built-in Application Programming Interfaces (APIs) for Plug-ins (i.e., functionality made available to the Internet user through the browser interface **20** and via the Plug-in functionality), such as those provided with Netscape Navigator and Microsoft Internet Explorer.

[**0104**] A Function Window is preferably created by disabling or hiding the initial browser window (in which the Plug-in resides) the first time the Plug-in is called by the browser **72**. To do this the Plug-in must first identify the handle of the initial browser window. A window member of the NPWindow structure is passed to the BIO Library from the browser **72** as a second argument to the NPP_SetWindow Function and is the handle to the Plug-in window (NPWindow and NPP_SetWindow are part of the Netscape and Internet Explorer API for Plug-Ins). To retrieve the initial browser window handle from this window member, the GetParent function (a Windows function) is continuously called until the present value for that function call represents one level below the desktop window. For example, a statement such as "m_Handle=GetParent(m_Handle)" executed in a loop may provide the desired functionality and result, where the value of m_Handle is initially equal to the value for the handle of the NPWindow structure, and will eventually return the handle for the initial browser window.

[**0105**] The initial browser window is then hidden and/or disabled so that the Internet user cannot close the Function Window and cause the BIO Library to crash. To hide the initial browser window from the user and/or disable that window from user-driven events, WM_SHOWWINDOW and/or WM_ENABLE messages (both Windows constants)

may be sent to the initial browser window with values to hide and/or disable that browser window. This can be accomplished by, for example, using the PostMessage or SendMessage function (Windows functions) to send a message to the initial browser window using the browser window handle. Alternatively, the ShowWindow and EnableWindow functions (Windows functions) may be used to achieve the same results.

[0106] The final step is to create a new browser window that the Internet user can use to continue surfing the web after the initial browser window has been hidden and/or disabled. This may be accomplished, for example, by calling any of the following Netscape and Internet Explorer Plug-in API functions: NPN_GetURL, NPN_PostURL, NPN_GetURLNotify, NPN_PostURLNotify, with the target parameter set to _new, _blank, or any window name that does not already exist. The NPP argument of the above functions is the NPP structure that was provided by the browser **72** to the Plug-in for the Function Window. Another way of doing this is to send a WM_COMMAND message to the browser **72** that corresponds to any command the browser **72** might use to open a new window such as a New Window command or Open In New Window command, for example. A new window may also be opened using the Dynamic Data Exchange (DDE) Support provided by the browser. NPN_GetURL, NPN_PostURL, NPN_GetURL-Notify, NPN_PostURLNotify and the NPP structure are part of the Netscape and Internet Explorer API for Plug-Ins and the WM_COMMAND is a Windows constant.

[0107] The BIO Plug-in may now control features and functionality of the browser **72**. The first step is to subclass any of the browser windows or any of the windows the browser uses (collectively, BIO windows) that are to be controlled in accordance with the present invention.

[0108] After the browser window or any of it's children has been subclassed, menus may be added to the browser interface **20** by retrieving a pointer to the browser window menu using the GetMenu function (a Windows function). Once the pointer to the menu's handle is obtained, the menu functions such as ModifyMenu, AppendMenu, InsertMenu, etc. (Windows functions), may be used to add any desired menus to the browser window. Any commands assigned to a menu must be handled by the BIO message handling procedure used to subclass the BIO window, taking care not to use the same command identifier in creating a menu as any included in the browser **72**.

[0109] Alternatively or additionally, an interface object toolbar **42** may be added to the browser interface **20** by retrieving the handle of the window to which the toolbar **42** is to be added (hereinafter referred to as the Frame Handle) using standard windows functions—typically the window will be the BIO window or a frame window that is a child of the BIO window. A window is then created using the Frame Handle as its parent. For example, to add a dialog bar (which is a form of a toolbar) as an interface object **40**, an object derived from or of type CDialogBar (a Microsoft Foundation Class) may be created and it's Create method called using the Frame Handle. If resources such as, for example, images, toolbars, dialogs, etc., are being used and the browser **72** does not share the same resources as the BIO Library, the browser's resources are temporarily replaced with the BIO Library resources before any data from the BIO Library resource may be loaded. The BIO Library resources may then be replaced with the browser's original resources.

[0110] As new browser windows are opened, the interface object **40** may be added to those new windows. This may be accomplished by creating a timer using SetTimer (a Windows function) that would call a user-defined function that would use the FindWindowEx function (a Windows function) to check every child of the desktop window for a window with the same class name as the Function Window. For those new windows that do not already have the modified interface, i.e., that do not include the interface object **40**, the necessary handles are retrieved and the same changes made to those windows as were made to the original BIO window.

[0111] Finally, when the Internet user wishes to close the browser **72**, it must be determined if all of the browser windows are closed except the Function Window, and if they are, the Function Window may be closed. This may be accomplished, for example, by listening for a WM_CLOSE message (a Windows constant) in the BIO window message handling procedure or by setting a timer that periodically checks the number of open browser windows. The original Function Window may be closed by sending it a WM_CLOSE message.

[0112] In accordance with the present invention, a BIO Library (i.e., a Plug-in) may be loaded and its functionality provided in the browser interface **20**, automatically, i.e., without requiring the user to positively access a particular Internet site, i.e., to surf to a web page that calls the Plug-in. For example, Netscape has a key in its windows registry identified as Automation Startup. Upon activation, Netscape loads all of the OLE controls that are listed in the Automation Startup key. By placing a reference or a call to the library file **74** (and thus to BIO Library and Plug-in that defines an interface object **40**) in the Automation Startup key, the library file may be loaded every time a user launches or activates a Netscape browser. Included in that library file **74** may be instructions to create an instance of the interface object **40** in the browser interface **20** and an instruction for the browser **72** to establish a connection to a predetermined Internet site **130**. Using this technique, a user does not have to choose to visit a specific Internet site **130** to load a BIO Library. The library file **74** needs to be kept open at least until the Plug-in may be loaded in the browser **72** for display and access via the browser interface **20**. One way to do this is to increment a reference counter associated with the library file **74** so that when Netscape unload the OLE controls listed in the Automation Startup key, the library file **74** will not be unloaded because it has a higher reference number.

[0113] The library file **74** may be loaded as a Plug-in using DDE to periodically look for a Netscape DDE Server using a timer or a loop. When a return is received by the browser **72** from the Netscape DDE server, Netscape is ready to receive commands and may be loaded with the Plug-in. DDE may then be used to send a command, such as WWW_OPENURL, to the browser **72**, which will cause the Plug-in to load as discussed herein.

[0114] Another method for hiding the Netscape Plug-in window that is used for BIO Library is to remove it from the task bar (i.e., where the Windows "Start" button is located)

and place it off of the visible screen. One way to remove it from the Task Bar is to call SetWindowLong and change the window style of the Plug-in window to a toolbox window. Toolbox window's do not appear in the task bar. The Netscape Plug-in window may be placed off screen by calling MoveWindow and providing coordinates that are not in the visible range for the users desktop.

[0115] The BIO window message handling procedure that is used to replace the original browser message handling procedure must know which window a message is intended to reach and what to do with a message once the BIO window message handling procedure receives that message. A preferred way to do this is to create a map that links one piece of information to another. For the present invention, a map that links window handles to structures that contain important information to that window is preferably used. For example when the BIO Library adds the interface object **40** to a new browser window, a new entry in the map is created that links the BIO window's handle to a structure that contains information useful for that BIO window. One of the pieces of information contained in the structure would preferably be the browser's original window message handling procedure for the BIO window. It is necessary to maintain the browser's original window message handling procedure so that if the BIO window message handling procedure does not know how to handle a message, it can pass the message to the browser's original window message handling procedure.

[0116] When a message is received by the BIO window message handling procedure, the first argument that is passed to the procedure is the handle of the window that received the message. To retrieve the structure that contains all of the data specific to that window, a lookup in the map is performed using the window handle as a key. The returned structure will contain all of the stored window specific information, such as the original window message handling procedure.

[0117] When controlling the browser interface **20**, the present invention changes how the browser **72** works. Almost anything an Internet user can do with a browser works by sending a message to a browser's window or child window. Objects or windows that send messages include the menus, buttons, combo boxes and almost anything else with which the Internet user can directly interact, i.e., interface controls. Messages for example can be broken up into four components: 1) the handle of the window intended to receive the message; 2) the msg value of the message; 3) a wParam, whose use is usually dependent on the msg value; and 4) an IParam, whose value is also usually dependent on the value of the msg value.

[0118] For example, clicking on a button in a browser's window might send a message that contains the WM_COMMAND, which is a Windows constant, msg value to a browser window's window message handling procedure. The lower two bytes of the wParam variable in that message would then be a number that is used to identify which button was pressed.

[0119] By subclassing a browser's window or child window, as described above, any messages that are sent when a user interacts with any of the interface controls may be intercepted. Once a message is intercepted, the BIO window message handling procedure can interpret it and react to it. If the functionality of the interface control is to remain the same (i.e., not added to, deleted from, or modified by the present invention), the message may be passed back to the original window message handling procedure. In this way, virtually all of the interface controls that exist in the browser **72** may be controlled. In addition, interface controls may be added to the browser interface **20** and assigned command identifiers (which are passed in the wParam). The BIO window message handling procedure can then interpret the wParam and provide the functionality of the Interface control that is to be added. In addition, functionality may be removed by simply having the inventive window procedure do nothing if it receives a command identifier associated with an interface control that is to be removed from the browser. That command may thus be prevented from passing to the browser window's original message handling procedure.

[0120] Using the various embodiment of the present invention, as discussed in detail above, an Internet user may create a browser interface **20** having user-defined interface controls. Then, by setting the parent of the window for that browser interface **20** to a window that has been subclassed, any message from the new (i.e., controlled) window will be handled in the BIO window message handling procedure. This can be used to add any interface object **40** such as a toolbar, dialogbar, floating dialog etc., to the browser interface **20**.

[0121] The following illustrative, non-limiting application examples are provided to further describe the present invention. A Plug-in or ActiveX control that stays persistent as an Internet user traverses the Internet may add an interface object **40** to the browser interface **20** that enables a user to download their "bookmarks" or "favorites" from a database located on the Internet. The interface will be added directly into the browser interface and will allow the user to visit the "bookmarks" or "favorites" links that they downloaded, using the interface object **40** provided by the Plug-in or ActiveX control. This interface object **40** will serve a similar function to the current "favorites" or "bookmarks" menu items and toolbars on existing browsers. The beneficial difference is that since the bookmarks will be downloaded from a database on the Internet, users have access to their bookmarks on any computer's browser that is capable of loading the Plug-in or ActiveX control.

[0122] The present invention may also be used to generate revenue based on placing advertising "links" included in the "favorites" or "bookmarks" on the browser interface **20** via the Plug-in or ActiveX control. Consumer targeting could be based on, for example, information stored in databases, such as name, age, sex, income, race, education and geographic location, and preferences such as favorites and bookmarks or other preferences that are stored on the database or exist on the browser **72**.

[0123] A Plug-in or ActiveX control that stays persistent as the user traverses the Internet may be used to add an interface object **40** to the browser interface **20** that permits a user to download their "address book" from of a database located on the Internet. Such an interface object **40** may be added directly into the interface of the browser interface **20** and will allow the user to send e-mail as well as retrieve stored information for contacts listed in their "address book".

[0124] The present invention may also be used to earn revenue based on placing advertising "links" included in the "address book" on an interface object **40** of the Plug-in or ActiveX control. Consumer targeting could be based for example on information stored in databases, such as name, age, sex, income, race, education and geographic location, and preferences such as favorites and bookmarks or other preferences that are stored on the database or exist on the browser.

[0125] The present invention may use a Plug-in or ActiveX control to add an edit box on the browser interface **20** that allows a user to type a search directly into the browser instead of having to visit a web page that allows the user to search.

[0126] Additionally, the Plug-in or ActiveX control that stays persistent may poll, or periodically seek—at user, web site or program selected intervals—information from a preferred web site, even though the user is surfing a different web site. As the preferred web site is polled, the preferred web site can send updated information to the interface object on the user's browser, such as near real-time notification of the receipt of mail, continuous updating of stock prices, or other time sensitive information, such as, for example, news feed headlines, sports scores for selected favorite sports teams, and the like. The preferred web site can control, if desired, the timing of the polling, so as to control the traffic at the preferred web site during peak usage periods by extending the time interval between polls.

[0127] Since the shell created by the library file, as described herein, is an environment within which applications can be run, or information displayed, any information or program can be added to the interface of the browser using the present invention. The shell is independent of the browser interface, the content of the browser, and even the content of the shell itself. In short, the shell is an adaptable piece of functionality that does not, in the extreme, even need to be visible to the user. Thus, in use, the shell can be empty and receive its contents from a web site, or the shell could get Plug-ins, or the shell could even get new library files and learn to parse new information "on the fly" as the shell receives new contents from a web site or user. Thus the present invention provides significant opportunities to direct desired information from a preferred site to the user even as the user visits other sites. Of course, the more user-specific functionality provided by a web site via the customizable interface of the present invention, the greater user loyalty that web site can engender.

[0128] In another embodiment of the present invention, and with reference now to FIGS. **11-17**, a shopping assistant button may be added to a toolbar of the Internet browser interface to facilitate on-line shopping at a supported merchant web site. Functionality of this embodiment, including defining the shopping assistant button, is provided by computer code transmitted from a server and stored on the user's computer. The computer code monitors the Internet navigation of the Internet browser to determine if the browser is at a supported merchant web site, provides an indicator for the shopping assistant button when the Internet browser is at a supported merchant web site, and fills out a supported merchant check-out web page.

[0129] As user herein, the term "on-line shopping" refers to the process by which a user of an Internet browser my purchase merchandise and/or services over the Internet using a computer. An on-line merchant (e.g., a seller of merchandise and/or services over the Internet) may provide a web site hosted by a merchant server via which a user may access one or more web pages using an Internet browser. Various merchandise and/or services may be offered for sale by the merchant via the one or more web pages. A shopper may view the merchandise, description of the services, select various merchandise and/or services for placement into an electronic shopping cart, view the contents of the shopping cart, and check-out (as described in more detail below).

[0130] The browser interface **20** depicted in **FIG. 11** includes an interface object **40** comprising an interface object toolbar **42** and a shopping assistant button **144** that provides various functionality to the browser interface **20**, including, by way of non-limiting example, the ability for a user (the terms user and shopper being used interchangeably herein) of the browser interface **20** to create and edit an electronic wallet (discussed in more detail below), disable the shopping assistant button **144** (and shopping functionality from the browser interface **20**), list supported merchants, query about the shopping assistant button **144**, pull-down menu **44**, and functionality provided by the shopping assistant button **144**, and query for help about use of the shopping assistant button **144** and the functionality provided thereby. Shopping assistant computer code or software (e.g., a .dll or .exe. file, javascript, etc.) is provided by a server and stored on the user's computer **50** and operable in connection with the browser and browser interface **20** to define and provide the shopping assistant button **144** and other functionality provided in accordance with the present invention, and as described in more detail below.

[0131] In a preferred embodiment, the shopping assistant button **144** and functionality provided thereby is provided when the interface object **40** is added to the browser interface **20**, as described in detail above with regard to the various embodiments of the present invention. That is, when the interface object **40** is added to the browser interface **20**, the shopping assistant button **144** and associated computer code are transmitted or downloaded to the user's computer **50**. Alternatively, the shopping assistant button **144** and functionality may be selectively added to the browser interface **20** upon request by the user, i.e., before or after the interface object **40** is added.

[0132] The shopping assistant functionality provided in accordance with this embodiment of the present invention is provided, at least in the first instance, by a server **102** (which may comprise one or more computers, located physically proximate each other, or physically separate from each other) (see, e.g., **FIG. 10**) that may be accessed when the user provides a predetermined Internet address or URL in the URL address window **26** of the browser interface **20**. The server **102** is preferably a secure server (i.e., one that is only accessible via a Secure Socket Layer (SSL) connection). At that predetermined Internet address, the user may access one or more web pages, at least one of which enables a user to set up a wallet in accordance with this embodiment of the present invention. Exemplary wallet set-up web pages are depicted in FIGS. **12-14** and discussed in more detail below.

[0133] The shopping assistant functionality is also provided by shopping assistant computer code or software

downloaded by the server **102** to the user's computer **50** and operable in connection with the browser interface **20** and shopping assistant button **144**. The shopping assistant code may be a .dll or .exe file, for example, javascript, or other known types of computer code or software files. The shopping assistant code monitors the Internet navigation of the Internet browser by intercepting the Internet address (or domain or URL) each time the user causes the browser to navigate to a different Internet address. The shopping assistant code also compares the intercepted Internet address with the Internet addresses of supported merchants by comparing the intercepted address with a supported merchant file containing Internet addresses of supported merchants. The supported merchant file is preferably downloaded by the server **102** and stored on the user's computer **50**. Each time a user logs into the server **102**, the server **102** compares its version of the supported merchant file with the version stored on the user's computer **50** (that version information may me transmitted by the user during the login process, for example). The server **102** downloads to the user's computer an updated version of the supported merchant file, if necessary. The shopping assistant code also intercepts each web page received by the browser and determines the type of web page (e.g., billing web page, merchant home page, merchandise page, etc.) by the HTML code and http request headers.

[**0134**] The shopping assistant code also defines the shopping assistant button **144** and pull-down menu **44**, and controls the communication between the browser, the server **102** and the merchant server for on-line shopping.

[**0135**] The shopping assistant button **144** may be provided as part of the interface object **40** in accordance with the various embodiments of the present invention. However, for a user to utilize and access the shopping assistant functionality, the user must first set up a wallet. As used herein, the term "wallet" refers to a file (that may be encoded or encrypted) particular to a user and comprised of information (also referred to herein as data) specific to that user, provided by that user, and stored in a wallet database **104** provided on a data storage device **106** (e.g., hard drive, optical disk, etc.) of the server **102** (see, e.g., **FIG. 10**). The wallet database **106** is structured so that the information provided by the user when setting up the wallet is stored in predefined fields or field names. Similarly, a supported merchant data file is structured using the same fields or field names, to the extent that the merchant check-out web page(s) require the same information as was provided during the wallet set-up process.

[**0136**] To set up a wallet, the user must provide certain information in response to certain wallet set-up web pages provided by the server **102**. It is known to persons skilled in the art that an Internet browser receives data from a server, typically HTML data, for display by the browser interface **20**. Thus, reference herein to web pages being provided by a server refers to the download or transmission of data from the server for display by an Internet browser.

[**0137**] An exemplary wallet set-up web page is depicted in **FIG. 12** and generally designated as **1000**. That web page **1000** may be accessed, for example, via the "edit wallet" option on the pull-down menu **44** of the shopping assistant button **144**. By selecting a predetermined link on the wallet set-up web page **1000**, such as, for example, a "Set up your Wallet" link **1002**, a wallet data entry web page **1010**, such as is depicted in **FIG. 13**, is downloaded by the server **102**. At the wallet data entry web page **1010**, the user enters certain information in a plurality of fields **1012** (either pull-down menu or alpha-numeric entry). The information may be, by way of non-limiting example, credit card type, number, expiration date, user's first and last name, billing address, phone number, user ID, user password, and various other information particular to the user. When setting up a wallet, each user is also required to enter a security key (see, e.g., **FIG. 16**) which is necessary, in addition to the user's ID and password, to access and use the user's wallet. When the user has completely filled out all required fields **1012** in the wallet data entry web page **1010**, the user may select a "Finished" button (not shown), which transmits the user-entered information to the server **102**. In response, the server **102** downloads a wallet summary web page **1020** that includes a summary **1022** of the user's newly entered account information for review by the user prior to finally setting up the user's wallet, as depicted in **FIG. 14**. If the user information is correct, the user may select a link on the wallet summary web page **1020** such as, for example, a "Click here to continue . . ." link **1024**, that causes the user information (data) to be transmitted to the server **102** and stored in the wallet database **104** on the data storage device **106**. During the wallet set-up procedure just described, data is being transmitted between the server **102** and the user's computer **50** (see, e.g., **FIG. 10**). For example, web pages **1000** (**FIG. 12**) and **1010** (**FIG. 11**) may both be transmitted by the server **102** for display by the user's browser interface **20** when the user selects the "Set up your wallet" link **1002**. When the user has completed the information required on the wallet data entry web page **1010** and selects the "Finished" button (not shown), the user information or data may be transmitted to the server **102**. Upon receipt of the user information from the user's computer **50**, the server **102** may transmit the wallet summary web page **1020**. Alternatively, each of the above-described web pages may be transmitted separately by the server **102** and in response to a user-initiated action (e.g., upon selection by the user of a link or button).

[**0138**] The server **102** may be that of a content provider **100** or other Internet service provider such as, for example, a search engine provider or ISP, collectively referred to herein as the provider **100**. On-line merchants are selected by the provider **100** in connection with which the shopping functionality of the present invention may be used. A supported merchant file contains a list of all supported merchants, including their respective Internet addresses. That file may be used by the shopping assistant computer code to determine if a user has navigated to a supported merchant web site by comparing the URL entered into the URL address window **26** (or provided in a link) with the urls contained in the supported merchant file.

[**0139**] Once a merchant (a merchant is generally designated **200** in **FIG. 10**) has been selected, the provider **100** determines the merchants' check-out process, including the layout of check-out web page(s) (see, e.g., **FIG. 17**), and information required to complete the merchant's check-out process and check-out web page(s). The provider **100** creates a supported merchant rules and mapping file for each selected merchant (also referred to herein as a supported merchant) that defines a plurality of merchant field names that correspond to the user data required by the merchant during its check-out processes and its check-out web

page(s). For example, and with reference to **FIG. 17, a** supported merchant check-out web page **2100** requires that a shopper provide billing information including, for example, payment method, credit card number, expiration date, and personal information for the shopper (e.g., name, address and telephone number). The merchant field names correspond to wallet field names defined in the user's wallet. Each supported merchant rules and mapping file may contain different merchant field names, as each merchant may require different information during its check-out process. However, all merchant field names correspond to wallet field names.

[**0140**] The supported merchant rules and mapping file maps the wallet data (i.e., wallet field names) to the required merchant data (i.e., merchant field names). For example, an entry in the merchant rules and mapping file defining a first name field and last name field may be structured as follows:

[**0141**] rmi_pageFields[1]=new rmi_fieldMap("contactFirstName", "text", "wallet_b_fname_first");

[**0142**] rmi_pageFields[2]=new rmi_fieldMap("contactLastName", "text", "wallet_b_fname_last");

[**0143**] where "wallet_b_fname_first" is the wallet database field name for the user's first name.

[**0144**] However, in the supported merchant file, the user's first name is identified as "contactFirstName". Thus, the present invention maps the fields in the wallet database to the fields in the supported merchant file so that each merchants order form may be automatically filled out when a user shops at a supported merchant web site.

[**0145**] With reference now to FIGS. **15-17**, an exemplary check-out process will now be described. As different merchants may have different check-out processes and provide different check-out web pages, the following description is merely illustrative and not intended to limit the scope of or otherwise define the present invention. In addition, the following description does not track the entire on-line shopping process (which may also vary from merchant to merchant), but only the check-out process, which generally begins after the user has selected a "Check Out" or other similar button on a merchant web page.

[**0146**] When a user launches their Internet browser, the interface object **40** automatically establishes a connection between the user's computer **50** and the server **102**. Various user data is downloaded by the server **102** to the user's computer **50**. The user may cause the Internet browser to navigate to any Internet address by typing the address in the URL address window **26** (see, e.g., **FIG. 11**), or by selecting a link on a web page. Regardless of how the browser is caused to navigate the Internet, the shopping assistant code intercepts the URL for each Internet site navigated to by the user, and compares that URL with the supported merchant data file. If a match is found, the shopping assistant code provides a user perceptible indicator on the shopping assistant button **144** such as, for example, a yellow circle, that indicates that the user has navigated to a supported merchant web site. The shopping assistant code also makes a request to the server **102** for a download of a file containing rules and mapping data for that merchant and that provides details on that merchant's check-out process.

[**0147**] The shopping assistant code also intercepts each web page received by the browser to determine the type of web page, e.g., a billing page, e-mail address request, credit card information request, etc., by looking at the HTML code and the http request headers. That information was obtained during the provider's selection of merchants and creation of the supported merchant data files. Using the merchant's rules and mapping file, the shopping assistant code can determine the specific page being served and how the user's information should be provided in that page.

[**0148**] If the shopping assistant code determines that the user's wallet is required for a particular page, it opens a security key window **3000** in the user's browser interface **20** asking the user to enter a security key (SK) in a data entry window **3020**, as depicted in **FIG. 15**. Once the user has entered a SK and selected a "Continue" button **3030**, that security key is securely transmitted, preferably using https, by the user's computer **50** to the server **102**. The server **102** compares the user data (e.g., ID and SK) with data previously stored in that user's wallet on the wallet database **106**. If the data matches, the SK is verified and an authorization is transmitted by the server **102** to the shopping assistant code; a rejection is transmitted if the data does not match. The authorization may be provided as a wallet information window **3100**, such as is depicted in **FIG. 16**. When a SK is successfully verified by the server **102**, a "secure cookie" is transmitted over SSL to the secure server, and the user's on-line shopping session using his/her wallet begins when the "secure cookie" is set. The "secure cookie" is aggressively timed out, and is preferably valid for up to one (1) hour.

[**0149**] If the user desires to continue the check-out process using the shopping assistant functionality of the present invention, the user may select a "Continue" button **3130**. If the user desires to check-out using the merchant's check-out process, the user may select a "Cancel" button **3140**. When the user selects the Continue button **3130**, the shopping assistant code retrieves the user's wallet from the wallet database **104** on the server **102**, and automatically fills in the merchant's check-out web page **2100**, depicted in **FIG. 17**. The user's wallet is securely transmitted by the server **102**, preferably using https. The user's wallet, and thus the user's personal data, need not be stored on the user's data storage device **52**, but only in temporary memory (also referred to herein as main memory or RAM) **60**, and used by and in connection with the shopping assistant code and functionality of this embodiment of the present invention. The user's data may automatically expire, or it may be flushed from memory **60** when the user closes the browser.

[**0150**] Depending upon the type of web page displayed by the browser, the shopping assistant code provides the appropriate user data from the user's wallet (temporarily data in RAM) to the web page. For example, if the shopping assistant code determines the web page is a "billing address" web page, the user's billing address data will be copied from RAM to populate the fields in the billing address web page, and ultimately communicated to the merchant's server. During the check-out process, the shopping assistant code operates in connection with the merchant rules and mapping file and with the user's wallet to automatically provide user data required by the various merchant check-out web pages.

[**0151**] The wallet database **106** may contain only wallet data or alternatively, the wallet data may be located in a database with other data. While the wallet is preferably used

by and in connection with the shopping assistant button **144** and functionality, the wallet may also be used by applications other than shopping assistant such as, for example, bill paying, shopping without shopping assistant, and in connection with virtually any on-line transaction that requires that the user enter personal data. The wallet provides secure transmission and handling of the user's data and makes that data available for use by the user in connection with various on-line transactions without unnecessarily compromising the security of that data.

[**0152**] Once a user has set up a wallet, the functionality provided in accordance with this embodiment of the present invention is now available to the user as the user shops on-line. Preferably, the on-line shopping functionality is available for use in connection with a supported merchant regardless of the Internet site to which the user is connected (via the Internet browser). Thus, the user need not be connected to the provider's Internet site, or even to a supported merchant's web site. The shopping assistant code handles all communication between the user (the user's computer **50**), the server **102**, the wallet database **104**, and the merchant's Internet site and check-out web page(s).

[**0153**] In accordance with another embodiment of the present invention, a button **44** (i.e., interface object) may be added to a browser interface toolbar, such as an interface object toolbar **42** (see, e.g., **FIG. 4**) by a party other than the provider **102**. For example, a link may be provided on a corporate web site that enables visitors to that web site to add a button **44** to their interface object toolbar **42** that would provide a one-click link to that corporate web site from the browser interface **20**. Corporations, or any entity hosting a web site, may sponsor the provider **100** because the addition of the corporate button to the interface object toolbar **42** enables a user to return to that corporate web site from any other web site with a single mouse click. The addition of the corporate button to the browser interface (i.e., the toolbar **42**) thus establishes an affinity between a user and the corporate web site. The corporate sponsor designs the button (e.g., look (one-click button or pull-down menus), icon, etc.), and provides the button design and URL data to the provider **100** for incorporation in a database maintained by the provider and associated with browser interface. Thus, a corporate sponsor may have a button defined in the provider database.

[**0154**] If a visitor to the corporate web site wants to install the corporate button in their browser interface **20**, but does not have a browser interface toolbar **42** installed, the present invention also enables installation of the browser interface toolbar **42** and corporate button **44** in the browser interface **20**. On the other hand, the corporate button **44** may be added to a toolbar **22** of the browser interface **20**, or may be added as a button **44** of an interface object toolbar **42**.

[**0155**] The present invention may also be used as a teaching tool. For example, the interface object toolbar **42** may include a plurality of buttons **44** which may be caused to change color, blink, highlight, etc., to direct a user's attention to that button or to the functionality provided by that button.

[**0156**] The present invention may also enable a provider or corporate sponsor to notify a user of certain properties of the provider or corporate sponsor when that user visits a competing website. For example, when a user visits an automotive website, the corresponding provider property (e.g., provider Autos) would be highlighted to alert the user of the availability of that feature at the provider website. Software may be provided on the user's computer to intercept the URL each time the user causes the browser to navigate to a web site. That intercepted URL may be compared with a file containing a plurality of urls for the provider properties.

[**0157**] Importantly, determining the web site to which the user has navigated at any particular instant is determined by the user's computer alone. Thus, no information regarding the user's activity on the Internet is communicated to any server.

[**0158**] Generally, two-steps are required to add a merchant to the user's merchant list. First, the user must establish an on-line account with a merchant via that merchant's web page. In so doing, the user may be asked to enter a user login identifier, password and account number(s). Once that data has been accepted by the merchant, an on-line account is set-up for the user. Second, the user must link the on-line merchant account to data for that user in a secure database. For example, a user may access a web page via which merchants may be selected, and data specific to that user and merchant (e.g., that user's login identifier, password, account number, etc.) may be entered. That data may then be verified (by communication with the merchant web site) and stored in the secure database for the user. That (those) step(s) may be carried out using another web page (i.e., one that may not be associated with the merchant). Once those two steps are completed, the user may add the merchant to the merchant list, and automatic login to the user's account at a particular merchant may be effected by simply selecting that merchant from the merchant list.

[**0159**] Each time a user activates (launches) his/her browser, the computer code provided in accordance with the present invention will automatically communicate with a predetermined Internet site to retrieve data specific to that user. That retrieved data may be used completely or partially to provide functionality to the browser and to add interface objects to the browser interface. Thus, the data retrieved will define the automatic login interface object, including an up-to-date merchant list.

[**0160**] Referring next to FIGS. **18A-18B**, a preferred embodiment of an automatic login method of the present invention will now be discussed in detail. The method depicted in FIGS. **18A-18B**, and discussed in detail below, considers that the automatic login interface object has already been added to the browser interface, as described in detail above. That method also considers that the computer code providing the automatic login (and other) functionality in accordance with the present invention has been communicated to the user's computer, and is operable in connection with a processor thereof and in connection with the user's Internet browser. At step **1810**, a user selects a merchant from the merchant list provided via the automatic login interface object, which may comprise a button and a pull-down menu. At step **1820**, the computer code (also referred to as the client) causes the browser to open a hidden window within which the automatic login process will be carried out. The computer code also communicates a first request comprising a merchant identifier and a cookie to a server, preferably a secure server, such as depicted in **FIG. 19** and

discussed in more detail below. The secure server receives the first request and determines if the cookie is a secure cookie, at step **1822**. If the cookie is not a secure cookie, the user has not logged in securely for the present Internet session, or a predetermined amount of time has passed since the user securely logged in. In either case, the secure server communicates one or more web pages to the user's browser (which open as new browser windows within the user's browser) prompting the user to enter certain data to securely log in, at step **1830**. One of the data the user must enter is a security key, which the secure server receives and determines if it is the correct and/or valid security key for that user, at step **1832**. If an incorrect security key is entered, the secure server determines if the user has attempted to re-enter the security key more than a predetermined number of times, at step **1834**. If the user has exceeded the permissible number of retires, the secure login process will terminate (be terminated by the secure server) and the user's browser will return to display the web page displayed prior to the user selecting the merchant from the merchant list of the automatic login interface object (button), step **1850**. If the user has not exceed the permissible number of retires, and has not selected to cancel the secure login process, at step **1836**, the user may attempt to re-enter a security key; the secure server again determining if a correct security key has been entered, at step **1832**. If, after a predetermined number of attempts a correct security key is not entered, or the user elects to terminate the secure login process by selecting a cancel button, the secure login process will terminate (be terminated by the secure server) and the user's browser will return to display the web page displayed prior to the user selecting the merchant from the merchant list of the automatic login interface object (button), step **1850**.

[**0161**] If the user successfully enters a correct security key, the secure server communicates an approval message to the computer code, step **1840**, which then communicates a second request to a feed server **1842** for the user's credentials specific to the selected merchant. The second request may include a user identifier and a merchant identifier that may be interpreted by the feed server to locate the user and the user's credentials in the database. If the feed server matches the second request with a database entry, the user's credentials (e.g., login identifier and password for the merchant, web site identifier (e.g., url) for the merchant, and merchant rules file) are communicated by the feed server to the user's computer. The computer code receives the credentials and causes the user's browser to navigate to the merchant's web site (as identified by the url) and automatically populate the login fields with the user's login identifier and password and according to the merchant rules file, thus effecting an automatic login of the user to the merchant web site.

[**0162**] The above-describe automatic login process is carried out in a transparent manner by the computer code provided in accordance with the present invention. In addition, the present invention provides the ability to control the browser interface and to thus control navigation by a user in such a way that rules for controlling the navigation, in this case, for automated login to a merchant web site, can be changed on the server side without changes to the client. The present invention may advantageously utilize the Document Object Module (DOM) available to the browser to effect changes to the rules as just described. That is, after the user selects a merchant from the merchant list, the computer code carries out the automatic login process without requiring further input from the user (except for the case where the use is not securely logged in). With continued reference to FIGS. **18A-18B**, the automatic login carried out by the computer code provided in accordance with the present invention will be discussed in more detail. At step **1844**, the computer code determines if it has received the credentials in response to the second request. If not, the computer code generates an error message that is displayed y the user's browser, step **1860**. Thereafter, the computer code causes the hidden window (opened at step **1820**) to close and the automatic login process to terminate. The display in the browser then returns to the web page displayed prior to the user selecting the merchant from the merchant list. If the computer code has received the credentials, as determined at step **1844**, the computer code launches another hidden window to carry out the remainder of the automatic login process, at step **1870**. At any time before the automatic login process is complete, the user may cancel the process by selecting an appropriate button displayed by the computer code, e.g., a cancel button displayed while the computer code is carrying out the automatic login process. If the user elects to cancel the automatic login process, at step **1872**, the computer code will close all hidden windows opened during the process, and the display by the browser returns to the web page displayed prior to the user selecting the merchant from the merchant list. If the user does not manually terminate the automatic login process, the computer code may cancel the process if a predetermined amount of time has passed before the process is completed. For example, after receiving the credentials, the computer code may be unable to cause the browser to navigate to the merchant web site due to equipment or network problems. In that case, the computer code will terminate the automatic login process, step **1874**, and display an error message to the user that the automatic login could not be completed, step **1878**. At that point, the computer code may offer the user an opportunity to attempt the automatic login process again, step **1878**, at which point the process returns to step **1844** and proceeds as described herein.

[**0163**] If the computer code determines that the automatic login process has not automatically times out, step **1874**, the computer code determines if it has received a login confirmation from the merchant web site, step **1876**. To do this, the computer code evaluates the HTML data received in the hidden window and looks for a predetermined word, words, message, or other indicator that the login at the merchant web site has been successful. For example, a merchant web site may display a message after a user has successfully logged in. That message may be included as part of the merchant rules file communicated by the feed server to the computer code as part of the credentials. The computer code can thus determine if an automatic login has been successful. If it has, the computer code displays the web page from the merchant web site in the main browser window, step **1890**, thus effectively causing the user's browser to navigate to the merchant web site and directly to a web page for that user's account.

[**0164**] Referring next to **FIG. 19, a** system for facilitating automatic login by a user to a web site is there depicted. The system comprises a server, which may be a single server, or a plurality of servers **1900**, **1920**, **1940**, as a routine matter of design choice. In a preferred embodiment, the system comprises a server with a data storage device and having

software stored thereon. The software communicates computer code to a user's computer for adding automatic login functionality to the user's Internet browser, as described in detail herein.

[0165] In the embodiment depicted in **FIG. 19, a** secure server **1900** carries out a secure login process in connection with the computer code. The secure server **1900** has a database with a plurality of user data and corresponding security key information. The secure server **1900** can receive a cookie from a user and determine if the cookie is a secure cookie with a valid security key. If so, the secure server **1900** can communicate an affirmation or authorization to the computer code on the user's computer.

[0166] A feed server **1920** has a database with a plurality of user data, including merchant login and password data, and merchant rules data, for each merchant supplied by the user. When the computer code on the user's computer receives the affirmation or authorization from the secure server, the computer code communicates a second request to the feed server including cookie having a user identifier and merchant identifier. The feed server determines if there is a match between the user identifier and merchant identified in the database on the feed server and, if so, communicates credentials to the user's computer. In a preferred embodiment, the credentials comprise a user login identifier, a password, ad a merchant rules file, all specific to a particular merchant. The computer code uses the credentials to cause the user's browser to navigate to the merchant's web site and to the login web page at that site, to automatically populate the login fields (login identifier and password), and effect a login according to the merchant rules file (e.g., select a "login" or other similar button). At this point, the user's computer may be connected via the Internet to the merchant's server **1940**. Once the computer code determines that the automatic login has been successful, the merchant web page for the user's account is displayed by the browser.

[0167] It will be obvious to persons skilled in the art and from the disclosure provided herein that the various embodiments of the present invention may be carried out using software provided on the user's computer and on the various servers.

[0168] It is to be understood that the present invention may be implemented utilizing any number of computer technologies. For example, although the present embodiments are disclosed as operable in connection with the Internet, the present invention may be utilized over any computer network, including, for example, a wide area network. Similarly, the user computer **50** may be any device that may be coupled to the network, including, for example, personal digital assistants, web-enabled cellular telephones, hard-wired telephones that dial into the network, mobile computers, personal computers, Internet appliances and the like. Furthermore, the servers described herein may be of any type, running any software, and the software modules, objects and plug-ins described herein may be written in any programming language. Lastly, the database and storage devices described herein may utilize any storage technology, including, for example, local computer memory, network attached storage, and any known storage medium, such as magnetic or optical.

[0169] Thus, while there have been shown and described and pointed out fundamental novel features of the invention as applied to preferred embodiments thereof, it will be understood that various omissions and substitutions and changes in the form and details of the disclosed invention may be made by those skilled in the art without departing from the spirit of the invention. It is the intention, therefore, to be limited only as indicated by the scope of the claims appended hereto.

What is claimed is:

1. A method of facilitating automatic login by a user to a web site using an Internet browser and a computer, said method comprising the step of communicating computer code to the computer to add automatic login functionality to the browser that uses user data to automatically log the user in to the web site.

2. A method as recited by claim 1, wherein said communicating step comprises communicating computer code to add a button that adds automatic login functionality to the browser that enables a user to access a merchant list and to automatically login to a web site of a merchant on the merchant list by selecting that merchant from the merchant list.

3. A method as recited by claim 2, wherein the button provides access to the merchant list via a pull-down menu.

4. A method as recited by claim 2, wherein the computer code facilitates automatic communication of a request to a server when a user selects a merchant from the merchant list.

5. A method as recited by claim 4, wherein the request includes a secure cookie.

6. A method as recited by claim 4, wherein the request comprises a first request and a second request including a user identifier and merchant identifier.

7. A method as recited by claim 6, wherein the first request includes a secure cookie.

8. A method as recited by claim 1, wherein the computer code monitors the Internet navigation of the browser by intercepting an Internet address for each Internet site to which the browser is caused to navigate.

9. A method as recited by claim 2, wherein the computer code determines if a merchant is a supported merchant.

10. A method as recited by claim 1, wherein the computer code provides an indicator when the browser is caused to navigate to a supported merchant web site.

11. A method as recited by claim 2, further comprising the steps of:

receiving a first request from a user;

determining if the first request is a secure request;

if the first request is a secure request, communicating a response to the user; and

if the first request is not a secure request, communicating a secure login web page to the user for display by the Internet browser.

12. A method as recited by claim 11, wherein the determining step comprises determining if the first request includes a secure cookie.

13. A method as recited by claim 11, further comprising the steps of:

receiving a second request for a user credential; and

communicating the user credential to the user.

14. A method as recited by claim 13, wherein the user credential is stored on a server, and wherein the user credential is at least one of a login identifier and a password,

and wherein the user credential is specific to a merchant and to a web site of the merchant.

15. A method as recited by claim 13, wherein the second request includes at least one of a user identifier and a merchant identifier.

16. A method as recited by claim 14, wherein the user credential includes a merchant rules file for the selected merchant.

17. An Internet browser interface displayable by an Internet browser on a computer display, said browser interface being modifiable by a user of the browser to add functionality to enable a user to automatically log the user in to a web site using user data.

18. An Internet browser interface as recited by claim 17, wherein said functionality is provided by computer code operable in connection with a processor of the computer.

19. An Internet browser interface as recited by claim 18, wherein said functionality is provided by computer code operable in connection with a processor of the computer for adding an automatic login button to said browser interface that adds automatic login functionality to the browser that enables a user to access a merchant list and to automatically login to a web site of a merchant on the merchant list by selecting that merchant from the merchant list, and wherein the button provides access to the merchant list.

20. An Internet browser interface as recited by claim 19, wherein the computer code facilitates automatic communication of a request to a server when a user selects a merchant from the merchant list.

21. An Internet browser interface as recited by claim 20, wherein the request includes a secure cookie.

22. An Internet browser interface as recited by claim 20, wherein the request comprises a first request and a second request including a user identifier and a merchant identifier.

23. An Internet browser interface as recited by claim 22, wherein the first request includes a secure cookie.

24. An Internet browser interface as recited by claim 18, wherein said computer code is further operable in connection with the processor for monitoring the Internet navigation of the Internet browser by intercepting an Internet address for each Internet site to which the Internet browser is caused to navigate.

25. An Internet browser interface as recited by claim 18, wherein said computer code is further operable in connection with the processor for determining if a merchant web site is a supported merchant web site.

26. An Internet browser interface as recited by claim 19, wherein said computer code is further operable in connection with the processor of the computer for:

communicating a first request;

communicating a second request for a user credential; and

receiving the user credential.

27. An Internet browser interface as recited by claim 26, wherein the user credential is stored on a server, and wherein the user credential is at least one of a login identifier and a password, and wherein the user credential is specific to a merchant and to a web site of the merchant.

28. An Internet browser interface as recited by claim 26, wherein the first request includes a secure cookie.

29. An Internet browser interface as recited by claim 28, wherein the second request includes at least one of a user identifier and a merchant identifier.

30. An Internet browser interface as recited by claim 29, wherein the user credential includes a merchant rules file for the selected merchant.

31. A system for facilitating automatic login by a user to a web site using an Internet browser and a computer, wherein user data is necessary to login to the web site, said system comprising:

a server connectable to the Internet and comprising:

a data storage device; and

a processor operable in connection with software stored on said data storage device for communicating computer code to the user's computer to add automatic login functionality to the browser, wherein the computer code adds automatic login functionality to the browser that uses user data to automatically log the user in to the web site.

32. A system as recited by claim 31, wherein the computer code adds a button to the browser that adds automatic login functionality to the browser that enables a user to access a merchant list and to automatically login to a web site of a merchant on the merchant list by selecting that merchant from the merchant list.

33. A system as recited by claim 32, wherein the button provides access to the merchant list via a pull-down menu.

34. A system as recited by claim 32, wherein the computer code facilitates automatic communication of a request to a server when a user selects a merchant from the merchant list.

35. A system as recited by claim 34, wherein the request includes a secure cookie.

36. A system as recited by claim 35, wherein the request comprises a first request and a second request including a user identifier and a merchant identifier.

37. A system as recited by claim 36, wherein the first request includes a secure cookie.

38. A system as recited by claim 34, wherein said processor is further operable in connection with software for determining if the request includes a secure cookie.

39. A system as recited by claim 31, wherein the computer code monitors the Internet navigation of the browser by intercepting an Internet address for each Internet site to which the browser is caused to navigate.

40. A system as recited by claim 39, wherein the computer code determines if a merchant web site is a supported merchant web site.

41. A system as recited by claim 31, wherein said server is a secure server.

42. A system as recited by claim 32, wherein said processor is further operable in connection with software for:

receiving a first request from a user;

determining if the first request is a secure request;

if the first request is a secure request, communicating a response to the user; and

if the first request is not a secure request, communicating a secure login web page to the user for display by the Internet browser.

43. A system as recited by claim 42, wherein said processor is further operable in connection with software for determining if the first request includes a secure cookie.

44. A system as recited by claim 42, wherein said processor is further operable in connection with software for:

receiving a request for a user credential; and

communicating the user credential to the user.

45. A system as recited by claim 44, wherein a user credential is stored on a server, and wherein the user credential is one or more of a login identifier and a password, and wherein the user credential is specific to a merchant and to a web site of the merchant.

46. A system as recited by claim 44, wherein the second request includes at least one of a user identifier and a merchant identifier.

47. A system as recited by claim 45, wherein the user credential includes a merchant rules file for the selected merchant.

48. A method of controlling an Internet browser interface displayable by an Internet browser on a display of a computer, the Internet browser enabling a user of the computer and Internet browser to access and navigate the Internet and to receive and display on the computer display one or more web pages from one or more Internet sites, including the display of a web page from a predetermined Internet site, the Internet browser having at least one Internet browser toolbar having at least one toolbar button providing a predetermined functionality to the user of the computer and Internet browser, said method comprising the steps of:

(a) providing, at the predetermined Internet site, access to a program for controlling the Internet browser interface; and

(b) making available for downloading by the predetermined Internet site, a file for displaying a user toolbar making additional functionality available to the user as part of the Internet browser interface, such that once the user toolbar is displayed, the user toolbar remains displayed, and said additional functionality remains available to the user, regardless of the Internet site to which the Internet browser is caused to navigate, wherein the file adds automatic login functionality to the browser that uses user data to automatically log the user in to the web site.

49. A method as recited by claim 48, wherein the user toolbar includes an interface object and is customizable by the user to provide user-selected functionality in the user toolbar.

50. A method as recited by claim 48, wherein the file comprises an ActiveX control.

51. A method as recited by claim 48, wherein the file comprises a Plug-in.

52. A method as recited by claim 49, wherein the interface object is a toolbar button.

53. A method as recited by claim 49, wherein the interface object is a search window that enables the user to initiate a search at the predetermined Internet site regardless of the Internet site to which the computer is connected via the browser at the time the search is initiated.

54. A method as recited by claim 48, wherein the predetermined Internet site maintains user-specific information for a plurality of users, including the user of the computer and Internet browser, said method further comprising the step of making available for downloading by the predetermined Internet site, information specific to the user of the computer and Internet browser for defining all or part of the user toolbar and wherein all or part of the display of the user toolbar is dependent upon the information, specific to the user of the computer and Internet browser, downloaded by the predetermined Internet site.

55. A method as recited by claim 54, further comprising the step of making available for downloading by the predetermined Internet site, additional information, specific to the user of the computer and Internet browser, for defining all or part of the user toolbar, and wherein all or part of the display of the user toolbar is dependent upon the downloaded information.

56. An Internet browser interface displayable by an Internet browser on a display of a computer, the Internet browser facilitating connection between the computer and one or more Internet sites, including a predetermined Internet site, the Internet browser having at least one Internet browser toolbar having predetermined functionality available to a user of the computer and Internet browser, the Internet browser further facilitating display on the computer display of one or more web pages from the one or more Internet sites, including display of web pages from the predetermined Internet site, said Internet browser interface including a user toolbar that is displayed as part of the Internet browser interface and along with the Internet browser toolbar while the Internet browser is activated regardless of the Internet site to which the computer is connected via the Internet browser, wherein the user toolbar makes additional functionality available to the user as part of the Internet browser interface, such that once the user toolbar is displayed, the user toolbar remains displayed, and said additional functionality remains available to the user, regardless of the Internet site to which the Internet browser is caused to navigate, wherein said additional functionality adds automatic login functionality to the browser that uses user data to automatically log the user in to the web site.

57. An Internet browser interface as recited by claim 56, wherein said user toolbar includes an interface object and is customizable by the user.

58. An Internet browser interface as recited by claim 57, wherein said interface object is a toolbar button.

59. An Internet browser interface as recited by claim 57, wherein said interface object is a search window that enables the user to initiate a search at the predetermined Internet site regardless of the Internet site to which the computer is connected via the Internet browser at the time the search is initiated.

60. An Internet browser interface as recited by claim 57, wherein said interface object comprises an ActiveX control that enables user customization of said interface object.

61. An Internet browser interface as recited by claim 57, wherein said interface object comprises a Plug-in control that enables user customization of said interface object.

62. An Internet browser interface as recited by claim 57, wherein the predetermined Internet site maintains user-specific information for a plurality of users, including the user of the computer and Internet browser, wherein said interface object causes the Internet browser to establish a connection to the predetermined Internet site when the Internet browser is first activated and to receive information specific to the user of the computer and Internet browser from the predetermined Internet site, and wherein all or part of said user toolbar is dependent upon the information, specific to the user of the computer and Internet browser, received from the predetermined Internet site.

63. An Internet browser interface as recited by claim **62**, wherein said interface object causes the Internet browser to periodically re-establish a connection to the predetermined Internet site while the Internet browser is activated to receive additional information, specific to the user of the computer and Internet browser, from the predetermined Internet site, and wherein all or part of said user toolbar is dependent upon the received information and additional information.

* * * * *