the transaction between CLI **702** and network management system **100**. in Step **834**, after the CLI response message **716** has been transmitted, major row **602** corresponding to CLI request message **704** in partial response table **600** is deleted.

FIG. **9** further illustrates steps **816–832**, the process of adding and deleting the major and minor rows of partial response table **600** in response to message elements **400**. The process of FIG. **9** is adapted for handling messages **300** directed to attribute **706** and attribute **708**. Characteristic **708** receives message element **400** at step **902**. Step **904** determines whether the attribute requested by message element **400** is available at characteristic **708**. If step **904** determines the attribute is available at characteristic **708**, step **906** fulfills the request by sending a message element **400** containing the data requested to body **206**. Step **918** places the data from the message element in parameters list **606** in the major row **602** corresponding to CLI request message **704**.

Step **920** determines whether the outgoing parameters list **606** is complete, after parameters list **606** has been updated in step **924**. The parameters list **606** will be complete after all of the attribute or action data has been received from characteristics **706**, **708** so that CLI response message **716** may be sent. If parameters list **606** is not complete, the process continues at step **922** where body **206** waits for additional message elements sent from characteristic **708**. As additional message elements are received from characteristic **708**, the data is placed in parameters list **606** at step **918**. Since CLI request message **704** may include a request for data from multiple characteristics, it should be noted that the process of FIG. **9** may be executed repeatedly for each CLI request message **704**.

If step **920** determines parameters list **606** is complete, managed object **204** is ready to send CLI response message **716** to CLI **702**. Step **924** responds to CLI request message **704** with CLI response message **716**. Step **926** deletes major row **602** from the partial response table **600** because managed object **204** has responded to the original CLI request message **704**. Multiple outstanding request messages to the managed object **204** will result in multiple major rows **602** in the partial response table **600**. Major rows **602** of the partial response table **600** are deleted every time managed object **204** responds to the associated major CLI request message **704**.

If step **904** determines that the attribute data member requested by the message element **400** received at characteristic **708** must be retrieved from a separate network entity, step **907** sends a request message element to body **206** requesting managed object request message **710** to body **206**. Step **908** creates minor row **604***a* in response to step **907**. Minor row **604***a–***604***n* points to and is pointed to by major row **602**. The basis for the forward and backward pointers is a hash of the OID **304**, XID **302** and ID **402**. Step **910** generates managed object request message **710** at managed object **204**. Managed object request message **710** is sent to network element **118** which contains the desired data members at characteristic **712**. Managed object **502** of network element **118** responds to managed object request message **710** with the data requested using the same process that managed object **204** uses to respond to CLI request message **704**. Managed object **502** may formulate additional request messages to be sent to additional network elements **110–120** or other network entities. When managed object **502** has retrieved or owns all of the data requested by managed object request message **710**, managed object **502** responds with managed object response message **714**.

Step **912** receives managed object response message **714** from network element **118**. OID **302**, XID **304**, identify the major row **602** of partial response table **600** with which the managed object response message **714** is associated. ID and data **308***a–***308***n* of managed object response message **714** identify the particular ID **402** of minor rows **604***a–***604***n* with which the managed object response message **714** is associated. Step **913** deletes the associated minor row or rows **604***a–***604***n* in response to received managed object response message **714**. Deletion of minor row **604***a* indicates managed object **204** has received a response to the managed object request message **710**.

Step **914** parses managed object response message **714** into message elements and passes them to their associated characteristics **708** at body **206**. Step **916** updates characteristic **708** and the data members of attribute **2** at step **916**. Characteristic **708** then fulfills the original message element request as received in step **902**. In the preferred embodiment, the parameters list **606** is updated by message element **400** from characteristic **708**. In an alternative embodiment, the parameters list **606** may be updated after parsing the managed object response message **714**, directly from the data received from managed object **502**. After the data members of characteristic **708** have been updated, flow control of the process is passed to step **918**, and the process proceeds as described above.

The process of FIG. **9** may be further adapted for subjugate messages. Subjugate messages are messages sent in response to response messages. For example, managed object **204** may send an additional action request message in response to managed object response message **714**. In such a case, characteristic **708** would receive the message element containing the data from managed object **502**. Characteristic **708** would determine that additional action (i.e., switching an additional cross-connect network element) is necessary and would send an additional message element **400** to body **206** requesting that an additional managed object request message be sent to the necessary network element **108***a–***108***n*. Another minor row **604***b* would be created, and the process repeats itself until the parameters list **606** is complete. Upon completion of the parameters list, the CLI response message **716** is generated and major row **602** is deleted from partial response table **600**. Subjugate action messages provide for dynamic network configuration management because network manager **106** is able to change the implementation of CLI request message **704** depending on the response messages received from network elements **108***a–***108***n*.

The present invention may be implemented using hardware, software or a combination thereof and may be implemented in a computer system or other processing system. In fact, in one embodiment, the invention is directed toward a computer system capable of carrying out the functionality described herein. An example computer system **1001** is shown in FIG. **10**. The computer system **1001** includes one or more processors, such as processor **1004**. The processor **1004** is connected to a communication bus **1002**. Various software embodiments are described in terms of this example computer system. After reading this description, it will become apparent to a person skilled in the relevant art how to implement the invention using other computer systems and/or computer architectures.

Computer system **1002** also includes a main memory **1006**, preferably random access memory (RAM), and can also include a secondary memory **1008**. The secondary memory **1008** can include, for example, a hard disk drive **1010** and/or a removable storage drive **1012**, representing a floppy disk drive, a magnetic tape drive, an optical disk drive, etc. The removable storage drive **1012** reads from

and/or writes to a removable storage unit **1014** in a well known manner. Removable storage unit **1014**, represents a floppy disk, magnetic tape, optical disk, etc. which is read by and written to by removable storage drive **1012**. As will be appreciated, the removable storage unit **1014** includes a computer usable storage medium having stored therein computer software and/or data.

In alternative embodiments, secondary memory **1008** may include other similar means for allowing computer programs or other instructions to be loaded into computer system **1001**. Such means can include, for example, a removable storage unit **1022** and an interface **1020**. Examples of such can include a program cartridge and cartridge interface (such as that found in video game devices), a removable memory chip (such as an EPROM, or PROM) and associated socket, and other removable storage units **1022** and interfaces **1020** which allow software and data to be transferred from the removable storage unit **1022** to computer system **1001**.

Computer system **1001** can also include a communications interface **1024**. Communications interface **1024** allows software and data to be transferred between computer system **1001** and external devices. Examples of communications interface **1024** can include a modem, a network interface (such as an Ethernet card), a communications port, a PCMCIA slot and card, etc. Software and data transferred via communications interface **1024** are in the form of signals which can be electronic, electromagnetic, optical or other signals capable of being received by communications interface **1024**. These signals **1026** are provided to communications interface via a channel **1028**. This channel **1028** carries signals **1026** and can be implemented using wire or cable, fiber optics, a phone line, a cellular phone link, an RF link and other communications channels.

In this document, the terms "computer program medium" and "computer usable medium" are used to generally refer to media such as removable storage device **1012**, a hard disk installed in hard disk drive **1010**, and signals **1026**. These computer program products are means for providing software to computer system **1001**.

Computer programs (also called computer control logic) are stored in main memory and/or secondary memory **1008**. Computer programs can also be received via communications interface **1024**. Such computer programs, when executed, enable the computer system **1001** to perform the features of the present invention as discussed herein. In particular, the computer programs, when executed, enable the processor **1004** to perform the features of the present invention. Accordingly, such computer programs represent controllers of the computer system **1001**.

In an embodiment where the invention is implemented using software, the software may be stored in a computer program product and loaded into computer system **1001** using removable storage drive **1012**, hard drive **1010** or communications interface **1024**. The control logic (software), when executed by the processor **1004**, causes the processor **1004** to perform the functions of the invention as described herein.

In another embodiment, the invention is implemented primarily in hardware using, for example, hardware components such as application specific integrated circuits (ASICs). Implementation of the hardware state machine so as to perform the functions described herein will be apparent to persons skilled in the relevant art(s).

In yet another embodiment, the invention is implemented using a combination of both hardware and software.

While the invention has been particularly shown and described with reference to preferred embodiments thereof, it will be understood by those skilled in the relevant art that various changes in form and details may be made therein without departing from the spirit and scope of the invention.

What is claimed is:

1. In a network management system having a network manager and at least one network element, a message handling process, comprising the steps of:

receiving a client request, from a client application, at a first managed object;

determining if said client request can be fulfilled at said first managed object;

generating a managed object request corresponding to said client request at said first managed object in response if said step of determining determines that said client request can not be fulfilled at first managed object;

sending said managed object request to a second managed object;

receiving a managed object response from said second managed object; generating a client response fulfilling said client request; and

sending said client response to said client application.

2. In a network management system having a network manager and at least one network element, a message handling system, comprising:

means for receiving a client request, from a client application, at a first managed object;

means for determining if said client request can be fulfilled at said first managed object;

means for generating a managed object request corresponding to said client request at said first managed object in response if said step of determining determines that said client request can not be fulfilled at first managed object;

means for sending said managed object request to a second managed object;

means for receiving a managed object response from said second managed object;

generating a client response fulfilling said client request; and

means for sending said client response to said client application.

3. In a network management system having a network manager and at least one network element, a message handling process, comprising the steps of:

receiving a client request at a first managed object, at least part of said client request being fulfilled by a second managed object;

creating a major row in response to receiving the client request; and

creating a minor row associated with a managed object request sent to said second managed object, said minor row;

(a) having an index that associates said minor row with said major row, and

(b) correlating a response to said managed object request with said client request.

4. In a network management system having a network manager and at least one network element, a system for message handling, comprising:

means for receiving a client request at a first managed object, at least part of said client request being fulfilled by a second managed object;

means for creating a major row in response to receiving the client request; and

means for creating a minor row associated with a managed object request sent to said second managed object, said minor row;

(a) having an index that associates said minor row with said major row, and

(b) correlating a response to said managed object request with said client request.

* * * * *

US005964830A

United States Patent [19]

Durrett

[11] Patent Number: 5,964,830

[45] Date of Patent: Oct. 12, 1999

[54] **USER PORTAL DEVICE FOR THE WORLD WIDE WEB TO COMMUNICATE WITH A WEBSITE SERVER**

[76] Inventor: **Charles M. Durrett**, 511 George, Birmingham, Mich. 48009

[21] Appl. No.: **08/699,843**

[22] Filed: **Aug. 20, 1996**

Related U.S. Application Data

[60] Provisional application No. 60/002,633, Aug. 22, 1995.

[51] **Int. Cl.**[6] **G06F 9/00**; G06F 9/06

[52] **U.S. Cl.** **709/200**; 709/203; 709/212; 709/216; 709/217

[58] **Field of Search** 395/200.3–200.33, 395/200.36, 200.42, 200.46–200.49, 186, 187.01, 188.01, 200, 670–675; 709/200–203, 212–213, 216–219

[56] **References Cited**

U.S. PATENT DOCUMENTS

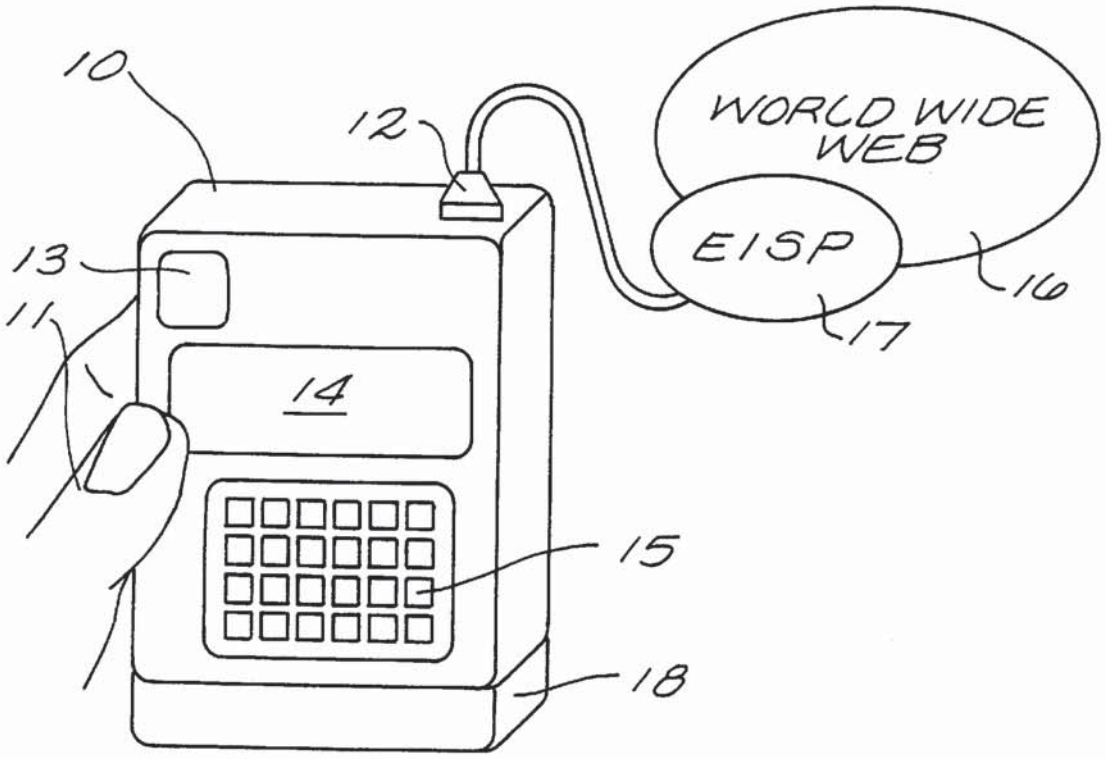
5,287,103	2/1994	Kasprzyk et al.	340/825.52
5,455,953	10/1995	Russell	395/187.01
5,530,852	6/1996	Meske, Jr. et al.	395/200.36
5,572,643	11/1996	Judson	395/200.48
5,630,133	5/1997	Hotea et al.	395/671
5,721,916	2/1998	Pardikar	395/200.47

Primary Examiner—Zarni Maung
Assistant Examiner—Bharat Barot
Attorney, Agent, or Firm—Mark E. Ogram P.C.

[57] **ABSTRACT**

An improved network and use of computer based capabilities in which a virtual disk server is employed to give a user portal device access to software object elements and a non-volatile memory. The user portal device includes a rudimentary stand-alone capability. To operate, the user portal device communicates with the virtual disk server and downloads the software object elements which form the operating correlation of objects, the user portal device communicates data and software object elements to the virtual disk server which stores the data and software object elements in its own non-volatile memory such as a disk drive apparatus.

15 Claims, 8 Drawing Sheets



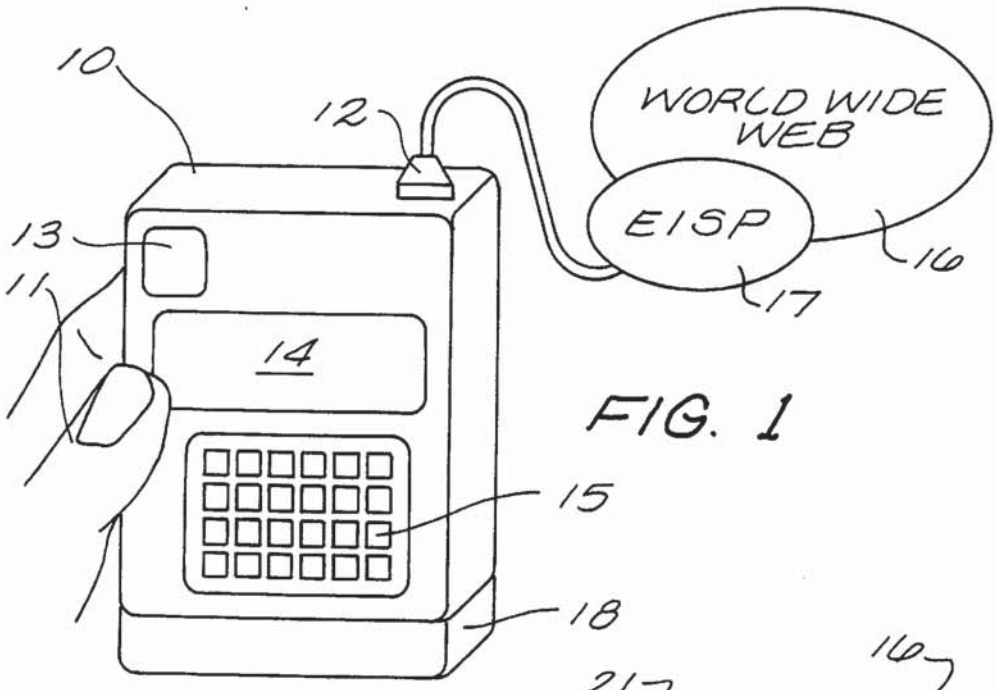


FIG. 1

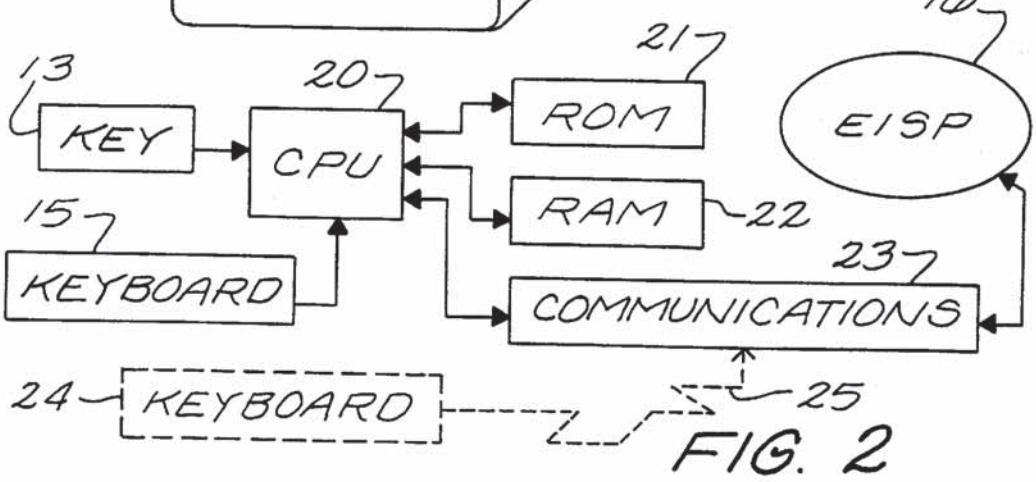


FIG. 2

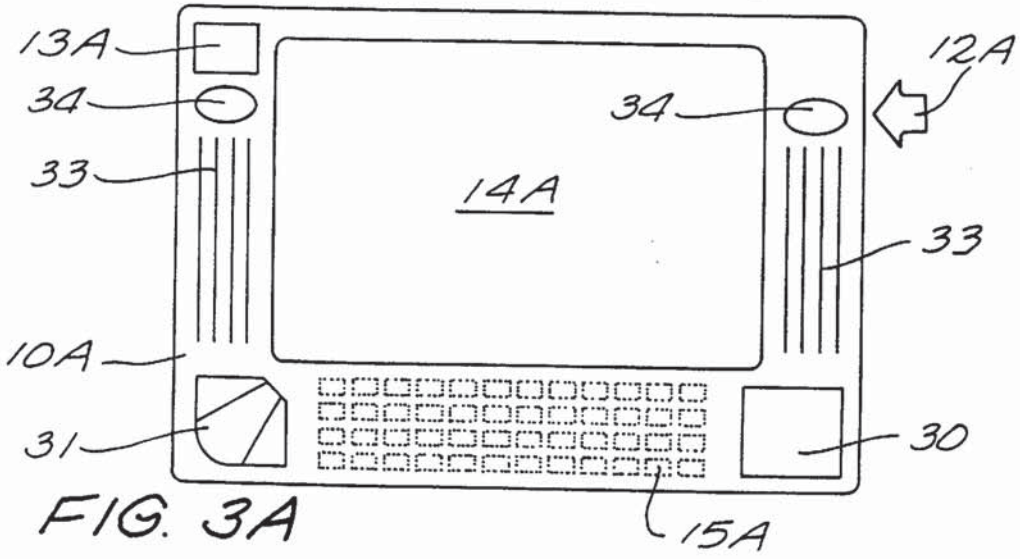


FIG. 3A

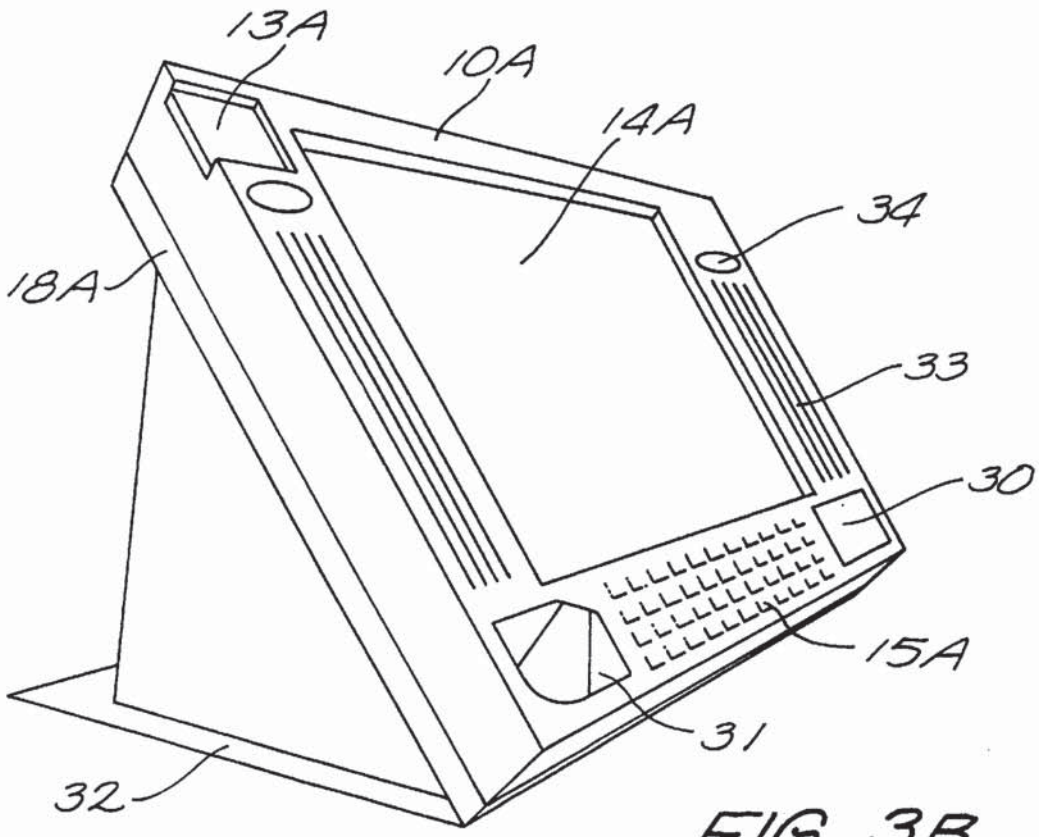


FIG. 3B

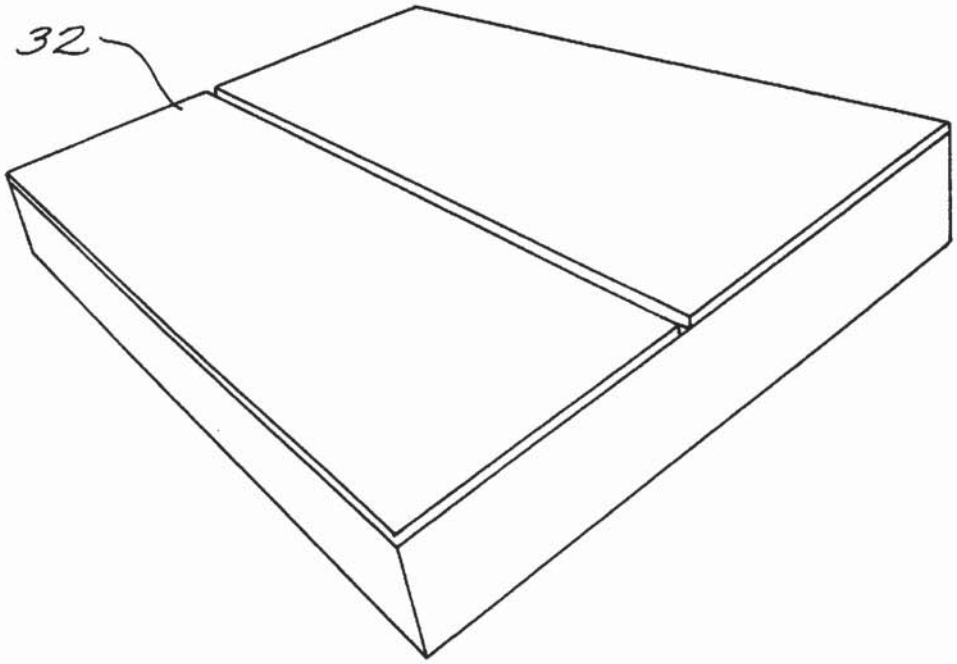


FIG. 3C

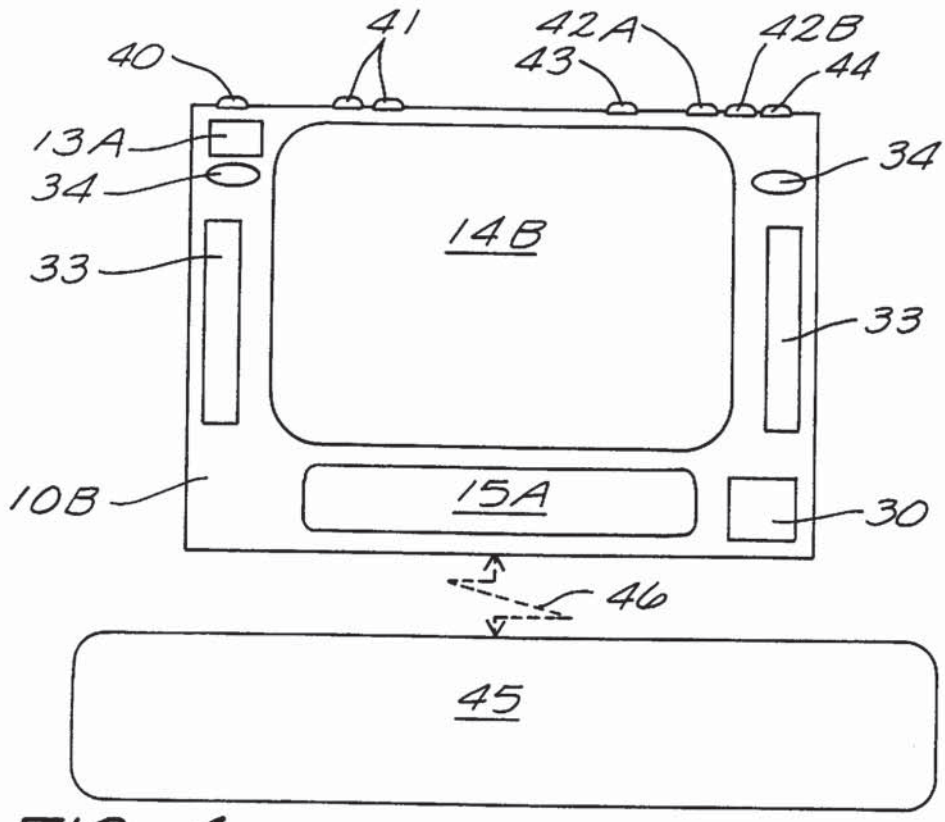

41
42A
42B
40
43
44
13A
34
34
14B
33
33
10B
15A
30
46
45

FIG. 4

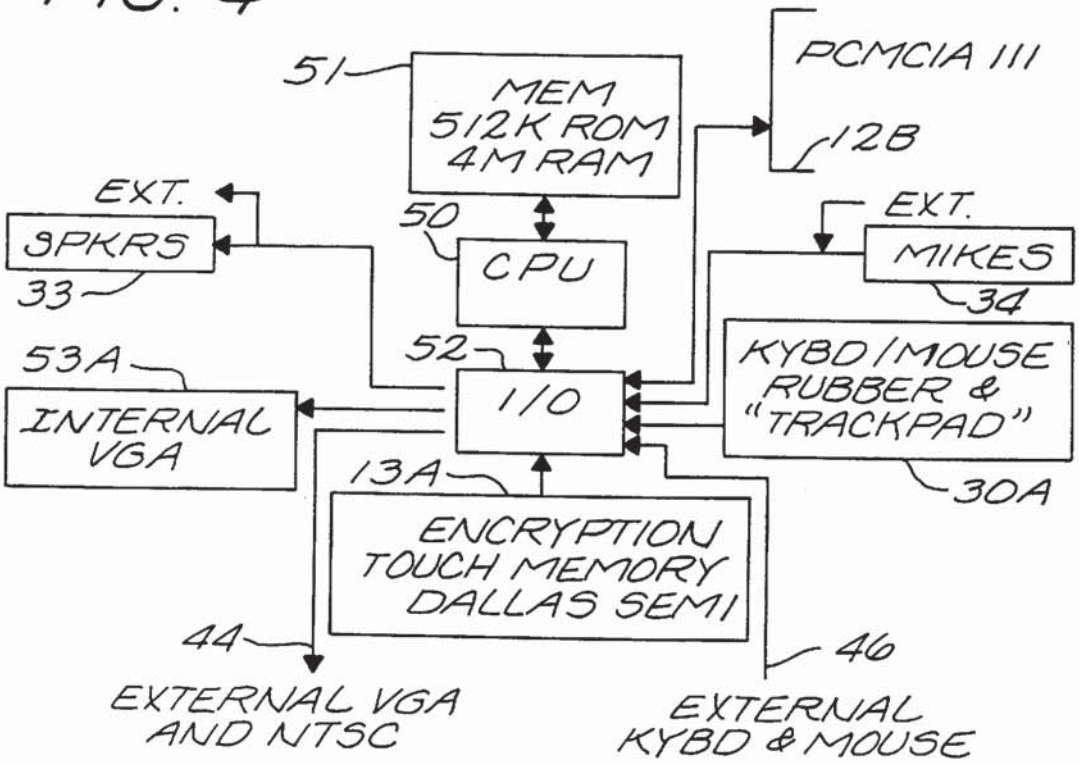

51
MEM
512K ROM
4M RAM
PCMCIA 111
12B
EXT.
SPKRS
33
50
CPU
EXT.
MIKES
34
53A
52
KYBD/MOUSE
RUBBER &
"TRACKPAD"
INTERNAL
VGA
I/O
30A
13A
ENCRYPTION
TOUCH MEMORY
DALLAS SEMI
44
46
EXTERNAL VGA
AND NTSC
EXTERNAL
KYBD & MOUSE

FIG. 5A

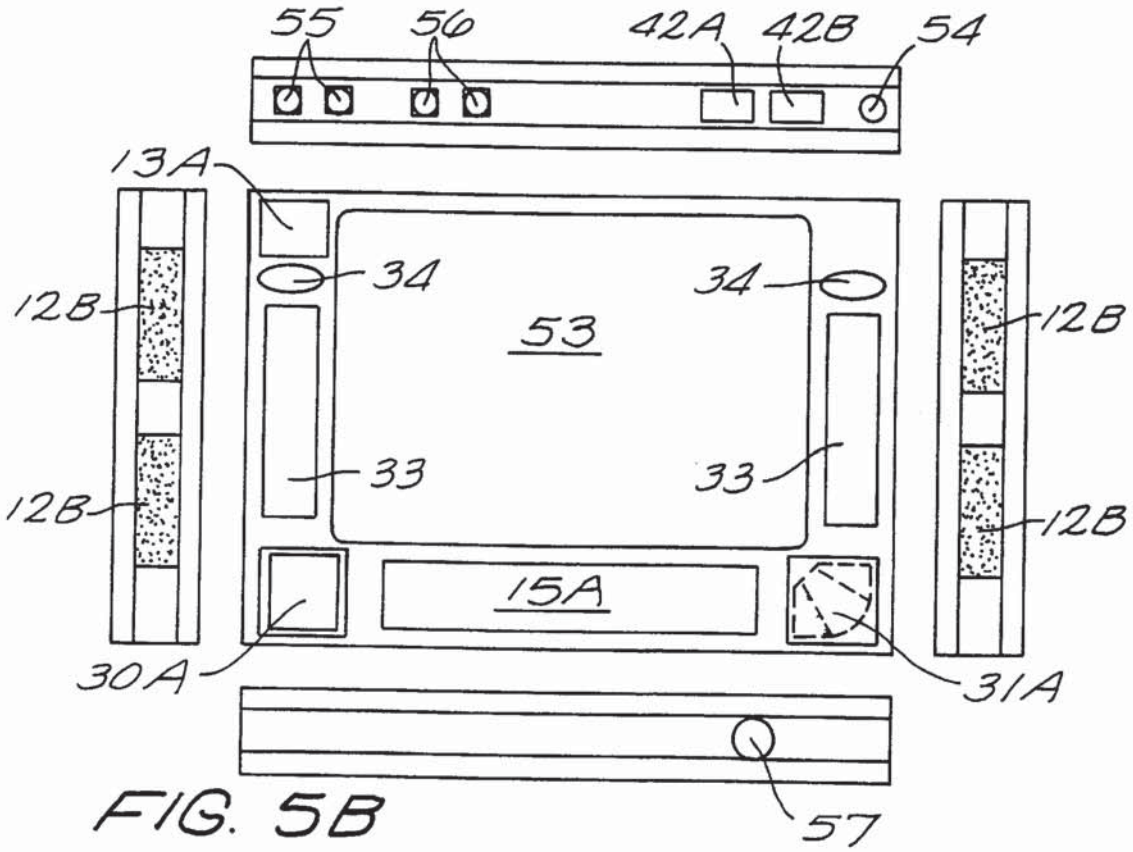


FIG. 5B

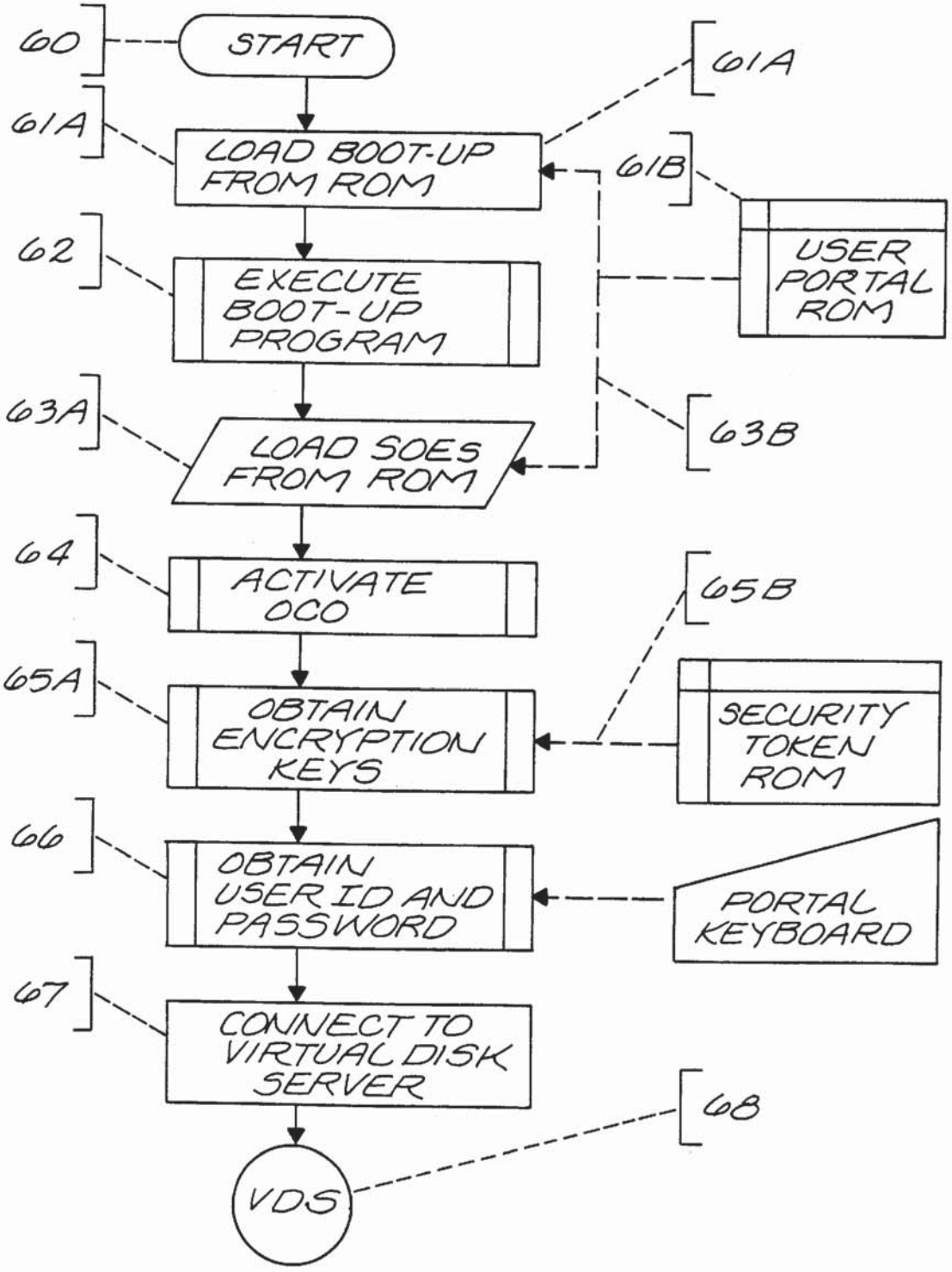

60
START
61A
61A
LOAD BOOT-UP FROM ROM
61B
USER PORTAL ROM
62
EXECUTE BOOT-UP PROGRAM
63A
LOAD SOEs FROM ROM
63B
64
ACTIVATE OCO
65B
65A
OBTAIN ENCRYPTION KEYS
SECURITY TOKEN ROM
66
OBTAIN USER ID AND PASSWORD
PORTAL KEYBOARD
67
CONNECT TO VIRTUAL DISK SERVER
68
VDS

FIG. 6A

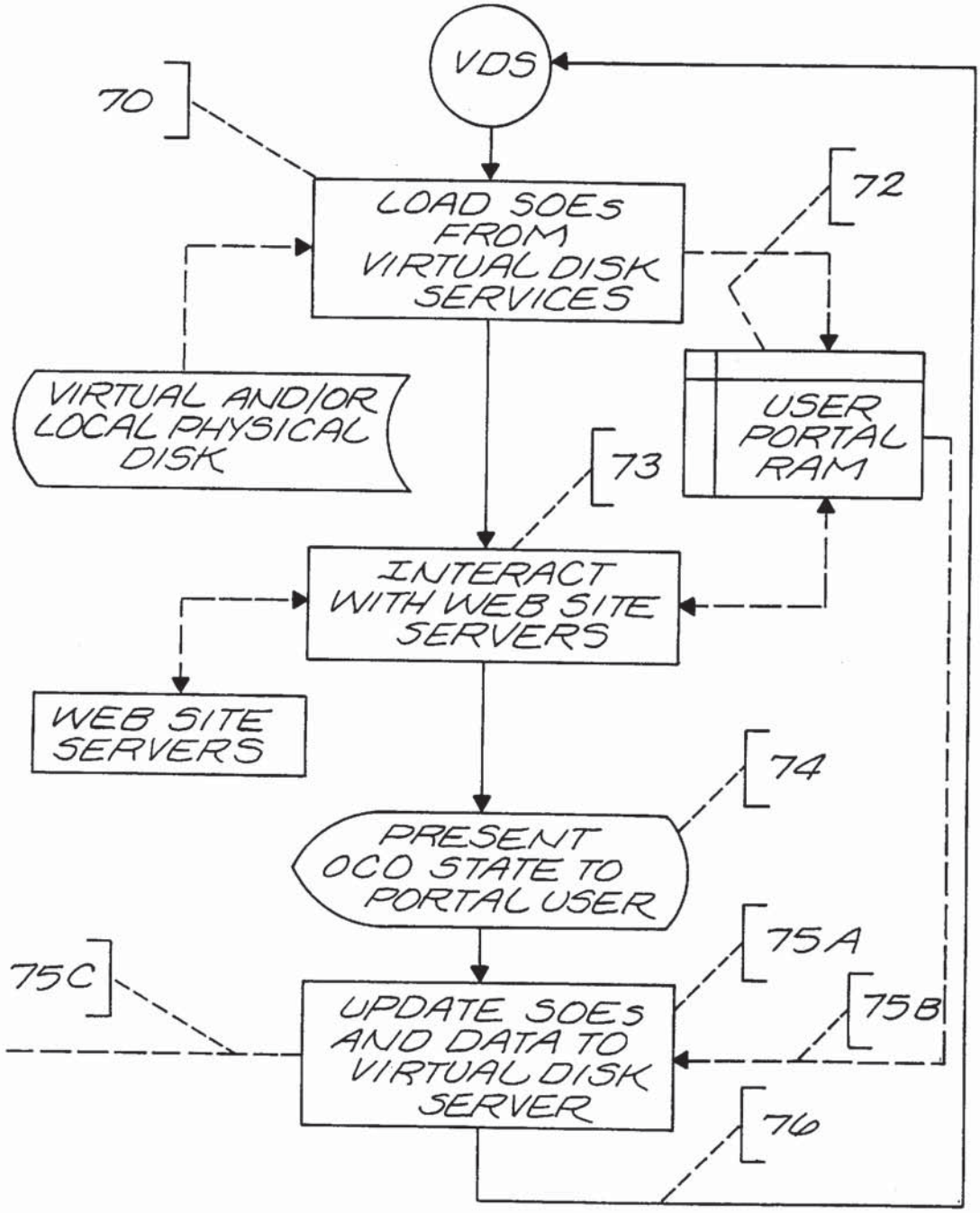

VDS
70
LOAD SOEs FROM VIRTUAL DISK SERVICES
72
VIRTUAL AND/OR LOCAL PHYSICAL DISK
USER PORTAL RAM
73
INTERACT WITH WEB SITE SERVERS
WEB SITE SERVERS
74
PRESENT OCO STATE TO PORTAL USER
75A
75C
75B
UPDATE SOEs AND DATA TO VIRTUAL DISK SERVER
76

FIG. 6B

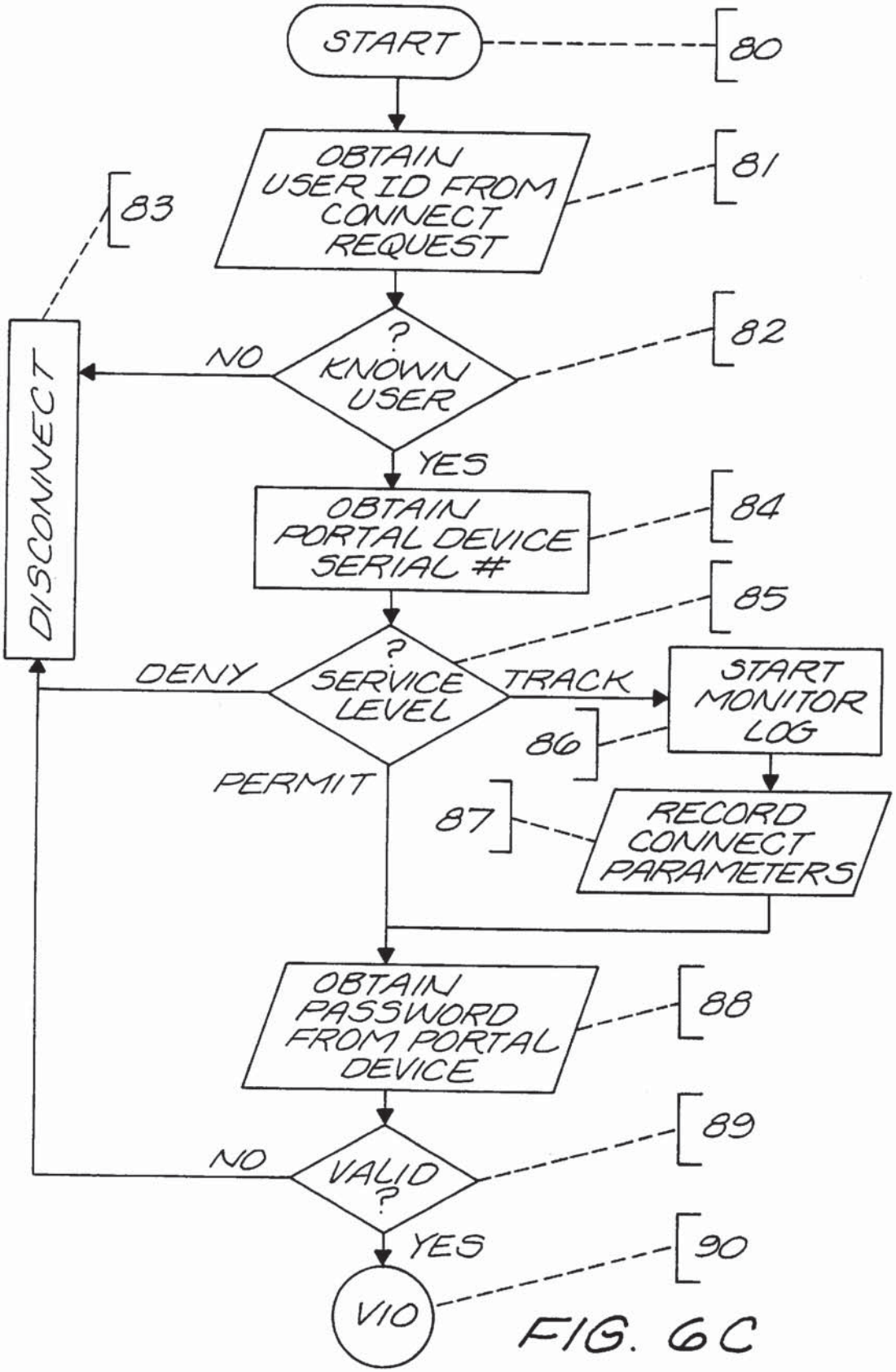

START
80
OBTAIN USER ID FROM CONNECT REQUEST
81
83
DISCONNECT
NO
KNOWN USER ?
82
YES
OBTAIN PORTAL DEVICE SERIAL #
84
85
DENY
SERVICE LEVEL ?
TRACK
START MONITOR LOG
86
PERMIT
87
RECORD CONNECT PARAMETERS
OBTAIN PASSWORD FROM PORTAL DEVICE
88
NO
VALID ?
89
YES
90
VIO

FIG. 6C

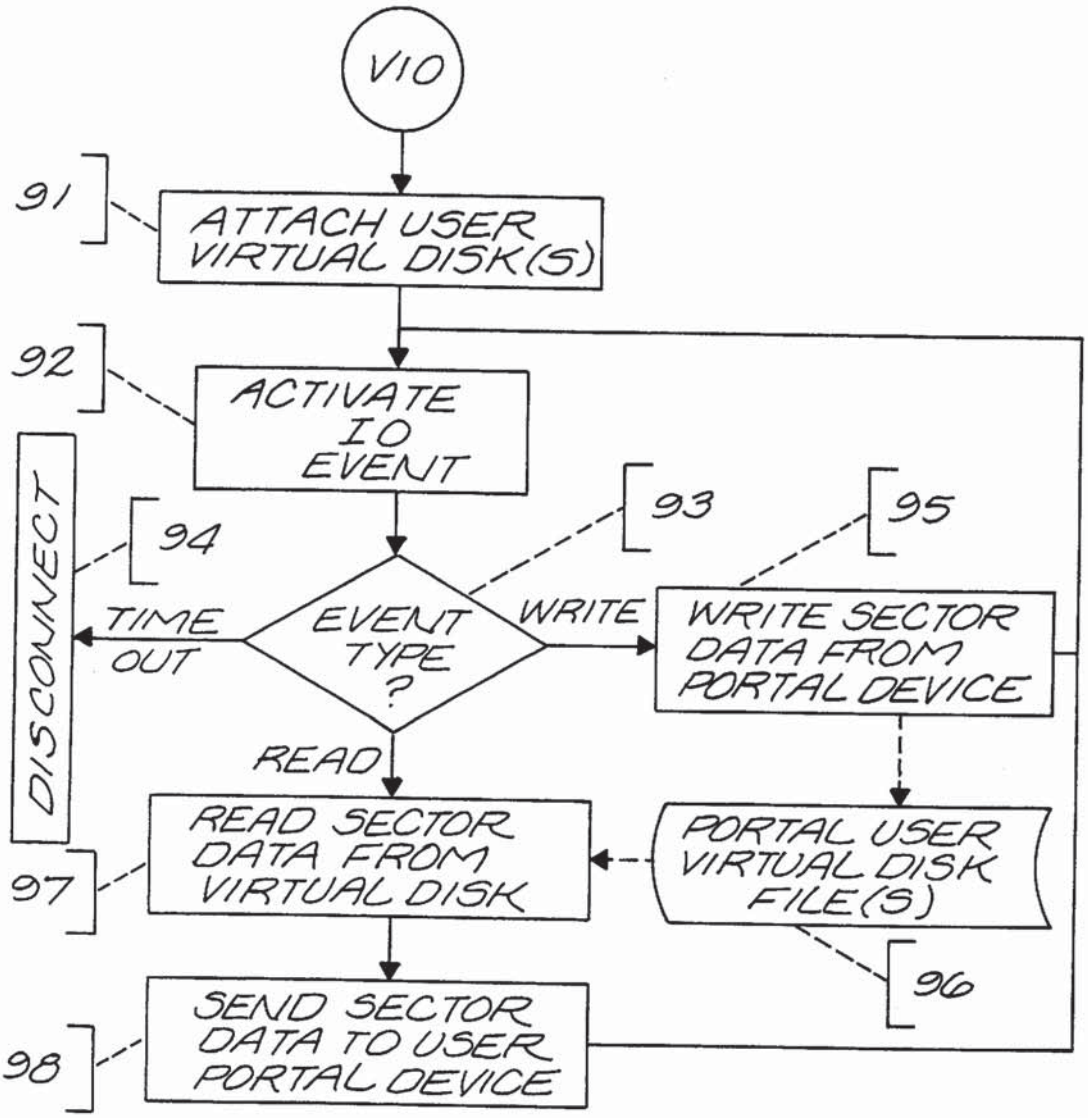

VIO
91
ATTACH USER VIRTUAL DISK(S)
92
ACTIVATE IO EVENT
DISCONNECT
94
93
95
TIME OUT
EVENT TYPE ?
WRITE
WRITE SECTOR DATA FROM PORTAL DEVICE
READ
READ SECTOR DATA FROM VIRTUAL DISK
97
PORTAL USER VIRTUAL DISK FILE(S)
96
SEND SECTOR DATA TO USER PORTAL DEVICE
98

FIG. 6D

USER PORTAL DEVICE FOR THE WORLD WIDE WEB TO COMMUNICATE WITH A WEBSITE SERVER

This is a continuation-in-part of United States Provisional Patent application Ser. No. 60/002,633, filed Aug. 22, 1995, and entitled "The WebBook—A WWW Surf Board".

BACKGROUND

This invention relates generally to computers and more particularly to computer based devices which are adapted for use on distributed network systems.

The World Wide Web (WWW) presents a user friendly face to the large quantities of information and community present on the Internet.

Currently, general purpose computer hardware and software are used to provide the user access to the WWW. The user employing this technology must be knowledgeable of many complex and technical factors that are not necessary if only WWW access is desired. Also, the user must pay for general purpose capabilities in his/her hardware and software that are unnecessary for WWW access.

The complexities and costs of current technical means of WWW access creates a knowledge and financial barrier to many who could benefit if a simple and inexpensive means of access were available.

It is clear that there is a need for a computer based device which capitalizes upon the strengths of the WWW and other distributed computer systems utilizing similar technologies.

SUMMARY OF THE INVENTION

This invention creates an improved usage of network and computer capabilities by positioning the network as the central element and viewing the various computer based devices as peripherals to the network. Within this overall configuration there are three kinds of computer based peripherals referred to as "user portal devices", "virtual disk server" and "web site server".

The virtual disk server provides user specific non-volatile storage for the user portal device through the network so that physical disk drives are not required on the user portal device. The web site server is any of the computers connected to the network that respond to stand WWW protocols and browsers.

The user portal device includes only a rudimentary standalone capability. In the preferred embodiment, the user portal device contains only read-only-memory (ROM) and random-access-memory (RAM) and no physical local disk.

In this preferred embodiment, the ROM contains the initial startup environment and a number of software object elements (SOEs) which are composed of program code and data. The SOEs are capable of behaviors which connect to a virtual disk server for the loading of SOEs maintained by and for the user. The SOEs executed predominately in the preferred embodiment's RAM and form its operating correlation or objects (OCO).

Without the assistance of a virtual disk server and its non-volatile storage, the user portal device is unable to form an OCO sufficiently capable of accessing web site servers and utilizing their content. In one embodiment of the invention, however, a physical disk can be locally added to the user portal device to execute the storage role of the virtual disk server.

Further, in one embodiment of the invention, the user portal device is able to obtain additional SOEs from a distributed computer system (such as the World Wide Web or Internet) once its OCO has been sufficiently established to support the technical protocols needed to effect such an access. The virtual disk server can be used for further connectivity to the network or the user portal device can go directly to a web site server without the virtual disk server's further support.

As desired and direct by the user of the user portal device, SOEs can be preserved in the user's storage maintained by the virtual disk server for later use in establishing a future OCO.

The present invention is a special purpose information appliance specifically designed to access the information resources of the WWW including SOEs present therein. The apparatus is a small portable unit consisting of an integral display and keyboard about the same size as a notebook computer. In one embodiment the integral display and keyboard are deleted and an external monitor and keyboard are used.

Preferably, the basic unit has between 8 and 32 megabytes of RAM and 1–4 megabytes of ROM. There is no local hard drive for non-volatile storage unless the user elects to install such capability.

The base user portal device is essentially useless unless it is connected to a network having at least a virtual disk server. The user has access to virtual disk drives that are independent of the specific user portal device being used. The provider of the virtual disk server may also elect to provide other virtual support for legacy operating systems such as Windows, MacOS and Unix variants should sufficient customers be interested in having these capabilities.

The Central Processing Unit (CPU) of the preferred embodiment is preferably inexpensive but computationally powerful. In this embodiment the processor is a stack based machine (SBM) rather than a register based machine (RBM). The invention relies on an SBM to effectively support the software technology used for the SOEs (Java) and to efficiently execute the SOEs forming the OCO.

In the preferred embodiment communication between the user portal device and the virtual disk server is protected by encryption. The keys used for this encryption are contained in a physical token such as a special removable ROM as indicated by the drawings. When a user connects with the virtual disk server standard account and password mechanisms are used to identify the user to the virtual disk server. After successful identification the virtual server connects the user portal device with the encrypted storage maintained for the user.

Virtual disk information is only realizable in-the-clear at the user portal device and only if the keys match those used to record the virtual disk information. The virtual disk server is unable to decrypt the user information being maintained in a virtual disk.

The user portal device has a numerically unique identification established a manufacturing time. This value is used by the virtual disk server to deny service to user portal devices that may have been stolen, misplaced or reported as destroyed.

The preferred embodiment of this invention includes a variety of capabilities:

1. The unit is a special purpose device produced only to exploit the information and SOEs available in the WWW and similar networks. It is therefore less expensive, simpler in both construction and operation, and more efficient than other more general purpose and complex technical means of utilizing the WWW.

2. The need for local physical disk drives is removed. The user saves SOEs and other information without worrying about data backups or disk space problems.

3. The user's information is protected and private. By employing an identity based key approach established by a physical token, the unit does not require the user to use a specific user portal device but can use any user portal device as defined by this invention. Local disk drives that the user may elect to install are also protected by encryption as well as mirrored by the virtual disk server. The built-in encryption and data protection capabilities makes this invention suitable for WWW based commerce and other sensitive applications.

4. The user portal device's operation is determined by the SOEs making up its OCO at any point in time. Because SOEs are assembled dynamically, application software updates are transparently applied to the user's environment. Also long term enhancements made to the capabilities being offered by Web site servers can be communicated dynamically through the network permitting users to remain current with the evolution of WWW utility.

5. The user portal device will efficiently execute SOEs because its CPU is a SBM. The software technology underlying the SOEs, Java, is also stack based so the user portal device will always have a technical performance advantage.

6. The user portal device is physically secure because its economic value can be reduced by denial of service from virtual disk servers once it reported stolen. Additionally, information kept on local disk drives is encrypted so that it will not be compromised if the unit is lost.

While those of ordinary skill in the art recognize a variety of configurations for the local computer, the following describes the specification for the preferred embodiment:

Processor	Patriot Scientific (ShBoom), 75–100 mhz;
Memory	4–8 MB DRAM, .5 MB ROM/EPROM;
Video	VGA 640 × 280 (Internal 6–7") LCD backlit (preferably color), VGA 640 × 280 (External - DB15), NTSC, Video - RCA Phono jack;
Keyboard	Internal - elastomer - 64–80 keys, External - AT standard;
Mouse	Internal - Trackpad + 3-button, External - PS/2 standard 2-button;
Sound	Input: Internal - Stereo Microphones, External - Stereo miniature phono jack, Output: Internal - Stereo Speakers, External - Stereo miniature phono jack, External - Single channel RCA Phono;
Authentication	Dallas Semiconductor "Touch Memory";
Network	Power line or UTP Ethernet;
Expandability	4 Type III PC Cards, 1 SIMM socket;
Power	AC Power adapter - 6V 1.0–1.5 A, Battery - 6V 1–2 AH with Motorola Cell phone compatible connector.

Another embodiment of the invention uses the following configuration:

Processor	Patriot Scientific (ShBoom), 100–150 mHz;
Memory	8–12 MB DRAM, .5 MB ROM/EPROM;
Video	SVGA 800 × 600 (External - DB15);
Keyboard	External - AT standard;
Mouse	External - PS/2 standard 2-button;
Sound	Input: Internal - Stereo Microphones External - Stereo miniature phono jack Output: Internal - Stereo Speakers External - Stereo miniature phono jack
Authentication	Dallas Semiconductor "Touch Memory";
Network	UTP Ethernet;
Expandability	4 Type III PC Cards, 1 SIMM socket;
Power	AC Power adapter - 6V 1.O–1.5 A.

Still another embodiment uses the following specification:

Processor	Patriot Scientific (ShBoom), 100–150 mHz;
Memory	8–12 MB DRAM, .5 MB ROM/EPROM;
Video	SVGA 800 × 600 (Internal 6–7") Color LCD backlit, SVGA 800 × 600 (External - DB15), NTSC Video - RCA Phono jack;
Keyboard	Internal - elastomer - 64–80 keys, External - AT standard;
Mouse	Internal - Trackpad + 3-button, External - PS/2 standard 2-button;
Sound	Input: Internal - Stereo Microphones External - Stereo miniature phono jack, Output: Internal - Stereo Speakers External - Stereo miniature phono jack External - Single channel RCA Phono;
Authentication	Dallas Semiconductor "Touch Memory";
Network	Power line or UTP Ethernet;
Expandability	4 Type III PC Cards, 1 SIMM socket;
Power	AC Power adapter - 6V 1.0–1.5 A, Battery - 6V 1–2 AH, Motorola Cell phone compatible connector.

The invention, together with various embodiments thereof, will be more fully explained by the attached drawings and the following descriptions.

DRAWINGS IN BRIEF

FIG. **1** is a perspective view of an embodiment of the invention illustrating its linkage with the World Wide Web.

FIG. **2** is a block diagram of the preferred embodiment's electronic interconnection.

FIGS. **3**A, **3**B, and **3**C are frontal, perspective (open), and perspective (closed) views respectively of the preferred embodiment of the invention.

FIG. **4** is a frontal view of an alternative embodiment of the invention.

FIGS. **5**A and **5**B are schematic and exploded views respectively of an alternative embodiment of the invention.

FIG. **6**A is a flow-chart of the operation of the preferred local computer.

FIG. **6**B is a flow-chart of the operation of the preferred access provider computer.

FIG. **6**C is a flow-chart of the virtual disk server responding to a connection.

FIG. **6**D is a flow-chart for the operation of the virtual disk server once a user has been validated.

DESCRIPTION OF FIGURES

FIG. **1** is a perspective view of an embodiment of the invention illustrating its linkage with the World Wide Web.

Local Computer **10** is easily held by operator **11** and is connected to the World Wide Web **16** via a PCMCIA device

(i.e. modem **12**) dealing through the Internet Service Provider (ISP) **17**.

Operator **11** identifies himself via key **13** which, in this embodiment, scans the fingerprint of the user **11** for proper identification. Other forms of key **13** are obvious to those having ordinary skill in the art.

Screen **14** is used to visually communicate information to user **11** while keyboard **15** is used for operator entry of data and instructions.

Battery **18** provides power for Local Computer **10**.

FIG. **2** is a block diagram of the preferred embodiment's electronic interconnection.

At the center of the operation is CPU **20** which receives operator data via keyboard **15** once proper operator identification has been established using data from key **13**.

Operational instructions for CPU **20** are utilized from a variety of sources including Read Only Memory (ROM) **21**, Random Access Memory (RAM) **22**, and via communications **23**. Note, no hard-drive is included as data storage normally associated with an on-board hard-drive is kept on the ISP **17**.

Communications **23** acts as the interface between CPU **20** and the ISP **17**. Communications **23**, in this embodiment includes a modem for telephone connection to ISP **17**.

Further, Communications **23** also is the interface with optional keyboard **24** which communicates, in this embodiment using infrared link **25**.

This entire system is readily portable and provides quick and secure access to the World Wide Web.

FIGS. **3**A, **3**B, and **3**C are frontal, perspective (open), and perspective (closed) views respectively of the preferred embodiment of the invention.

Local computer **10**A has as its primary communication mechanisms, screen **14**A and speakers **33**. Operator data is entered into local computer **10**A via keyboard **15**A, mouse **31**, microphone **34**, and pointing device **30**.

In this embodiment, keyboard **15**A is a membrane type in which individual keys are activated using a pen tip or other similar tool.

External data is received via interconnect **12**A which communicates, in the preferred embodiment, via the phone lines with the access provider computer.

Logging into the access provider computer and for activating local computer **10**A is accomplished via fingerprint key **13**A. Fingerprint key **13**A scans the user's fingerprint and correlates this to a file which is stored within the non-volatile memory of the access provider computer. In one embodiment, the "fingerprint" data serves as a "key" to de-crypt the data stored within the non-volatile memory.

Battery **18**A provides the electrical power for the entire local computer **10**A and is ideally rechargeable.

Cover **32** serves as a cradle (as illustrated in FIG. **3**B) and also as a protective cover for the face of local computer **10**A (as shown in FIG. **3**C).

FIG. **4** is a frontal view of an alternative embodiment of the invention.

This embodiment of the local computer **10**B incorporates the use of screen **14**B, speakers **33**, microphone **34**, pointing device **30** and authentication/key generation **13**A.

This embodiment of the invention also incorporates the ability to accept keyed input not only from membrane keyboard **15**A but also from a full sized keyboard **45** which communicates with the local computer **10**B via an infrared link **46**. This linkage of a full-sized keyboard **45**, in conjunction with linkage with a full-sized monitor via VGA output **44**, permits the user to easily enter data for use by the local computer.

In this application, where the user has linked a full sized monitor and a full sized keyboard, local computer **10**B is able to down-load from the access provider computer (or from another source on the Internet) software to operate a variety of programs including: word-processing, spread sheets, and computer-aided-design. These programs would not be resident on local computer **10**B but would be obtained when needed and would utilize the non-volatile memory of the access provider computer.

In this context, the access provider computer is a computer which is accessible to local computer **10**B, whether the access provider computer is linked via a direct phone line (a dedicated communications channel) or through the internet/World Wide Web.

Other attributes of this embodiment include the ability to accept external power **40** to either operate the apparatus or to recharge the internal batteries (not shown).

Game ports **41** permit the local computer **10**B to interface with a variety of apparatus.

Audio and visual input and output is accomplished in this embodiment via stereo In/Out **43**, TV input **42**A and TV output **42**B. These provide another source of communication from the local computer **10**B and the operator as well as mechanisms for the acceptance of input data which is storable in the non-volatile memory of the access provider computer (not shown).

FIGS. **5**A and **5**B are schematic and exploded views respectively of an alternative embodiment of the invention.

Referring to FIG. **5**A, a block diagram of an embodiment of the invention, communication to and from the unit is centralized through an Input/Output (I/O) apparatus **52**. I/O **52** accepts data input from a variety of sources including interconnects PCMCIA III **12**B, microphones **34**, mouse/trackpad **30**A, external mouse and keyboard **46**, and the encryption touch memory **13**A. Output from I/O **52** is communicated to speakers **33**, the internal VGA driver **53**, external VGA **44**, and interconnects **12**B.

CPU **50** is preferably a commercially available microprocessor with enhanced capabilities and with extensive computing power. CPU **50** receives and transmits its data for external use via I/O **52**. Internally used data is stored in memory **51** which, in this embodiment, consists only of a Read-Only-Memory (ROM) and a Random-Access-Memory (RAM). While the ROM size is kept extremely modest, preferably at only 512k, the RAM size is extended to four megabytes to give CPU **50** maximal operating capabilities.

With reference to FIG. **5**B, a view of six different sides of the alternative embodiment, the location of the various interconnects is shown.

Speakers **33** and microphones **34** are located on each side of screen **53**. At three corners are located the authentication keypad **13**A, touchpad **30**A, and three way mouse **31**A. Below screen **53** is the membrane keyboard **15**A.

On both the right and left side panels are located interconnects **12**B which allow the apparatus to be connected to a variety of peripheral mechanisms.

The front panel contains a connector **57** for a remote keyboard to be attached to the computer.

The rear panel includes audio-in connectors **55**, audio-out connectors **56**, television-in connector **42**A, and television-out connector **42**B. On/off switch **54** is used to start and stop the machine.

FIG. **6**A is a flow-chart of the operation of the user portal device from boot-up to connection with a virtual disk server.

After start **60**, the boot-up program **61***a* is loaded from the portal ROM **61***b* and executed by the portal computer **62**.

The boot-up program creates the environment for the software object elements that make up the operating correlation of objects and loads **63***b* the initial set of software object elements **63***a* from the portal ROM **61***b*. The boot-up program then activates the operating correlation of objects **64** which then control the remaining operation of the portal device. The software object elements are the same kind of artifacts as the software object elements originating from either the virtual disk server or from Web site servers.

In the preferred embodiment, the operating correlation of objects then obtains the encryption keys **65**A stored in the security token ROM **65**B. These keys are used to encrypt and decrypt the information stored by the virtual disk server for the user.

The user ID and password are queried from the operator **66** and the connection is then made with the virtual disk server **67**.

FIG. **6**B is a flow-chart of the operation of the user portal device after connection has been made with the virtual disk server.

The user portal device operates as a continuation **69** from the previous operation. That is, a connect is treated as a restoration of service after an interruption.

Software object elements needed for the operation of the user portal device are loaded **70** from the user's virtual disk **71** into the RAM in the portal device **72**. These software object elements are those not already present in the user portal device or newer elements than those present in the portal device.

Interaction with Web site servers continues in response to operator actions and preferences **73**. Data and software object elements from this interaction are recorded temporarily in the portal device RAM **72**. Software object elements compose not only the operating correlation of objects that act like an operating system for the user portal device but also provide functionality similar to various conventional application programs such as Web browsers, word processors, spread sheets and others.

The user portal device continuously presents the state of its operating correlation of objects to the operator **74**. Updated software object elements and data in the portal device RAM **75***b* are store in the virtual disk **75***c* by the operating correlation in the portal device **75***a*.

The operating correlation then loops **76** back to the beginning of FIG. **6**B.

FIG. **6**C is a flow-chart of the virtual disk server responding to a connection.

After start **80**, the user ID is extracted from the connection request **81** and compared with a list of know users **82**. If the user is not know then the virtual disk server disconnects **83**.

If the user ID is known to the virtual disk server, the serial number of the user portal device is examined **84** to determine the service level defined for the device. Decision **85** determines if service is to be denied (eg, stolen unit) then the virtual disk server disconnects **83**. If the service level is for tracking (eg, trying to locate a missing portal device based on usage) a monitor log is started **86** for the connection and the connect parameters (such as originating phone number is dial-in access is being made) recorded **87**.

If the service level is permit or tracking the virtual disk server then requests the password from the portal device **88**. The password is check for validity **89** and if it is invalid a disconnect occurs **83**; otherwise, the program connects **90** to FIG. **6**D.

If the password is correct then FIG. **6**D applies.

FIG. **6**D is a flow-chart for the operation of the virtual disk server once a user has been validated.

The user is attached to the virtual disk(s) **91**. This peruser non-volatile storage will typically be files on the computer system hosting the virtual disk server.

After attachment, the virtual disk server will await an input/output event from the user portal device **92**. After a period of time without any event **93**, the virtual disk server will disconnect **94**.

If the event type **93** is a write it is accompanied by the encrypted data to be written **95**. The data is placed in the virtual disk of the user **96**. If the event type **93** is a read, the virtual disk server reads the data from the virtual disk **97** and sends it to the user portal device **98**.

After processing an input-output event, the virtual disk server awaits the next event **92**.

In this manner, the operating correlation of objects is supplemented and increased by interacting with Web site servers in response to user preferences and operations. The user portal device grows in capability and utility over time and is not limited to the software packaged with the portal device or by the software object elements stored by the virtual disk server.

It is clear that the present invention creates a highly improved means for accessing and using the world wide Web.

What is claimed is:

1. A system of computers comprising:

a) a communication system for transmitting electronic data from one location to another location;

b) an access provider computer having,

1) a non-volatile memory having stored therein,

A) data defining software programs requiring use of the non-volatile memory, and,

B) remotely generated data,

2) an access provider operational means for,

A) receiving data from said communication system,

B) communicating selected groupings of said software programs and remotely generated data to said communication system, and,

C) storing in said non-volatile memory, data received from said communication system as remotely generated data; and,

c) a hand-held local computer having,

1) a volatile memory means for storing a software program,

2) communication means for accessing said communication system,

3) operator input means for establishing operator generated data,

4) a local computer operational means for,

A) via said communication means, receiving a selected software program from the non-volatile memory of said access provider computer system,

B) storing said selected software program in said volatile memory means,

C) executing said software program in said volatile memory, and,

D) via said communication means, communicating selected data generated by said software program and said operator input means to said access

provider computer for storage in said non-volatile memory of said access provider computer.

2. The system according to claim **1** wherein:

a) the non-volatile memory of said access provider computer includes data defining authorized serial numbers, and identification data; and,

b) the access provider operational means includes means for comparing received data to said authorized serial numbers and identification data from said non-volatile memory means.

3. The system according to claim **2** wherein the local computer operational means of said hand-held local computer includes means for communicating operator generated data to said access provider computer via said communication system.

4. The system according to claim **1** wherein said hand-held local computer further includes a read-only memory containing programs and identification data stored therein, and wherein said operator generated data includes identification data stored within said read-only memory.

5. The system according to claim **1** further including a power source contained within said hand-held local computer for powering said volatile memory, said communications means, and said operational means.

6. The system according to claim **5** wherein said hand-held local computer further includes:

a) a visual display apparatus for communication of data from said operational means to an operator; and,

b) a phone interconnect member for connecting a phone line to said communication means.

7. The system according to claim **1** wherein said hand-held local computer further includes a read-only-memory containing data indicative of a boot-up program, and wherein said operational means of said hand-held local computer includes means for:

a) installing said boot-up program from said read-only-memory into said volatile memory; and,

b) executing said boot-up program in said volatile memory.

8. The system according to claim **7** wherein said volatile memory and said read-only-memory constitute an entire data memory of said hand-held local computer.

9. A system of computers comprising:

a) a network of computers providing electronic data;

b) a network communication system for transmitting electronic data from one computer to another computer within said network of computers;

c) a dedicated communication means for communicating data from a single computer to another single computer;

d) an access provider computer having,

1) a non-volatile memory having stored therein data defining software programs requiring use of the non-volatile memory, and remotely generated data,

2) network accessing means for retrieving data from and communicating data to said network of computers via said network communication system,

3) an access provider operational means for,

A) receiving and transmitting data from and to said communication system,

B) communicating selected groupings of said software programs and remotely generated data to a remote computer via said dedicated communication means, and,

C) storing in said non-volatile memory, data received from said dedicated communication means as remotely generated data; and,

e) a hand-held local computer having,

1) a volatile memory for storage of a software program,

2) communication means for accessing said dedicated communication means,

3) operator input means for establishing operator generated data, and,

4) a local computer operational means for,

A) via said dedicated communication means, receiving a selected software program from said access provider computer system,

B) storing said selected software program in said volatile memory,

C) executing said software program in said volatile memory,

D) via said access provider computer, receiving data from said network of computers, and,

E) via said dedicated communication means, communicating selected data generated by said software program, said operator input means, and said data from said network of computers, to said access provider computer for storage in said non-volatile memory of said access provider computer.

10. The system of computers according to claim **9** wherein:

a) the non-volatile memory of said access provider computer includes data defining authorized serial numbers, user identifications; and,

b) the access provider operational means of said access provider computer includes means for comparing received data from said hand-held local computer to said authorized serial numbers.

11. The system of computers according to claim **10** wherein the local computer operational means of said hand-held local computer includes means for communicating operator generated data to said computer system via said dedicated communication means.

12. The system of computers according to claim **11** wherein said hand-held local computer further includes a read-only-memory containing data and programs and wherein said operator generated data includes identification data stored within said read-only-memory.

13. The system of computers according to claim **9** wherein said hand-held local computer further includes:

a) a visual display apparatus connected to said operational means for communicating data from said operational means to an operator; and,

b) a modem communicating between said operational means of said hand-held local computer and said dedicated communication means.

14. The system of computers according to claim **9** further including a read-only-memory containing a boot-up program and wherein said operational means includes means for:

a) installing said boot-up program from said read-only-memory into said volatile memory; and,

b) executing said boot-up program in said volatile memory.

15. The system of computers according to claim **14** wherein said volatile memory and said read-only-memory constitute an entire data memory of said hand-held local computer.

* * * * *

US006073142A

United States Patent [19]

Geiger et al.

[11] Patent Number: 6,073,142

[45] Date of Patent: Jun. 6, 2000

[54] **AUTOMATED POST OFFICE BASED RULE ANALYSIS OF E-MAIL MESSAGES AND OTHER DATA OBJECTS FOR CONTROLLED DISTRIBUTION IN NETWORK ENVIRONMENTS**

[75] Inventors: **Fred J. Geiger**, Park City; **William K. Wood**, West Weber; **Sonjaya T. Tandon**, Park City, all of Utah

[73] Assignee: **Park City Group**, Park City, Utah

[21] Appl. No.: **08/881,034**

[22] Filed: **Jun. 23, 1997**

[51] **Int. Cl.**[7] **G06F 7/00**

[52] **U.S. Cl.** **707/500**; 709/204

[58] **Field of Search** 707/500; 709/204, 709/205, 206

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,106,060	8/1978	Chapman, Jr.	358/402
5,204,939	4/1993	Yamazaki et al.	706/50
5,283,856	2/1994	Gross et al.	395/51
5,555,346	9/1996	Gross et al.	706/45
5,619,648	4/1997	Canale et al.	395/200.01
5,632,011	5/1997	Landfield et al.	395/326
5,675,733	10/1997	Williams	395/200.01
5,768,505	6/1998	Gilchrist et al.	395/200.31
5,796,394	8/1998	Wicks et al.	345/329

OTHER PUBLICATIONS

Amadi, A. O., "Automatic Filing and Retrievel of Offical Messages Using Global Mail Attributes and a Viewdata System with Symbollically Named Pages," Office Information Systems, pp. 11–18, Oct. 1988.

Ayre, R., "Evolving E–Mail: Will Client/Server Get the Message?," PC Magazine, vol. 10, No. 15, pp. 322–323, Sep. 10, 1991.

Berlind, D., "Don't Allow Rules of E–Mail to Get Broken," PC Week, vol. 12, No. 3, p. 55, Jan. 23, 1995.

Buyers Guide, "Choosing An E–Mail Package," PC User, No. 264, p. 31, Jul. 26, 1995.

Kramer, M., "Mail Scout Motto: Filter E–Mail," PC Week, vol. 12, No. 23, p. 65, Jun. 12, 1995.

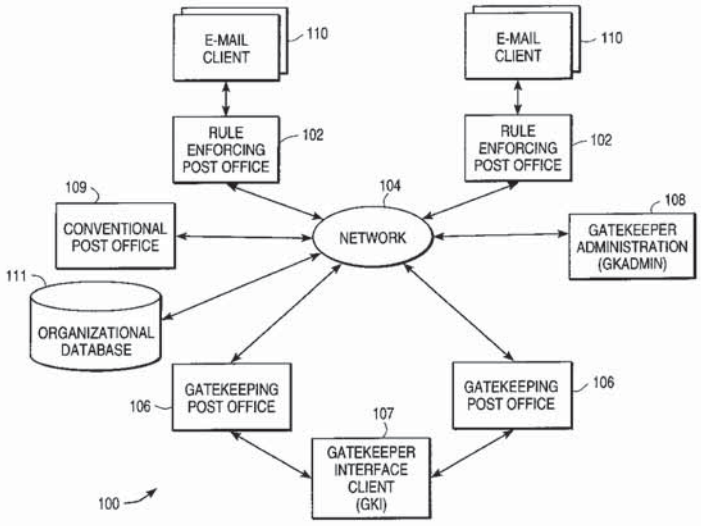
(List continued on next page.)

Primary Examiner—Stephen S. Hong
Assistant Examiner—Cesar B. Paula
Attorney, Agent, or Firm—Fenwick & West LLP

[57] **ABSTRACT**

A system, method and various software products provide for automatic deferral and review of e-mail messages and other data objects in a networked computer system, by applying business rules to the messages as they are processed by post offices. The system includes rule enforcing post offices that store a plurality of business rules derived from business communication policies. The rule enforcing post offices receive messages from client applications and from other post offices and apply the business rules with a rule engine. The rule engine determines a set of actions, specified by business rules that are fired, to be applied to each message. The rule engine provides the actions to a distribution engine, which executes a highest priority action. Actions include releasing, deleting, returning, forwarding, or gating the message. Gating forwards the message to a gatekeeper, an administrator assigned to review messages for conformity with business policies or for other reasons. The gated messages are received by the gatekeeper at a gatekeeping post office. A gatekeeper can review the gated messages, and then manually release, delete, return, or further gate the message. Alternatively, if the gatekeeper does not review a gated message with a specified time period, the message is automatically reviewed by the gatekeeping post office with its own set of business rules. Having multiple post offices with independent sets of business rules allows for distributed and hierarchical review and gating of the messages. The system can route any type of data object, and apply the business rules to such objects in a similar manner.

26 Claims, 23 Drawing Sheets



OTHER PUBLICATIONS

Marshak, D. S., "Separating the Wheat from the Chaff," Patricia Seybold's Office Computing Report, vol. 13, No. 11, pp. 1–16, Nov. 1990.

Pollock, S., "A Rule–Based Message Filtering System," ACM Transactions on Office Information Systems, vol. 6, No. 3, pp. 232–254, Jul. 1988.

Rizzo, J., "Macs to Achieve cc:Mail Parity," MacUser, vol. 11, No. 9, p. 109, Sep. 1995.

Rosenberg, J., Everhart, C. F., and Borenstein, N. S., "An Overview of the Andrew Message System," Information Technology Center, Carnegie Mellon University, pp. 99–108, Jul. 1987.

Turner, L., "Rules for Messaging," Network World, p. 65, Mar. 20, 1995.

Willis, D., "Messaging Stand–Off—Lotus Notes 4.1 Battles Microsoft Exchange Server In A Clash Of Messaging Titans," Network Computing, p. 50, Jun. 15, 1996.

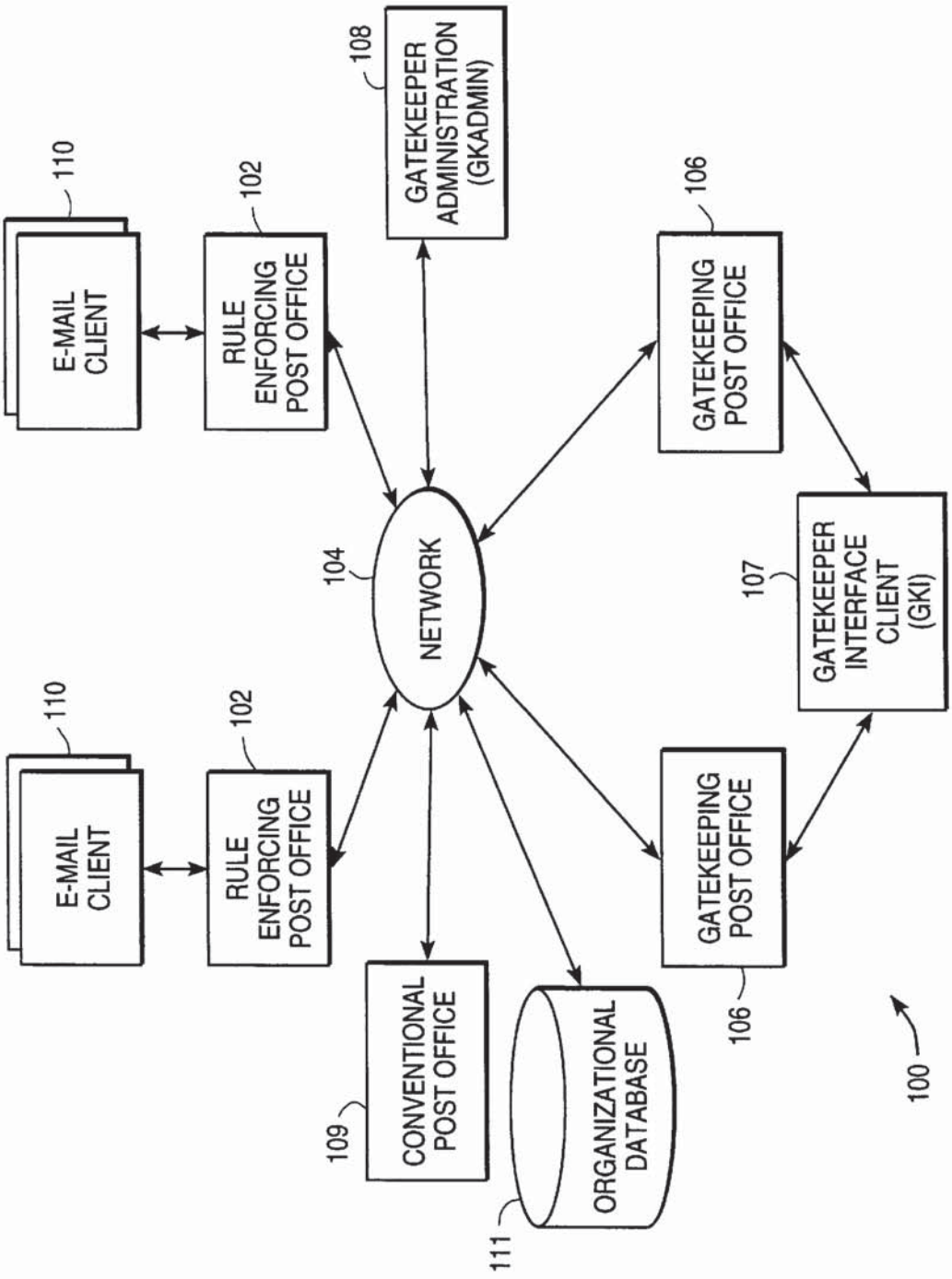

E-MAIL CLIENT
110
RULE ENFORCING POST OFFICE
102
GATEKEEPER ADMINISTRATION (GKADMIN)
108
GATEKEEPING POST OFFICE
106
NETWORK
104
GATEKEEPER INTERFACE CLIENT (GKI)
107
E-MAIL CLIENT
110
RULE ENFORCING POST OFFICE
102
GATEKEEPING POST OFFICE
106
CONVENTIONAL POST OFFICE
109
ORGANIZATIONAL DATABASE
111
100

FIG. 1

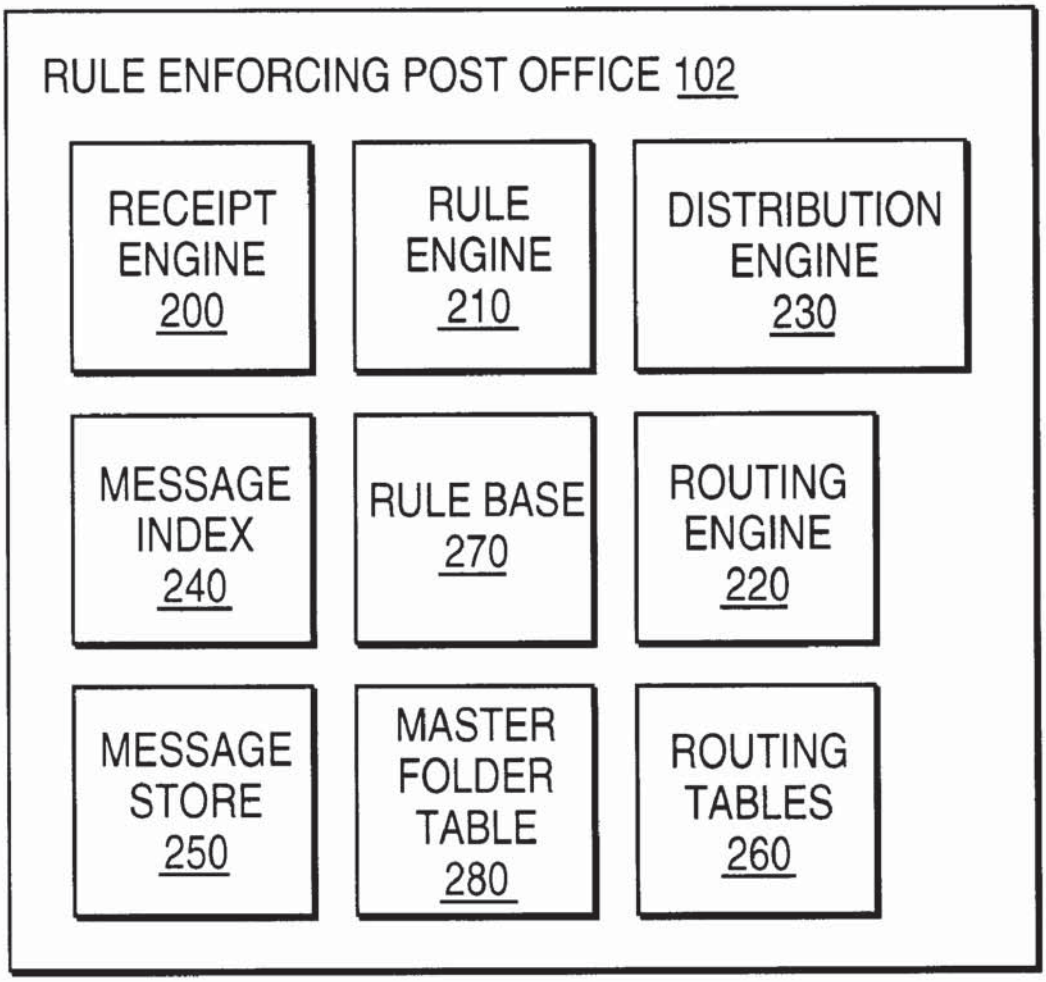

RULE ENFORCING POST OFFICE 102
RECEIPT ENGINE 200
RULE ENGINE 210
DISTRIBUTION ENGINE 230
MESSAGE INDEX 240
RULE BASE 270
ROUTING ENGINE 220
MESSAGE STORE 250
MASTER FOLDER TABLE 280
ROUTING TABLES 260

FIG. 2

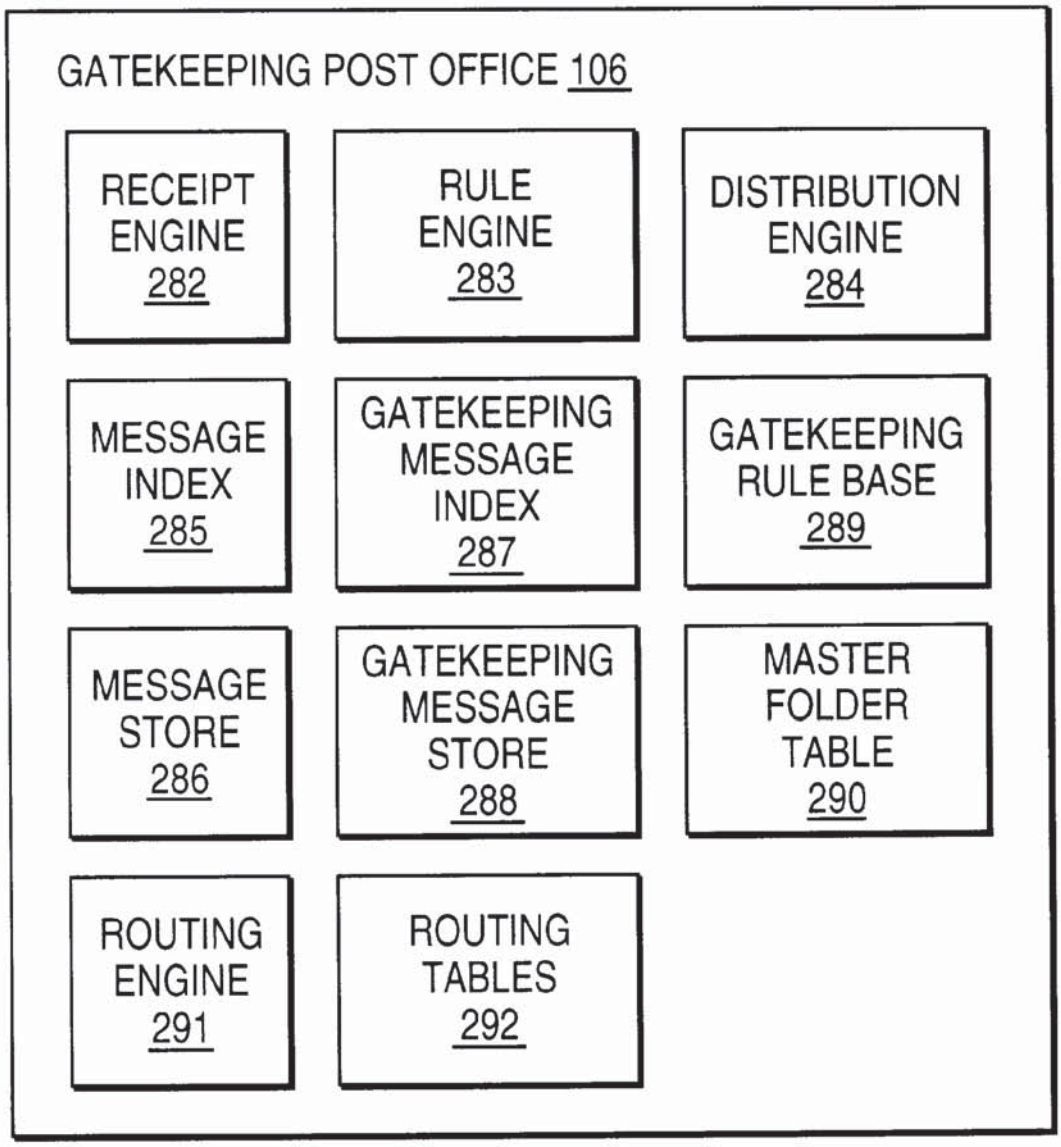

GATEKEEPING POST OFFICE 106
RECEIPT ENGINE 282
RULE ENGINE 283
DISTRIBUTION ENGINE 284
MESSAGE INDEX 285
GATEKEEPING MESSAGE INDEX 287
GATEKEEPING RULE BASE 289
MESSAGE STORE 286
GATEKEEPING MESSAGE STORE 288
MASTER FOLDER TABLE 290
ROUTING ENGINE 291
ROUTING TABLES 292

FIG. 3

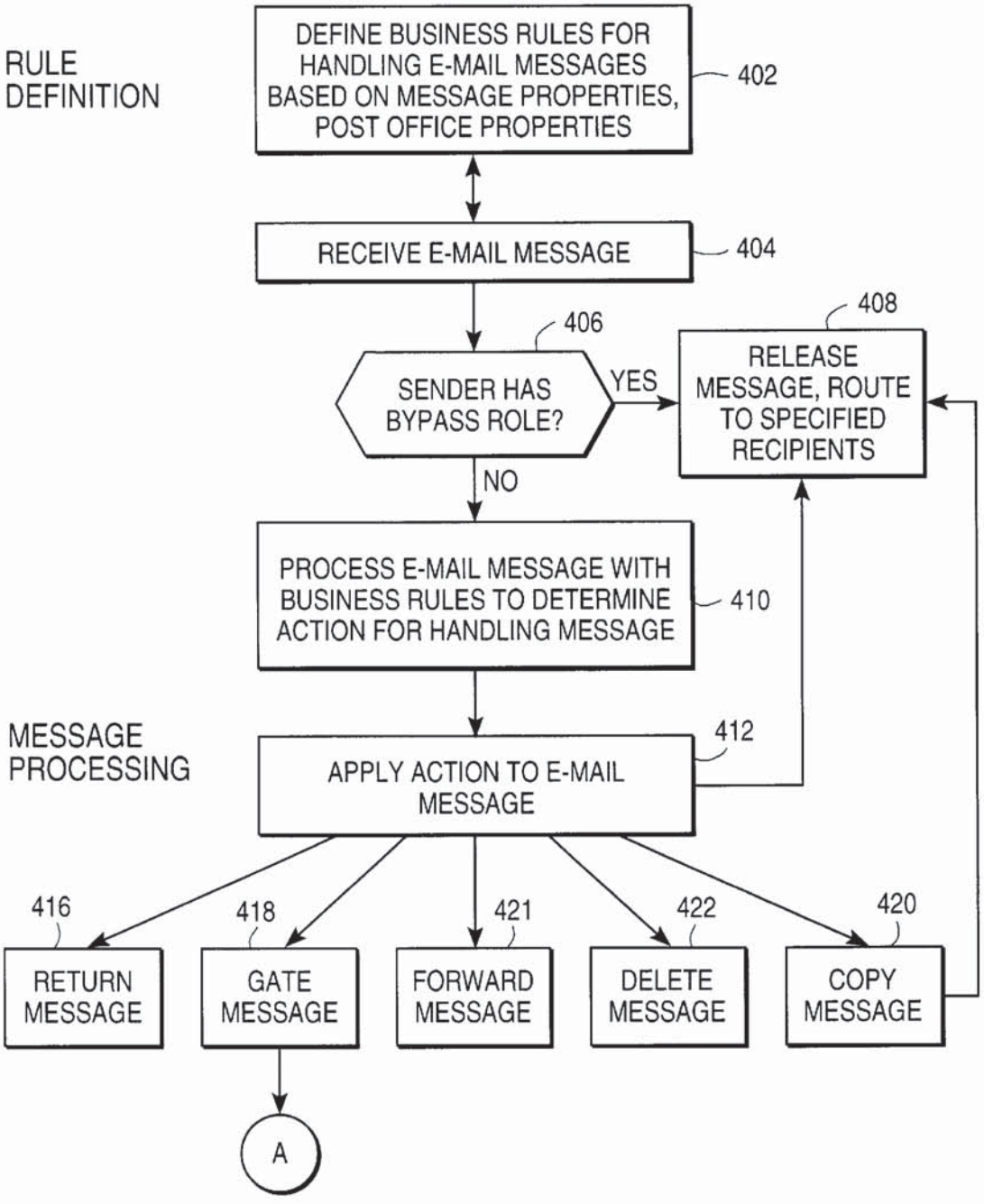

RULE DEFINITION
DEFINE BUSINESS RULES FOR HANDLING E-MAIL MESSAGES BASED ON MESSAGE PROPERTIES, POST OFFICE PROPERTIES
402
RECEIVE E-MAIL MESSAGE
404
406
SENDER HAS BYPASS ROLE?
YES
408
RELEASE MESSAGE, ROUTE TO SPECIFIED RECIPIENTS
NO
PROCESS E-MAIL MESSAGE WITH BUSINESS RULES TO DETERMINE ACTION FOR HANDLING MESSAGE
410
MESSAGE PROCESSING
412
APPLY ACTION TO E-MAIL MESSAGE
416
RETURN MESSAGE
418
GATE MESSAGE
421
FORWARD MESSAGE
422
DELETE MESSAGE
420
COPY MESSAGE
A

FIG. 4A

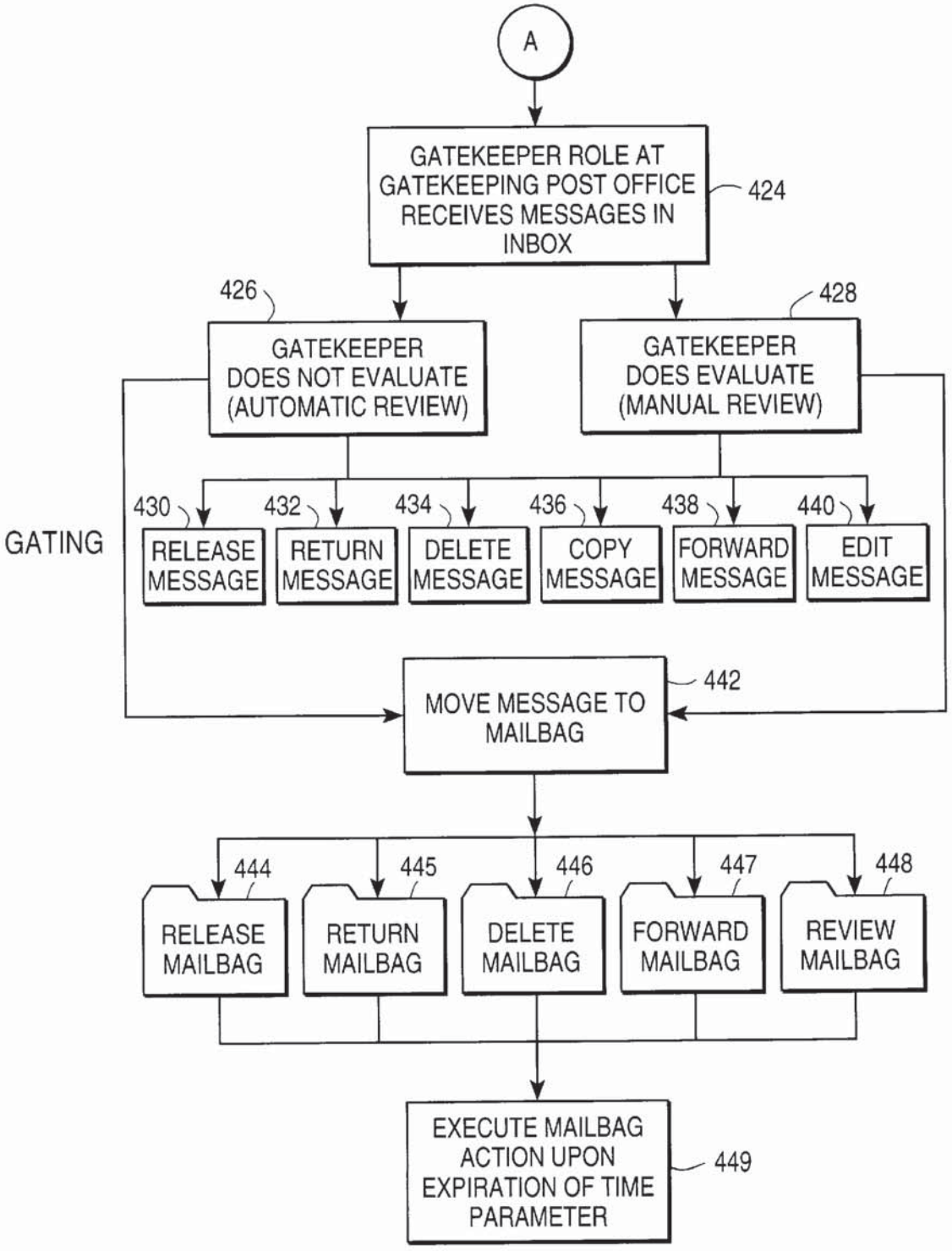
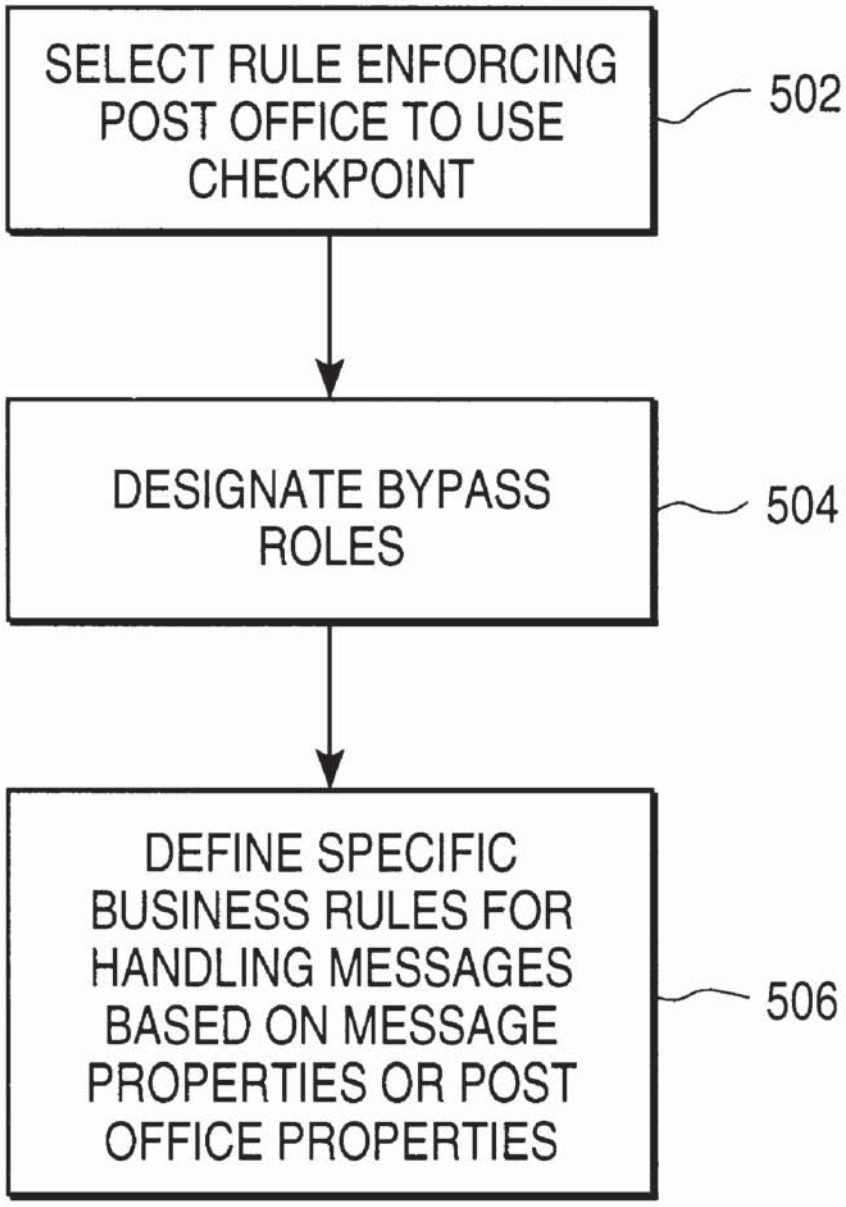

A
GATEKEEPER ROLE AT GATEKEEPING POST OFFICE RECEIVES MESSAGES IN INBOX
424
426
GATEKEEPER DOES NOT EVALUATE (AUTOMATIC REVIEW)
428
GATEKEEPER DOES EVALUATE (MANUAL REVIEW)
GATING
430
RELEASE MESSAGE
432
RETURN MESSAGE
434
DELETE MESSAGE
436
COPY MESSAGE
438
FORWARD MESSAGE
440
EDIT MESSAGE
MOVE MESSAGE TO MAILBAG
442
444
RELEASE MAILBAG
445
RETURN MAILBAG
446
DELETE MAILBAG
447
FORWARD MAILBAG
448
REVIEW MAILBAG
EXECUTE MAILBAG ACTION UPON EXPIRATION OF TIME PARAMETER
449

FIG. 4B



CONFIGURING A RULE ENFORCING POST OFFICE

FIG. 5

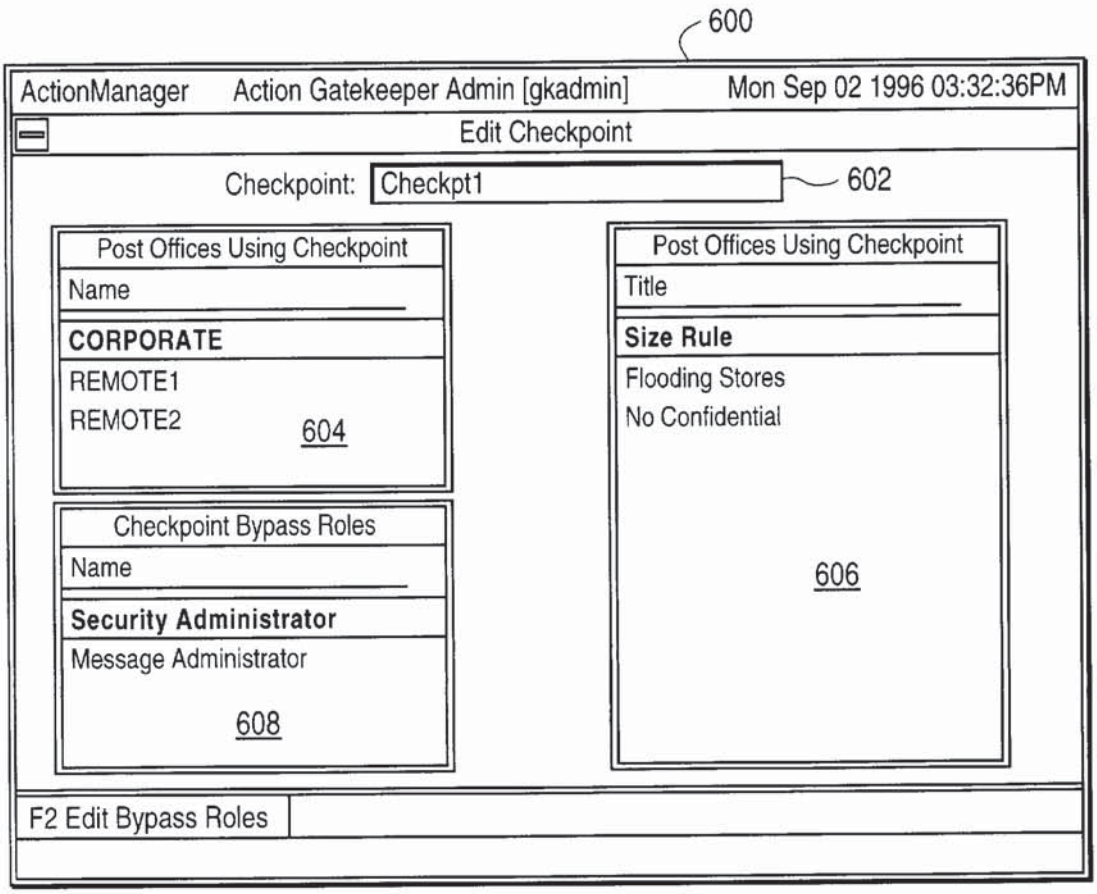

600
ActionManager
Action Gatekeeper Admin [gkadmin]
Mon Sep 02 1996 03:32:36PM
Edit Checkpoint
Checkpoint:
Checkpt1
602
Post Offices Using Checkpoint
Name
CORPORATE
REMOTE1
REMOTE2
604
Post Offices Using Checkpoint
Title
Size Rule
Flooding Stores
No Confidential
606
Checkpoint Bypass Roles
Name
Security Administrator
Message Administrator
608
F2 Edit Bypass Roles

FIG. 6

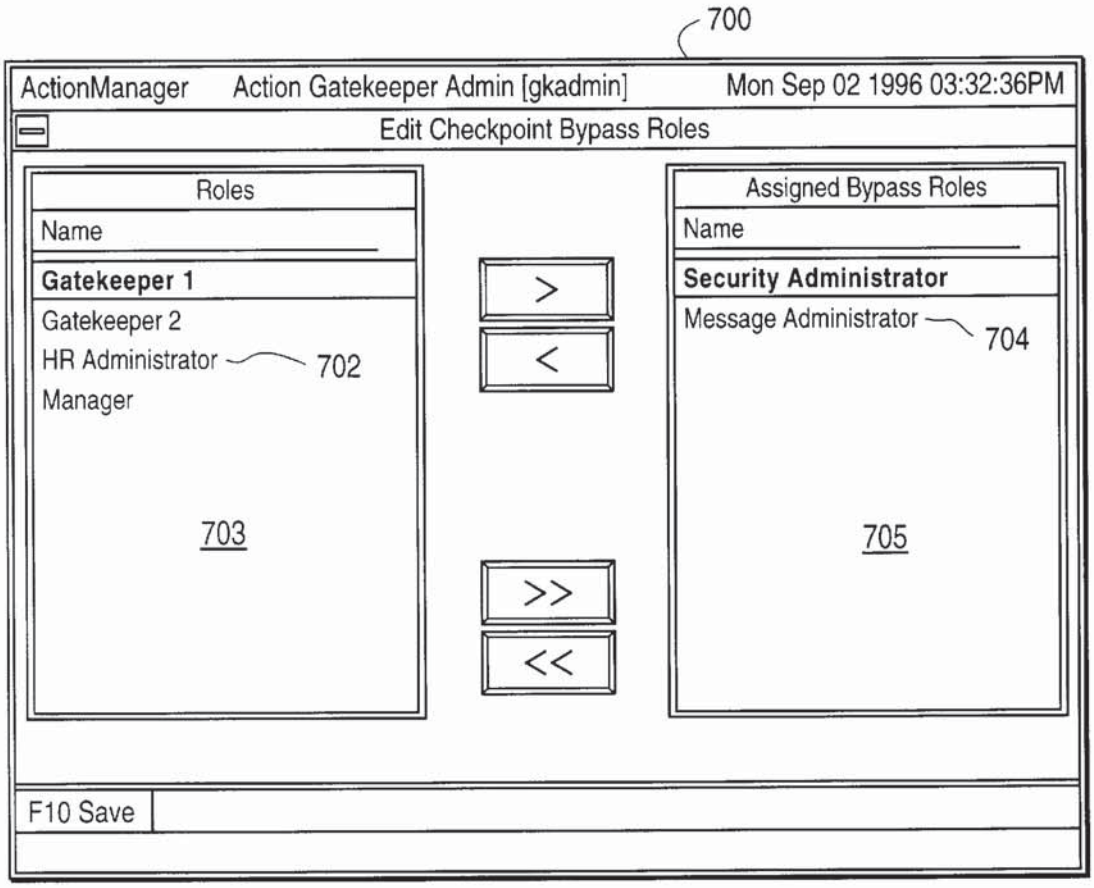

700
ActionManager
Action Gatekeeper Admin [gkadmin]
Mon Sep 02 1996 03:32:36PM
Edit Checkpoint Bypass Roles
Roles
Name
Gatekeeper 1
Gatekeeper 2
HR Administrator
702
Manager
703
>
<
>>
<<
Assigned Bypass Roles
Name
Security Administrator
Message Administrator
704
705
F10 Save

FIG. 7

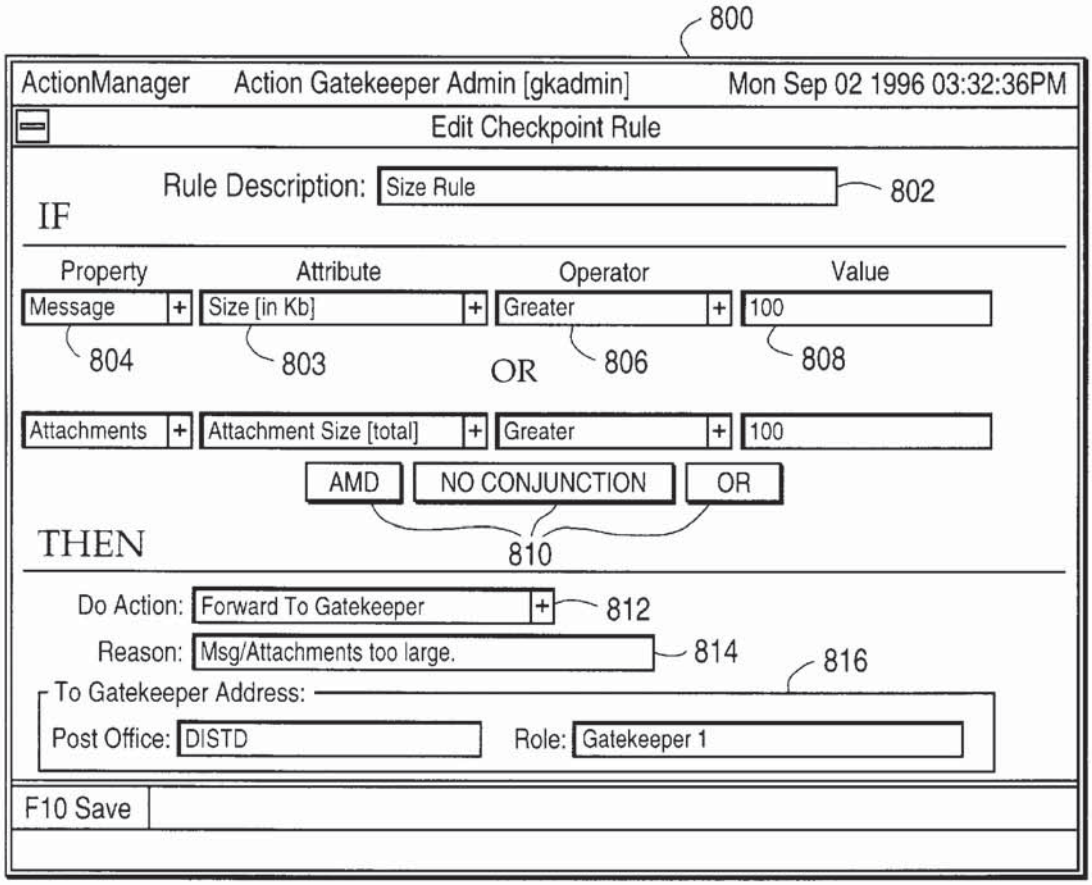

800
ActionManager
Action Gatekeeper Admin [gkadmin]
Mon Sep 02 1996 03:32:36PM
Edit Checkpoint Rule
Rule Description:
Size Rule
802
IF
Property
Attribute
Operator
Value
Message
Size [in Kb]
Greater
100
804
803
OR
806
808
Attachments
Attachment Size [total]
Greater
100
AMD
NO CONJUNCTION
OR
810
THEN
Do Action:
Forward To Gatekeeper
812
Reason:
Msg/Attachments too large.
814
816
To Gatekeeper Address:
Post Office:
DISTD
Role:
Gatekeeper 1
F10 Save

FIG. 8

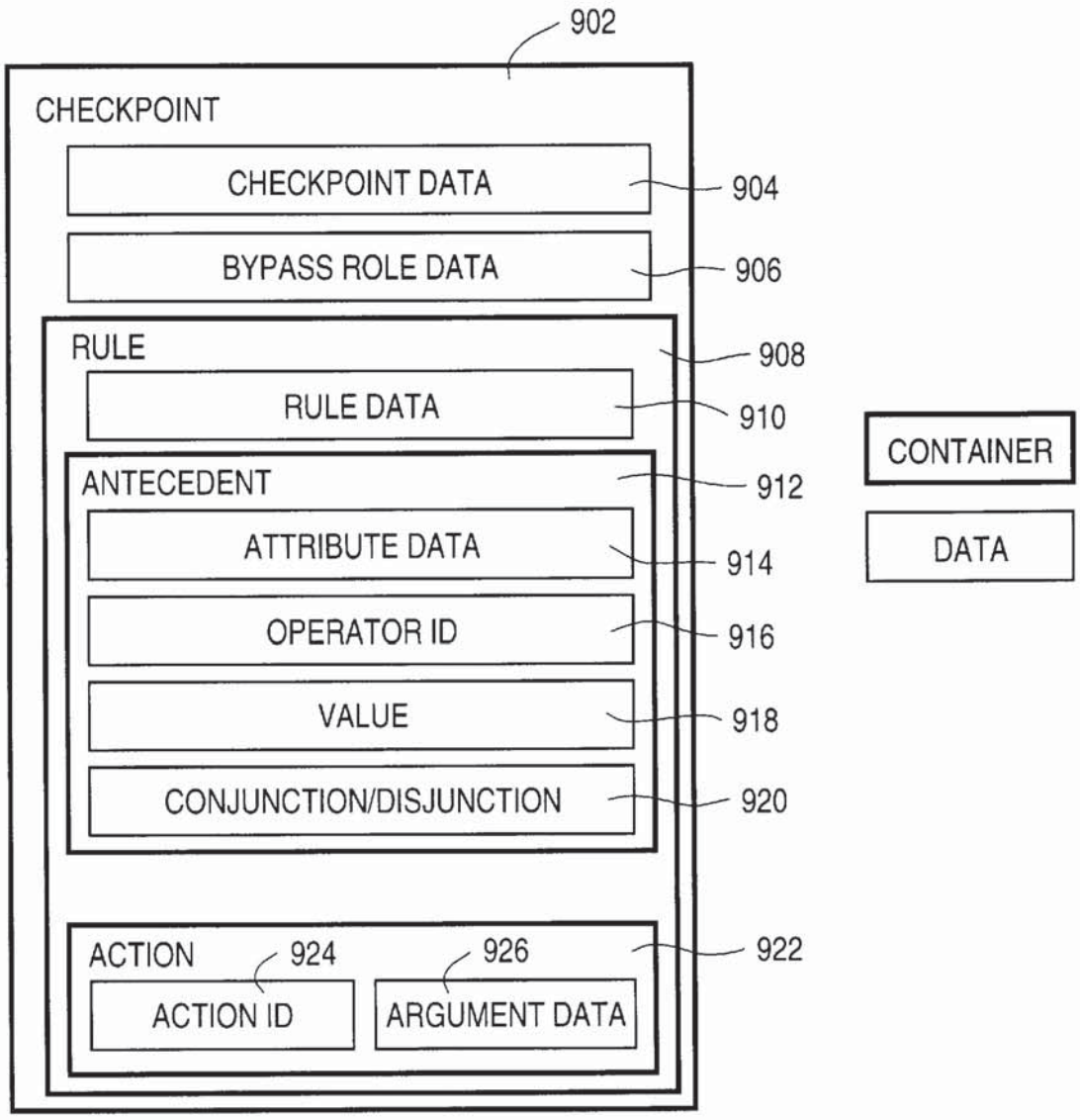
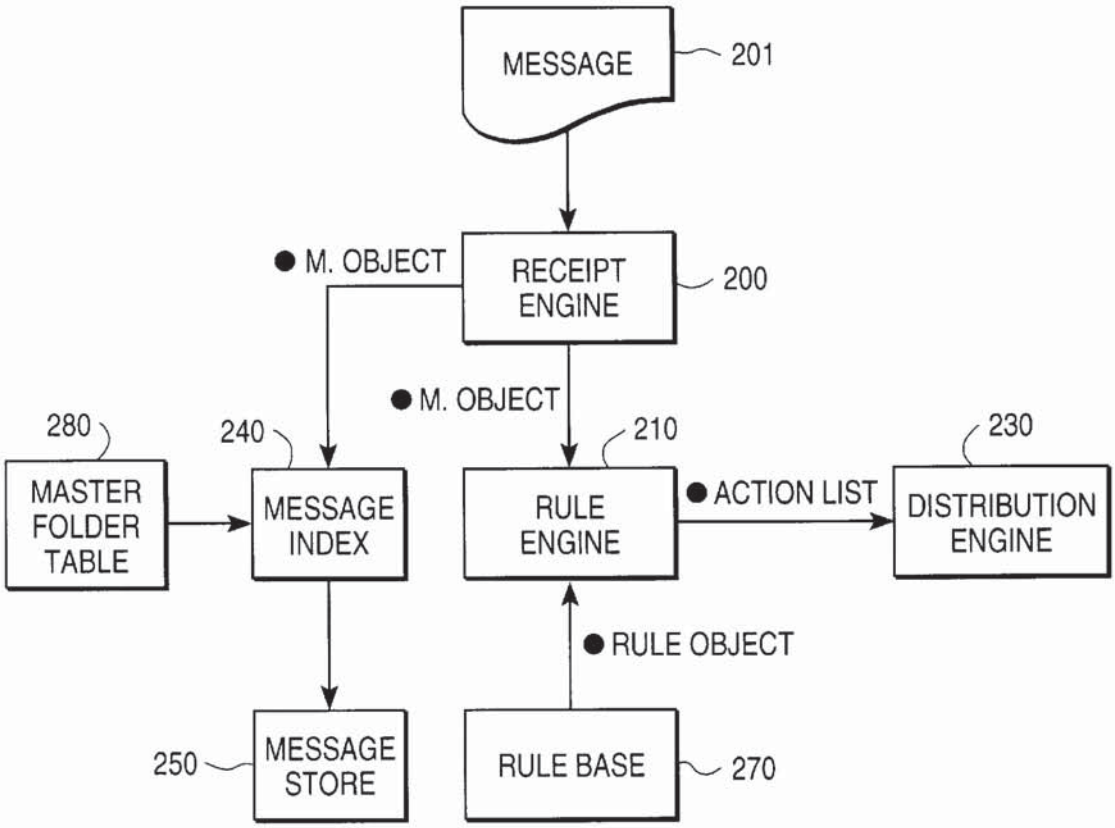
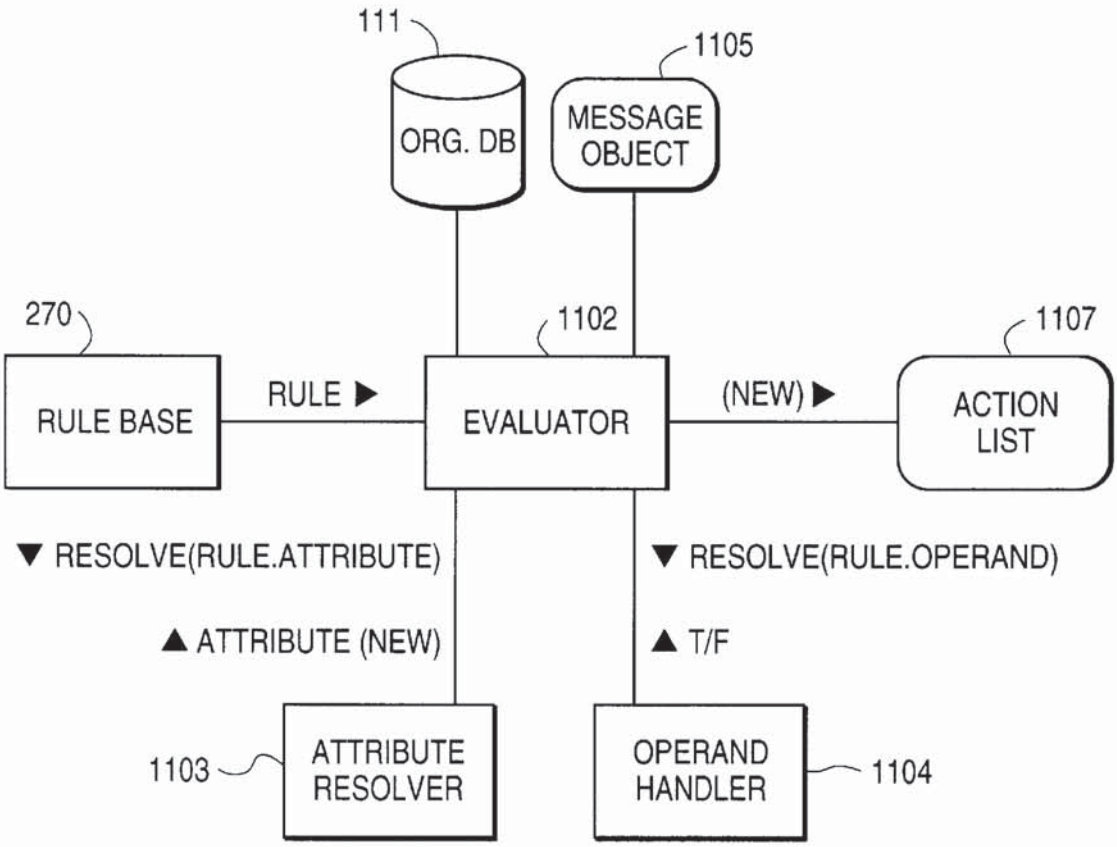

902
CHECKPOINT
CHECKPOINT DATA
904
BYPASS ROLE DATA
906
RULE
908
RULE DATA
910
ANTECEDENT
912
ATTRIBUTE DATA
914
OPERATOR ID
916
VALUE
918
CONJUNCTION/DISJUNCTION
920
ACTION
924
926
922
ACTION ID
ARGUMENT DATA
CONTAINER
DATA

FIG. 9



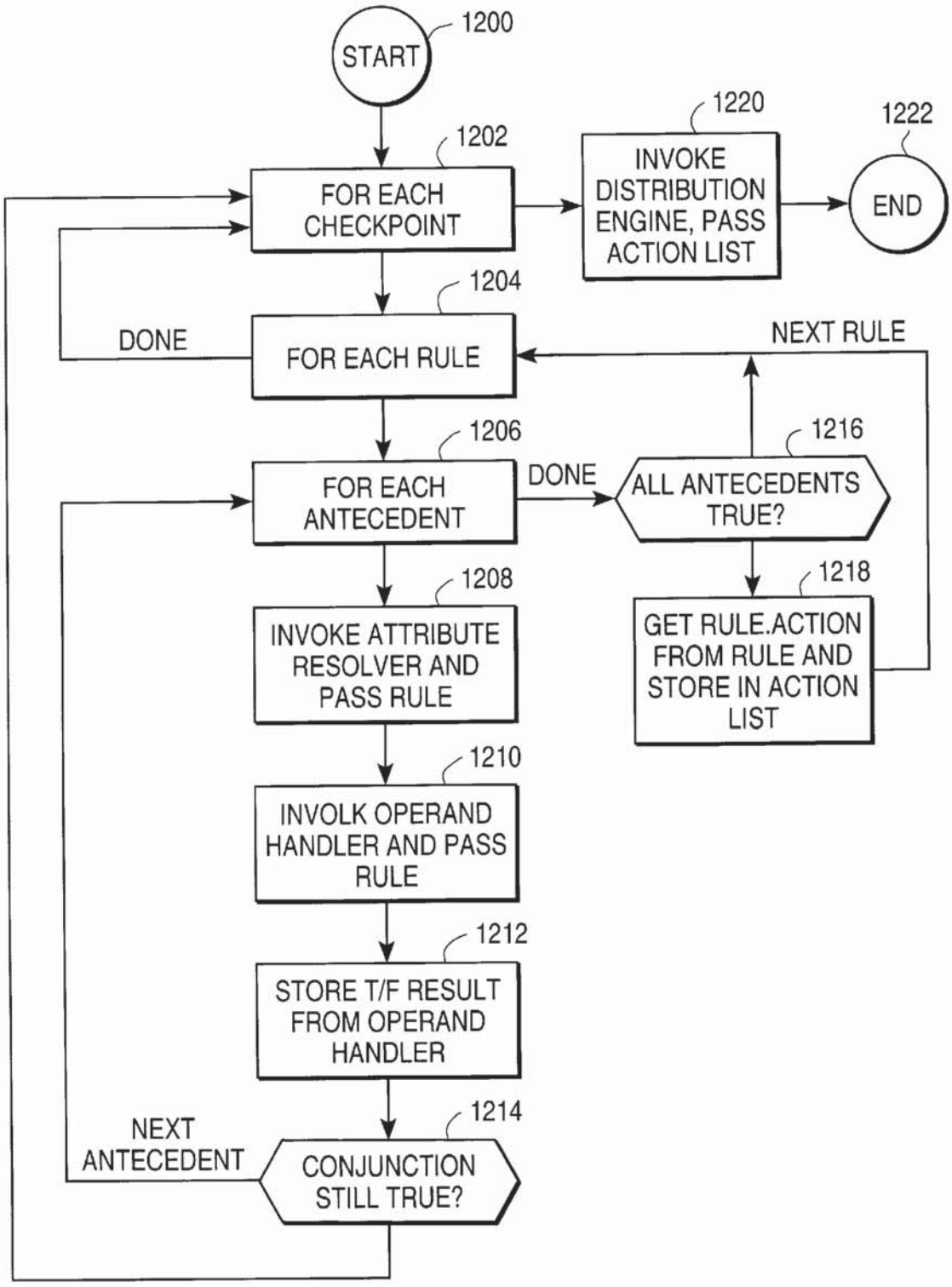
RULE ENFORCING POST OFFICE

FIG. 10



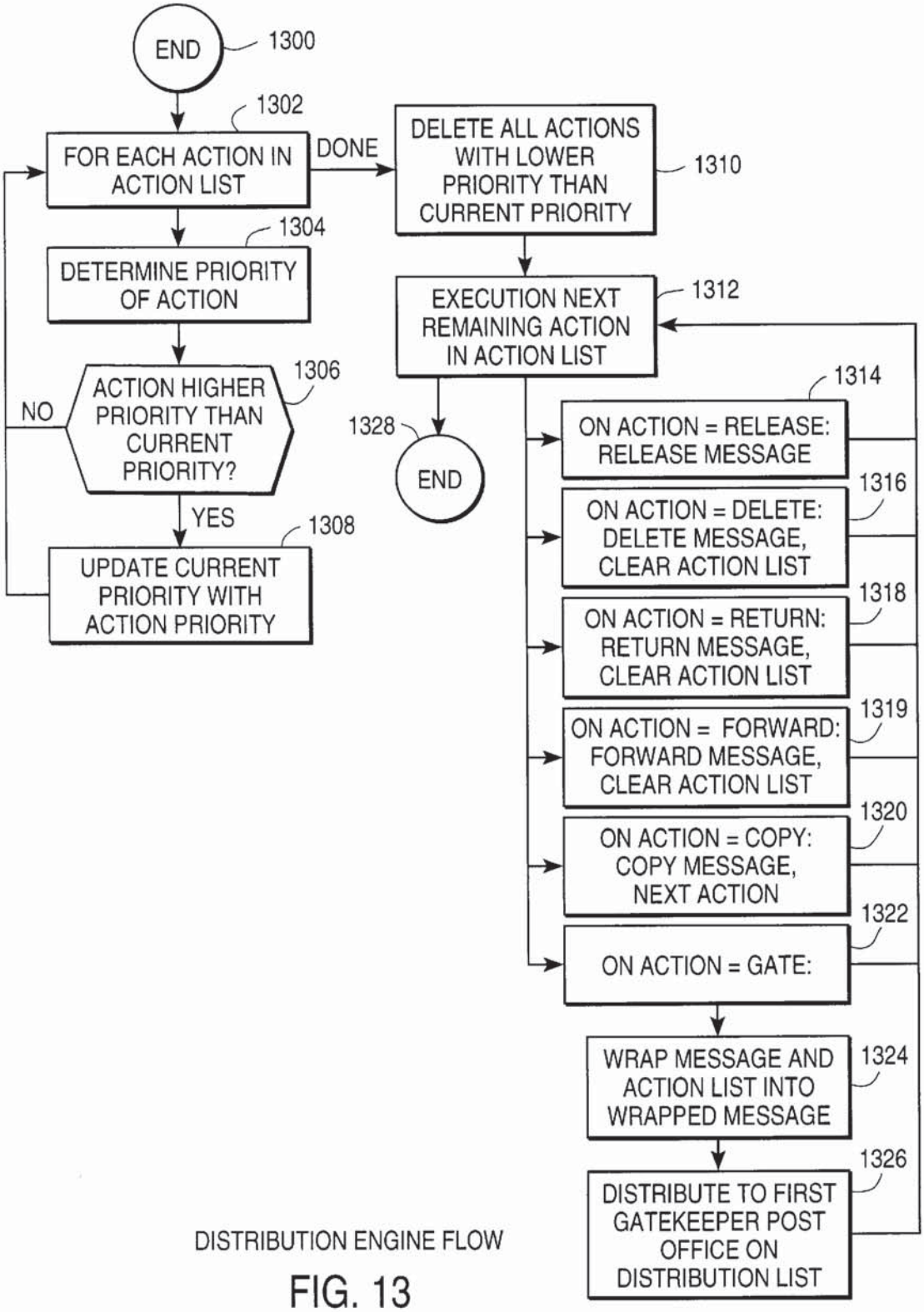
RULE ENGINE

FIG. 11



EVALUATOR FLOW

FIG. 12



DISTRIBUTION ENGINE FLOW

FIG. 13

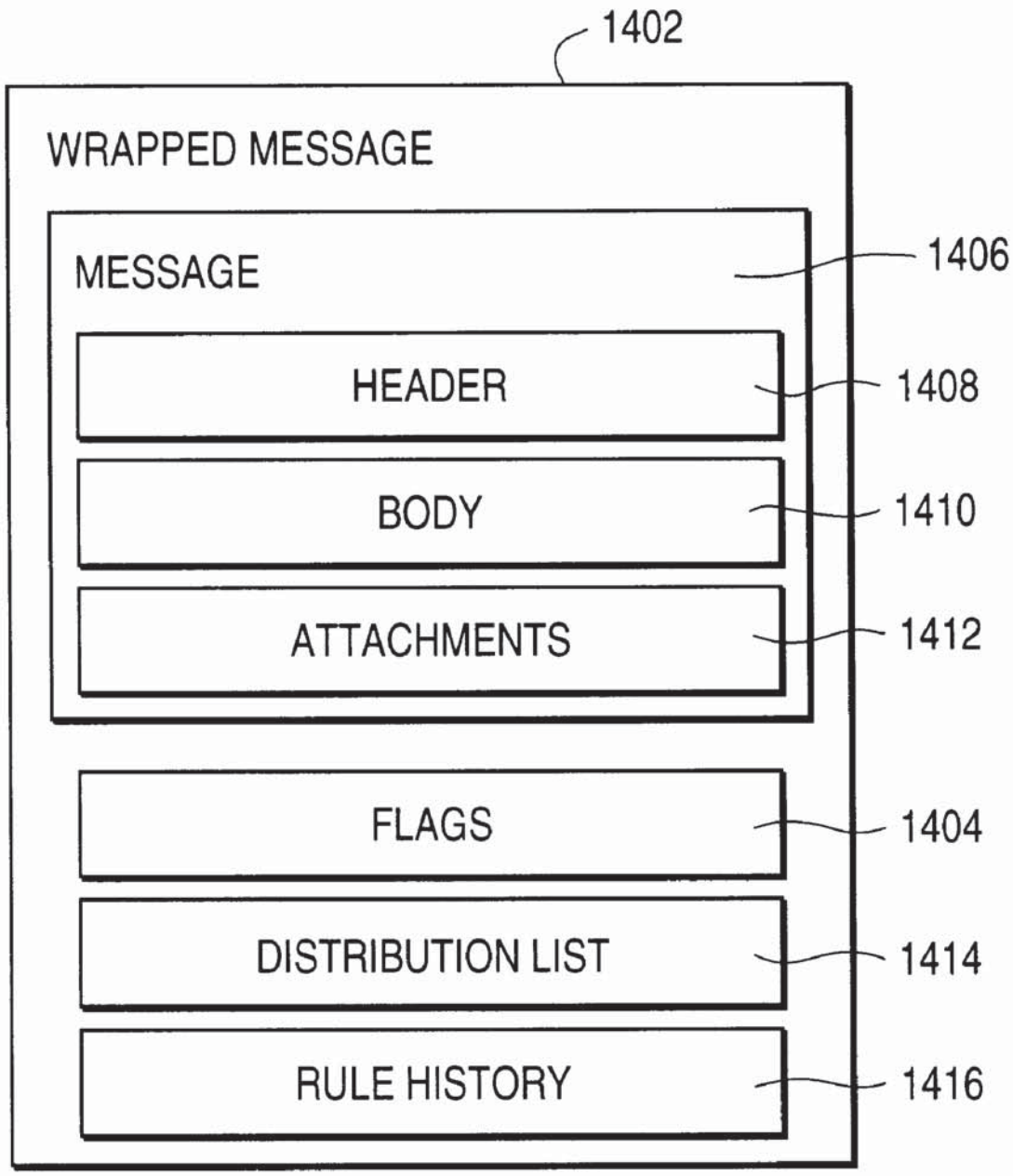
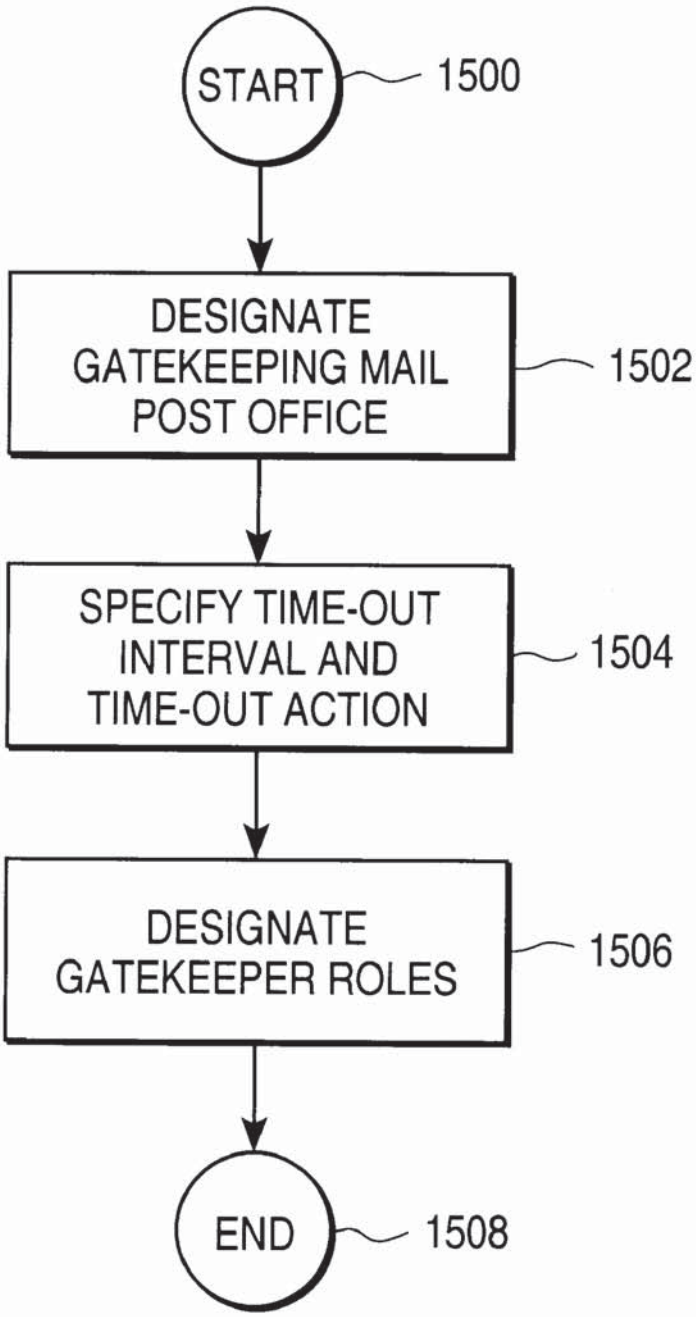

1402
WRAPPED MESSAGE
MESSAGE
1406
HEADER
1408
BODY
1410
ATTACHMENTS
1412
FLAGS
1404
DISTRIBUTION LIST
1414
RULE HISTORY
1416

FIG. 14



CONFIGURING A GATEKEEPING POST OFFICE

FIG. 15

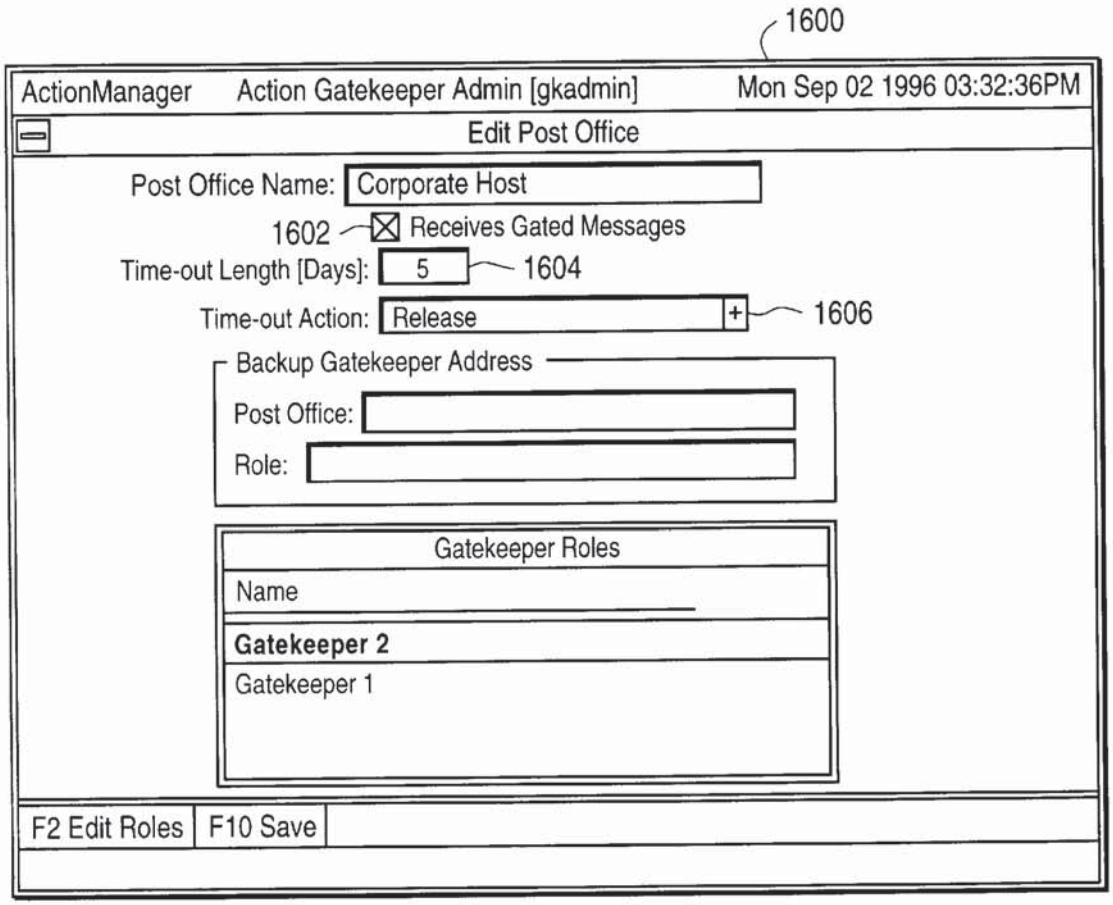

1600
ActionManager
Action Gatekeeper Admin [gkadmin]
Mon Sep 02 1996 03:32:36PM
Edit Post Office
Post Office Name:
Corporate Host
1602
Receives Gated Messages
Time-out Length [Days]:
5
1604
Time-out Action:
Release
1606
Backup Gatekeeper Address
Post Office:
Role:
Gatekeeper Roles
Name
Gatekeeper 2
Gatekeeper 1
F2 Edit Roles
F10 Save

FIG. 16

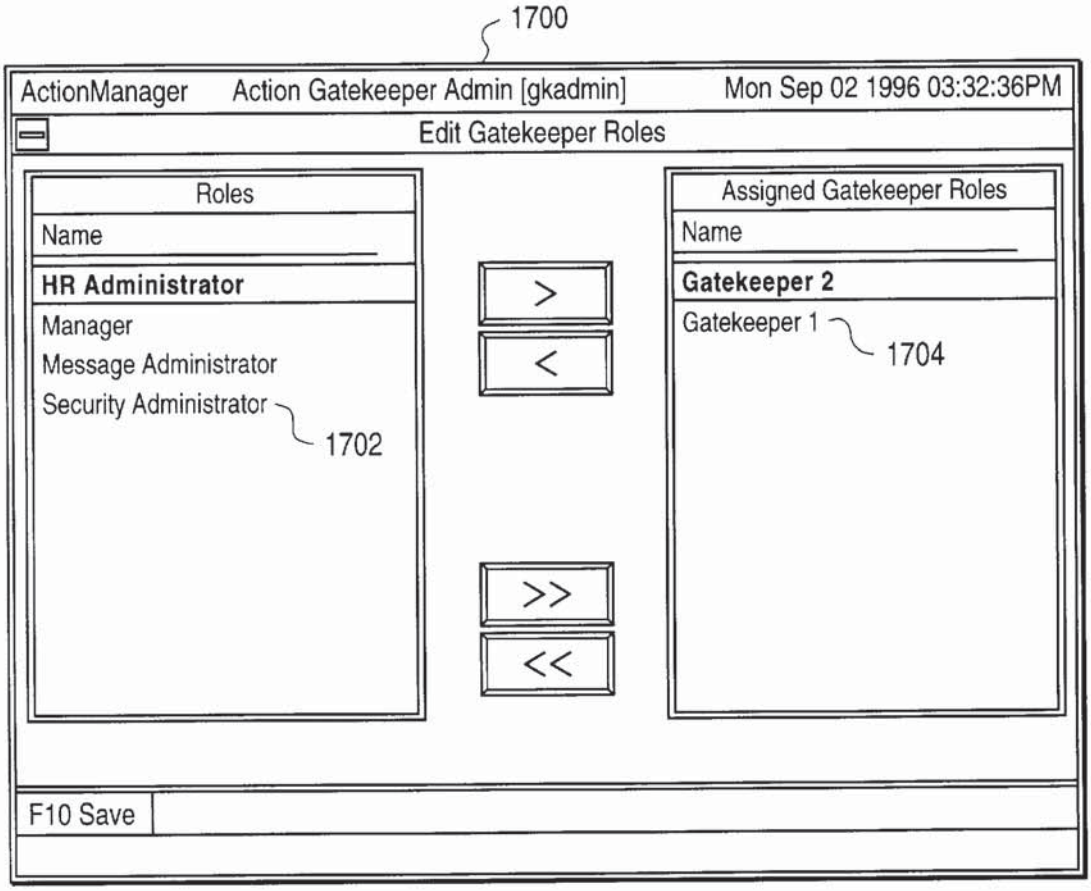
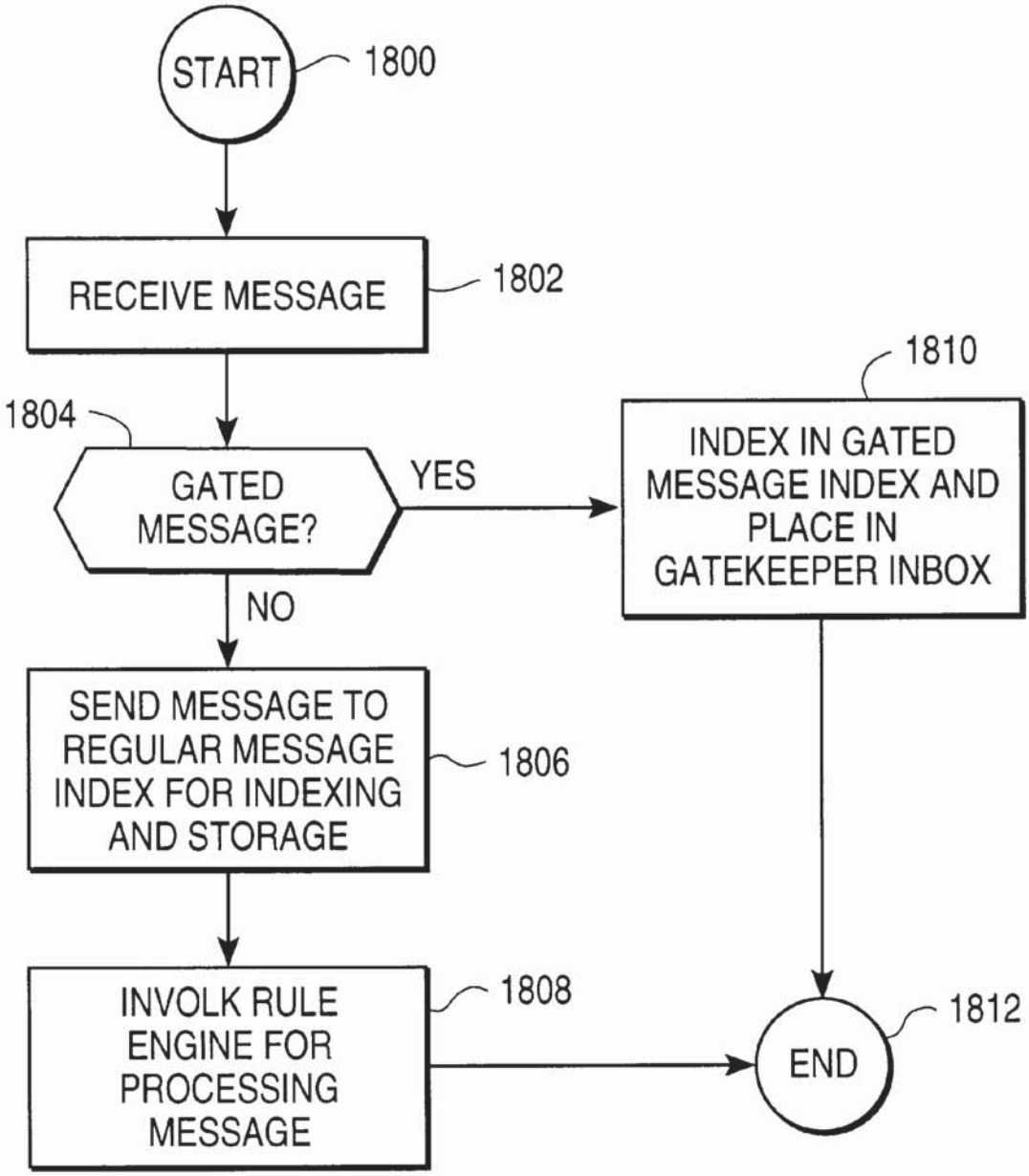

1700
ActionManager
Action Gatekeeper Admin [gkadmin]
Mon Sep 02 1996 03:32:36PM
Edit Gatekeeper Roles
Roles
Name
HR Administrator
Manager
Message Administrator
Security Administrator
1702
>
<
>>
<<
Assigned Gatekeeper Roles
Name
Gatekeeper 2
Gatekeeper 1
1704
F10 Save

FIG. 17



GATEKEEPING RECEIPT ENGINE

FIG. 18

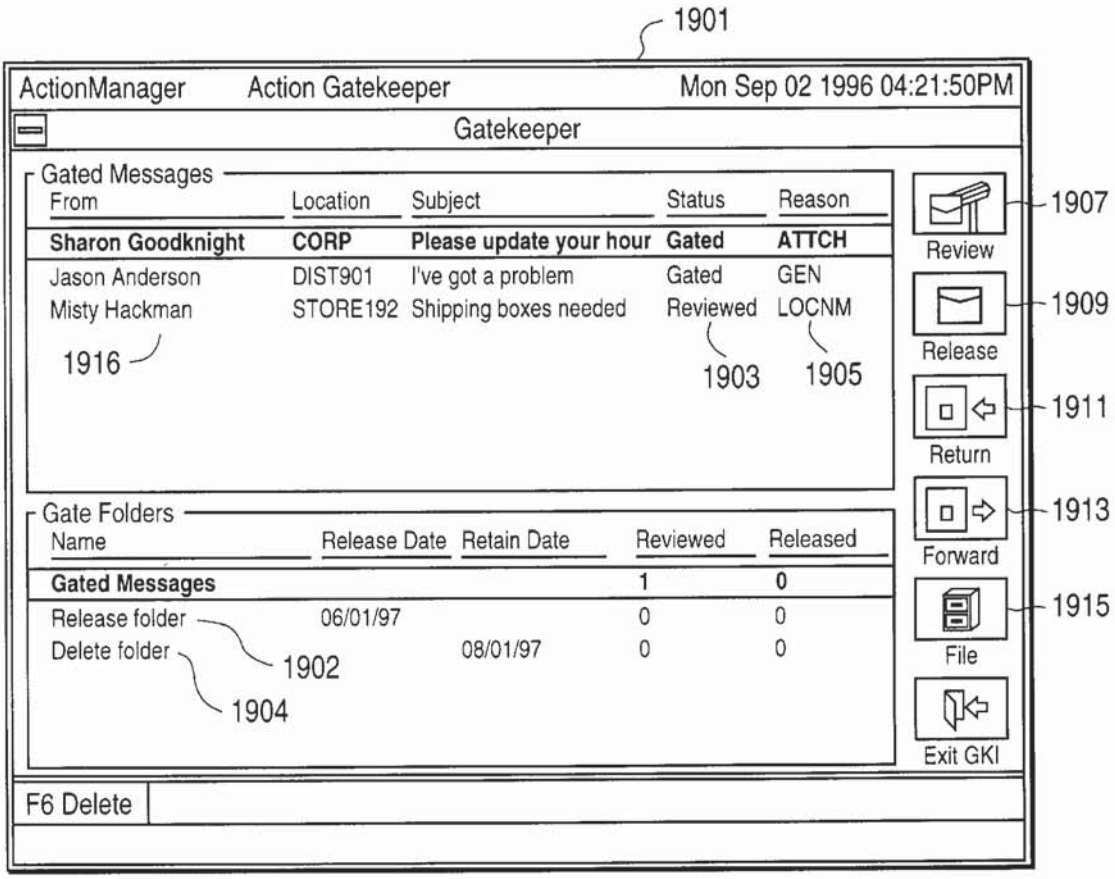

1901
ActionManager
Action Gatekeeper
Mon Sep 02 1996 04:21:50PM
Gatekeeper
Gated Messages
From
Location
Subject
Status
Reason
Sharon Goodknight
CORP
Please update your hour
Gated
ATTCH
Jason Anderson
DIST901
I've got a problem
Gated
GEN
Misty Hackman
STORE192
Shipping boxes needed
Reviewed
LOCNM
1916
1903
1905
Gate Folders
Name
Release Date
Retain Date
Reviewed
Released
Gated Messages
1
0
Release folder
06/01/97
0
0
Delete folder
08/01/97
0
0
1902
1904
Review
Release
Return
Forward
File
Exit GKI
1907
1909
1911
1913
1915
F6 Delete

FIG. 19

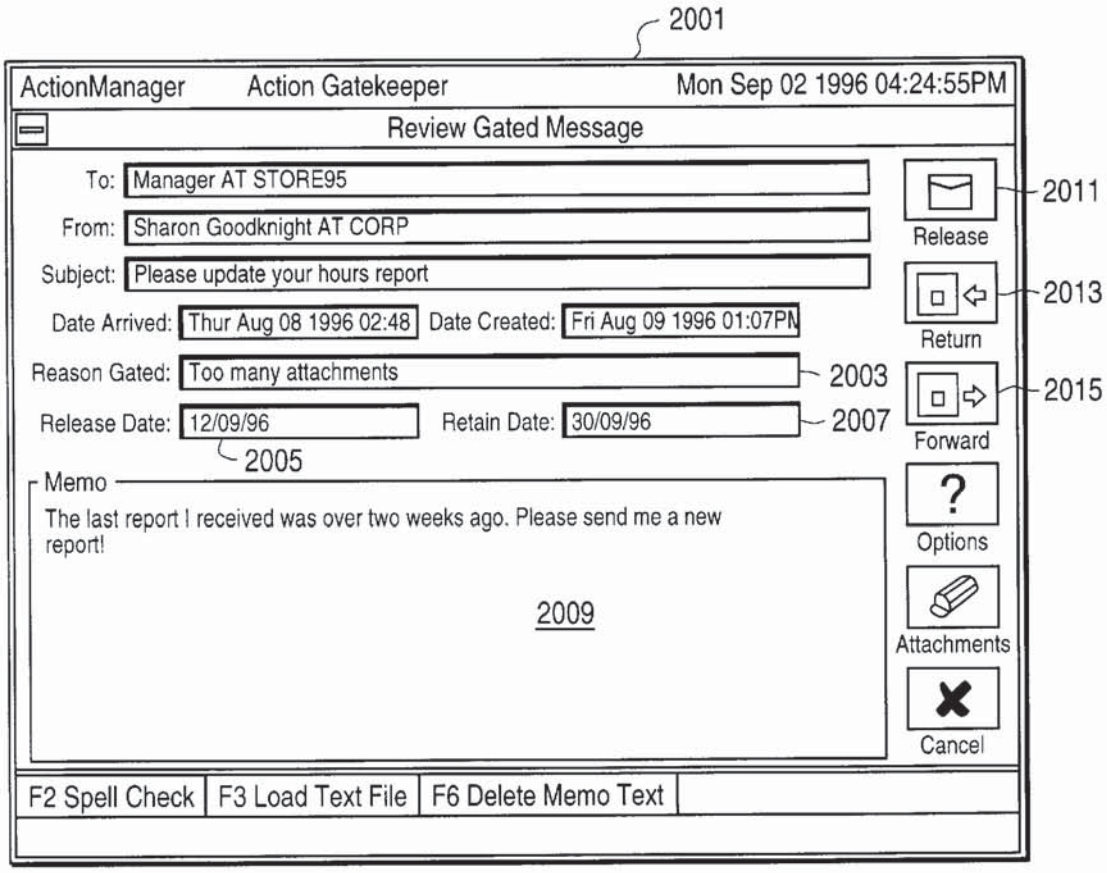

2001
ActionManager
Action Gatekeeper
Mon Sep 02 1996 04:24:55PM
Review Gated Message
To: Manager AT STORE95
From: Sharon Goodknight AT CORP
Subject: Please update your hours report
Date Arrived: Thur Aug 08 1996 02:48
Date Created: Fri Aug 09 1996 01:07PM
Reason Gated: Too many attachments
2003
Release Date: 12/09/96
2005
Retain Date: 30/09/96
2007
Memo
The last report I received was over two weeks ago. Please send me a new report!
2009
Release
2011
Return
2013
Forward
2015
Options
Attachments
Cancel
F2 Spell Check
F3 Load Text File
F6 Delete Memo Text

FIG. 20

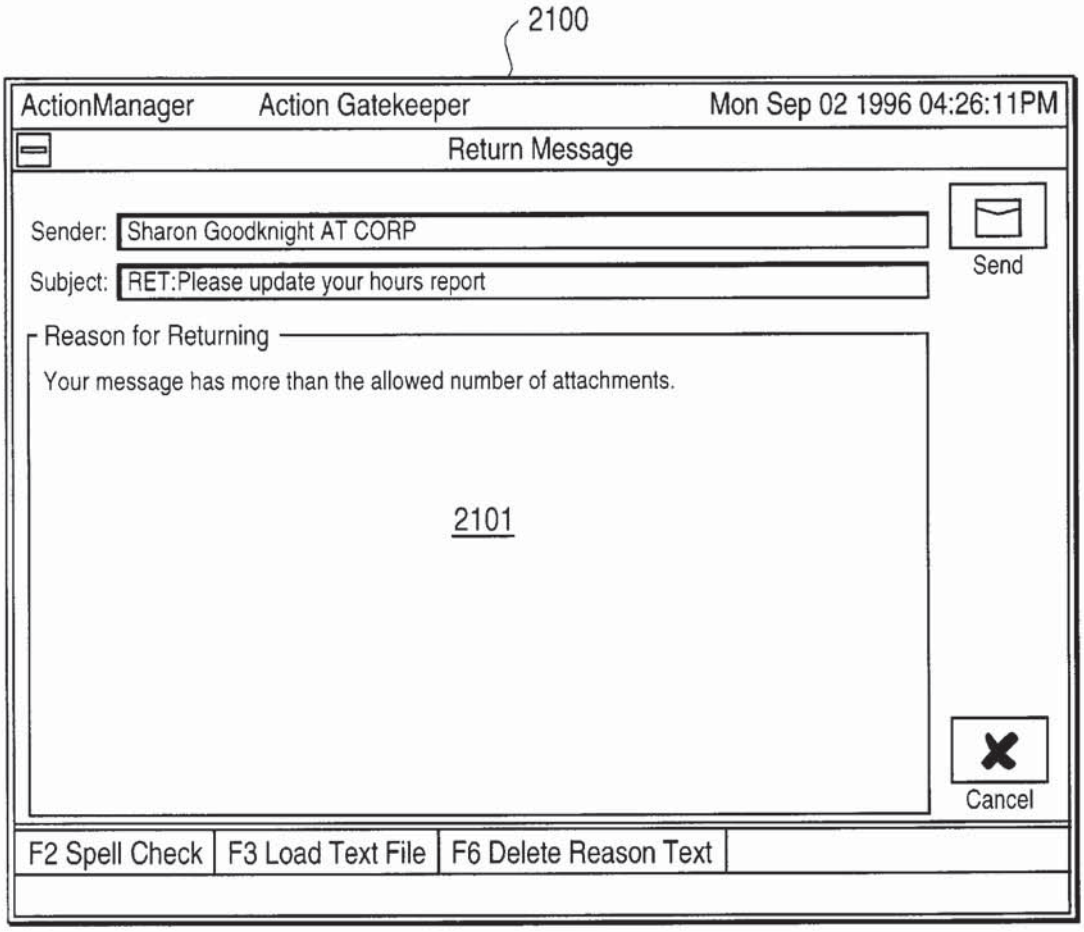
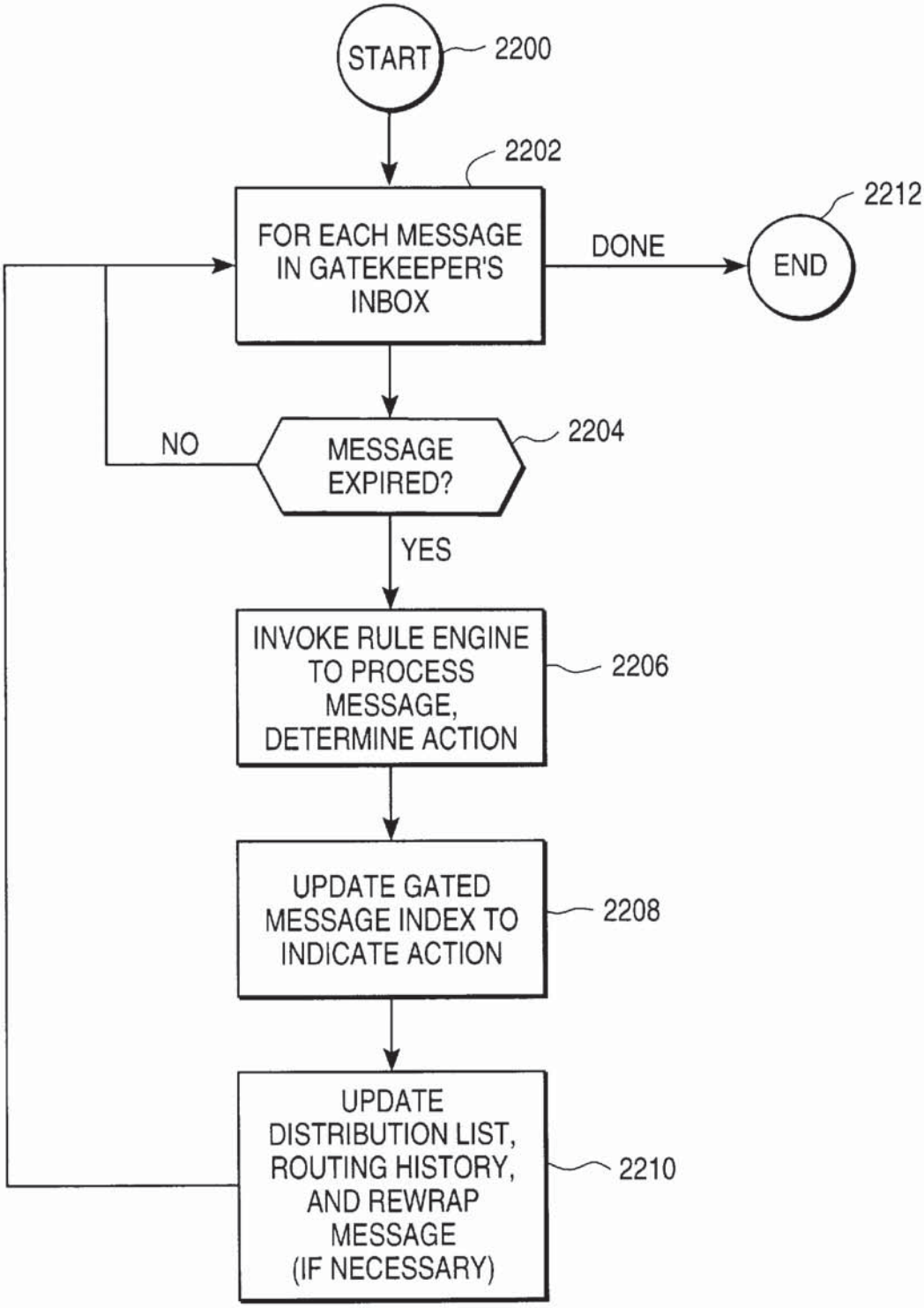

2100
ActionManager
Action Gatekeeper
Mon Sep 02 1996 04:26:11PM
Return Message
Sender: Sharon Goodknight AT CORP
Subject: RET:Please update your hours report
Send
Reason for Returning
Your message has more than the allowed number of attachments.
2101
Cancel
F2 Spell Check
F3 Load Text File
F6 Delete Reason Text

FIG. 21



GATEKEEPING INTERFACE CLIENT FLOW

FIG. 22

AUTOMATED POST OFFICE BASED RULE ANALYSIS OF E-MAIL MESSAGES AND OTHER DATA OBJECTS FOR CONTROLLED DISTRIBUTION IN NETWORK ENVIRONMENTS

FIELD OF THE INVENTION

The present invention relates generally to electronic messaging systems, and more particularly to electronic messaging systems that enable the application of business communication policies for controlling distribution of electronic messages and other data objects.

BACKGROUND OF THE INVENTION

Many corporate organizations have elaborate methods to control the flow of memorandum, publications, notices, and other printed information within the organization. An organization may limit the types of documents employees can distribute at work, and in some cases, control which persons within an organization communicate with each other. For example, the organization may prohibit the distribution of memoranda to all employees in order to reduce photocopying costs; it may except this rule for individuals with specific corporate positions, such as the president or chairman. As another example, an organization may filter documents that are to be sent to specific persons or departments, or it may automatically copy (archive) documents distributed by certain persons or departments. Finally, organizations ordinarily have rules that prohibit distribution of certain types of documents, such as those containing disparaging, sexist, or profane materials. These various rules are typically documented as part of the organization's business communication policies, and managed by the personnel, human resources, or other departments.

Most organizations today also use electronic messaging systems, or e-mail, for inter- and intra-company communications. Generally, an e-mail system comprises one or more post offices, zero or more mail servers and a relatively large number of e-mail client applications. The post offices are distribution mechanisms which receive e-mail messages from client applications (both within the organization and external thereto) and transfer these e-mail messages to other post offices associated with the specified recipients, who again are both within and external to the organization or system. Conventional post offices operate on a store and forward model, where an e-mail message is stored only temporarily for the duration it takes to route the message to the next post office(s).

In e-mail systems which use mail servers, post offices deliver incoming messages to a mail server which persistently stores the messages for the recipients. The recipients access the messages via the client applications. In some systems where mail servers are not used, the post offices deliver e-mail messages directly to the client applications. The e-mail client applications are end-user applications for creating, reading, and managing a user's individual e-mail account.

The fundamental operating paradigm of conventional post offices and mail servers is unabated delivery, which is intended to deliver an e-mail message from its sender to its recipients as directly as possible, with no interference from other users or administrators. Thus, a conventional post office receives and routes a message as quickly as possible, and does not purposely delay routing in order to process or otherwise delay the message. Existing e-mail protocols, such as Simple Mail Transfer Protocol (Internet RFC 821), all are intended to operate by unabated delivery.

With the increasing reliance on e-mail for all types of corporate communications, it is becoming increasingly desirable for an organization to be able to define and automatically enforce communication policies with respect to the handling of e-mail messages in their e-mail systems. An organization should be able to define specific business rules which implement its business communication policies, and apply these business rules to all e-mail messages on the e-mail system in order to monitor and control the distribution and handling of e-mail messages. In particular, it is desirable to provide such business rules within a post office, so that the business rules may be applied to all e-mail messages handled by the post office, regardless of their origin or destination.

Conventionally, however, organizations have not had the ability to define and automatically enforce communication policies with respect to the handling of e-mail messages by post offices. This is because conventional post offices are designed to implement existing e-mail protocols, which are based on unabated delivery. As a result these post offices are not designed to apply business rules to e-mail messages which either intentionally delay or prohibit delivery of e-mail messages. Delayed, intercepted, or prohibited delivery is antithetical to the unabated delivery concept, and thus, conventional post offices do not provide this ability. Similarly, conventional mail servers, which normally only serve messages to client applications, do not process or otherwise delay message delivery in order to apply business rules thereto.

At best, most available e-mail systems do allow the individual user of the e-mail client application to define how e-mail messages received or sent by that specific e-mail client application are to be handled. For example, the user of an e-mail client application may decide to store e-mail messages received from particular senders in various mailbags or directories. However, conventional e-mail systems do not enable the organization to define specific business communication policy based rules for the post office itself to use in order to control the delivery of e-mail messages. Since all e-mail messages are received and routed by a post office, it is desirable to provide such ability directly at the post office rather than at the individual client application, which would at best provide only limited, and inconsistent rule application.

It should be noted that from a technical standpoint, conventional post offices do employ routing rules for routing and addressing e-mail messages. These rules however, are merely physical, data, or transport layer protocol rules (layers **2**, **3**, **4** of the OSI model), and describe only the low level handling of the e-mail message. Control of the delivery of e-mail messages by the post office itself is at the application or session layer (layer **7**, **6**). Thus, existing low level filtering rules do not provide the desirable ability to define business rules which implement business communication policies for handling e-mail messages.

E-mail messages are but one type of data object which are communicated over networks. In addition to the use of e-mail systems, organizations use their internal and external networks to distribute and route many other types of data objects, such as database records, forms, application programs, and so forth. However, there as yet appears no mechanism by which an organization can control the distribution of such data objects in light of business communication policies. Accordingly, it is desirable to provide a generalized data server that includes the ability to define business rules for handling the distribution of various types of data objects.

SUMMARY OF THE INVENTION

The present invention overcomes the deficiencies of conventional systems and provides the beneficial ability to define business rules that implement business communication policies for controlling the handling of e-mail messages and other data objects by a data server. In the present invention, a data server is an e-mail type post office that includes the ability to directly distribute any general form of data in addition to e-mail messages. The present invention enables the data server to monitor e-mail messages and other data objects, and to selectively gate, delete, forward, copy, release, or return e-mail messages and other data objects by applying the business rules to the messages and data received at the post office for delivery to others. Gating messages re-route the e-mail message from its specified recipients to a gatekeeper at a gatekeeping post office. Here, the gated message is additionally reviewed (either manually or automatically) and further processed, which again may include any of the above actions.

In accordance with one embodiment of the present invention for distribution of email and related data formats, a post office includes a receipt engine, a database of business rules, a rule engine, and a distribution engine. The receipt engine provides support for conventional e-mail protocols, such as SMTP or POP, so as to be able to receive e-mail messages from both internal client e-mail applications, and external e-mail systems.

The database of business rules are used to implement business communication polices of the organization. Each business rule describes a particular action to be applied to an e-mail message in response to either attributes of the e-mail message or performance data of the post office. For example, a business rule may specify actions such as deleting the e-mail message, gating the e-mail message for further review, copying the e-mail message, returning the e-mail message to its sender without delivering it, forwarding the e-mail message to a new recipient, or releasing an e-mail message to its specified recipients. The attributes of an e-mail message which may trigger application of a business rule include, for example, the size of the e-mail message, the number of attachments, the size of individual or all attachments, the text of the message or its subject line, the inclusion of specific addresses or distribution lists, and other message-specific attributes. For example, a business rule to gate an e-mail for further review may be triggered for any e-mail message that is addressed to the president of the company. As another example, a business rule that returns an e-mail message to its sender may be triggered when the e-mail message or its attachments exceed a certain size, or that are addressed to a particular distribution list, such as "All Employees."

Business rules may also be used at the post office to direct incoming e-mails to particular employees for further handling. For example, a business rule may forward an e-mail message from outside of the company to a particular employee or user (e.g., Marketing Director) when the text of message matches specific keywords or other properties (e.g., text within the message matching product name keywords), even where the employee is not one of the originally specified recipients.

These various types of business rules enable the organization to apply very detailed, automatic control over all e-mail messages being handled by one or more post offices in the company's e-mail system.

Alternatively, a business rule may be triggered by conditions completely external to any particular e-mail message being handled, but rather in response to statistical performance factors for the post office. For example, a business rule to defer delivery of an e-mail message may be triggered when the average throughput of the post office exceeds a defined threshold. Thus, the business rules may be responsive to any discrete factors that reflect the business communication policies of the organization.

In accordance with the present invention, the rule engine operates with the database to apply the business rules to each e-mail message, in order to determine a set of actions (one or more) to be applied to the e-mail message.

In accordance with another aspect of the present invention, there are defined "bypass roles" that enable an e-mail message to be released (delivered) without further rule application by the rule engine. Bypass roles are associated with specific corporate positions, rather than specific persons (or e-mail addresses). For example, the corporate position of President may be a bypass role, such that all e-mail messages from the President are released without further application of the business rules.

The rule engine provides for each e-mail message it processes a set of actions to the distribution engine. The distribution engine is responsible for handling the e-mail message according to the set of actions. These actions may instruct the distribution engine to release the e-mail message, gate the e-mail message by delivering it to a specified gatekeeping post office, delete the e-mail message without delivering it, return the e-mail message to the sender, and so forth, as described above. In one embodiment, each action has a predefined priority level. The distribution engine then reviews the set of actions and applies the action having the highest priority level. For example, assume that as the result of different business rules being applied by the rule engine, the set of actions includes both deleting the message and copying the message. Deleting an e-mail message may be defined as a higher priority action than copying the message. Accordingly, the distribution engine would delete the e-mail message, without copying it.

More particularly, the present invention provides the ability to gate e-mail messages by altering the delivery of the e-mail message from its specified recipients to a gatekeeping post office, where they are stored for further review and handling. These gated e-mail messages may be manually reviewed by a corporate administrator designated as a gatekeeper, and either released, returned to the sender, deleted, forwarded, or copied. The gated e-mail messages may also be automatically processed by an independent set of business rules, and again various different actions may be applied. Because each gatekeeping post office may act independently, it may further gate the e-mail message to yet another gatekeeping post office. This enables distributed, network gatekeeping and review of the e-mail messages by any number of corporate officials designated as gatekeepers.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is an illustration of a gatekeeping e-mail message system in accordance with the present invention.

FIG. **2** is an illustration of the software architecture of a rule enforcing post office.

FIG. **3** is an illustration of the software architecture of a gatekeeping post office.

FIGS. **4**A and **4**B are flow diagram of the overall process of gatekeeping e-mail messages.

FIG. **5** is a flow diagram of the process of defining business rules within the rule base.

FIG. **6** is an illustration of a user interface for selecting rule enforcing post office to use a checkpoint.

FIG. **7** is an illustration of a user interface for designating checkpoint bypass roles.

FIG. **8** is an illustration of a user interface for defining specific business rules.

FIG. **9** is an illustration of a code point model for storing business rules.

FIG. **10** is an illustration of the data flow within a rule enforcing post office.

FIG. **11** is an illustration of the data flow within a rule engine for processing a message using the rule base.

FIG. **12** is a flow diagram of the operation of the evaluator for evaluating an e-mail message with respect to the rule base.

FIG. **13** is a flow diagram of the operation of the distribution engine for handling e-mail messages based on the action list for each message.

FIG. **14** is an illustration of the structure of a wrapped message.

FIG. **15** is a flow diagram of a process for configuring a gatekeeping post office.

FIG. **16** is an illustration of a user interface for configuring a gatekeeping post office.

FIG. **17** is an illustration of user interface for assigning organizational roles to gatekeeper roles.

FIG. **18** is an illustration of the operation of the receipt engine of a gatekeeping post office.

FIG. **19** is an illustration of a user interface for accessing gated messages in a gatekeeper's inbox.

FIG. **20** is an illustration of a user interface for reviewing a gated message.

FIG. **21** is an illustration of a user interface for returning a gated message to its sender.

FIG. **22** is a flow diagram of the operation of the gatekeeping post office during automatic message processing.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

I. System and Operational Overview

Referring now to FIG. **1** there is shown one embodiment of a gatekeeping e-mail communication system of the present invention. An electronic communication system **100** operates on a conventional communications network **104**, which may be a LAN, WAN, MAN, or the Internet.

Communicatively coupled to the network through conventional e-mail protocols, such as SMTP, POP**3**, and the like are one or more rule enforcing post offices **102** ("REPO 102") and one or more gatekeeping post offices **106** ("GPO 106"). In addition, a conventional post office **109** may be present to provide conventional mail transfer functionality for transferring e-mail messages to and from the REPOs **102**, and GPOs **106**. Additionally, conventional mail servers and conventional post office/mail server combinations may be present.

Communicatively coupled to REPOs **102** are conventional e-mail client applications **110**. The e-mail client applications **110** are capable of generating and receiving e-mail messages in a conventional manner for individual users. Users may use the e-mail client applications **110** to address e-mail messages to other e-mail client applications **110** associated with any of the post offices **109**, **102**, **106**. In this disclosure the terms "e-mail message" and "message" are used interchangeably and are intended to have the same meaning.

A gatekeeper administration program **108** ("GKADMIN 108") is used by system administrators to configure the various REPOs **102** and GPOs **106** of the system. A gatekeeper interface client **107** ("GKI 107") is used by individual gatekeepers to access and review gated messages. A gatekeeper is a human administrator assigned a gatekeeper role and having responsibility for reviewing gated messages.

An organizational database **111** stores organizational information, including an organizational hierarchy of organizational roles and the individuals assigned to such roles. This information is used by the REPOs **102** and GPOs **106** to apply various rules to messages based on the role of the sender or recipient.

II. Overview of a Rule Enforcing Post Office

A conventional post office acts as a mail transfer agent, and provides only for mail transfer based on a temporary store and forward model. A REPO **102** is a post office that, in addition to conventional functionality for transferring messages, enforces an organization's business communications policies by applying to each e-mail message received (either from local e-mail clients applications **110** or other remote post offices) business rules which define actions to be applied to an e-mail message according to various attributes of the message or the state of the REPO **102** itself.

Referring now to FIG. **2**, there is shown the software architecture of one embodiment of a REPO **102**. A REPO **102** includes functional modules for receiving, processing, and distributing e-mail messages to and from e-mail client applications **110** and other post offices (including REPOs **102**, GPOs **106**, and conventional post offices **109**). These functional modules are provided by a receipt engine **200**, a rule engine **210**, and a distribution engine **230**, respectively. Supporting data storage includes a message index **240** and a message store **250**. Also included are a routing engine **220** and routing tables **260**.

The receipt engine **200** provides a standard communication protocol interface (e.g., SMTP, MHS) to other post offices in order to receive e-mail messages from such post offices, and from any e-mail client applications **110** directly associated with the REPO **102**. The receipt engine **200** temporarily indexes and stores received messages in the message index **240** and message store **250** during processing of the message. The message store **250** generally provides for temporary storage of message objects for subsequent distribution by the distribution engine **230**. The message index **240** provides interfaces to the rule engine **210** and the distribution engine **230** to enable these modules to retrieve e-mail messages from the message store **250**, to enable the rule engine **210** to access stored messages for processing prior to any distribution.

The routing engine **220** cooperates with the routing tables **260** in a conventional manner to determine, given recipient or addressee information of an incoming message, the proper post office and mail protocols for further delivery of the message. The routing engine **220** provides the routing information to the distribution engine **230** when necessary to continue delivery of the message.

The rule engine **210** operates in conjunction with a rule base **270** to process incoming messages with business rules defined in the rule base **270**. The rule base **270** stores one or more sets of business rules. Business rules are input into the rule base **270** using the GKADMIN **111**. In this disclosure, "business rule" and "rule" are used interchangeably and are intended to have the same meaning. Each set of business rules is referred to as a "checkpoint," and a REPO **102** is alternatively known as a checkpoint post office. Each business rule specifies an action to be applied to a message by

the distribution engine **230**. The actions are output by the rule engine **210** in the form of an action list which is read and interpreted by the distribution engine **230**.

Instead of operating under the conventional unabated delivery paradigm, the distribution engine **230** handles e-mail messages in accordance with actions specified for the messages in the action list. The distribution engine **230** supports conventional mail protocols for delivering messages to other post offices, both conventional and servers such as the REPO **102** and GPO **106**. One of the significant actions of the distribution engine **230** is to "gate" an e-mail message by delivering it to one or more GPOs **106** instead of delivering it to any of the recipients specified by the sender. This gating action enables administrative review of the e-mail message by a gatekeeper administrator prior to any delivery.

A REPO **102** may also include a mail server component, so that client applications may communicate with the mail server to receive messages delivered to the REPO **102**. In this embodiment, when the REPO **102** receives a message for a specified recipient associated with the post office, it transfers the message to an inbox for the mail server, which in turn makes the message available to the appropriate mail client application. In this embodiment, a master table folder **280** is optionally provided that defines, typically for each user associated with the REPO **102**, a collection of folders (also called "mailbags" herein) for indexing stored messages. In this variation, the message store **250** is used for persistent storage of messages, enabling them to be served to client applications by the mail server.

III. Overview of a Gatekeeping Post Office

A GPO **106** is a post office that provides for administrative review and processing of gated messages. A GPO **106** provides for both manual review by a gatekeeper—a person designated to review gated messages—and automatic review and processing using its own set of business rules. Processing applies various actions to gated messages, including deleting the message without delivering it to the specified recipients, returning a message to its sender, copying a message prior to sending, editing a message, forwarding the message to a new recipient, and releasing a message for distribution to its specified recipients. A GPO **106** provides for both immediate execution of these processing actions or delayed execution. Thus, the primary function of the GPO **106** operates contrary to the conventional unabated delivery model by adding an additional processing and review layer between the sending of the message and its receipt (if ever) by the originally specified recipients.

There are two basic embodiments of a GPO **106**. First, a GPO **106** may provide only gatekeeping functionality, as a post office which only receives gated messages, but otherwise is not used to rule process other messages as does a REPO **102**. Second, a GPO **106** may also include the functionality of a REPO **102**, and receive both gated messages and non-gated messages, and provide all of the rule processing functionality of a REPO **102** on such incoming non-gated messages. For the purposes of this disclosure, this integrated embodiment will be discussed first.

Referring now to FIG. **3**, there is shown one embodiment of a GPO **106**. Like a REPO **102**, a GPO **106** includes a receipt engine **282**, a rule engine **283**, and a distribution engine **284**. The GPO **106** extends the functionality of the receipt engine **282** and distribution engine **284** to handle messages which have been gated to the GPO **102**, including providing functionality for automatically reviewing messages.

The receipt engine **282** operates in a manner similar to that described above to receive messages and to provide these messages to other components for indexing, storage, and processing. Because the GPO **106** receives gated messages from other post offices, the receipt engine **200** includes additional functionality for handling gated messages. This additional functionality includes distinguishing gated messages from non-gated messages as messages are being received, and providing gated messages to the inbox of a gatekeeper while providing non-gated messages to the receipt engine **282** for rule processing as described above. Further, the GPO **106** includes additional functionality, for example, as part of its program executive, to automatically review messages on a periodic basis and perform defined actions upon such messages.

The rule engine **210** operates in conjunction with a gatekeeping rule base **289**, which stores a set of business rules for handling gated messages, and with a rule base **270** as described above for handling non-gated messages. The rules for the gatekeeping rule base **289** may be different from the rules applied at one of the REPOs **102** in the rule base **270** of the GPO **106**, and in particular can be specifically adapted by the administrator with the intent of handling gated messages received from a REPO **102** or other GPOs **106**.

To support the receipt and review of gated messages outside of the conventional delivery paradigm, a GPO **106** provides a gatekeeping message index **287** and gatekeeping message store **288** which is used to store gated messages prior to review and processing. The gatekeeping message index **287** and gatekeeping message store **288** are preferably separate from the message index **285** and message store **286** used for storing messages during normal rule processing and transfer. This is because during the gatekeeping phase storage is transient, typically for only as long as necessary to process the messages. In contrast, the gatekeeping message index **287** and gatekeeping message store **288** are used for persistent storage of gated messages until reviewed and processed, which may require storage for extended periods of time (e.g., 30 days).

FIG. **2** illustrates this embodiment with two distinct message indices and message stores. A regular message index **285** indexes all regular (non-gated) messages received by the GPO **106**, which are then stored in the regular message store **286**. Gated messages however are indexed in a gatekeeping message index **287**, and stored in the gatekeeping message store **288**. The GPO **106** distinguishes a gated message from a non-gated message by whether or not it is a wrapped message.

Messages in the gatekeeping message index **287** are categorized into a number of folders for subsequent handling. In this disclosure the term "mailbag" is used interchangeably with the term "folder." The master folder table **290** defines these folders and their parameters. The master folder table **290** includes an inbox for each gatekeeper, into which gated messages for that gatekeeper are placed. Useful folders which may be defined include a return folder, a delete folder, and a release folder; other folders may also be created as needed by a gatekeeper. Each folder has a unique folder ID which is stored with the message in the message index **240** and a folder name.

In one embodiment, each folder has two time parameters associated with it: a release date and a retain date. The release date is the date on which all messages in the folder are released. The retain date is the date up to which all messages in the folder are retained, and then deleted on the specified date. Using these attributes, a gatekeeper can create any number of folders to release or delete gated messages after various times. The earliest time parameter is

always executed first. FIG. **19**, for example, shows a release folder **1902** with a release date of Jun. 1, 1997, and a delete folder **1904** with a retain date of Sep. 1, 1997.

In another embodiment, more sophisticated processing is provided with the folders. In this embodiment, each folder also has a time parameter, and an action associated with the folder. A folder action is a gatekeeper-defined action to be applied to the messages in the folder according to the time parameter. The actions include gating, forwarding, copying, deleting, or returning the messages. In addition, complex action sequences may also be defined by a gatekeeper, such as copying and then deleting messages. This flexible definition of the folder actions enables a gatekeeper to precisely control the processing applied to gated messages.

The time parameters may be either:

1. A delta time: This parameter specifies an amount of time after which a message in the folder is acted upon according to the folder action. The delta time is measured by the GPO **106** relative to a timestamp of the message.

2. A time interval: a periodic time interval for acting upon messages in the folder. All messages in the folder are acted upon according to the folder action at the same time upon expiration of the interval.

3. Fixed time: a specific date and/or time at which all messages in the folder are acted upon according to the folder action.

Generally, the inbox has a delta time parameter associated with it, while the other folders have a time interval parameter.

Each folder may also be associated with a gatekeeper role, which identifies the owner of the folder as the gatekeeper who created the folder. This allows the gatekeeper who created the folder to reset the time parameter and action.

An embodiment including two segregated message indices **285**, **287** is desirable because it enables the rule engine **283** and the distribution engine **284** to very quickly distinguish gated messages which tend to have relatively long or persistent storage, from non-gated messages which are stored only on a temporary basis, without having to use processing time to filter the distinct types of messages from each other.

A gatekeeper accesses gated messages through the GKI **107**. With this application, the gatekeeper manually reviews a gated message in an inbox assigned to that gatekeeper, and either directly acts upon the message to release, return, delete, copy, or edit the message, or delays disposition of the message by moving the message with one of the other folders which will be automatically processed at a later time. The GKI **107** also supports automatic processing of gated messages, as further explained below.

The REPOs **102** and the GPOs **106** are preferably implemented as software products executing on conventional server-class computers, such as Sun Microsystems Inc.'s SPARC™ based workstations and server, or IBM compatible computers based on Intel Inc.'s Pentium™ processors. The servers operate in conjunction with conventional operating systems, such as UNIX™, or Microsoft Corp.'s Windows95™ or WindowNT™.

IV. Operational Overview

Referring now to FIGS. **4**A and **4**B, there is shown a flowgraph of the overall process of processing and gating e-mail messages in accordance with the present invention. Operation of an e-mail system in accordance with the present invention may be understood as having three distinct phases: rule definition, message processing, and gating.

During the rule definition phase, an administrator or gatekeeper defines **402** various business rules for handling e-mail messages. In one embodiment, the rules are defined using the GKADMIN **108** which can directly edit the rule base of any GPO **106** or REPO **102**. The business rules are preferably consistent with the business communication policies used by the business, or may extend or modify such communication policies. These business rules are stored in the rule base **270** of one or more REPOs **102**. In addition, business rules for handling gated messages are created and stored in the gatekeeping rule base **289** of one or more GPOs **106**. It is anticipated that the business rules stored in the REPOs **102** will be different in some respect than those stored in the GPOs **106**, since the function of a GPO **106** is to further process messages that have already once been processed and gated by a REPO **102**.

In addition to defining various business rules for specific actions to be applied to messages, the administrator may also define bypass roles. A bypass role is an organizational position within the organization for which a message from a sender with a bypass role is passed through a REPO **102** without any other rule processing. For example, the president of the company would be a typical bypass role, and all messages from the president would then not be subject to the other rule processing or gating operations of a REPO **102**. Thus, bypass roles are used to define the exceptional case of unabated delivery in the context of the invention. The bypass roles are defined with respect to company position, and not with respect to actual user names. This allows a change in the holder of the bypass role, without the administrator having to manually redefine the bypass role, as would be the case if the bypass ability were directly associated with a user name. The role information for bypass roles is held in the organizational database **111**.

In the message processing phase, the REPOs **102** and GPOs **106** are operational for receiving and distributing messages. A message is received **404** at a REPO **102** from another post office or client application. With the REPO **102**, the receipt engine **200** indexes the message in the message index **240** and provides it to the rule engine **210**. The rule engine **210** checks **406** whether the sender has a bypass role. If so, the message is released **408** by the distribution engine **230**. The message may be delivered to the specified recipients at other post offices, or it may be gated at a REPO **102** that does not recognize the particular bypass role.

If the sender of the message does not have a bypass role, the rule engine **210** processes **410** the message with the business rules in the rule base **270** to determine the appropriate action(s) for handling the message. Typically, at least one business rule will be satisifed, and thereby specify the action to be applied to the message. The action is communicated to the distribution engine **230**, which applies **412** the action to the message. If there are multiple actions, the distribution engine **230** selects a highest priority action, and applies it to the message. The distribution engine **230** may release **408**, return **416**, gate **418**, copy **420**, forward **421**, or delete **422** the message. When a message is gated **418**, it is not delivered to its initially specified recipients. Instead, the gating action specifies a gatekeeper role at a GPO **106** who is to review the message. Accordingly, the distribution engine **230** sends the message to this gatekeeper. When a message is forwarded **421**, it is delivered to a new recipient, typically one other than a specified recipient; the new recipient is specified by the applicable business rule which was satisfied.

A REPO **102** operates continuously in this mode of receiving and processing messages. A GPO **106** which includes rule processing functionality on incoming messages also operates in this manner with respect to non-gated messages.

The gating phase is applied by a GPO **106** to messages that have been gated **418** by a REPO **102** or other GPO **106**. A gated message is received **424** at a GPO **106**, indexed in the gatekeeping message index **287** and initially placed in an inbox for the gatekeeper to whom the message has been gated.

The present invention provides for two types of gatekeeping review, manual review and automatic review. Each of these types of reviews may result in actions that are immediately executed, or actions for which execution is delayed.

During manual review, the gatekeeper evaluates **428** any or all of the messages in the inbox, and may immediately execute an action on the message, including releasing **430** the message, returning **432** the message to its sender (with or without an explanation as to why it was returned and not delivered), deleting **434** the message, copying **436** the message, and gating the message by forwarding **438** it to yet another gatekeeper, or otherwise forwarding **438** the message to another recipient for further handling. In addition, the gatekeeper can edit **440** a message, for example to remove offensive language, delete an attachment, or the like. After editing, the gatekeeper can release **430** the message for further delivery (to the specified recipients or other recipients), or return **432** it to the sender with an explanation of why the message was not delivered.

Alternatively, the gatekeeper may decide to delay execution of an action such that it is applied to a set of messages instead of individually. For delayed execution, the gatekeeper moves **442** a message to a mailbag which has a defined time parameter. The mailbag may have a specifically defined action, such as a release mailbag **444**, a return mailbag **445**, a delete mailbag **446**, a forward mailbag **447**, or a review mailbag **448**, or it may simply have a release and retain date. The GPO **106** periodically checks the time parameters of the various mailbags and executes **449** an appropriate action for each mailbag upon expiration of its time parameter. For example, a delete mailbag **446** may have an interval time parameter of **30** days; all messages in this mailbag are deleted once every **30** days. Alternatively, a release mailbag **444** may have a specific release date, and the GPO **106** checks to see if the current date matches the release date, and if so, releases the messages in the mailbag. This approach lets a gatekeeper manually review some messages and immediately act on some messages, while delaying actions on other messages.

In addition, the gatekeeper may review the message and not perform any action on it at all, including not releasing the message. In this instance, the message simply remains in the gatekeeper's inbox.

Automatic review is also provided by the GPO **106**, for example as part of its basic functionality or program executive. For those messages which the gatekeeper does not evaluate **426** or which remain in the inbox, the GPO **106** operates a daemon process, and periodically wakes up and processes all messages in the gatekeeper's inbox. The inbox has a timer associated with it. Here however, the timer is treated as relative value, instead of an absolute period. When a gated message is received, the timer is added to the message date to define an expiration date for the message. Thus, each message in the inbox may have the different expiration date, whereas in each mailbag, all of the messages therein will have a same expiration date.

When the GPO **106** traverses the inbox, it checks the expiration date of each message against the current date to determine if the message has expired. If the message has expired, the GPO **106** invokes the rule engine **283**, which processes the message against the rules in the gatekeeping rule base **289** to determine an appropriate action. This action may be any of the actions described above (release, delete, copy, forward, return). Alternatively, the action may be to move the message into one of the mailbags, such that the GPO **106** executes **449** the appropriate action for that mailbag upon expiration to the time parameter for that mailbag, as described above.

The operations of the GPO **106** and REPO **102** are concurrent and independent of each other.

As noted, a gatekeeper at a GPO may manually gate a message to another GPO **106**. Also, a message may be automatically gated by one GPO **106** to another GPO **106**, where it may be reviewed independently by another gatekeeper, or again, automatically by that second GPO **106**. This feature allows for either hierarchical or distributed gatekeeping of messages by any number of independently operating GPOs. That is, a message may be processed through a series GPOs **106**, each of which may apply its own set of rules for processing and further handling the message. Thus, while a first or second GPO **106** may not delete or return a message, an nth GPO **106** may in fact delete the message or simply release it to its specified recipients.

The three phases of rule definition, message processing, and gating are illustrated in FIGS. **4**A and **4**B in a linear fashion, but it is understood that the distinct phases can operate concurrently with each other. That is, once rules are defined, the message processing phase operates as its own process on any number of REPOs **102**. Similarly, the gating phase occurs concurrently at various GPOs **106**. Thus, each phase contributes independently to the overall operation of the system.

The operation of the REPOs **102** and GPOs **106** is here described with respect to e-mail messages. However, the present invention and the described embodiments are not limited to processing e-mail messages. The present invention and the described embodiments also operate on any other types of data objects that may be distributed over a network. For example, a data server may be used in an organization to route data objects such as forms, spreadsheets, applets, code objects, database information, or any other type of data directly from a sender to any number of specified recipients, without being attached to an e-mail message. The present invention enables the business rules to be applied to any such data objects at a REPO **102**, and gated, if a rule is satisfied, to a GPO **106** for further review. For example, a business rule may be defined to return to sender any code object larger than a specified size, or to gate forms having a specific data type. This general ability of the present invention to apply business rules to any type of data object provides substantial control and flexibility in the management of corporate communications.

V. Establishing Business Rules

Referring now to FIG. **5**, there is shown a flow diagram of the process of defining business rules for a rule base **270** or GRB. Rule definition proceeds with the administrator selecting **502** a REPO **102** for using a particular checkpoint or set of business rules. FIG. **6** illustrates one user interface **600** for selecting a REPO **102**, as provided by the GKADMIN **108**. Checkpoint field **602** indicates the name of a selected checkpoint. The post office or post offices which are available to use the checkpoint are listed in window **604**. The user selects one or more of these post offices to use the named checkpoint. The business rules defined within the checkpoint are listed in window **606** so that the administrator can determine the content of the checkpoint.

The administrator then designates **504** the bypass roles applicable to the checkpoint. The current bypass roles are

listed in window **608**. The administrator defines new bypass roles by designating various positions within the organization. FIG. **7** illustrates a sample user interface **700** for designating bypass roles. Available roles **702** within the organization are shown in window **703**. To obtain this role information, the GKADMIN **108** queries the organizational database **111** for all available roles **702**. These roles **702** are then listed by the GKADMIN **108** in window **703**. The administrator selects one or more of these roles **702** and moves them over to the assigned bypass roles window **705**, where they are listed as bypass roles **704**. The implementation of bypass roles within the REPO **102** may vary. In one embodiment, bypass roles are implemented in the same fashion as other business rules, further described below.

The administrator next defines **506** any number of business rules for the checkpoint. These rules are stored within the rule base **270** of the REPO **102** or GPO **106** for which the checkpoint is being defined. Each rule defines a specific action to be taken when an attribute of a message or data object satisfies an operator with respect to a user-defined value. The attributes may be static attributes of the message, such as message size, identity or number of recipients, number of destinations, message priority, message type, the text of the message, the subject line of the message, number of attachments, or the like.

Attributes may also be defined by performance or statistical attributes of the REPO **102** itself. Useful performance or statistical data include the number of messages processed per hour by the post office, the average size of messages processed per hour, the average number of attachments per message, largest message size in past hour, largest attachment size, and so forth. Those of skill in the art will appreciate that many other performance or statistical measures may be used as attributes for rule processing.

Each data object has various properties, such as being a message, attachment, and so forth. Each property has a name and an attribute. For example, the attachment property of a message has attributes such as the number, size, largest size, smallest size attachment. Any of the various attributes of a property can be the basis for an action by the distribution engine **230**.

Operators include arithmetic operators (=, <, >, _, ^{2}, ^{3}) and content-oriented operators (has does not have; is in: is not in). Values may be numeric, text, or labels.

Business rules within a checkpoint may be classified (for purposes of explanation) into various classes: attachment rules, destination rules, destination level rules, message rules, sender rules, and post office rules according to the property of interest. Tables 1 through 6 describe each of these rule classes. Each table lists the available operators and values used to define the rule for each property and attribute.

TABLE 1

Attachment Rules

Property	Attribute	Operator	Value
Attachments	Total count	Equals, Not equal, Greater or equal, Less or equal, Greater, Less	Specific number
	Memo count (number of memo attachments)	Equals, Not equal, Greater or equal, Less or equal, Greater, Less	Specific number
	File count (number of file attachments)	Equals, Not equal, Greater or equal, Less or equal, Greater, Less	Specific number
	Other count (number of attachments of, types other than, memo or file	Equals, Not equal, Greater or equal, Less or equal, Greater, Less	Specific number
	Attachment size, (total) (total size of all attachments)	Equals, Not equal, Greater or equal, Less or equal, Greater, Less	Specific size in (kilobytes)
	Largest, attachment size	Equals, Not equal, Greater or equal, Less or equal, Greater, Less	Specific size in (kilobytes)
	Smallest, attachment size	Equals, Not equal, Greater or equal, Less or equal, Greater, Less	Specific size in (kilobytes)

TABLE 2

Destination Rules

Property	Attribute	Operator	Value
Destination	Contains list	Equal, Not equal	Yes or No
	Number of recipients	Equals, Not equal, Greater or equal, Less or equal, Greater, Less	Specific number
	Number of locations	Equals, Not equal, Greater or equal, Less or equal, Greater, Less	Specific number
	Distribution list	Has, Does not have	Specific distribution, list
	Locations	Has, Does not have Is In, Is Not In	Specific post office Specific distribution, list
	Role	Has, Does not have	Specific role

TABLE 3

Destination Level Rules

Property	Attribute	Operator	Value
Destination, Level	Level name	Equals, Not equal, Greater or equal, Less or equal, Greater, Less	Specific level

TABLE 4

Message Rules

Property	Attribute	Operator	Value
Message	Revisable	Equals	Yes or No
	Confidential	Equals	Yes or No
	Priority	Equals, Not Equal	Low, Medium, or High
	Size (in Kb)	Equals, Not equal, Greater or equal, Less or equal, Greater, Less	Specific size (in kilobytes)
	From/MSG Type	Equals, Not equal	A Form or A Message
	From Program	Equals, Not equal	Yes or No
	To Program	Equals, Not equal	Yes or No

TABLE 5

Sender Rules

Property	Attribute	Operator	Value
Sender	Role	Has, Does not have	Specific role
	Post Office	Equals, Not equal Is In, Is Not In	Specific post office Distribution list

TABLE 6

Post Office Rules

Property	Attribute	Operator	Value
Post Office	Messages Per Hour	Equals, Not equal, Greater or equal, Less or equal, Greater, Less	Specific number
	Average Message Size	Equals, Not equal, Greater or equal, Less or equal, Greater, Less	Specific number
	Largest Attachment Size	Equals, Not equal, Greater or equal Less or equal, Greater, Less	Specific number

FIG. **8** illustrates a sample user interface **800** of the GKADMIN **108** for defining business rule. Each rule has a user-supplied name **802** or description for easily identifying the rule. Each rule has an antecedent and a consequent. To define the antecedent, the administrator selects the property **804** and attribute **803** upon which the rule operates from drop down menus containing alternatives for these aspects of the rule. The administrator also selects an operator **806** as appropriate for the designated property the available operators are updated by the GKADMIN **108** according to the selected property and attribute. The administrator then supplies or selects a value **808** of the appropriate type. The GKADMIN **108** provides for type checking to ensure that the rule is well formed.

A business rule may have multiple conjuncts or disjuncts; the administrator indicates whether these apply and their type through buttons **810**.

Once the antecedent is defined, the administrator defines the rule consequent by specifying the action to be applied to a message satisfying the antecedent conditions. The action is specified in action menu **812**, and includes, as set forth previously, the actions of releasing, deleting, gating, copying, forwarding, or returning the message. Other user defined actions may also be implemented. A reason **814**, a user input text entry, may be optionally specified. This is particularly useful when the action is to return the message to the sender, or to gate the message, and thereby inform either the sender or the gatekeeper of the reason(s) why a message was not delivered.

Where the specified action is to gate the message, as shown in the example of FIG. **8**, the administrator specifies the destination by indicating in the gatekeeper address **816** the destination GPO **106** to receive the message, and the particular gatekeeper role within that GPO **106** for reviewing the message. When the action is executed, the message is delivered to the gatekeeper having the specified role.

As indicated by the various possibilities in Tables 1–6, there is a large variety of different types of business rules that may be constructed. Table 7 includes examples of useful business rules for use at a REPO **102**:

TABLE 7

Sample Business Rules

Antecedent				Consequent
Property	Attribute	Operator	Value	Action
Message	Size	>	100 kb	Return to Sender
Message	Text	Has	"Product X"	Forward to Marketing Dept.
Attachments	Number	>	10	Return to Sender
Destination	Role	Has	President	Gate to Gatekeeper
Sender	Role	Has	President	Copy and Release
Server	Message Load	>	100 messages	Return to Sender

The rules may be internally stored with the rule base **270** or gatekeeping rule base **289** by any of a number of useful implementing data structures. In one embodiment, the rules may be stored in tables in a relational database.

In another embodiment, the rules are encoded in code point structures, as more completely described in "Data Management Using Nested Records And Code Points," U.S. Pat. No. 5,634,123. FIG. **9** illustrates the code point model for a checkpoint and set of rules. In this embodiment, checkpoints **902** are container records that encapsulate further containers for rules. Each checkpoint container **902** includes checkpoint data **904**, for example, specifying the checkpoint name, the number of rules, number of bypass roles, and other checkpoint specific data. Bypass role data **906** may also be specified using links to the organizational database **111**.

Each rule container **908** encapsulates one or more antecedent containers **912** and an action container **922** for the rule action. Each antecedent container **912** includes data records for the attribute **914**, operator **916**, and value **918** specified for the rule. The attributes, operators, and actions are encoded with identification numbers, which the rule engine **210** engine uses to decode their specific meaning. There is alsoa value stored **920** indicating whether or not there is another conjunct or disjunct in the rule. The action container **922** includes the encoded action data **924**, and any

arguments **926**, such as destination address for gating or returning the message.

VI. Applying Business Rules to E-mail Messages

Referring now to FIG. **10**, there is shown an illustration of the data flow within a REPO **102**. Processing within a REPO **102** begins with the receipt of message data **201** by the receipt engine **200**. For receiving an e-mail message a receipt engine **200** provides conventional post office functionality. Generally, the receipt engine **200** establishes a connection with a client e-mail application or other message source, and exchanges parameters for initialization of the data transfer. The receipt engine **200** then receives the message data **201**. Upon completion of the receipt of the message data **201**, the receipt engine **200** terminates the connection. This process is repeated each time a message is received.

As the message data **201** is being received, the receipt engine **200** constructs a message object out of the message data **201**. Each message object has interfaces for accessing and manipulating data from its header components, message text, attachment, and other attributes and properties. After the message object is constructed, the receipt engine **200** passes a handle to the message object to the rule engine **210**, which can then obtain the message object for applying the rules thereto. The receipt engine **200** also provides the message object to the message index **240** for indexing and storage in the message store **250**.

Some of the messages that a REPO **102** receives may have already been processed by another REPO **102** and gated for delivery to a GPO **106**. The messages are identified by the receipt engine **200** as being wrapped messages, and are provided by the receipt engine **200** directly to the distribution engine **230** for distribution. Wrapped messages are further explained below.

The rule engine **210** receives the message from the receipt engine **200**, and processes it using the rules from the checkpoint assigned to the REPO **102**. For each message, the rule engine **210** generates an action list of one or more actions to be performed by the distribution engine **230** on the message. The action list may be associated with the message by an object handle or other association.

The distribution engine **230** traverses the action list for the message, and applies the action having the highest priority.

Rule Processing

Referring now to FIG. **11**, there is shown the data flow within a rule engine **210** for processing a message. The rule engine **210** holds references to the rule base **270** for obtaining rules therefrom. The rule engine **210** also holds a reference to the distribution engine **230** for invoking distribution of a message and passing the distribution engine **230** the action list **1107** for the message.

The rule engine **210** comprises three components, an attribute resolver **1103**, an operand handler **1104**, and an evaluator **1102**. The evaluator **1102** controls the overall evaluation of a message against all rules in a checkpoint. For each rule, the evaluator **1102** uses the attribute resolver **1103** to determine the attributes of the rule and the corresponding attributes of the message, and uses the operand handler **1104** to determine whether the attributes of the message satisfy the operand and value of the rule. The attribute resolver **1103** returns a new attribute object with its value to the evaluator **1102**, and the operand handler **1104** returns a truth value.

Because a message may satisfy any number of rules within a checkpoint, the evaluator **1102** outputs an action list object **1107** which stores the actions from those rules that were satisfied. The action list **1107** includes a message ID which identifies the message and a list of actions as encoded values with their optional arguments.

As noted above, the evaluator **1102** performs a high level true/false analysis over an entire checkpoint with respect to a set of rules. Referring to FIG. **12** there is shown a flow diagram of the operation of the evaluator **1102** for evaluating a particular message.

The evaluator **1102** receives a handle to the message from the receipt engine **200**. The evaluator **1102** then traverses **1202** the rule base **270** for each checkpoint assigned to it. For each rule in the checkpoint **(1204)**, the evaluator **1102** evaluates the antecedents of the rule with respect to the attributes of the message or performance attributes of the REPO **102**. For each antecedent **(1206)**, the evaluator **1102** invokes **1208** the attribute resolver **1103** and passes in the rule.

The attribute resolver **1103** is a state machine the calls the appropriate interface of a rule to obtain an attribute, and interfaces of a message to obtain an attribute thereof. The attribute resolver **1103** also holds reference to the receipt engine **200**, the distribution engine **230**, and the organizational database **111** in order to obtain role information, or other attribute information. The attribute resolver **1103** returns an attribute object which includes the value of the attribute extracted from the message, or from the distribution engine **230**, and its type.

The attribute resolver **1103** operates as follows: The attribute resolver **1103** receives a reference to a message object and a reference to a rule object. The attribute resolver **1103** gets the attribute of the rule object. The attribute resolver **1103** then calls either the message object or the distribution engine **230** in order to obtain the appropriate attribute information. As noted above, the attribute information may be external to the rule, such as the performance data of the distribution engine **230**, or other other statistical data from the post office.

For message attributes, the attribute resolver **1103** requests the message to provide the attribute information for the specific attribute. For example, the attribute resolver **1103** will determine that the attribute of particular rule is the "message size." The attribute resolver **1103** then invokes the message to determine the corresponding attribute information from the message. In this example, the attribute resolver **1103** would obtain the size of the message. The attribute resolver **1103** then stores this message attribute information in a local cache.

For an aggregate or performance attribute of the distribution engine **230**, the attribute resolver **1103** calls the distribution engine **230** to obtain the required data. For example, the attribute of the rule may be the server load, in which case the attribute resolver **1103** calls the distribution engine **230** to provide this data, which is again stored in a local cache.

With the attribute information, the attribute resolver **1103** constructs an attribute object that stores the attribute data and its type. The attribute resolver **1103** then returns this attribute object to the evaluator **1102**.

For bypass roles, the attribute resolver **1103** invokes the organizational database **111** to determine the role of a particular user, and this information is stored in the attribute object.

After receiving the attribute object from the attribute resolver **1103**, the evaluator **1102** invokes **1208** the operand handler **1104**, and passes the rule and the attribute object to the operand handler **1104**. First, the operand handler **1104** determines whether it is necessary to query the rule object to obtain more information than is available directly from the message, such as the uncompressed size of an attachment, an aggregate value that covers the domain of the message, or

the run time environment of the REPO **102** or GPO **106**. The operand handler **1104** calls the attribute object to obtain the value of the attribute. The operand handler **1104** calls the rule object to obtain the value specified in the rule. The operand handler **1104** then compares the attribute value with the rule value using the operand from the rule. If the comparison is true, the operand handler **1104** returns "true" to the evaluator **1102**, meaning the rule was satisfied. Otherwise, the operand handler **1104** returns "false."

For example, assume a rule:

IF message size>100 kb THEN Return message.

Further assume a message with a message size of 50 kb.

Here the attribute resolver **1103** receives the message object, obtains the message size, and constructs an attribute object storing the value of "50", and passes this object back to the evaluator **1102**. The evaluator **1102** calls the operand handler **1104** with the attribute object and rule object. The operand handler **1104** calls the rule object and obtains a value of "100" from the rule, and an operator of ">". The operand handler **1104** performs the greater than comparison of 50>100, which evaluates to "false." This result is returned to the evaluator **1102** as the evaluation of the current antecedent.

The evaluator **1102** stores **1212** the result from the operand handler **1104**, and then tests **1214** whether the conjunct of the rule is true. If the rule conjunct is true, the evaluator **1102** proceeds with the next antecedent. If the rule conjunct is not true, the evaluator **1102** proceeds with the next rule, since the current rule had a false conjunct, and thus failed to be satisfied.

When all antecedents are evaluated, the evaluator **1102** determines **1216** if all of the antecedents are true. If so, the evaluator **1102** obtains **1218** the action from the rule and stores the action in the action list, constructing a new action list object as necessary. The evaluator **1102** proceeds to evaluate the next rule.

Once all rules in a checkpoint are evaluated, the evaluator **1102** proceeds with the next checkpoint until done. The evaluator **1102** then invokes **1220** the distribution engine **230** and passes the action list to it for handling.

Referring to FIG. **4**A, bypass role processing **406** is handled by a set of rules, each of which define a bypass role in the antecedent, and a release action as the consequent. These rules are stored in a primary checkpoint which is evaluated **(1202)** as the first checkpoint during rule processing by the evaluator **1102**.

The operation of the rule engine **210** is equally applicable to data objects other than e-mail messages, with the rule engine **210** and its components obtaining the appropriate attributes of a data object for applying a business rule thereto.

Action List Processing

Once the evaluator **1102** completes evaluation of a message with respect to the rule base **270**, and invokes the distribution engine **230**, the distribution engine **230** completes the processing by applying the actions in the action list to the message. Referring now to FIG. **13** there is shown a flow diagram of the operation of the distribution engine **230** in processing messages.

The distribution engine **230** receives the action list and message from the rule engine **210**. For each action in the action list **(1302)** the distribution engine **230** determines **1304** the priority of the action Priority information is held in a separate priority object; each action has a default priority level which may be adjusted by the administrator. The default priority levels (highest to lowest) for actions are ordered follows: Gate, Return, Forward, Copy, Delete, Release. The distribution engine **230** calls the priority object, passing the action from the action list, and obtains the priority level for the action.

The distribution engine **230** also maintains data for the current priority level, which is initially set to 0. The distribution engine **230** compares **1306** the current priority level with the action's priority level. If the action priority is higher than the current priority, the distribution engine **230** updates **1308** the current priority with the action priority, and continues with the next action. If the current priority is higher, then the distribution engine **230** simply continues.

This process identifies the highest priority action or actions to be executed. Lower priority actions will not be executed. For example, if the highest priority action is to delete the message, then the lower priority release action will not be executed. Similarly, if the highest priority action is to gate the message, then there is no reason to copy the message. Accordingly, the distribution engine **230** deletes **1310** all actions with lower priority than the current priority level.

The distribution engine **230** then executes **1312** the remaining actions in the action list, in order.

If the action is "release," the distribution engine **230** releases **1314** the message, sending it to the next distribution point for delivery to its specified recipients. It is still possible that the message will not be delivered to such recipients because it may be gated by another REPO **102** or GPO **106** within the system. The distribution engine **230** then continue **1312** with the next action in the action list.

If the action is "delete", the distribution engine **230** deletes **1316** the message, clears the action list and continues.

If the action is "return", the distribution engine **230** returns **1318** the message to its sender, clears the action list, and continues.

If the action is "forward", the distribution engine **230** forwards **1319** the message to recipients specified in the action for the applicable rule, clears the action list, and continues.

If the action is "copy", the distribution engine **230** copies **1320** the message, and then releases the message for distribution.

Finally, if the action is "gate", the distribution engine **230** first wraps **1324** the message object and the action list into a wrapped message. A wrapped message is a special type of message within the system. FIG. **14** illustrates the structure of a wrapped message **1402**. The wrapped message **1402** contains a distribution list **1414** of all the GPOs **106** that are to review the message based on the rules that fired. This distribution information is encoded in the action list, since each gate action includes the address of the gatekeeper and GPO **106** that is to receive the gated message. The last addressee on this distribution list **1414** is the REPO **102** (or GPO **106**) that is sending the message for gating. The distribution engine **230** extracts this distribution information from the action list and stores it in the distribution list **1414**. The usefulness of placing the REPO **102** last on the distribution list **1414** is explained further below.

The wrapped message **1402** also contains a rule history **1416** of the reasons why the message was removed from the normal distribution stream. Each rule is identified by a rule ID and a timestamp indicating when the rule was fired. This information is used by the GKI **107** to inform the gatekeeper of the reasons the message was gated.

The wrapped message **1402** contains flags **1404** indicating the status of the message. The wrapped message contains a flag that indicates that the message should be delivered

unabated from this point forward to the designated GPO **106**. This allows the message to be immediately delivered to the GPO **106** without further processing.

Finally, the wrapped message **1402** includes the original message data **1406**, including its header **1408**, body **1410**, and any attachments **1412**.

The flags **1404**, distribution list **1414** and rule history **1416** are preferably encoded after the message data **1406**; in this manner these portions of the wrapped message are ignored by conventional post offices, which deliver the message in a conventional fashion. Only a REPO **102** or GPO **106** programmed in accordance with the present invention processes this wrapper information.

Accordingly, after constructing the wrapped message **1402**, the distribution engine **230** distributes **1326** it to the first GPO **106** on the distribution list **1414**.

As noted above, the REPO **102** may itself receive messages from other REPOs **102**. If these messages are wrapped messages, then they have already been gated and need not be reviewed by the rule engine **210** of the recipient REPO **102**. The flag information in the wrapped messages indicates this status to the receipt engine **200**. The receipt engine **200** provides the wrapped message directly to the distribution engine **230**, which exports an interface for directly releasing a message.

VII. Gatekeeping at a Gatekeeping Post Office

Configuring the GPO

Referring now to FIG. **15** there is shown a flow chart of the process of configuring a GPO **106**. In one embodiment, a GPO **106** is configured with the GKADMIN **108**, instead of from within the executable of the GPO **106** itself. This lets an administrator easily configure any number of GPOs **106** on the network, without having to load and execute each one individually, which may not always be possible, since some GPOs **106** will execute on remote computers from the one being used by the administrator.

Configuration generally begins with designating **1502** a post office as a GPO **106**. Any REPO **102** may be designated as a GPO **106**, and operate accordingly. This results in enabling the extended functionality of the GPO **106**. For the purposes of this disclosure, the REPO **102** becomes a GPO **106** when so configured. The approach provides considerable flexibility to the administrator in configuring the mail system, depending on the performance, server loads, and other considerations. FIG. **16** illustrates a user interface **1600** from the GKADMIN **108** for designating a post office as a GPO **106**. The administrator checks the "Receives Gated Messages" box **1602** to designate a REPO **102** as being a GPO **106**. In addition, at any time a GPO **106** may be changed back to a REPO **102**, so that it no longer provides gatekeeping functionality.

The administrator then specifies **1504** an interval time parameter and an optional time out action: for example, as illustrated in FIG. **16**, fields **1604** and **1606** respectively, are the parameters associated with the gatekeeper's inbox in the GPO **106**. The interval time parameter is the number of minutes, hours, days, or other interval of time for which gated messages are left in the inbox to allow the gatekeeper to manually review the messages. This time value is added to the timestamp of each gated message as it is received at the GPO **106** to produce an expiration date for the message. As described above, the expiration date is then used by the GPO **106** to automatically initiate a further action to be applied to the message. The optional time out action may be used to specify the action to be applied to all expired messages, rather than individual actions as would be applied from the business rules in the gatekeeping rule base **289**.

The administrator also designates **1506** gatekeeper roles. These roles are used to determine which members of the organization will have access to gated messages for further evaluation and disposition. FIG. **17** illustrates a sample user interface **1700** of the GKADMIN **108** for designating gatekeeper roles. An organizational role **1702** is selected, and associated with a gatekeeper role **1704**. The roles used for assignment are stored in the organizational database **111**. The GKADMIN **108** creates a separate table of gatekeeper roles and stores this with the GPO **106** being configured.

Receiving Messages

FIG. **18** illustrates the operation of the receipt engine **282** for receiving messages, when a GPO **106** also includes rule processing functionality for handling non-gated messages. The gatekeeping rule engine **283** operates to receive **1802** both conventional and gated (wrapped) messages from either conventional post offices, other GPOs **106**, or REPOs **102**, and processes these messages accordingly for handling by a gatekeeper (gated messages) or the distribution engine **284** (non-gated messages). The low level operation of the receipt engine **282** for setting up and receiving a message is essentially the same as described above. The receipt engine **282** differs in that it performs additional processing on a gated message.

The receipt engine **282** determines **1804** if the message is a gated message by whether it contains the additional wrapper information of the distribution list, flags, and rule history information. If the message is a gated message, then the receipt engine **282** provides **1810** the gated message to the inbox of the gatekeeper to whom the message is addressed. This gatekeeper can then review the gated message at a later point in time. The receipt engine **282** updates the gatekeeping message index **287**, including storing in the gatekeeping message index **287** the expiration date for the message, based on the timestamp of the message and upon the interval time parameter of that particular gatekeeper's inbox.

If the message is not gated and hence not wrapped, then the receipt engine **282** sends **1806** the message to the regular message index **240** for indexing and storage, and then invokes **1808** the rule engine **210** to process the message in the same manner as described above, using the rule base **270**. In this manner, wrapped (gated) messages are specifically diverted from further rule processing at this stage, but rather, stored in the inbox of the gatekeeper until such time as they are manually or automatically reviewed.

Manual Review of Gated Messages

Each GPO **106** provides for manual review of messages by one or more persons having been designated the role of gatekeepers. At each GPO **106** there may be any number of assigned gatekeeper administrators (individuals responsible for reviewing gated messages). When a REPO **102** sends a gated message, the rule that fires indicates which gatekeeper is to review the message. The distribution engine **230** of the REPO **102** sends the message to the GPO **106** associated with the designated gatekeeper. As noted above, such a gated message is placed in the inbox of the gatekeeper for review.

In one embodiment, the gatekeeper accesses gated messages through a separate utility program, the GKI **107**, operates like a mail client, but includes additional functionality. Access through a separate utility program is desirable because the gatekeeper can thereby access any number of GPOs **106** for performing manual review of gated messages, instead of having to execute built-in functionality of each GPO **106**.

When the gatekeeper accesses a message in the inbox, the GKI **107** automatically unwraps the message and provides