

context, owners of a process can be product managers, purchase agents, account managers, customer service representatives, etc. An individual also may act as a Participant (also termed a Member or Recipient). Members interact with one or more specific tasks in a process, such as interviewing candidates, bidding on proposals, participating in meetings, etc. The owner of a process can assign specific roles to the participants. Owners may be participants also.

In one embodiment, with respect to groups, an Owner can add or remove members from the group; carry out moderator options; exercise author rights over all group messages; delete group messages; and exercise all Member privileges. A group Member, in contrast, can send or receive group messages; invite others to join public groups; unsubscribe from the group; and create a sub-group.

Notifications, in one approach, are alert messages that are sent to users when a predefined activity has occurred in a process. For example, notifications may be issued as a result of the following activities: Response to a building block by a user; change in process or task status; change in process or task due dates; more than 50% of people have polled; more than 60% of users have confirmed for the meeting; and others. Reminders are prompts sent to users. Examples of reminders include: Inform a user that a due date is fast approaching; inform all users about an important process development; etc. Both notifications and reminders generally are sent outside the context of a transportable application, for example, by a separate e-mail message directed to the recipient. In contrast, notes, as described herein, are text messages that are selectively embedded within a transportable application to draw something to the attention of the recipient when the recipient opens the transportable application.

Notifications may be pull or opt-in notifications, or push notifications. With pull notifications, a user defines (or sets rules) when to receive notifications and reminders. The system automatically sends a notification if the specified definition is satisfied. With push notifications, an owner sends notifications and reminders to users regardless of whether the user has requested for the same. Here the owner overrides the notification preference of the recipient.

In one embodiment, notification processes are configured so that a participant receives notifications on any updates immediately. In this approach, users have the option to opt-in to receive any updates. The user receives a single notification on any updates since the last read. In one specific approach, taking any of the following actions on a process or task triggers delivery of a notification: Adding new tasks; response to building

block; closing of tasks; closing of processes; change in process status or due date; change in task status or due date.

In one sub-approach, only the tasks for which a user is in the recipient list trigger notification updates. If a different task changes, for which a user is not in the recipient list, the user is not notified.

In another embodiment, notification processes are configured so that a participant receives notifications on any updates to specific tasks immediately. Users can receive notifications on any updates to a single task or group of tasks. A list of active tasks is provided to the user from which the user can select tasks on which to be notified.

The notification processes also may be configured to send a process-level reminder to all recipients. Specifically, the owner of the process can configure a transportable application to send ad-hoc reminders everyone in a recipient list whenever an important process event occurs. For example, assume that Michael is interviewing at Alpha Company and an interview process transportable application is currently used for scheduling interviews. A manager at Alpha receives information during the interview process that Michael has a competing offer and needs to decide whether to accept it within the next week. The manager, who is an owner of the transportable application, can immediately notify all the participants that they should schedule interviews for Michael and decide on the candidate. In a related approach, the notification processes are configured to send process-level reminders to selected recipients in the recipient list.

In another embodiment, the system is configured to send a task level reminder to all recipients in a recipient list. The owner of the task can send ad-hoc reminders to everyone in a recipient list whenever an important task event occurs. For example, assume that an offer letter to candidate John Q. Public is under discussion in the "Offer task" of a transportable application. The salary to be offered to the candidate is still under discussion among the managers. However, the Director of Sales needs to provide the sales headcount to the VP of Marketing & Sales next week and as such needs to finalize the offers quickly. She sends a notification to the participants in the "offer task" to come to a consensus quickly on the offer and go further with the hiring. In a related feature, a participant can send a task level reminder to select recipients among the recipients of the transportable application.

Another feature provides scheduled process update notification. A user can schedule to receive process update notifications periodically or on a specified date and time. Periodic update options for the user to select are daily (options within a day), and weekly (options within a week). For example, assume that Bob is the Director of Business

Development at Alpha Company and his team is working on new business deals with a lot of startups. Bob would prioritize on his updates based on the importance of the deal. So he schedules some deals for weekly updates while others for daily updates.

In another embodiment, the system is configured to send scheduled process "due date" reminders to all users in a recipient list of a transportable application. In this feature, the owner of a process can schedule specific "due-date" reminders to be sent to everyone either at the process or at the task level. The owner can send the reminder either on a particular date or a specified period before the due date (e.g., two days before, two weeks before, etc). For example, assume that a product management team has defined new features for a particular product release and requires approval from other functional areas (such as Engineering, Sales, Business Development etc.). The features need to be frozen by a certain date so that development on the product can commence. The project lead schedules a "due-date" reminder to be sent to everyone a week before the deadline to ensure that the activities are completed by the due date.

In other features and embodiments, a participant or user can "opt-in" to receive summary of notification changes; "opt-in" to receive selected notifications immediately; send a scheduled task "due date" reminder to everyone in a recipient list; send a scheduled task "due date" reminder to select recipients in a recipient list; and "opt-in" for a scheduled process due date reminder.

In one implementation approach for the foregoing features, the event-based messaging system described herein is configured to enable building blocks and associated notification event handlers to communicate. One or more events may issue as a result of another event. Responses to events are carried out by a notification event handler that is associated with each kind of response event. Response-based notifications are generated by each such handler. Each notification is an event, and the each notification event handler comprises logic that determines which users need to receive notifications and when. In an embodiment, each event handler uses a notification API to generate a list of users to notify, and the event handler then sends the list to an event daemon that dispatches the notifications. As a result, an event-based messaging system facilitates generating rule-based notifications in response to any change in any attribute of a transportable application.

In one implementation of response-based notifications, as outlined above, each user may "opt in" to receive notifications at a task level and at a process level. Hence, each user can subscribe to changes in particular tasks or to any change in the process. Further, each user can associate a notification frequency value with each subscription.

FIG. 7 is a flow diagram of one embodiment of a process of carrying out response-based notifications. In block 702, a response is issued to a building block of a transportable application. For example, a first participant in a group collaboration or other activity enters text, graphics, a button selection or some other value in response to a query provided in a building block.

In block 704, in response, a database query is issued to obtain a list of users who have requested notifications for the current building block. A notification time value is obtained for each user in the list; the notification time value indicates when to notify each individual on the list. In block 706, the list is passed to the parent object of the current building block, which may be another building block or a container object, with a request to carry out notifications.

In block 708, a list of recipients associated with the parent building block or container object is retrieved and compared to the list of users who qualified for notifications at the child building block level. Only those users who qualified for notifications at the child building block level are then considered. For each user who qualified, if that user has a notification time value that indicates a delayed notification is necessary, then no action is taken since the child's notification time overrides any notification time that may be associated with the parent.

In block 710, for each user in the child notification list that qualifies for an immediate notification, then a database query is carried out to determine that the user is active in the system and does not have a notification already pending. This is done to avoid duplicate notifications. If these tests result in a determination that the user is entitled to a notification, then control is passed to block 712.

In block 712, a status value for the user associated with the parent building block or container is changed to Updated, and a current time value is stored in a notification time value in the database.

In block 714, users in the recipient list of the parent block or container who did not qualify for child level notifications are considered. The status value for each such user is changed to Updated, and a current time value is stored in a notification time value in the database. In block 716, the notification message is dispatched to all qualifying users in the parent and child notification lists.

### **1.5.2 Notifications Based on Rules and Attributes**

In a related approach that is integrated with an event management system, each building block can publish attributes about itself to the rest of the system, and publishes event that alert the system when such attributes change. Further, users may create and



store rules based on these attributes that cause such users to receive a notification when the rules are satisfied. In addition, users may be notified at a particular time if a rule is satisfied. For example, a user can be notified if a project status reaches “complete”, and the user can also be notified if the project status is not “complete” one month after the project began.

In one implementation approach, database 208 comprises a rules table having the following columns: Rule ID; Block ID; User ID; Attribute; Threshold value; Comparator; Time flag indicating whether the rule is time-based; Event ID if the rule is time-based; Action type. In one embodiment, the Rule ID field does not store a unique key value, because the same Rule ID can encompass several rules that are evaluated simultaneously.

Database 208 further comprises an alert log table having the following columns: Block ID; User ID; Note; Read bit. An Alert Waiting bit is provided in a user status table. Each block is associated with a presentAttributes method that returns one or more attributes, types, comparators (if applicable), and description values for each attribute. An interface is accessible from each transportable application with which a user can build the rules and set threshold values and comparators. Each rule may be characterized in terms of Boolean values, number comparisons (equals, less than, greater than), string equals comparisons, etc. Using the interface, a user may edit the rules that have been created, and attach a time value and recurrence period to a rule.

In response to a user creating a rule using the interface, a servlet of application server 202 enters the rule into the database 208, and attaches a rules event to the building block in which the rule was created. The rules event subscribes to attribute changes in the building block. The servlet also deletes any old rules in the database for the same building block.

Thereafter, when an attribute changes in the block, the rules event is invoked. Processing the rules event involves first retrieving all rules for that block from the database 208, evaluating the rules as designed to result in creating and storing a list of rule identifiers that evaluated to TRUE, and generating a rules-passed event that includes the list.

Actions can subscribe to the rules-passed event. Each such action has an associated rule ID value. If a rule matching the associated rule ID value is fulfilled, then the action is executed.

An Alert Notification event object is provided and has a handler process that determines if its rule has passed. If so, the handler sends a notification to the user if needed, and records the notification in a notification log.

In one approach for displaying notifications, when a user opens and reads a transportable application, a flag message is displayed that informs the user that a new notification exists. The flag message may be a hyperlink. The user selects the flag message. In response, the system displays the notification in a pop-up dialog with which the user may scroll through one or more notifications. Each alert then is marked as read.

#### **1.6 Object Communications—Programmatic Methods**

In one embodiment, system 200 uses two distinct types of internal communication mechanisms. Non-event driven sharing of data is carried about in Building Block and Container interactions and Container-to-Container communications. Event-driven publish-subscribe exchanges are carried out between disparate objects within the system.

Non-event-driven data sharing is used in cases in which communications require detailed knowledge of the hierarchy of objects or the need to transfer essentially the entire data of such objects. In order to ensure the successful delivery of these communications, a unique identification system is provided for all objects that will communicate within the system. In one embodiment, each object in the system has a unique global identifier, as described further in this document in the section entitled “Directory Integration—Global Object Identifiers.” Using global identifiers and associated mapping tables, container objects for contained objects can be determined.

In an alternative embodiment, a global object identifier is associated only with container objects. The relative position of a contained object within the container object is used as a unique identifier of the contained object. As a result, each contained object is accessed only through its immediate Container. For example, a poll Building Block within a task in a Process Container would have the id: <Process Container ID>\_<task index>\_<poll BB index>, or alternatively, <Task Container ID>\_<poll BB index>, if the Process Container was not needed in order to deliver the message. A benefit of this mechanism is that there could be ACLs applied on a particular Container that may affect access to a contained object.

In one embodiment, a data-sharing communication mechanism is used in order to aggregate data from multiple Building Blocks in order to form a composite view. For example, in the case of a Poll Discussion, the Data Access Component for the poll building block and the discussion building block are joined by a composite Building Block in a particular way in order to show both the poll and discussion data together. In order to join such a composite view, the Container collects multiple Data Access Components from the blocks and delivers them to the Composite Building Blocks.

In one specific embodiment, containers or other objects in the platform implement a `DataSharingInterface` in order to achieve communication. The interface is defined as:

```
public interface DataSharingInterface {
    public DAC  getDataAccessComponent (RelativeID
target, UniqueID requester, UserID user);
}
```

The Data Gathering Service Manager implements a `DataGatheringInterface` in order to extract the DACs of the objects that a component may want to access. This interface is defined as:

```
private interface DataGatheringInterface {
    public      Vector      gatherDataAccessComponents (UniqueID
requester, UserID user, UniqueID[] fromList);
}
```

In one example embodiment, the Data Gathering Service Manager loops through each element in the `fromList`, determines the Container, sends the container to a `DataSharingInterface` and calls the `getDataAccessComponent` method with the `RelativeID` of the specified Blocks. The Container implementing the `DataSharingInterface` gathers the appropriate DAC from the specified Block. This process allows Building Blocks across Containers to share data, and also allows the sharing object to limit the amount of data that should be sent out to the requester.

### **1.7 Object Communications—Event-Driven Methods (Event Handling System)**

In one embodiment, the system described herein provides an event handling service as represented by event processor 112 of FIG. 1B, and event daemons 216 of event service 146 of FIG. 2B. In this embodiment, one or more event daemons 216 are communicatively coupled to event service 146. The event service 146 is communicatively coupled by link 218 to database server 208. The event daemons serve to offload certain separable functions from the application server 202. For example, in an embodiment, event daemons are responsible for mail event queuing and handling, bounced-mail handling, and generating personalized transportable application content, based on a user's e-mail client profile.

Alternatively, a generic event handling system is provided to enable different components of the system to communicate. In one embodiment, an event handling system enables the system to act when a specific event occurs within a transportable application, act when a specific event does not happen, and facilitates authoring rules to carry out the foregoing. Actions may include generating notifications, generating reminders, forwarding a transportable application, other automated actions, delivering a message to subscribers, etc. In one embodiment, actions may comprise anything that can be carried out programmatically. The event handling system may comprise an object framework, message format and implementation classes.

FIG. 17 is a block diagram illustrating elements of an event handling system, in one example embodiment. Event handling system 1700 is hosted in application server 202 and comprises an event router framework 1702, event broker framework 1704, and event timer framework 1706. Event router framework 1702 performs message routing, selects a transport mechanisms for messages that are sent, and serves as an entry point for other components of the system that need to use events. Examples of transport mechanisms include JMS, HTTP posts, etc. Event broker framework 1704 performs rule evaluation that involves the filtering of event messages and invoking action classes, and can store event messages in a table of database 208. Event timer framework 1706 enables creating event messages at a specified time, for processing time-based rules.

Events are programmatically represented by event messages. In general, event messages contain information about what occurred and the state of objects that relate to the event. Standard events include creating, updating, deleting, and changing the state or status value of a transportable application. For example, an update event for the poll building block may contain the building block identifier, the user name of the person who added a response, the response value, the time and date of the response, the total number of responses, and the total number of recipients. However, each building block may generate any desired events having any desired data or content. Events may be time-based. For example, events are generated or created by invoking particular methods when the prescribed time for an event arrives.

Each building block has a method which, when called by another program element, returns a list of events that it can generate. This enables other program elements to identify and subscribe to events.

In one specific embodiment, each event message comprises a header and a body. The header comprises metadata, and the body comprises information that identifies the container and building block that generated the event, the name of the event, etc. In one

specific embodiment, the header comprises a fromDestination value that identifies the originating system; a toDestination value that identifies a destination system for the message; a message type value; a timestamp value that identifies a date and time at which the event occurred; a message action value; and a tracking identifier value. The body encapsulates another header (“inner header”) and inner body or payload that contains event-specific data. Events and their data may be defined by an XML schema.

The message type value enables an event message to specify whether it is a system event, application event, etc. Examples of system events including replication events, system administration events, initialization events, etc. Application events may be events generated by transportable applications, connectors, groups, etc. Each event type has a corresponding schema that defines the elements of the inner header and payload for that event type.

Event messages may be persistent. Persistent event messages are stored in an event table in the database 208. Events can be made persistent by programmatically setting a “Persist” flag in the event message header. Alternatively, the event type definition may specify that all event messages of that type are persistent.

In one embodiment, containers generate events that are published to the event handling system. The event handling system applies rules to determine whether received events should result in an action. If the rules are satisfied by the events or other data, then actions result.

Rules may be associated with building blocks or containers. Rules may be saved in association with a template of a transportable application.

Rules may be subject to author control or participant control. In rules with author control, only users who are authors of a transportable application template can modify or deactivate the rules. In rules with participant control, any participant who receives a transportable application that is instantiated from a template having the rule can modify the rule.

Rules may be designated as active or inactive. Active rules are visible within a template of a transportable application and within an active transportable application.

Each rule comprises an association with one event through a coarse-grain filter, a fine-grain filter that has one or more conditions, zero or more constants, one or more actions or handler. Rule constants can comprise a static string or may be defined as reusable expressions.

Rule conditions may be created as coarse-grain filters or fine-grain filters. Coarse-grain filters determine whether a particular event message maps to or is associated with a

pertinent set of rules for the event. Thus, coarse-grain filters carry out filtering only on a header portion of an event message. Coarse-grain filters support, for example, static strings or wildcards for filtering events based on header elements. An example of a coarse-grain filter is, "EventType=createResponseVoteRequest". This filter would pass only event messages that result from an end user issuing a vote in a poll building block. The coarse-grain filter "SenderID=1222" would pass only event messages created as a result of actions by a specific user (user "1222").

A fine-grain filter is a filter that contains conditions used to decide whether an associated action should be fired or not for a particular rule; the action is invoked only if all conditions in the filter are satisfied. Conditions in a fine-grain filter may be applied against any data in a message or against dynamically retrieved data. Fine-grain filters generally are defined by a custom class that implements an interface, or specific programmatic expressions that invoke methods. In one embodiment, fine-grained filters are defined as Xpath statements according to the format specified in the document "xpath.html" that is available at this writing in the "TR" folder of the "www.w3.org" directory and domain on the Internet. An example of an Xpath statement is "/message/body/poll/currentCount/text() > 5," which states that the value of the variable "currentCount" of the text() method of the poll building block shall be greater than "5".

Actions are implemented as handler classes that can invoke any programmatic method or routine. In general, the handler classes are implemented within a building block that generates the events that include the actions associated with the handler classes. In one embodiment, during rule editing, a rule author may select one of a plurality of standard actions that are provided by a graphical rule editor. Alternatively, custom actions can be created by preparing appropriate program code that is uploaded to application server 202 and registered with the event handling system. Examples of standard actions include: system action for notification; system action for closing a transportable application; system action for unclosing a transportable application; system action for updating a status field of a transportable application; system action for creating a new page or transportable application based on a saved template; system action to change a role for a particular user for a particular page; system action to rename a page; system action to show a page; system action to hide a page; system action to open a page; system action to close a page.

Rules may be defined in XML format and attached to a building block, a page, or to a template for a transportable application. In one embodiment, rules may be created using a graphical Rules Editor, which is accessible from the transportable application

editor described herein, when building blocks, pages, or templates for transportable applications are authored. The Rules Editor is also accessible from within a transportable application that has been opened. Rule editing involves selecting a condition template from a scope of available condition templates, providing values for variables in the condition template, and selecting result actions. Rule editing may be carried out at any time during the lifecycle of a transportable application template or instance.

Rules may be evaluated or “fire” one or more times.

In one specific embodiment, to carry out event-driven messaging, containers for transportable applications or groups implement an EventHandler interface 340 of FIG. 3. In one embodiment, the EventHandler interface is defined as:

```
public interface EventListener {
    public void handleEvent(Message msg);
}
```

For containers of transportable applications in order to react to create, respond, and edit events, and for GroupContainers to add, modify and delete members, and for FolderContainers to add, modify or delete files, the following event handling process is carried out. First, the appropriate event is passed to an EventManager that forwards the event to a particular ActionManager for the specific type of event, for example, a ResponseActionManager. The ActionManager then calls a handleEvent method on the appropriate EventListener. In these cases it is clear which object is intended to act on this event. Accordingly, to require each container to subscribe to its create, respond, and edit events is superfluous and therefore point-to-point messaging may be used as an alternative. In point-to-point messaging, the ResponseActionManager calls the handleResponseEvent, making the Container design easier.

For general event handling when Building Blocks are generating information useful for other Building Blocks, a publish-subscribe model is used, in which objects subscribe to certain events from the EventManager. Subscribers implement the EventListener interface and handle the appropriate event.

Details of an embodiment of an event processing system are now provided. In one embodiment, an event processing system comprises tables in a database that are configured according to the database design and schema described herein, and programmatic objects that implement functions of the API described herein.

In general, in one approach, an Event Message may be published on many occasions, which identify the type of the message. Every Message object has a specific message type value (“MsgType”) associated with it. The specific message type will trigger the proper action, associated with the message type. In one specific approach, message type values are omitted, and each message provides attributes as name/value pairs that are accessible in a global memory space.

In one embodiment, a database schema that supports message processing comprises an attribute table and message table. The attribute table may have the following structure:

Name	Field	Field Description	Required	Key	Type
	MSGID	Id of the Message	Yes	Yes	NUMBE R(19)
ME	ATTRNA	Name of the Attribute	Yes	Yes	VARCH AR(64)
LUE	ATTRVA	Value of the Attribute			VARCH AR(1024)
PE	ATTRTY	Type of the Attribute	Yes		VARCH AR(16)

The MSGID and ATTRNAME fields are included in the primary key. The ATTRTYPE field is used to store information about internal type of the attribute on the app server side. This information is used to transfer the value of the attribute to the required type.

The message table may have the following structure:

Name	Field	Field Description	Required	Key	Type
	MSGID	Id of the Message	Yes	Yes	NUMBE R(19)
E	MSGTYP	Type of the Message	Yes		VARCH AR(64)
	OBJID	Id of the Object	Yes		NUMBE R(19)
E	OBJTYP	Type of the Object	Yes		NUMBE R(3)
ID	SENDER	Id of the sender, who posted the message			NUMBE R(19)
TYPE	SENDER	Type of the Sender, which posted the message (user, group, etc.)			NUMBE R(3)
E	MSGTIM	Time when message was created	Yes		DATE
	STATUS	Status of	Yes		NUMBE



	the Message			R(3)
E	EXPTIM Expiration time of this message	Yes		DATE

The MSGID field is the primary key. An index is created on the combined OBJID and OBJTYPE fields. The STATUS field represents an internal parameter and is hidden from the API. The STATUS field stores the result of database transaction and processing of the event.

An example class structure that implements an appropriate API is set forth in APPENDIX 1.

FIG. 17C is a flow diagram of a process of evaluating and acting on an event message. In block 1720, a transportable application type message is created. In general, block 1720 is carried out by a container object.

In block 1722, the event handling system determines how to route the event message. For example, in block 1724, the event handling system determines whether the event is synchronous or asynchronous. If the event is synchronous, then it is sent to a message broker 1734 that is defined by the event broker framework 1704.

Message broker 1734 determines whether the event is persistent, as shown by block 1736. If so, then control passes to block 1738 in which the event is stored in the database or otherwise made persistent. Thereafter, and if the event is determined as not persistent at block 1736, control is passed to block 1740, in which one or more coarse-grain filters are located. The filters are selected based on the message type, and applied to the event message.

If the event message matches one of the coarse-grain filters, then in block 1742, one or more rules with fine-grain filters are retrieved. Rule constants are extracted from the rules in block 1746. In block 1748, the fine-grain filters are applied to the event message. If a match occurs, then in block 1744, the associated action is performed.

Referring again to block 1724, if the event message is asynchronous, then control passes to block 1726 in which the event message is dispatched using a transport mechanism. The event message is sent over a durable or non-durable topic, as appropriate, as shown by block 1728, 1730, 1732. Thereafter, the event message is processed at the message broker, as shown by block 1734, in the manner described above. Concurrently, a notification message is received at block 1750, and in response a notification is sent to the end user, at block 1752.

**1.8 Object Communications—External Systems**

### 1.8.1 Enterprise Application Integration Using Connectors

According to one embodiment, mechanisms for enterprise application integration, using connectors, are provided to enable the system to connect to existing (“legacy”) applications of an enterprise that uses the system. The mechanism for connectivity may use a synchronous or an asynchronous approach. In a synchronous approach the client makes a request and waits for a response before it proceeds. Synchronous approaches can use HTTP, HTTPS, RMI, CORBA. Asynchronous approaches do not have this limitation and typically use asynchronous messaging implementations.

FIG. 18A is a block diagram of a first enterprise application integration approach that uses an asynchronous approach.

Application server 202 and other servers 1802 that comprise the transportable application system as described herein are communicatively coupled using JMS 214 to an adapter 1804. The adapter 1804 is communicatively coupled to an existing asynchronous Enterprise Application Integration (EAI) bus 1806. Commercially available examples of EAI bus 1806 are produced by Vitria, TIBCO, IBM, WebMethods/Active, etc. The bus 1806 is communicatively coupled through one or more connectors 1808A, 1808B, 1808C to corresponding legacy applications in the form of an enterprise application 1810, Web server 1812, mainframe 1814, etc. In this example, adapter 1804, EAI bus 1806, and connectors 1808A, 1808B, 1808C are compatible and generally are provided by one of the foregoing vendors.

FIG. 18B is a block diagram of the system of FIG. 18A wherein a custom connector is used. The custom connector 1818 is substituted for JMS 214 and adapter 1804. In this configuration, an asynchronous solution is provided and use of JMS is not required. As a result, a particular JMS implementation is not required. The custom connector 1818 may be created and implemented, for example, using a software development kit (SDK) from the party that supplies the EAI bus 1806.

FIG. 18C is a block diagram of an application-server centric integration approach for providing a synchronous integration solution. In this approach, servers 202, 1802 are communicatively coupled through one or more Java 2 Enterprise Edition (J2EE) connectors 1820A, 1820B, 1820C to corresponding applications 1810, 1812, 1814. J2EE connectors, as defined by Sun Microsystems, provide a standard architecture for connecting Java 2 systems and applications to legacy information systems. Application 1810 may be Siebel, SAP, PeopleSoft, etc., or any other external application.

Alternatively, in FIG. 18C a Java Connector Architecture (JCA) construct may be used as connectors 1820A, 1820B, 1820C. There may be multiple instances of a connector for each external application.

FIG. 18D is a block diagram of an enterprise application integration approach that provides synchronous integration through one or more synchronous protocols. The servers 202, 1802 are communicatively coupled to the legacy systems through the synchronous protocols. Examples of such synchronous protocols include HTTP, RMI, CORBA, SOAP, etc. In the case of CORBA, a bridge 1822 may be used to convert CORBA messages and objects to Common Object Model (COM) format.

FIG. 18E is a block diagram of an enterprise application integration approach that uses event-based communication. Application server communicates through event daemon 216 to event service 146. Within or in association with event service 146, incoming event messages are passed to a Java object to XML converter 1832, yielding an XML representation of the information in the daemons. The SML information is transformed using engine 1836, with input from an XSL stylesheet 1838, to yield transportable XML information. The transportable XML information is passed to transport adapter 1840, which outputs the XML information using one or more synchronous protocols 1842. The synchronous protocols communicate with the legacy systems as in FIG. 18D.

In operation, in one specific embodiment, as illustrated in the top half of FIG. 18E, the application server 202 uses JMS point-to-point mode to generate events and communicate them to event daemon 216. When the event daemon 216 processes an event, event service 146 instantiates a Java object and uses converter 1832 to transform the Java object into an XML string. It then uses this XML, transformation engine 1836, and an XSL stylesheet 1838 associated with the Java object to output the expected XML schema for a receiving partner system. The method of transport for the XML can be HTTP, IIOP, or SMTP.

Referring now to the bottom half of FIG. 18E, the reverse occurs when an XML message arrives from a partner system. For example assume that the transport mechanism is HTTP. A servlet is invoked and uses transformation engine 1836 and the corresponding XSL 1838 for that XML message to convert it to an XML representation that is expected by application server 202. An XML to Java object converter 1842 is then used to instantiate a corresponding Java object for the event daemon 216. The object is encapsulated as an event message. An event handler for that object is then invoked from

event handling system 1830 when the event daemon processes the event. Each XML message must have its own object representation, XSL, and event handler.

This arrangement has the advantage that XML messaging is becoming the method of choice for inter-operability between business-to-business systems that exchange data. Application-server vendors are coming out with their own XML-based messaging systems for such exchanges, e.g., WebLogic's Collaborate.

In the approaches of FIG. 18B, FIG. 18C, the disclosed connectors generally act as gateways for external applications to create, retrieve, update and delete application business objects of the system through an object interface mechanism. The connectors also receive notifications for changes to such objects through the event management system and notification system, or by polling. The connectors also enable application server 202 and other components of the system to retrieve and update data from external applications. In one specific embodiment, the connectors enable objects associated with building blocks to retrieve and update objects that are hosted in external applications or systems.

FIG. 18F is a block diagram of providing another embodiment of an enterprise application integration approach. One or more enterprise applications 1810A, 1810B, 1810C are communicatively coupled to JMS queues 214 either directly, as in the case of application 1810A, or indirectly through an EAI bus 1806, as in the case of applications 1810B, 1810C. The direct connected application 1810A has an adaptor 1840B that can queue objects to the JMS queues 214, for receipt by a corresponding proxy adaptor 1840A of a connector framework 1854. EAI bus 1806 has a similar corresponding adapter 1841B and proxy adaptor 1841A.

Adaptor 1841B and proxy adaptor 1841A may be configured to operate with any desired EAI bus 1806 or similar product, e.g., webMethods, Vitria, SeeBeyond, etc.

Connector framework 1854 communicates through an API 1852 to connector building blocks 1850, which may be included in a transportable application to give that application the ability to communicate with enterprise applications. The API 1852 may provide create, read, update and delete functions for business objects and transportable application objects. Such operations are subjected to access controls as described herein. In one embodiment, notifications and event rules can be set on connector building blocks 1850 to enable taking actions or creating other transportable applications when the connector building blocks change or generate events. A generic connector building block provides an XSL translation function equivalent to XSL transformation engine 1836 of FIG. 18E, and can display connector data. One or more Extensible Style Documents

(XSDs) describe the business objects of the enterprise applications in a manner equivalent to XSL 1838.

Thus, the integration framework as described herein provides both direct integration and integration through an existing EAI bus. As a result, adapters can be constructed without platform changes. Both outbound and inbound operations are supported. The framework provides the ability to programmatically create transportable applications and add pages from enterprise systems. Business objects are described in XSD's rather than in source code. Adapter configuration information is described in XML. Asynchronous messages, through the JMS queues, are used for communication with enterprise applications or an EAI bus.

A building block can synchronously query an enterprise application adapter for all business objects matching given criteria. The building block can synchronously request data for a business object from the external system if the data is not found in the cache. Further, an enterprise application can request the system to create a transportable application using external data. The enterprise application can send a notification that a business object has changed, causing the system to update the cache.

An advantage of using this approach is that the building blocks in general will not need to store such objects persistently since the building blocks may retrieve a copy of the external data, at any time, through the connectors, with less impact on performance than if persistent storage is used.

Connectors as disclosed herein may conform to any appropriate communication mechanism for external business objects. For example, the protocols proposed by BizTalk.org, RosettaNet, EBXML, etc., may be used.

In one specific implementation, connectors are implemented in one or more programmatic classes that conform to the following API description:

Connector class. Building blocks can retrieve any external business object as an instance of a connector business object class. The connector business object class provides methods to produce an XML representation of the object, modify the object, etc. Building blocks can use XSL stylesheets to present the business objects through generic HTML presenters or use custom presenters. The business objects may be implemented as cached data access objects.

Subscribe method. Registers a subscription for a business object for later use. Receives, as parameters, a name of a business object, and one or more name/value pairs that identify an instance of the business object. Returns a key to identify the subscribed

object; the key, which may be persistently stored, is passed to all other methods of the connector class.

**GetBizObject** method. Retrieves a business object. Receives, as parameters, a name of a business object and a key value. Returns the requested instance of the object.

**unsubscribe** method. Drops a subscription to a business object. Receives a name of the business object and the key value.

A Connector Business Object class provides a base class for all business objects, and defines a **getXMLString** method, **setXMLField** method, **update** method, and **registerNotification** method. The **getXMLString** method returns an XML string method of a named business object. The **setXMLField** method sets the XML field in a business object based on a field name and a value for the field. The **update** method stores all changes made to the business object through the **setXMLField** method.

The **registerNotification** method registers a rule with the event handling system. It receives, as parameters, an array of fine-grain filters that comprise Xpath expressions, and an action to invoke when the filters are satisfied. The rule registered with the event handling system is created using the specified list of fine-grain filters and the type of event message that is generated by the connector system when the business object changes.

### **1.9 Security Processes; Access Control**

In one embodiment, a security framework is provided having a plurality of security services and interface definitions. The security framework enables an end user to configure and define security features to use when authenticating users and authorizing them to access data. Thus, in this context, security and access control refer both to authenticating users for access to the system as a whole, as well as verifying that a particular authenticated user is authorized to retrieve or modify specific data in the database.

The security framework also enables one user to develop transportable applications with another party and have some of the data to be shared amongst the users associated with that party. The security framework comprises a plurality of interfaces, each of which providing a contractual set of features and responsibilities to the consumers of the interface.

In one embodiment, the security framework is implemented using access control service 136 and security service 120. Access control processes applicable to the embodiments described herein are described in co-pending application Ser. No.

09/861,008, filed May 17, 2001, the entire contents of which are hereby incorporated by reference as if fully disclosed herein.

A GateKeeper interface provides a data consumer with the ability to retrieve and configure information that defines relationships among security domains. This interface provides information about the hosting domain and other domains that have a trusted relationship with this domain. Also, specific users can be managed through this interface so that only specific individuals within an organization have access to data within a hosting organization.

A PortalGate interface provides a consumer with the ability to authenticate a user using a username and password, SSL, PKI, etc. Further, the interface provides the consumer with the ability to query whether a user is still valid, for example, by checking to see if a user is still valid and has not been revoked. The interface also enables another program element to query whether a user has access to a specific data object or object instance.

An Access Control (AssetGuardian) management interface defines one or more contracts between a consumer and an entitlements database. An entitlements database stores information about users and what they have access to. In one embodiment, the entitlements database is maintained separate from database 208 of FIG. 2A, to improve security of the entitlements database.

A Security Provider interface provides components to control end-to-end security. In one embodiment, a PKI enabler interface and an authorization interface each has an implementation that can be defined by configuration where each implementation represents a way to access a PKI or an authorization scheme. Each of these implementations can be loaded simultaneously so that one or more schemes can be used at the same time.

Programmatic classes within the security framework are configured to provide security against intrusion. For example, the classes are typed as final to prevent a hacker from providing implementations to an abstract class or extend and override a non-final class with dangerous or risky code. Therefore, the security framework has its own interfaces and extends other trusted interfaces in packages that are trusted, e.g., the java.security package available from Sun Microsystems.

The data that the framework manages is composed of hierarchies of assets. Assets are defined as objects that exist in a department or an enterprise that need to be protected. For example, assets include transportable applications, pages, building blocks, and objects that encapsulate field data values for any of the foregoing. Each asset in a hierarchy can

have permission assigned to it on behalf of a user. The mechanism by which an asset has a user and permission composed for it is termed a security label. Since each asset in a hierarchy can have its own label and levels of access can be applied across a hierarchy, the labels are termed multi-level. Therefore, the security framework is a multi-level security label system.

Contracts within the Security Framework may be defined using the Interface Definition Language (IDL). IDL enables a framework to expose its interfaces and contract data as well as error handling capabilities. In one embodiment, types of IDL syntax that are used in the framework include IDL Exceptions, Structs, and Interfaces. IDL Exceptions are defined so that generated Java source, or any other language that has an IDL binding, will have error handling capabilities defined at the package level. IDL structures are compiled into Java objects, which are typed as final. The security framework composes the contract objects in its model package. The framework model package defines the objects that are used in communications or invocation of interfaces' methods. IDL Interfaces contain the methods that can be invoked as well as the error-handling signature, which completely defines the contract of the interface.

One important benefit to using IDL is that most application and transaction servers use IDL as a way to initially introduce interfaces and implementations into the container. Another benefit is that the IDL to Java conversion process produces client and server side stubs and skeletons so that an end to end implementation is more easily created.

The Gatekeeper Interface contract states that of the security configuration for a given security domain, which is determined at the organizational level, all parameters that allow security integration across multiple domains can be retrieved. The interface, for example, supports the retrieval of the CrossDomainList, which is a list of X.500 distinguished names ("DNs") representing external organizations that have a trusted relationship with this domain. Additionally, CrossDomainDN is a list interface list of all the users (by the DN) who have trusted access to this domain. If this list is null then normal authentication mechanisms are used to determine if a user has access to this domain. If it is not then the intention of this managed list is to provide the users who can access this domain. When the user DN is determined then it can be cross-referenced to the list. If the user DN is not present then the user's authentication must fail whether they can authenticate properly or not. A commercial example of an authentication system that may be used is WebLogic.



An important contract of this interface is the management of the domain and whether it is a secure domain or not. The `getMode()` method informs the consumer of the method if the domain represented by the interface is secure or not. In fact the value returned is not a Boolean but a string, which contains definite values of “Secure”, “Not Secure”, and options text for any granularity in between. In this way a security administrator can define as many security levels as they require.

The PortalGate Interface provides the system with a trusted path. When a user authenticates to the security framework, by one of a variety of mechanisms, a session is established specifically for that user. Because IDL has been used to define the contract of the interface the session trusted path can be managed in an ORB, Application server, or transaction server container. Invoking a method in this interface checks operation that are attempted after the user authenticates.

In one embodiment, four types of authentication mechanisms are provided. PKI verification provides the framework with the ability to participate in a single sign on arrangement with a PKI environment. To invoke such verification, an object or method passes the name of the user, a digitally signed version of their name, and the symbol `PKI_VERIFICATION` as parameter values. SSL-only verification is like weak verification, discussed below, in that the user name and password are passed as parameters. It simply informs the framework that an SSL connection is being used to send information to and from the interface. Certificate-based SSL verification is like PKI verification in that the name of the user is passed in one parameter and their SSL certificate is passed in another parameter. The certificate is then validated with the CA of the SSL certificate (either Verisign, Cylink, or Entrust). Weak verification passes a name and password as parameters and provides relatively low security.

Each data parameter is provided as a mutable type, e.g., a byte array, so that the data within it can be deleted once it is used. A `checkVerification()` method returns true if the user is still authenticated and has not been revoked from the environment. A first `checkGuard()` method checks to see if the user has access to a specific asset (either at the type level or the instance level) given a specific permission. A second `checkGuard()` is the same as the first except that the variable parameter allows the framework to accept extra data to further scrutinize the access check. For example, an application component may want to verify that a user not only has read permission to sales data but that they only have access to the Northeast sales region and not any other. In this case the application component can pass a value that is effectively a SQL where clause or an XML document

which describes the SQL where clause. A getName() method returns the authenticated users DN. This can be used for further checking or for personalization purposes.

For purposes of facilitating use of the AssetGuardian interface, all assets within the framework are contained within an organizational hierarchy. The framework composes and manages organization objects as X.500 organization objects. Within the organizational definitions there are users, roles, permissions, security labels, and security preferences. The entire framework also has an audit trail, which is not bound, at an organizational level.

Based upon organizational hierarchy roles, permissions, resultant security labels, preferences, and assets exist at nodes within the tree. Assets themselves are hierarchical structures in that they can represent complex types (such as containers, databases, database tables, etc.). Each asset can have its own security label and each label can be assigned a level. The security framework provides a LabelComparator interface and implementation that provides for the interpretation of the level of a security label as it is applied to one or more assets.

The security provider interface allows the framework to dynamically load an implementation that supports a Public Key Infrastructure vendor. PKI vendors support encryption, decryption, digital signature, and signature verification. They also provide key management, certificate issuance and management, as well as user authentication for single sign on.

The interface supports S/MIME and non-S/MIME security operations as well as the management of security recipients. Recipients in a security context are those persons who have a public X.509 encryption certificate and can have data of any sort encrypted specifically for them. Operational the interface and its implementation manage a stack of recipients, which is pushed before an operation occurs. When recipients are defined they can have data encrypted for them. In the case of a signing operation the user who has a connection to the PKI and managed through the connect method in this interface has their private signing certificate used.

To apply access controls to transportable applications, in one specific embodiment, the following processes are used. A transportable application is created as otherwise described in this document. The transportable application is defined as an asset having an asset identifier that is obtained by calling a method of the java.security package. One or more access control definitions (or "labels") are created by the author using the "makeAccessControlLabelModel()" method of the java.security package. Each access control definition identifies read, write, and update permissions. Each recipient has

his or her own access control definition that defines one or more limited permissions. Thus, the intersection of the transportable application access control definition and the recipient access control definition indicates whether a particular recipient can access an application.

At the time a recipient attempts to open or read a transportable application that is secured, the recipient is first prompted to log in to the system. The access control labels are checked to identify the recipient's individualized permissions. In one embodiment, a checkAsset method of the Asset Guardian interface is used. Access is denied when permission is not allowed.

Access controls specifically applicable to database access are now described. In general, in one embodiment, access to database 208 is restricted and is based on the role-based permissions provided by the security framework for different object types. A Java class encapsulates information needed to carry out an access control request or verification, including session identifier, user name, action type, and object type. This information is used when calling an authentication API of the security framework.

Further, classes and methods responsible for access to containers and folders, database queries or row selections, inserts, updates, and deletions are configured to carry out access control verification on the objects that are the subject of such operations, before carrying out such operations. Carrying out access control verification refers to calling a method of the security framework that can determine whether a particular user is authorized to access a particular named object or asset. Each such class and method is provided with methods that can check for access authorization and generate exceptions if access is denied.

In conjunction with access control each asset can have encryption and digital signature attributes applied so that transactions based against the asset can be encrypted and or digitally signed. For example, when a user is interacting with a transportable application, each time that a client 120 generates a network request that includes data for a field of a building block, the client can digitally sign the request. Upon receiving the request, application server 202 can verify the signature before the request is processed. In one embodiment, each HTTP request that is generated by a client and that includes field data relating to a transportable application, page or building block is digitally signed. Each HTTP request that is received at an application server 202 is checked to determine if the request contains a digital signature. If so, the digital signature is extracted from the request and verified. If verification is successful, the request is redirected to a service

routine, i.e., processed normally by the application server. In one alternative, information collected in the extraction process may be logged or stored in an audit trail.

Extracting digital signatures from an HTTP request stream may be implemented using software systems that are commercially available from PrivateWire. Verifying digital signatures that have been extracted may be implemented using software systems that are commercially available from Entrust, Inc.

Access controls may be modified as a result of events that are processed using the event handling system described herein. For example, an action associated with an event may be to modify an access control of a transportable application to become either broader or stricter in some way, or to enable a new recipient to have access to the transportable application.

In one embodiment, instance-scoped role-based access control is provided for transportable application. Such control is "instance-scoped" because access controls are determined and can be defined uniquely for each user for each instance of a transportable application. Such a mechanism provides much more detailed access control, as compared to class-scoped access control using J2EE mechanisms, which provides only method-level checking per user per transportable application class. In one embodiment, when a transportable application is authored, or after the transportable application becomes active, an author can add, modify or delete users from access controls specified for the transportable application and pages within the transportable application.

Access to JSPs and servlets in the system is controlled through membership of users in roles. Roles may be "page-scoped," that is, defined at the page level within transportable application. Thus, access to instances of assets such as building blocks and associated rule descriptors are determined based on the role that a user is assigned to for the page instance on which the building block and rule descriptor instances are created.

In another embodiment, directory auto-registration is provided. When the system has been configured with knowledge of the existence and location of a directory server, a user may log in to the system using a user name, password or other credentials that are stored in the directory server. After locating such credentials in the directory server at the time of the user's first login, the system automatically registers those credentials in database 208. Thereafter, the user can log in to the system without reference to the directory server. In another feature, bulk user registration may be carried out by an administrator, by loading a formatted file that contains the user information. Self-registration is also facilitated.

### **1.9.1 Cluster-Specific Encryption and Request Routing**

In one embodiment, all message identifiers that are sent from the system to a client are encrypted. In another embodiment, the encryption process associates each message identifier with a processor cluster or database cluster that is responsible for processing the message. Using such a process, a message identifier in a URL that is meant for one cluster cannot be processed by another cluster. This is beneficial in the event that a malicious user redirects the URL toward another cluster, e.g., by changing the URL to point to the new cluster and keeping the same arguments, and the URL and the message it carries are decrypted at the destination. In one encryption approach, a database identifier or cluster identifier is embedded in the message in order to provide more security.

In one past approach, the format for a message ID is **<prefix>\_<encrypted message id>**, where “prefix” is a number that determines the seed for the TwoFish encryption algorithm. In one sub-approach, the seed value may be hard-coded in program source code for the functions that carry out the encryption process. A disadvantage of this approach is that such code is installed on all client installations, so that all clients use the same seed, or each new client installation will need a new software release. Moreover, a message destined for one client could be redirected to another client’s cluster, and because the system decrypts all messages that are received by the cluster a security vulnerability exists. In this context, a “client installation” refers to a particular instance of system 200 that is licensed to or used by an enterprise, organization or similar entity.

Therefore, in another approach, different seeds are used for encryption for different client installations, and each incoming message is checked to see if it is meant for the cluster. If not, it is discarded. Each client is assigned a database ID, which is unique. A global identifier replaces the message ID.

Further, in an improved approach, each client uses a different seed that is determined by the database ID of the cluster and is derived from a base seed by addition of the database ID to the base seed. Since the seed consists of 16 bytes, a long time interval must elapse before any two clients can get the same seed.

The prefix in the message contains the database ID. The database ID is also present in the encrypted message ID in encrypted form. Thus, if a malicious user attempts to change the database ID to another cluster in the hope that it would be a valid URL, it is most unlikely that the decrypted message will resolve to a valid global identifier. Even if it does, the database ID component is highly unlikely to match the prefix, and therefore the system would discard the message if the two do not match.

In this approach, all incoming messages pass through a sanity check mechanism wherein the system initially compares the prefix of the message with the database ID of the originating client installation. If they do not match the message is discarded. If they match, the message identifier is decrypted. The message identifier is a global identifier, and since the global identifier contains the database identifier, the database identifier is compared to the prefix and if they do not match, the message is discarded.

In another feature of this approach, to accommodate changes in the encryption algorithm or methodology itself, a version value is associated with each specific encryption methodology. Each client may use its own encryption algorithm, and the encryption version value is part of the prefix. In one embodiment, the format for the new prefix is **EV:DBId**, where "EV" designates the encryption version and "DBID" is the database identifier value. The prefix may be transformed, e.g., by bit shifting, so that the component values EV and DBID are not easily visible.

### **1.9.2 Sharing Transportable Application Data Among Multiple Sites**

In one embodiment, the system described herein is a distributed system in which multiple particular installations of the system can share transportable applications and associated data. For example, different companies could each set up the system and collaborate by sharing transportable applications and associated data.

In general, transportable application data is replicated at each of the participating sites or installations, enabling the user to receive a consolidated view of interaction with all sites. A user logs in to a portal home page and receives a view of all transportable applications directed to that user, from any originating system. Each transportable application is authored and updated only at one site. All portal operations are performed on the home site of the user. Portal operations include viewing group lists, group archives, and folders; and performing administrative functions such as assigning transportable applications to folders. Operations other than portal operations are performed on the home site of the respective object. Users can author transportable applications at any site.

Trust relationships are established among sites that participate in replication, using elements of the security framework described herein. Each user is designated to have a home site and all users are denoted as local or remote for a particular site. Data for a particular user is always replicated at the home site, enabling the user to obtain a consolidated, global view of all activity at the home site at all times. Accordingly, a user may log in once to a home site and need not log in multiple times to different sites or clusters.

In operation, a user logs in to the system. The user is transparently redirected to an application server 202 at a site that has home information about the user. At the home site, the user is presented with a personalized portal view appropriate for the user. The Personal Messages page and Group Archive pages present a list of all transportable applications involving the user, including those that are remotely located.

When a link of a particular transportable application is selected, the user is re-directed to the appropriate site, and information from that site is displayed in a new window. Authentication to the new site is carried out by passing a digital certificate with the user's security credentials to the new site, using the security framework, so that multiple logins are not required. If the selected transportable application is remotely located, the user can view and respond to it. The remote site recognizes that the user is a remote user and configures links for buttons in the user interface to reference the user home site. For example, the New Message link identifies a URL in the home site rather than the remote site.

A user may also author a new transportable application based on a template that is owned by a remotely located group. When such a template is selected, a new window is opened from the remote site and the user is re-directed to the remote home location of the group that owns the template. The new transportable application is authored at the remote site, i.e., its data is stored at a database of the remote site.

All folders of the user and all administrative tasks relating to the folders are carried out at the user's home site server.

Redirection, for the foregoing processes, is achieved by determining a URL of the remote site to which redirection is occurring. Accordingly, all sites that are cooperating as described above provide access to one another through a private TCP port in their firewalls, or through a security mechanism of the security framework.

The replicated data includes certain metadata about all transportable applications and groups that relate to the user. For example, for transportable applications that involve a user and are remotely stored, a local or home site receives metadata including subject, status, author name, updated timestamps, timestamp for the last time the user read the transportable application, message attributes, etc., Such information is retrieved from rows in a transportable application table of a database of the remote site. Metadata about groups is also replicated, including group hierarchy and membership data, group folders and content data, etc.

In one embodiment, objects that are replicated across sites, or across clusters of servers at a co-located location that implement different sites, implement a Replicable

interface. A replication manager receives a Replicable interface as an argument and transports it to one or more sites that need the object. The Replicable interface includes an export data method, apply data method, and methods to retrieve header information for messages to be sent. The export data method outputs an array of replication data objects, each corresponding to a single destination site. Each replication data object serves as a container for the transport of objects across sites and encapsulates state and type information.

The export data method is used at the home or master site of a replication event to export a version of the state of an object for reconstruction on a remote site. The apply data method is used by an empty Replicable object to import a state from the input object. This allows a Replicable object to be reconstructed on a remote site of a replication transaction with the state tailored to that remote site. A replication status object may be used to provide acknowledgment of replication messages. Replication is carried out as data is updated. Sites communicate using XML messages that are sent over HTTPS.

### **1.10 System Administration Processes**

System administration processes may include a reporting function that presents information that analyzes interaction of recipients with transportable applications. Such statistical data may be retrieved by or available to a server administrator, authors of transportable applications, etc.

### **1.11 Using Transportable Applications in Business Processes and Workflow**

In an embodiment, one or more transportable applications may be used to carry out complex business processes and workflows. In one specific embodiment, a process template designer mechanism enables a business process expert to create and publish templates of a process framework. A template for a process framework can include a general process description, process-specific fields (e.g., process status), templates of component tasks for transportable applications, a Process Style comprising an association of fonts, colors, images, layout, text that are applied to all building blocks and tasks of transportable applications that make up the process; required, preordered starter-tasks, and process-specific properties (e.g., notifications, access controls). The process template serves as a guide for an author's later design of a specific instance of a process. On-the-fly editing of templates and authoring of a process transportable application can also be initiated from within the template designer mechanism.



In one approach for implementation of the foregoing, a process designer uses a Process Composer software tool to create a process template. Thereafter, the process template may be retrieved and used during the authoring process to construct a process transportable application and send it out to a list of recipients.

The process composer enables a user to identify, select, and include tasks in a template for a process that has one or more transportable applications. Some such tasks may be designated as starter tasks. A starter task is included when an author begins to compose a process transportable application. A starter task may or may not be deemed required by the process template creator. A template also may include one or more "addable" tasks. An addable task is one that can be added to the process transportable application after it has been sent to recipients.

In one embodiment, a process template comprises a Java class that stores information about a process. The process template class determines the types of tasks that can be entered after a transportable application is sent, and also provides appropriate process creation information at the time that an author creates a transportable application based on the template.

The template class stores information identifying the types of tasks that are allowed and which tasks are starter tasks. For starter tasks, the template class stores whether or not each task is a required task. Additionally, some meta information about the overall process is stored, e.g., a process name, process description, and process style. In one specific embodiment, a process template class has the following member variables and corresponding accessor methods:

```

private String name;
private ObjectID authorID;
private String description;
private Vector styleIDs;
private boolean updateStyle;
private ObjectID templateID;
private Vector introFields;    // vector of IntroField
objects
private Vector requiredTasks; // vector of ProcessTask
objects
private Vector allowedTasks;  // vector of ProcessTask
objects

```

The process template class may also have the following methods:

```
public void ProcessTemplate()
public void store (DBTrans transaction)
public void remove (DBTrans transaction)
public static void remove (DBTrans transaction, ObjectID
templateID)
public static ProcessTemplate getTemplate(ObjectID
templateID)
```

The ProcessTemplate class may have an empty constructor such that a Process Servlet that creates the ProcessTemplate class is responsible to set the fields appropriately. For example, the servlet parses the HTTP Request and sets the values on the ProcessTemplate object.

The store() method takes in a transaction value, which can be null, and writes the object to the database. The method calls another method in a helper class to transform the member variables into XML format. The remove() method deletes the ProcessTemplate from the database. The transaction value can be null, in which case it constructs its own transaction and executes it. The static method removes the given template from the database. The getTemplate() method retrieves the ProcessTemplate with the given template identifier and passes it back to the calling method. It uses a helper class to construct the ProcessTemplate object from the XML information in the database.

An IntroField class represents, at an abstract level, any additional introduction field information the author wishes to provide, and the default value for this field and what kind of graphical user interface widget is used to display the field, e.g., drop-down, radio button, text area. It has the following member variables:

```
private String name;
private String defaultValue;
private String inputType; //html type of input e.g.
text, textarea, etc.; use constants
private Vector sizeInfo; // vector of sizes we want for
the html components like width
private String options = ""; //additional options
```

```
private Vector choices; // String Vector of choices for
drop down
private int position; //position on authoring page
```

A ProcessTask class represents a task from the process point of view and includes a task ID value and information about the task, e.g., whether it is a starter task or an allowed task. The ProcessTask object contains the following member variables and appropriate accessor methods:

```
private ObjectID taskID;
private int position;
private String name; //
private boolean isStarter = false;
private boolean isRequired = false;
```

A ProcessXMLUtils helper class aids in the writing and retrieval of the process XML to and from the database. It will implement an interface so that it can be passed into the XML parser to provide the appropriate call back routines. Further, the helper class includes the methods getXmlString and ProcessTemplate:

```
public static String getXmlString (ProcessTemplate)
public static ProcessTemplate getProcessTemplate(String
xmlString)
```

The getXmlString method receives a ProcessTemplate object and returns an XML representation of it, including all its member variables. The ProcessTemplate method receives a string of XML text and returns a ProcessTemplate object.

In one specific embodiment, each process template is stored in database 208 in a Process Template table that contains the following columns:

Name	Null?	Type
TemplateID	Not Null	Number (19)
AuthorID	Not Null	Number(19)
Position	Not Null	Number (3)
XML	Not Null	VARCHAR2(1024)
DateCreated	Not Null	Date
DateModified	Not Null	Date

Database 208 also comprises a table that associates styles and tasks with the template so that the system can update them if any changes occur in the task template or the style. This table can be used by any object that requires a parent to child relationship, and needs to update an object that is using it. In one specific embodiment, the style association table comprises the following columns:

Name	Null?	Type
ParentID	Not Null	Number (19)
ParentType	Not Null	Number(4)
ChildID	Not Null	Number (19)
ChildType	Not Null	Number(4)
Update	Not Null	Char(1)

Creating a Process Template involves selecting one or more tasks that can be used, certain attribute values for the tasks, and related process information that applies to all processes, such as the name and description of the process. In one specific embodiment, a servlet ("ProcessTemplateServlet") is used to save a process template to database 208. The Process Template Servlet has action parameters such as save, open, add, etc. Each action type will have a corresponding method to perform the appropriate action. A form submitted to the servlet includes parameter values that can be used to construct a process template object. In general, the servlet extracts parameters from the HTTP request that submits the form to the servlet, and constructs a process template object from them that can be written to the database 208.

The servlet then forwards a template creation request to a process template designer server page. In an embodiment, a Java Server Page (JSP) is used. The process template designer JSP iterates over the parameters from the form and displays them correctly on the page. The JSP also implements the submissions to the ProcessTemplateServlet to move up, move down, delete, insert tasks, and carry out any other defined operations.

The introductory fields may include various types of inputs and ordering. The XML syntax can be extended to incorporate other information, e.g. specifying a maximum number of choices for checkboxes.

Process templates may include rules that control, for example, issuance of notifications in responses to changes to tasks in the process. In one approach, the rules are represented as a vector of objects that are stored in association with the ProcessTemplate object. Additionally, task level rules can be added to a task using the process composer,

which stores the task-level rules as a vector of objects stored in association with a ProcessTask object.

## 2.0 MULTIPLE-PART ELECTRONIC MESSAGES

In general, in one aspect, the present invention is directed to multiple-part electronic messages. Each multiple-part electronic message comprises a plurality of parts that are associated as part of a single message. In one embodiment, a multiple-part electronic message is distributed, viewed and updated as part of a group collaboration application system. In this embodiment, one or more multiple-part electronic messages may be configured to implement business processes such as information review, planning, forecasting, etc. In another embodiment, multiple-part electronic messages are transported as part of a client-server electronic communication system. For example, the multiple-part messages may be created using an e-mail client and communicated using an e-mail transport server or related infrastructure. Thus, embodiments do not require proprietary equipment or special modifications for transport within an existing or "legacy" communication system.

Embodiments are not limited to e-mail as a communication media. The multiple-part messages may be communicated using other data communication mechanisms such as HTTP. Embodiments also are not limited to display at conventional e-mail clients. Multiple-part messages may be displayed using a personal digital assistant, wireless communication device, Internet appliance, etc.

FIG. 21A is a diagram of a first embodiment of a graphical user interface display of a multiple-page electronic message.

According to this embodiment, a user interface display window 2100 is generated by an e-mail processing application. The user interface display window 2100 includes a toolbar 2101, a message header display pane 2102, and a message body display pane 2104. Toolbar 2101 displays one or more buttons, links or other user interface widgets for selecting commands, options or tools of the e-mail processing application. Conventional commands such as File, Edit, View, Insert, Format, Tools, Actions, and Help may be provided for carrying out operations with respect to a message that is displayed in window 2100.

Toolbar 2101 also may include one or more buttons, links, or other user interface widgets for taking messaging actions with respect to the displayed message. For example, Toolbar 2101 may provide Reply, Reply To All, and Forward options. Selecting the Reply option instructs the e-mail processing application to generate a new message in

reply to the currently displayed message, directed only to the sender of the message. Selecting the Reply To All option instructs the e-mail processing application to generate a new message in reply to the currently displayed message, directed to the sender of the message and all recipients of the message. Selecting the Forward option instructs the e-mail processing application to generate a new message that is directed to a new recipient and that includes a copy of the currently displayed message. In other embodiments described herein, new messages are not generated, and reply content or forwarded content is consolidated in the original message.

The message header display pane 2102 displays message header information. For example, the message header display pane 2102 may display the name of the sender of the message, the names of recipients, the subject of the message, the date that the message was sent, etc.

Message body display pane 2104 comprises one or more message pages 2106, 2112, 2114, 2116, etc. In FIG. 21A, for purposes of illustrating a simple example, four message pages 2106, 2112, 2114, 2116 are shown. However, in other embodiments, a message may comprise any number of message pages and message body display pane 2104 may include any number of message pages.

Each message page comprises a page navigation region and a page body. For example, a first message page 2106 comprises a page navigation region 2108 and a page body 2110.

In one embodiment, each page navigation region is graphically displayed such that the page navigation region is a contiguous part and integral with its associated page body. Further, each page navigation region is graphically displayed such that every page navigation region is selectable, using a graphical cursor that is movable using pointing device such as a mouse, whenever any particular page body is displayed.

For example, in the embodiment of FIG. 21A, message pages 2106, 2112, 2114, 2116 are displayed as simulated overlays wherein the first message page 2106 appears to be on top of a stack of message pages. Although message page 2106 is the top message page, the page navigation regions of all other message pages 2106, 2112, 2114, 2116 are visible adjacent to message page 2106 and are selectable at any time during which message page 2106 is displayed.

When a particular page navigation region is selected, the message page associated with the selected page navigation region becomes fully visible. For example, if the Bookings Pie Chart message page 2114 is selected, it becomes fully visible and appears to be the top page in a stack of pages. The message body 2110 of message page 2106

becomes hidden, although its page navigation region 2108 remains visible. Processing operations to carry out such functions may be executed by a server that is communicatively coupled with a client that is displaying the message window 2100, according to processes that are described further herein.

Each page navigation region may carry a label that identifies the contents of the message page that is associated with the page navigation region. For example, in FIG. 21A, the message displayed in window 2100 generally relates to a First Quarter Financial Summary. A first message page 2106 presents Profit & Loss information, as indicated by a "Profit & Loss" label in the page navigation region 2108 of the first message page. A second message page is labeled "Balance Sheet," and other message pages may have any other desired labels. Such labels may comprise text, numeric values, graphic images, icons, hyperlinks, or any other indicator element or other information.

The page navigation regions may be color-coded, for example, according to a topical key, an order of priority, and industry-standard color arrangement, etc.

For purposes of illustrating an example, in FIG. 21A, the page navigation regions are shown as arranged along a top edge of a message page. However, the page navigation regions may be arranged along a bottom edge, left edge, right edge, or other side edge.

If a message has a plurality of message pages, and message display window 2100 has insufficient space to display all the message navigation regions associated with the message pages in a row, then the message body pane 2104 may comprise an indicator that additional message pages and navigation regions are available for display and selection. For example, message body pane 2104 can display an arrow, dot, or other icon adjacent to the right-most message navigation region. Selection of the arrow, dot, or other icon causes a server or other element to generate message body pane 2104 such that one or more of the other message navigation regions are displayed, and such that previously visible message navigation regions scroll or slide to the side, up or down to make room for the newly displayed message navigation regions.

Using this configuration, an e-mail message is displayed in a structured fashion. Its content may be organized so that one set of related information is collected in a particular page, and another set of related information is collected in another page. Specific information is accessible simply by selecting a navigation region that corresponds to the specific information. Extensive scrolling or searching for such specific information becomes unnecessary.

FIG. 21B is a diagram of a second embodiment of a graphical user interface display of a multiple-page electronic message.

In the embodiment of FIG. 21B, message window 2100 further includes a static content pane 2105 that may display text notes, graphic images, banner advertisements, or any other desired static content. In one embodiment, data for static content pane 2105 is obtained from the static content region of electronic media that associated with a group collaboration application.

FIG. 22A is a diagram of a third embodiment of a graphical user interface display of a multiple-page electronic message.

A message display window 2200 is provided in the embodiment of FIG. 22A. As in FIG. 21A and FIG. 21B, message display window 2200 may include a toolbar 2201, a message header pane 2202, and a message body display pane 2204.

Optionally, in certain embodiments the message display window 2200 also comprises a message toolbar 2207 that provides command options for generating and working with messages that are displayed in the message body display pane 2204. For example, in one specific embodiment, message toolbar 2207 provides New, Forward, Note to Author, Note to All, View Recipients, Edit, and Notifications command options.

Message toolbar 2207 may be implemented separate from toolbar 2201 in embodiments that interoperate with unmodified e-mail processing applications. For example, in an embodiment that interoperates with Microsoft Outlook as an e-mail processing application, the toolbar 2201 is generated by Microsoft Outlook and controls its functions, whereas message toolbar 2207 is generated as part of a displayed message by a separate server. This arrangement enables use of the electronic media, group collaboration applications, and multi-page electronic messages disclosed herein without modification of the e-mail processing application. Alternatively, the electronic media, group collaboration applications, and multi-page electronic messages may be integral to the e-mail processing application, and the functions of toolbars 2201, 2207 may be combined in a single toolbar.

Operation of functions of toolbar 2207 are described further herein in connection with a description of a server structure that may be used to implement the processes described herein.

Selecting the New function from message toolbar 2207 is a request to generate a new message that may contain one or more message pages. Selecting the Forward function is a request to forward the currently displayed message, including all message pages, to a new recipient. Selecting the Note to Author function is a request to create a static note that is visible only to the author of the currently displayed message when that



author displays or re-displays the message. An example of such a note is action request 2284 of FIG. 22C.

Message window 2200 of FIG. 22A further includes one or more message pages 2206, 2214, 2216. Each message page has an associated message navigation region and a message page body. For example, message page 2206 includes a navigation region 2208 and a page body 2210. In the embodiment of FIG. 22A, each navigation region extends laterally from its associated page body, such as to the left of the page body. Alternatively, the navigation regions may extend to the right of the message body. Each navigation region is arranged so that it is continuously visible whenever a particular page body is displayed.

The page body may contain any desired text or graphics, or a combination thereof. In the example of FIG. 22A, page body 2210 includes a page title 2220 and a text block 2222. Additionally or alternatively, there may be other text, graphics, icons, images, hyperlinks to other resources, etc.

Page body 2210 may also comprise one or more dynamic content regions that display dynamic content and are supported by one or more active application elements that are executed by a supporting server. In one specific embodiment, page body 2210 comprises one or more application building blocks that have been selected from an application starter set or library. Each building block comprises a pre-defined, self-contained module of executable program instructions that can be linked together with other building blocks to form a complete executable application.

Typically, each building block performs a discrete function, such as group discussion; polling; interactive Web pages; file sharing; inline document viewing; table generation; rating generation; surveys; approval lists; schedules; images; image galleries; invitations; information fields; connections to external systems and applications; and others. When a page containing a building block is selected using its navigation region, a supporting server re-displays an image corresponding to the application with graphical elements relating to the selected building block. Such graphical elements may include headers, text, graphic images, radio buttons or other user input widgets, as appropriate for the function to which the building block relates.

FIG. 22B is a diagram of a fourth embodiment of a graphical user interface display of a multiple-page electronic message that includes a dynamic content region.

As in FIG. 22A, the embodiment of FIG. 22B includes a message display window 2200, toolbar 2201, a message header pane 2202, a message body display pane 2204, a message toolbar 2207 that provides command options, and one or more message pages

2206, 2214, 2216 each having an associated message navigation region and a message page body. Message display window 2200 also comprises a dynamic content region 2210 that contains dynamic information.

In the example of FIG. 22B, the dynamic content region 2210 is based on a comment building block that facilitates gathering comments from a plurality of members of the group. As members of the group receive the message shown in message display window 2200, each member may select dynamic content region 2210 and enter one or more comments. The entered comments are stored in a database of a supporting server. Whenever any other group member or other recipient receives the message, opens it, and views message page 2206, the server obtains a then-current copy of the dynamic content, such as all comments entered to date, and displays it as part of dynamic content region 2210.

Dynamic content region 2210 operates as a discrete window within message page 2210. If the content associated with the dynamic content region 2210 overflows the dynamic content region 2210 when it is displayed, a user may select a navigation tool 2212 to view additional content.

Likewise, if not all the content associated with a particular message page will fit in the message page when it is displayed, the message page may include an indication that other content can be obtained. For example, in FIG. 22B, message page 2206 has a downwardly extending navigation region 2214 that displays a navigation tool 2216. By selecting appropriate icons within the navigation tool 2216, a user can instruct a supporting server to retrieve and display different parts or additional parts of a message page.

In one embodiment, if a message has a plurality of message pages, additional page navigation regions are displayed generally in a column arranged on the left edge or right edge of the message display pane. In embodiments in which the elements in the message display window are rendered based on source code in HTML, the message display window has potentially infinite length. In these embodiments, the message display window may include any number of page navigation regions. If not all the message navigation regions are viewable in on a screen display, the user can scroll the screen display to view further page navigation regions.

Alternatively, to remove the need for scrolling, there may be a pre-defined maximum number of pages that appear in a message display window at any one time.

o

FIG. 22C is a diagram of a further embodiment of a graphical user interface display of a multiple-page electronic message that includes an indicator of additional pages.

As in FIG. 22A, the embodiment of FIG. 22B includes a message display window 2200, toolbar 2201, a message header pane 2202, a message body display pane 2204, a message toolbar 2207 that provides command options, and one or more message pages 2206, 2214, 2216 each having an associated message navigation region and a message page body. Message display window 2200 also comprises a “more pages” navigation region 2250, which is present when a message has a plurality of message pages, and message display window 2100 has insufficient space to display all the message navigation regions associated with the message pages in a row.

Selection of the “more pages” navigation region 2250 causes the server or other element to re-generate message body pane 2104 such that one or more of the other message navigation regions are displayed, and such that previously visible message navigation regions scroll or slide to the side, up or down to make room for the newly displayed message navigation regions. Such re-generation may result in removing the “more pages” navigation region 2250 from the display.

The “more pages” navigation region 2250 may comprise an indicator that additional message pages and navigation regions are available for display and selection. For example, the region can display an arrow, dot, or other icon adjacent to the right-most message navigation region.

FIG. 22D is a diagram of a further embodiment of a graphical user interface display of a multiple-page electronic message.

The embodiment of FIG. 22D includes a message display window 2200, toolbar 2201, a message header pane 2202, a message body display pane 2204, and message toolbar 2207 that provides command options. One or more message pages 2252, 2254, 2256 are provided, each having an associated message navigation region 2252A, 2254A, 2256A, and a message page body 2252B, 2254B, 2256B. Each of the message pages 2252, 2254, 2256 is displayed in an overlay manner such that one of the message pages, e.g., message page 2252, appears to be the topmost message page, and such that other message pages 2254, 2256, etc., appear to be stacked beneath the topmost message page.

In this arrangement, each message navigation region 2252A, 2254A, 2256A is continuously visible and may be selected at any time. When a message navigation region is selected, the corresponding message page is internally designated as the topmost

message page, and the message body display pane 2204 is re-generated such that the new topmost message page appears on top and its contents are visible.

In FIG. 22D, for purposes of illustrating a simple example, three (3) message pages are shown. However, embodiments may comprise any number of message pages.

### 3.0 LINKING AND AGGREGATING MESSAGES

According to one embodiment, the system and processes described herein facilitate linking and aggregating messages, such as transportable applications.

FIG. 20 is a block diagram illustrating a plurality of messages that are linked across different folders. In one embodiment, a persistent URL may be used to identify the messages. In FIG. 20, non-underlined message labels identify messages (e.g. "Message 1"), and underlined message labels identify hyperlinks to other messages. Such links may be used to hyperlink messages in the same folder. For example, message 1 of Inbox 1 is linked to message 3 of Folder 1. Links may also associate messages in different folders, such as message 10 of Folder 4 that is linked to message 12 of Folder 5. Links may associate messages in an "in-box" to messages in a folder, as in the case of message 1 and message 3. Links may associate messages in a folder and messages in an application, as in the case of message 7 of Folder 3, which is referenced in Web Page 2050. One message may contain links to multiple messages. For example, message 10 of Folder 4 has links to message 11 and message 12.

The interconnection between messages enables serialized and parallel decision making within a messaging system. Further, since the URL is not dependent upon the position within the messaging system, such as within a folder or on a web page, the position of the message may be changed without destroying the link between messages. For example, message 9 may be moved from Folder 4 to Folder 5 and still maintain a link from Web page 2050.

When two or more messages are linked in the foregoing manner, they form a message web. In one embodiment, message webs are networks of messages that are related by a topic or activity. Such networks aggregate knowledge that is generated within a context of the activity. For example, message webs linked over a network can provide information that does not require significant webmaster interaction or detailed internal system knowledge. Moreover, a current status of content or other attributes may be captured within the message web. This allows the health or age of the content to be communicated to a system administrator or other interested individual. Accordingly, the most current or active information can be highlighted. The content of messages within a

message web may be involved in the same context and may include content modification, process interaction, choice making, and activity launching among a group or participants.

The links may be configured to be unidirectional or bi-directional. For a unidirectional link up, there is not a corresponding link at the destination back to the source location of the unidirectional link. In a bi-directional link up, there may be a corresponding link at the destination back to the source location of link creation.

FIG. 23A is a flowchart of a process for linking messages, according to one embodiment. In general, message links may be constructed manually by using user actions to link to a message. Further, a URL of a message or message web can be presented to an end user, suitable for copying and linking. Message authors or recipients can link messages. Ad hoc tasks may be served by letting a user connect steps within the task. Ad hoc knowledge organization may be served by letting a user connect related messages to each other. Message linking provides a streamlined mechanism for copying a message's URL to the clipboard and avoids obscure multiple step processes to locate a message's URL.

In block 2302, the process of FIG. 23A begins when a recipient, author or other user of a transportable application wishes to link one transportable application to another. In block 2304 the user selects a Link function button within the message. The Link button may be displayed, for example, as part of command buttons 282 of FIG. 2C. In response, in block 2306 the user is prompted to select a linking method. The prompt of block 2306 may be a dialog box or wizard. Block 2306 also may involve adding a List building block to the transportable application, wherein items in the list of the building block comprise references to linked messages. Thus, the List building block serves as a mechanism for maintaining links to other messages.

In one method, the user may copy the URL of another message to the clipboard provided by the operating system, in block 2308. Control returns to block 2304, and the user then pastes the copied URL to a link field that is provided in the prompt of block 2306, as shown by block 2310. Alternatively, the user may select one or more messages from a personal folder or list, or from a group folder or list, as shown by block 2312. In another alternative, the user may search for messages and select the right ones, as indicated by block 2314; this may involve opening another window that has a personal folder or list of messages. In another alternative, the user may drag a message from another context into the current message or into the dialog box or wizard, as in block 2316.

After selecting a linking method and a linked method, in block 2318, the user is prompted to change one or more link attributes, as appropriate. Such attributes may include cross-linking, link labels, link description, access control, etc.

In block 2320, definitions of messages are modified to add links. In one embodiment, block 2320 involves updating the List building block of the transportable application to add a reference to the linked message. As a result, the selected messages become linked, at block 2322. A user who opens the transportable application can view the List building block and link to another message by selecting a message that is in the list.

FIG. 23B is a flowchart of a process of automatically linking messages in another embodiment. Linking of messages may occur automatically or indirectly as a result of actions by a user within a task or as a result of workflow execution, *e.g.*, as a part of creating or performing a next step of the task. Such links may be constructed automatically or indirectly by the application server 202. Thus, error rate can be reduced by automatically linking messages together, where steps or messages are related.

In block 2324, a workflow process generates a new message that is related to the current message. For example, block 2324 may comprise a first transportable application generating another transportable application in response to an event that is generated by user interaction with a page or building block. Alternatively, in block 2326, a workflow or user event determines that a link is needed among two or more messages.

In block 2328, optionally, the user may be prompted to indicate whether the messages should be linked. The prompt may take the form of requesting confirmation of a proposed link of messages. If the user indicates that the messages should not be linked, then in block 2330, no link is created. Alternatively, if the link is confirmed, then in block 2332 and block 2334 the user is optionally prompted to modify one or more link attributes, as in block 2318 of FIG. 23A. Control then passes to block 2320 and block 2322, as in FIG. 23A.

Messages may be linked based on the message's content or context. For example, a name, electronic mail address, a group, or company name, may be used to recognize and match a message with another message on an associated page or in an associated folder. In another example, a message that is named similarly to another message with the same or similar recipient list may be tied together to form a Message Web.

FIG. 23C is a flow diagram of a process of automatically creating message links in response to a change in an object. Such automatic linking may relieve an author of adding common links by hand.

In block 2336, a change occurs to one or more objects to which automatic links can be made. For example, a building block or page of a transportable application is deleted, modified, renamed, or created. In block 2338, block 2342, and block 2346, branch points are carried out in accordance with the kind of change that occurred. In block 2338, if the object is deleted, then in block 2340, links are removed from messages and from the list of links in the message. For example, the list building block of the message is updated to delete links. In block 2342, if the object is edited or renamed, then the list of automatically generated links is reviewed. For each link that is identified in the list, the link is followed to the linked message. Any link in the list of linked messages that contains the old name is updated with the new name, as shown by block 2344. If a new object is created, as in block 2346, then in block 2348, its content is marked as changed. A background task or process is scheduled and dispatched to look for references to the new object.

As a result, affected links are changed, as shown by block 2352. If all the tests of block 2338, block 2342, and block 2346 are negative, then any change that has occurred is not relevant to linking, so no action is taken, as shown in block 2350.

FIG. 23D is a flow diagram of a process of updating message links in response to changes in message content. In block 2354, a change to one or more items of message content is detected. For example, a recipient of a transportable application updates the application with new dynamic content, and in response, a building block of the application generates an update event. In block 2356, the changed content is search to identify one or more recognizable object references that could be the subject of a link to another message. For example, the changed content is searched to identify an e-mail address or user name, organization name, message title, etc.

In block 2358, if an e-mail address or user name is identified, a link is created to instantiate an e-mail to the user. For example, an HTML "mailto:" link may be created in the List building block that references the user. In block 2360, if an organization name is identified, a link to a Web page for that organization is created, e.g., in the List building block. In block 2362, if a message title of another message or transportable application that is known in the system is identified, then a link to that message is created, e.g., in the List building block. In block 2364, if other linkable content is identified, then an appropriate link to that content is created in the list. An example of other linkable content may be a digital song, an image, etc. In block 2366, the list of links is updated with any link that has been created in the preceding steps.

FIG. 23E is a flow diagram of a process of suggested message linking. In one embodiment, message links may be suggested, and either accepted or denied by a user, especially when adding a step to an ad hoc task. This combines the advantages of manual and automatic message linking in order to enhance ad hoc task/data linking. Lists of suggested links may be built into the logic built into a message web, or are provided in a message tab (as in the aggregated content under a tabbed presentation), or are provided in a message template, or are provided in a message web template. Templates can be used to allow new instances of message tabs, messages, or message webs to be selected by an end user. End users may create templates by using running instances of message tabs, messages, or message webs and saving their structure and optionally their content. Templates may be categorized and shared with others. In one embodiment, the process of FIG. 23E is useful to allow a user to optionally add a link to complete, enhance, continue, or add to a current task. For example, a user may schedule an event in one task and once a time is agreed upon for the event, may continue the task by enabling users in a group to purchase tickets to the event.

In block 2370, a user finishes a step in a task that is defined as a message. For example, a user completes providing input to a building block of a transportable application. In response, in block 2372, the input is analyzed and the user is presented with a list of other messages that are likely to follow the completed step; ordered by context-determined relevance. For example, if the user has completed rating a job applicant in a poll building block of a human resources transportable application, the system determines that a salary offer application is likely to follow next. Therefore, the user is presented with the salary offer application in the list. Alternatively, a new message is created by the user, or automatically, while the user is in the context of another message.

In either case, in block 2376, the user indicates whether to link the new message with the original message, as by selecting a user interface button. If the user requests linking, as tested in block 2378, then automatic message linking is carried out, as described herein in connection with FIG. 23B. If no linking is requested, then none is carried out, as shown by block 2379.

When a user traverses a hyperlink from within e-mail messages to a URL or Web document, according to one embodiment, the Web page or other HTML content is displayed within the e-mail client window, as in FIG. 12. The content may comprise on-page navigation controls (Home, Back, Forward), since browser controls are not provided in conventional e-mail clients.



FIG. 24 is a flow diagram of a process of displaying HTML content in an e-mail client with browser navigation features. In block 2402, the system is requested to display a transportable application that contains a link to HTML content. In block 2404, the system determines whether, in displaying the HTML content, it should spawn a browser window and display the content therein, or display the content within the e-mail client window. If the test of block 2404 is true, then in block 2406, the HTML message display is supplemented with navigation functions that are normally available in a browser.

In one implementation, block 2406 involves displaying a Home button, which causes the original page of the dynamic portion of the message to be shown, a Back button, and a Forward button. FIG. 12 illustrates examples of such buttons. The Home button is implemented as a self-referencing URL to the dynamic content portion of the transportable application. The Back and Forward buttons are implemented as JavaScript elements such that when each button is selected, JavaScript is invoked to carry out the functions. The JavaScript elements link back to application server 202 to determine what URL to load, based on a link traversal history that is maintained by that server. As shown in block 2408, when a link is traversed, application server 202 is updated with the current and next links in a link history that is associated with the current transportable application. In block 2409, a next page of the transportable application is displayed. Control flows back to block 2404 to render that page in the same manner.

In this configuration, browser controls may be used to navigate links within the messaging system. Thus, a user can easily navigate between a current message and another message, and then return to the current message. This navigation may occur within the same window.

FIG. 25A is a block diagram of a linked collection of related message webs, referred to herein as a message web ring. Such rings may provide a complete list of related message webs. A user may navigate through the ring searching for desired information. Message web rings may also provide a higher level of aggregation to organize a project's tasks. In the example of FIG. 25A, a first message web 2501 comprises messages 2510, 2512, 2514. Message 2510 is the home message of message web 2501, which acts as the head of a message web ring that includes a second message web 2502 and a third message web 2503. The second and third message webs 2502, 2503 each include respective message web home messages 2504, 2506. Each message web home message 2510, 2504, 2506 has a Next link that identifies the next message web in the ring, and a Previous link that identifies the previous message web in the ring. The

links among home messages 2510, 2504, 2506 form a ring in which message web 2501 is the head and message web 2503 is the tail.

In this configuration, linked messages may be navigated between multiple applications. For example, a user may navigate within an application between messages or navigate between applications and then access multiple messages within another application. This allows aggregation of messages to be performed within a single application or folder and also amongst other folders.

FIG. 25B is a flow diagram of a process of creating a message web ring. In block 2520, a message web is created that resides within an area, group, or project context that may have other message webs. In block 2522, a test is carried out to determine whether the current message web is the first message web within the current context. If so, then in block 2524, the Next and Previous links of the current message web are set to be equal and to refer to the current message web.

If not, then the current message web is threaded integrated into a ring structure with other message webs, in a position between the head and tail of the ring, in block 2526. In particular, in one embodiment, the Previous link of the current message web is set to the value of the Previous link of the home message of the tail message of the ring. The Next link of the home message of the tail message web of the ring is set to point to the home message of the current message web. The Next link of the home message of the current message web is set to the home message of the head message of the ring. The Previous link of that message is set to point to the new message web. As a result, as shown by block 2528, the message web ring's head and tail are updated to include the new message web, and the ring is therefore updated, as indicated by block 2530.

FIG. 26A is a block diagram illustrating messages in a message web having shared address lists. In one embodiment, a message web may share a list of addresses among its member messages. Individual messages may have their unique addressees extended or restricted in comparison to a shared list of addresses. Shared lists may be reused, which can avoid initially generating an address list for each step in a task. Further, a user can be easily added to an existing task by adding the user to a shared address list and forwarding a message to the user within the message web. Access to the remainder of the message web is then achieved by way of links and navigation tools as described herein. Subsequent steps that are added will then include the added user.

In particular, a list of recipients of the message may be changed between linked messages, such that a first set of recipients can be defined in one message and another set of recipients can be defined in another message. For example, in FIG. 26A, in a first

message 2601 the recipients are Bob, Carol, and Dave. However, in message 2603, which is linked to message 2601, the recipients are defined as everyone in message 2601 plus Harry. Thus, a recipient list may be shared between messages. Additionally, messages may be configured to define business logic, other roles having the same sharing relationship between messages. The link between message 2604 and message 2603 demonstrates one form of “side-bar” or private conversation that could occur within a message web in which the link between messages is unidirectional.

FIG. 26B is a flow diagram of a process of generating a list of recipients of a transportable application. In block 2605, a list of users is generated from a role description. For example, an initial set of recipients of a transportable message, such as message 2601, may be generated based on a role that comprises a set of user names. Assume that the author of the message 2601 addresses the message to “Project X Group” and that group name is associated with a set of users {Bob, Carol, Dave}. As a result, the recipient list of message 2601 is Bob, Carol Dave.

In block 2606, the system determines if the list generated in block 2605 contains a reference to another list. If so, in block 2608 the current recipient list is expanded to include all recipients who are named in the referenced list, and duplicates are removed. In block 2610, the system determines if the recipient list generated in block 2605 contains a reference to an individual. If so, then the individual is added to the current recipient list, and duplicates are removed. In block 2614, the system determines if the recipient list generated in block 2605 includes instructions to exclude a user or list. If so, then in block 2616, the referenced list is expanded, and its members are removed from the list generated in block 2605. If the list generated in block 2605 has more instructions, then they are processed in similar manner. As a result, a new recipient list is generated, as indicated in block 2620.

FIG. 27A is a block diagram illustrating that the content of messages that are linked can be changed, with automatic propagation of changed content to linked messages. This allows content to be targeted to certain groups and separated from other groups. Changes to content may flow back, forth, and among linked message members. The linking provides connections that workflow processes can use to identify targets and sources of data that are needed within independent messages. Links used for sharing data and role information may be made unavailable to the end user.

In the example of FIG. 27A, a first message 2701 is linked by link 2702 to message 2703. The Meeting Date value of message 2701 is linked to a field “Schedule.Choice.Best” of message 2703, so that changes to that field are propagated to

first message 2701. Message 2703 is also linked to message 2704 and sends the same field value to it. Thus, changes in message 2703 flow along links 2702, 2708 to other messages in a message web.

FIG. 27B is a flow diagram of a process of updating data among linked messages in a message web.

In block 2726, the system determines that a field of a transportable application has been updated. For example, user input results in a change to a data entry field of a building block. In block 2728, the system determines whether any other system objects, such as data objects, fields, or building blocks have subscribed to the field that has changed. If so, then the subscribers are marked as needing to be refreshed, in block 2730. For every subscriber needing to be refreshed, block 2732, control is passed to block 2724 to mark the subscriber field as no longer needing to be refreshed. The field is then interpreted starting at block 2710. When all subscriber fields have been refreshed, control passes to block 2734 in which the process of FIG. 27B is complete.

In block 2710, a field definition and value within a message are interpreted to identify a link or other reference to other messages. In block 2712, the system determines whether the field definition comprises a link to data in another field. If so, then in block 2714, the system verifies that the linked field is updated, and fetches data from the linked field. Block 2714 may involve dynamically retrieving field data from database 208. In block 2716, the system tests whether the field definition contains an instruction to embed data from another field. If so, then the embedded field is identified and tested to determine if it is updated, as in block 2718. Data from the embedded field is fetched, and the field definition property is cleared since the data is then embedded.

In block 2720, the system tests whether all data in the field definition has been resolved to static form. If not, then control is transferred to block 2712 to resolve any remaining references. If so, then in block 2722, any business rules in the field definition are applied to compute the final field value.

Accordingly, data from one field of a message that is linked in a message web may propagate to fields of other linked messages, automatically and in response to user input or other actions that cause changes to data objects.

FIG. 27C illustrates one application of the processes of FIG. 27A, FIG. 27B in which a linked collection of related messages are used to aggregate data from a child message up to a parent message. The result may be displayed in any message of the tree to show activity in the lineage of a tree. Messages 2742, 2744, 2746, 2748 participate in a tree rooted at message 2740, which receives data values from all child messages and

aggregates them. Message 2740 is used to collect donations from a group of recipients. Similar donation collection messages may be created from any donation collection message and sent to a different set of recipients. Each message shows the donations committed to by its recipients, as well as the donations committed to by all its descendants. Arrows show the flow of data up through the message hierarchy. Message 2740 is linked to and subscribes to donation fields of child messages 2742, 2744, 2746, 2748. Each such message may also be a message web.

FIG. 28A is a diagram of a graphical user interface display in which a transportable application includes a plurality of tabs that switch between the content of message web members. Transportable application 2800 comprises a header area 2802 and a message web header 2804 having a plurality of tabs 2806. In the example of FIG. 18A, the tabs 2806 include an agenda tab link 2808, action items link, invitation link, poll link, etc. Agenda link 2808 accesses an agenda that is presented within the context of transportable application 2800, e.g., in a display pane 2810. By clicking on one of the tabs, the action items for that message content may be viewed or changed without leaving the context of the enclosing message. There may be different access control capabilities for the content under each tab. Participants may link other message webs to header 2804 by selecting an Add New Tab link 2812. In response, the system prompts the user to identify a message web to link, as in FIG. 23A.

Thus, message web members that share a common address list may be aggregated into a single message for purposes of presentation. Messages that are added, modified, and deleted as parts of the message web are reflected in the aggregated presentation. Some tasks are better navigated by direct access to steps instead of linear progression from one step to the next. By aggregating message web members within a single presentation, available members can be easily seen and accessed directly. Aggregated message web presentations may also provide a per role control panel for a task such that messages with diverse recipient or access control lists can be presented in an aggregated manner.

FIG. 28B is a flow diagram of a process of adding tabs to a transportable application of the type shown in FIG. 28A. In block 2812, a user action adds to a collaboration activity within a message. For example, assume that a recipient selects Add New Tab link 2812 of FIG. 28A. In response, members are assigned to the new collaboration, in block 2814. For example, the recipient list of the current transportable application may be added to a message recipient list for the new collaboration. In block 2816, content for the new collaboration is created. In one embodiment, the content is

created by prompting the user to select whether to extend the collaboration as a new part of an existing message, or as a new message web member, as indicated by block 2818.

If a new message web member is requested, then in block 2822, a new message is sent to the recipient list, and appropriate links are created to insert the new message into the message web. If a new part of the same message is selected, then in block 2820, the new content is added to the current message. Recipients of the current message notified appropriately, and the new content is highlighted when such recipients read the message. In either alternative, as shown by block 2824, the new message may be associated with a new tab in the current message.

FIG. 28C illustrates a notification message of a message web. Message 2800 of FIG. 28A is shown with New icons 2830, 2832 indicating areas that are new since the time the user last looked at this message. The user may set conditions for such notifications by selecting a Notifications link 2841. In response, the system displays a dialog box 2840 having notification activation links 2842, 2844. The user may turn on notifications relating to a particular message tab with notification activation link 2844 and may turn on notifications about an entire message web with link 2842. When such notifications are set, the icons 2830, 2832 are displayed when content of the page or message web changes.

Thus, a single notification message serves to alert a user about changes or additions spanning multiple messages within a message web. Individual changes or additions across messages are highlighted within the notification message. The notification message can be the message web home, any message web member, a digest of changes within a message web, or a list of message webs.

For example, assume a user working on a multiple step task wants to be kept informed about changes within that task. By signing up for message web notifications, a user cuts down on the number of notifications he receives by aggregating indications of what has changed into a single notification message. The user does not get multiple notifications corresponding to every change within the message web. No further notification is sent until a user has looked at the changes indicated by a previous notification.

Message webs that are delivered to recipients may be received in an e-mail client and identified by a special subject designation. For example, a special message that represents a message web may be displayed within the same containers, lists, or folder as a regular email message. An icon representing the message could be different, and menu functions available for the special message might be different from those available for a

regular message. The message subject line may include a topical subject, appended with, “- Message Web,” or a similar designation. Message Webs may take the place of individual messages within message folders. These may be individual messages, but a special message may be referred to as the “Message Web Home”. Message Webs may exist within the same folders as messages.

FIG. 29 illustrates a method for tracing deleted messages using links. In one embodiment, a message deleted from within a message web does not destroy the integrity of the message web; instead, message links are automatically repaired. Automatically generated message placeholders may also be used in the place of deleted messages in order to maintain the integrity of a message web. These placeholders can act as patch panels, giving user options of where to link to, especially when the self-repair process cannot definitively decide what the correct links are out of a broader set. Additionally, links to a deleted message may be removed or disabled from their source.

Within the Internet, broken links are an extremely common problem. By knowing that message webs form a unit, a manager can either rethread the ends of links pointing to deleted messages or utilize a placeholder message shell through which existing links can traverse. The shell has little or no content other than the links from it that the previous message contained. The shell can also offer suggestions, when the shell manager is not sure of what links are important. Links that are automatically deleted when a message is deleted result in fewer broken links.

In block 2902, a message in a message web is marked for deletion. In block 2904, the system determines what kind of deletion to carry out. In an embodiment, deletion may involve complete elimination of the deleted message, or retaining a message shell as a placeholder. If a placeholder is retained, then in block 2908, the content of the message is deleted, and the existing message web links are retained. Optionally, a summary or decision portion of the message may be retained.

If the message is completely eliminated, then in block 2906, all links from the deleted message to other members within the same message web are identified. Each message web member is visited and all links to the deleted message are identified. The links are replaced with links to the next message in the message web, subject to access control privileges and elimination of self-referencing links. The updated links are entered in an automatically maintained link list. In block 2910, links to other message webs are similarly updated. In block 2912, the deleted message is added to a list of deleted messages or “dead URLs.” Optionally, a crawl of the Internet may be scheduled to search

for external links to the deleted message, since such links become “broken” upon deletion of the message. As a result, the message is deleted.

FIG. 30 illustrates a message web map that may be used in a graphical user interface of a messaging system that supports linked messages, in one example embodiment. The user interface graphically depicts messages as connected to each other. Applications containing multiple messages may also be linked together. In the example of FIG. 30, a visualization of an entire message web available to a user is provided. With such a message web map, a user can locate a message, jump directly to another message (even if not directly linked), see what messages have new content (relative to the user), and note the status of each message/step in a task or sub-task. Similarly, a map of other applications and folders may be represented.

For example, FIG. 30 depicts a folder 3002 for a particular project that contains a first message web 3006 dealing with an upcoming status meeting. The message web 3006 comprises a home message 3010 and child messages 3012, 3014, 3016, 3018. Each such message may be a transportable application as described herein. FIG. 30 also depicts a previous message web 3004 and next message web 3008 in a web ring. Thus, a user may receive a graphical view of complex message relationships and related message webs.

FIG. 31A, FIG. 31B, and FIG. 31C are diagrams of graphical user interface displays that illustrate an example of a recruiting process using a message web.

Assume that a user logs into the system and enters a name and password via, for example, a Web page. Once the password and user name are verified by application server 220, an electronic form or application editor is provided to the user. The user may author a transportable application providing recruiting functions based on a template. The user may specify one or more addresses, such as electronic mail addresses, for the participants of the message web in a field of the form. The user may also specify the subject of the Message Web in a field. The form may also include a link that allows a user to upload content into a static content region. Static content, for example, a candidate’s resume may be uploaded from a file residing on the user’s computer. The form may also include buttons that allow the user to send the contents of the form to application server 220 and to first preview the content before it is sent to the server.

Once the user submits the form to the server, in response, the server sends a transportable application 3100 (FIG. 23D) with the content specified in the form. In one embodiment, transportable application 3100 may include one or more static content regions and dynamic content regions 3101, 3103, 3120, 3110, 3125, which include and



capture content from regions of the electronic form. The regions may display headers, introductory text, substantive content, graphics, etc.

The transportable application 3100 also includes tabs 3105, 3106 that are associated with separate pages of the application. Tab 3105 specifies that transportable application 3100 includes a "Discussion". In this example, participants in the discussion can add comments into an interface region 3125. The tab 3106 allows one or more of the participants to link the current "Discussion" to, for example, a "Schedule" which can be used to set-up times to interview the candidate. A window 3160 may pop-up that allows a participant to choose "Schedule." Once the participant clicks on Schedule, the request is sent to the server.

In response, the server generates a new electronic form for providing a schedule. FIG. 31B illustrates an example electronic form 3900 that includes a region 3930 with fields 3905, 3906, and 3907 for specifying the Time, Date, and Name of the participants. The form 3900 may also include fields 3901 and 3902 and a dynamic content region 3903. The form 3900 may also include Send and Preview buttons 3941, 3942 that function in a manner similar to the buttons described above. Further, the form 3900 may include a link 3940 that enables a user to submit the content of the form to the server. When a participant presses the button 3940, the content of the form 3900 is sent to the server.

The transportable application 3100 is then be updated to reflect the content of the form 3900. FIG. 31C is a diagram of a screen display in which the transportable application reflects such updates. A tab 3107 is added to the transportable application 3100 to indicate that a "Schedule" has been initiated to interview the candidate. A participant may reply to the Schedule using the interface region and may also view the most current content in the dynamic region. The participants may use the tabs to navigate between the Discussion and Schedule. Each time a participant navigates between the tabs, the participant dynamically receives the most current content from the server.

A participant may add other transportable applications, pages or building blocks to the transportable application. In this way, multiple types of applications can be combined using the same message without the need for multiple instances of messages or tedious navigation through multiple message folders. Additionally, tabs similar may be added to the transportable application that allow multiple transportable applications, pages, or building blocks to be available, but marked as no longer active or useful to one or more of the participants ("grayed out").

#### 4.0 HARDWARE OVERVIEW

The approaches described herein may be implemented in hardware or software, or a combination thereof. In one embodiment, the approaches are implemented in computer programs executing one or more programmable computers. The programmable computers may be either general-purpose computers or special-purpose, embedded systems. In either case, program code is applied to data entered with or received from an input device to perform the functions described and to generate output information. The output information is applied to one or more output devices.

Each program is preferably implemented in a high level procedural or object-oriented programming language to communicate with a computer system. However, the programs can be implemented in assembly or machine language, if desired. In any case, the language may be a compiled or interpreted language.

Each such computer program is preferably stored on a storage medium or device (e.g., CD-ROM, hard disk, magnetic diskette, or memory chip) that is readable by a general or special purpose programmable computer for configuring and operating the computer when the storage medium or device is read by the computer to perform the procedures described. The system also may be implemented as a computer-readable storage medium, configured with a computer program, where the storage medium so configured causes a computer to operate in a specific and predefined manner.

FIG. 19 is a block diagram that illustrates a computer system 1900 upon which an embodiment of the invention may be implemented. Computer system 1900 includes a bus 1902 or other communication mechanism for communicating information, and a processor 1904 coupled with bus 1902 for processing information. Computer system 1900 also includes a main memory 1906, such as a random access memory ("RAM") or other dynamic storage device, coupled to bus 1902 for storing information and instructions to be executed by processor 1904. Main memory 1906 also may be used for storing temporary variables or other intermediate information during execution of instructions to be executed by processor 1904. Computer system 1900 further includes a read only memory ("ROM") 1908 or other static storage device coupled to bus 1902 for storing static information and instructions for processor 1904. A storage device 1910, such as a magnetic disk or optical disk, is provided and coupled to bus 1902 for storing information and instructions.

Computer system 1900 may be coupled via bus 1902 to a display 1912, such as a cathode ray tube ("CRT"), for displaying information to a computer user. An input device 1914, including alphanumeric and other keys, is coupled to bus 1902 for

communicating information and command selections to processor 1904. Another type of user input device is cursor control 1916, such as a mouse, a trackball, touch screen, keypad of a cellular telephone or PDA, or cursor direction keys for communicating direction information and command selections to processor 1904 and for controlling cursor movement on display 1912. This input device typically has two degrees of freedom in two axes, a first axis (e.g., x) and a second axis (e.g., y), that allows the device to specify positions in a plane.

The invention is related to the use of computer system 1900 for collaborative communications, multiple-part messages, and linking and aggregating messages. According to one embodiment of the invention, collaborative communications, multiple-part messages, and linking and aggregating messages is provided by computer system 1900 in response to processor 1904 executing one or more sequences of one or more instructions contained in main memory 1906. Such instructions may be read into main memory 1906 from another computer-readable medium, such as storage device 1910. Execution of the sequences of instructions contained in main memory 1906 causes processor 1904 to perform the process steps described herein. In alternative embodiments, hard-wired circuitry may be used in place of or in combination with software instructions to implement the invention. Thus, embodiments of the invention are not limited to any specific combination of hardware circuitry and software.

The term "computer-readable medium" as used herein refers to any medium that participates in providing instructions to processor 1904 for execution. Such a medium may take many forms, including but not limited to, non-volatile media, volatile media, and transmission media. Non-volatile media includes, for example, optical or magnetic disks, such as storage device 1910. Volatile media includes dynamic memory, such as main memory 1906. Transmission media includes coaxial cables, copper wire and fiber optics, including the wires that comprise bus 1902. Transmission media can also take the form of acoustic or light waves, such as those generated during radio wave and infrared data communications.

Common forms of computer-readable media include, for example, a floppy disk, a flexible disk, hard disk, magnetic tape, or any other magnetic medium, a CD-ROM, any other optical medium, punch cards, paper tape, any other physical medium with patterns of holes, a RAM, a PROM, and EPROM, a FLASH-EPROM, any other memory chip or cartridge, a carrier wave as described hereinafter, or any other medium from which a computer can read.

Various forms of computer readable media may be involved in carrying one or more sequences of one or more instructions to processor 1904 for execution. For example, the instructions may initially be carried on a magnetic disk of a remote computer. The remote computer can load the instructions into its dynamic memory and send the instructions over a telephone line using a modem. A modem local to computer system 1900 can receive the data on the telephone line and use an infrared transmitter to convert the data to an infrared signal. An infrared detector can receive the data carried in the infrared signal and appropriate circuitry can place the data on bus 1902. Bus 1902 carries the data to main memory 1906, from which processor 1904 retrieves and executes the instructions. The instructions received by main memory 1906 may optionally be stored on storage device 1910 either before or after execution by processor 1904.

Computer system 1900 also includes a communication interface 1918 coupled to bus 1902. Communication interface 1918 provides a two-way data communication coupling to a network link 1920 that is connected to a local network 1922. For example, communication interface 1918 may be an integrated services digital network ("ISDN") card or a modem to provide a data communication connection to a corresponding type of telephone line. As another example, communication interface 1918 may be a local area network ("LAN") card to provide a data communication connection to a compatible LAN. Wireless links may also be implemented. In any such implementation, communication interface 1918 sends and receives electrical, electromagnetic or optical signals that carry digital data streams representing various types of information.

Network link 1920 typically provides data communication through one or more networks to other data devices. For example, network link 1920 may provide a connection through local network 1922 to a host computer 1924 or to data equipment operated by an Internet Service Provider ("ISP") 1926. ISP 1926 in turn provides data communication services through the worldwide packet data communication network now commonly referred to as the "Internet" 1928. Local network 1922 and Internet 1928 both use electrical, electromagnetic or optical signals that carry digital data streams. The signals through the various networks and the signals on network link 1920 and through communication interface 1918, which carry the digital data to and from computer system 1900, are exemplary forms of carrier waves transporting the information.

Computer system 1900 can send messages and receive data, including program code, through the network(s), network link 1920 and communication interface 1918. In the Internet example, a server 1930 might transmit a requested code for an application program through Internet 1928, ISP 1926, local network 1922 and communication

interface 1918. In accordance with the invention, one such downloaded application provides for collaborative communications, multiple-part messages, and linking and aggregating messages as described herein.

The received code may be executed by processor 1904 as it is received, and/or stored in storage device 1910, or other non-volatile storage for later execution. In this manner, computer system 1900 may obtain application code in the form of a carrier wave.

#### 4.0 EXTENSIONS AND ALTERNATIVES

In the foregoing specification, the invention has been described with reference to specific embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

---

APPENDIX 1—CLASS STRUCTURE AND API FOR EVENT HANDLING

Class Message

```
java.lang.Object
|
+--com.zaplet.message.Message
```

---

```
public class Message
extends java.lang.Object
```

Class for implementing Messaging API - responsible for message content

Contains all data, related to the event message along with get/set methods, providing access to this data The most proper way of using the Message class functionality would be to instantiate the class and then add proper attributes, i.e name / value pairs for this class:

```
=====

        Message msg = new Message(msgType, objID, objType,
senderID, senderType, expTime);

        // adding name/value pairs here
        msg.addAttr(AttrName, Object);

=====
```

**Since:**  
Java v1.1.8

**Version:**  
1.0


**Author:**  
Vlad Silverman

**See Also:**  
com.zaplet.db.SelectAttributesByMsgId

<pre><b>Message</b>(java.lang.String msgType, com.zaplet.data.ObjectID objID, java.lang.String objType, com.zaplet.data.ObjectID senderID, java.lang.String senderType, java.util.Date expTime) Constructor Used to initialize a Message</pre>
--

void	<b>addAttr</b> (java.lang.String name, java.lang.Object value) addAttr() - adds attributes for the current message
java.util.Hashtable	<b>getAttr</b> () getAttr() - gets the m_attr class variable
java.util.Date	<b>getExpTime</b> () getExpTime() - gets the m_expTime class variable
com.zaplet.data.ObjectID	<b>getMsgID</b> () getMsgID() - gets the m_msgID class variable
java.lang.String	<b>getMsgType</b> () getMsgType() - gets the m_msgType class variable
com.zaplet.data.ObjectID	<b>getObjID</b> () getObjID() - gets the m_objID class variable
java.lang.String	<b>getObjType</b> () getObjType() - gets the m_objType class variable
com.zaplet.data.ObjectID	<b>getSenderID</b> () getSenderID() - gets the m_senderID class variable
java.lang.String	<b>getSenderType</b> () getSenderType() - gets the m_senderType class variable
int	<b>getStatus</b> () getStatus() - gets the m_status class variable
boolean	<b>isPersistent</b> () isPersistent() - gets the m_persistent class variable
static void	<b>main</b> (java.lang.String [] args) main() - main method provides functionality for unit testing of Message class
void	<b>setAttr</b> (java.util.Hashtable attr) setAttr() - sets the m_attr class variable
void	<b>setExpTime</b> (java.util.Date expTime) setExpTime() - sets the m_expTime class variable
void	<b>setMsgID</b> (com.zaplet.data.ObjectID msgID) setMsgID() - sets the m_msgID class variable
void	<b>setMsgType</b> (java.lang.String msgType) setMsgType() - sets the m_msgType class variable
void	<b>setObjID</b> (com.zaplet.data.ObjectID objID) setObjID() - sets the m_objID class variable
void	<b>setObjType</b> (java.lang.String objType) setObjType() - sets the m_objType class variable
void	<b>setPersistent</b> (boolean persistent) setPersistent() - sets the m_persistent class variable
void	<b>setSenderID</b> (com.zaplet.data.ObjectID senderID) setSenderID() - sets the m_senderID class variable
void	<b>setSenderType</b> (java.lang.String senderType) setSenderType() - sets the m_senderType class variable
void	<b>setStatus</b> (int status) setStatus() - sets the m_status class variable

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

  
**Message**

```
public Message(java.lang.String msgType,  
               com.zaplet.data.ObjectID objID,  
               java.lang.String objType,  
               com.zaplet.data.ObjectID senderID,  
               java.lang.String senderType,  
               java.util.Date expTime)
```

Constructor Used to initialize a Message

**Parameters:**

msgType -- String value of the Message type

objID -- int value of the object Id

objType -- String value of the object type

This value should be part of names, defined in ObjectType class

senderID -- value of the sender Id

senderType -- String value of the sender type

This value should be part of names, defined in ObjectType class

expTime -- expiration time for the current message

  
**setMsgID**

```
public void setMsgID(com.zaplet.data.ObjectID msgID)  
setMsgID() - sets the m_msgID class variable
```

**Parameters:**

msgID -- the value to be set

---

**getMsgID**

```
public com.zaplet.data.ObjectID getMsgID()  
getMsgID() - gets the m_msgID class variable
```

**Returns:**

msgID - the value of the class variable

---

**setMsgType**

```
public void setMsgType(java.lang.String msgType)  
setMsgType() - sets the m_msgType class variable
```

**Parameters:**

msgType -- the value to be set



---

**getMsgType**

```
public java.lang.String getMsgType()
```

getMsgType() - gets the m\_msgType class variable

**Returns:**

msgType - the value of the class variable

---

**setObjID**

```
public void setObjID(com.zaplet.data.ObjectID objID)
```

setObjID() - sets the m\_objID class variable

**Parameters:**

obj ID -- the value to be set

---

**getObjID**

```
public com.zaplet.data.ObjectID getObjID()
```

getObjID() - gets the m\_objID class variable

**Returns:**

objID - the value of the class variable

---

**setObjType**

```
public void setObjType(java.lang.String objType)
```

setObjType() - sets the m\_objType class variable

**Parameters:**

objType -- the value to be set

---

**getObjType**

```
public java.lang.String getObjType()
```

getObjType() - gets the m\_objType class variable

**Returns:**

objType - the value of the class variable

---

**setSenderID**

```
public void setSenderID(com.zaplet.data.ObjectID senderID)
```

setSenderID() - sets the m\_senderID class variable

**Parameters:**

obj ID -- the value to be set

---

**getSenderID**

```
public com.zaplet.data.ObjectID getSenderID()
```

getSenderID() - gets the m\_senderID class variable

**Returns:**

senderID - the value of the class variable

---

**setSenderType**

public void **setSenderType**(java.lang.String senderType)

setSenderType() - sets the m\_senderType class variable

**Parameters:**

senderType -- the value to be set

---

**getSenderType**

public java.lang.String **getSenderType**()

getSenderType() - gets the m\_senderType class variable

**Returns:**

senderType - the value of the class variable

---

**setStatus**

public void **setStatus**(int status)

setStatus() - sets the m\_status class variable

**Parameters:**

status -- the value to be set

---

**getStatus**

public int **getStatus**()

getStatus() - gets the m\_status class variable

**Returns:**

status - the value of the class variable

---

**setExpTime**

public void **setExpTime**(java.util.Date expTime)

setExpTime() - sets the m\_expTime class variable

**Parameters:**

expTime -- the value to be set

---

**getExpTime**

public java.util.Date **getExpTime**()

getExpTime() - gets the m\_expTime class variable

**Returns:**

expTime - the value of the class variable

---

**isPersistent**

```
public boolean isPersistent()
isPersistent() - gets the m_persistent class variable
```

**Returns:**

m\_persistent - the value of the class variable

---

**setPersistent**

```
public void setPersistent(boolean persistent)
setPersistent() - sets the m_persistent class variable
```

**Parameters:**

m\_persistent -- the value to be set

---

**setAttr**

```
public void setAttr(java.util.Hashtable attr)
setAttr() - sets the m_attr class variable
```

**Parameters:**

attr -- the value to be set

---

**getAttr**

```
public java.util.Hashtable getAttr()
getAttr() - gets the m_attr class variable
```

**Returns:**

attr - the value of the class variable

---

**addAttr**

```
public void addAttr(java.lang.String name,
                    java.lang.Object value)
                    throws MessageException
```

addAttr() - adds attributes for the current message

**Parameters:**

name - the name of the attribute to be added to the name/value list

value - the value of the attribute to be added to the name/value list

**Throws:**

MessageException -- in case of null parameters

---

**main**

```
public static void main(java.lang.String[] args)
main() - main method provides functionality for unit testing of Message class
```

**Class MessageService**java.lang.Object

|

+---com.zaplet.message.MessageService

public class MessageService

extends Object

Class for implementing Messaging API - responsible for services on Message data

Publishes/stores info about any object in the db

This class provides also transformation between internal objects used by the Zaplet platform and standard data types, used by JDBC layer

Possible ways of using the MessageService are outlined below:

=====

```

        Message msg = new Message(msgType, objID, objType,
senderID, senderType, expTime);

```

```

try {

```

```

    // adding name/value pairs here

```

```

    msg.addAttribute(AttrName1, Object1);

```

```

    msg.addAttribute(AttrName2, Object2);

```

```

    ....

```

```

// Now there are several choices:

```

```

//

```

```

// 1. we can just store the message and

```

```

// all its attributes in the database

```

```

    MessageService.publish(msg);

```

```

    ....

```

```

        OR

```

```

// 2. we can publish the message in the Db and

```

```

// fire the action, associated with the message

```

```

// The action should be implemented in the handle()

```

```

// method of the class, specified by 'msgType'

```

```

    MessageService.publishAndFire(msg);

```

```

    ....

```

```

        OR

```



static void	<b>main</b> (String[] args) main() - main method provides functionality for unit testing of MessageService class
static void	<b>publish</b> (Message msg) publish() - stores a new Message this method will verify first m_Persistent flag of the Message object and store the Message in the Db only if this flag is true
static void	<b>publishAndFire</b> (Message msg) publishAndFire() - activates the handle() method of the specific handler. Handler is associated with the type of the current message.

, clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait



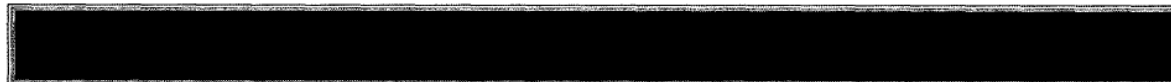
**SEQUENCEQUERY**

public static final String SEQUENCEQUERY



**MessageService**

public MessageService()



**publish**

public static void **publish**(Message msg)  
 throws MessageException

publish() - stores a new Message

this method will verify first m\_Persistent flag of the Message object and store the Message in the Db only if this flag is true

**Parameters:**

msg -- a Message object

**Throws:**

MessageException -- in case of one of db transactions failed

**insertMessage**

private static int **insertMessage**(DbTrans trans,  
Message msg)

insertMessage(msg) - inserts a message in the database

**Parameters:**

trans -- database transaction

msg -- a Message object

**Returns:**

inserted number of rows

---

**insertAttributes**

```
private static int insertAttributes(DbTrans trans,
                                   Message msg)
```

insertAttribute(msg) - inserts all attributes (name/value pairs) of a message in the database

**Parameters:**

trans -- database transaction

msg -- a Message object

**Returns:**

inserted number of rows

---

**publishAndFire**

```
public static void publishAndFire(Message msg)
                               throws MessageException
```

publishAndFire() - activates the handle() method of the specific handler.

Handler is associated with the type of the current message. Before calling the handler this method will store the Message object in the database

**Parameters:**

msg -- the Message object

**Throws:**

MessageException -- in case the event message can't be fired. Possible reason for this - the handle can't be found

---

**getHandler**

```
protected static MessageHandler getHandler(String msgType)
                                       throws MessageException
```

getHandler() - get the name of the class, which should handle the current message

The name of the class is related to the type of the message

**Parameters:**

msgType -- the type of the message

**Returns:**

MessageHandler - returns an object which implements MessageHandler interface

**Throws:**

MessageException -- in case the handler can't be found

---

**getUniqueId**

```
protected static int getUniqueId(String query)
                               throws MessageException
```

Get unique primary key id for an object, specified in the query parameter. Use the database sequencer

**Parameters:**

query -- a SQL query string ot get next id

**Returns:**

int - the next id number

**Throws:**

MessageException -- in case of DB communication failure

---

**main**

```
public static void main(String[] args)
                       throws Exception
```

main() - main method provides functionality for unit testing of MessageService class

This procedure instantiates a new message and sets message type of the current message to the HandlerTest class located in com.zaplet.message package Three attributes of different types are added to the current message After fire() method will be called on the current message the handle() method of the HandlerTest class will be executed This method is implemented just for testing purposes It will print out the names, values and types of all attributes, associated with the current message

**Class SystemHandler**

```
java.lang.Object
|
+--com.zaplet.message.SystemHandler
```

---

```
public class SystemHandler
```

```
extends java.lang.Object
```

```
implements MessageHandler
```

Class for implementing Messaging API Contains handle method

**Since:**

Java v1.1.8

**Version:**

1.0

**See Also:**

Message



	<p><b>SystemHandler()</b>                  Constructor Used to initialize a SystemHandler</p>

void	<p><b>handle(Message msg)</b>                  handle() gets the list of all observers of the current message, i.e gets the list of all messages, associated in the current message for every associated message activates the fire method, which in turn: stores associated message if it is persistent activates the handle of the associated message</p>

<p>clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait</p>	

--	--

**SystemHandler**

public **SystemHandler()**  
 Constructor Used to initialize a SystemHandler

--	--

**handle**

public void **handle(Message msg)**  
 handle() gets the list of all observers of the current message, i.e gets the list of all messages, associated in the current message for every associated message activates the fire method, which in turn: stores associated message if it is persistent activates the handle of the associated message

**Specified by:**

handle in interface MessageHandler

**Parameters:**

msg - Message object

**Interface MessageHandler**

**All Known Implementing Classes:**

SystemHandler

---

public interface **MessageHandler**  
 Interface for Messaging API Contains methods to be implemented by every specific Handler class

**Since:**

Java v1.1.8

**Version:**

1.0

**See Also:**

`com.zaplet.db.SelectAttributesByMsgId`

---

void	<b>handle</b> ( <u>Message</u> msg) handle method



**handle**

public void **handle** (Message msg)

handle method

**Parameters:**

msg -- a Message object

## CLAIMS

What is claimed is:

1. A method for processing a request to display an electronic message, the method comprising the computer-implemented steps of:
  - generating first message data, wherein the first message data defines at least a first message portion and one or more selection regions for one or more other message portions of a multiple-part electronic message having a plurality of message portions;
  - providing the first message data to a client;
  - receiving from the client a request for a second portion of the electronic message selected from among the other message portions;
  - generating second message data which, when processed at a user interface of the client, causes the client to display the second portion of the electronic message; and
  - providing the second message data to the client.
2. A method as recited in Claim 1, wherein generating first message data further comprises the steps of generating first message data that defines a plurality of message portions each having a corresponding selection region and that defines a message user interface region that comprises all corresponding selection regions and the first message portion.
3. A method as recited in Claim 1, wherein the first message data includes user interface definition data which, when processed at the user interface, causes the user interface to display the first portion of the electronic message in a first panel.
4. A method as recited in Claim 3, wherein the user interface definition data includes data which, when processed at the user interface, causes the user interface to display a first identifier of the first portion of the electronic message in the first panel.
5. A method as recited in Claim 4, wherein the first identifier indicates content of the first portion of the electronic message.

6. A method as recited in Claim 3, wherein generating the first message data and generating the second message data further comprises the steps of generating second user interface definition data which, when processed at the user interface, causes the user interface to display a plurality of continuously visible selection regions, each associated with a different portion of the multiple-part message and the first portion of the electronic message.

7. A method as recited in Claim 6, wherein the second user interface definition data comprises data which, when processed at the user interface, causes one or more other identifiers to be displayed in association with the continuously visible selection regions to identify corresponding portions of the electronic message.

8. A method as recited in Claim 1, wherein the first message data comprises selection region definition data which, when processed at the user interface of the client, causes the client to display a plurality of selection regions that extend outwardly laterally from the first portion of the electronic message.

9. A method as recited in Claim 1, wherein the first message data comprises selection region definition data which, when processed at the user interface of the client, causes the client to display a plurality of selection regions that extend outwardly upwardly from the first portion of the electronic message.

10. A method as recited in Claim 1, wherein the second message data comprises selection region definition data which, when processed at the user interface of the client, causes the client to display a plurality of selection regions that extend outwardly downwardly from the first portion of the electronic message.

11. A method as recited in Claim 1, wherein the first message data and the second message data comprise one or more hypertext markup language (HTML) instructions.

12. A method as recited in Claim 1, wherein the first message data comprises selection region definition data which, when processed at the user interface of the client, causes the client to display a toolbar of functions for manipulating the multiple-part electronic message within a user interface panel that contains the first message portion.

13. A method as recited in Claim 1, wherein the first message data further comprises one or more executable application building blocks, and further comprising the steps of:

executing the one or more application building blocks to result in creating and storing one or more then-current dynamic data values as part of the first message portion;  
providing the one or more dynamic data values to the client as part of the first message portion.

14. A method as recited in Claim 1, further comprising the steps of:  
retrieving one or more then-current dynamic data values from a database;  
rendering the dynamic data values as part of the first message portion;  
providing the one or more dynamic data values to the client as part of the first message portion.

15. A method as recited in Claim 1, wherein the step of generating first message data comprises the steps of generating first message data that defines at least a first message page, one or more selection regions for one or more other message pages of a multiple-page electronic message having a plurality of message pages, and a plurality of sub-pages of the first message page.

16. A method as recited in Claim 15, further comprising the steps of:  
receiving a selection of a sub-page of the first message page;  
generating third message data that defines the selected sub-page of the first message page and which, when processed at the user interface, causes the user interface to display the selected sub-page of the electronic message;  
providing the third message data to the client.

17. A method providing a multiple-part electronic message, the method comprising the computer-implemented steps of:

- generating first message data that defines a multiple-part electronic message and includes at least a first message portion and one or more selection regions for one or more other associated message portions;
- providing the first message data to a first client;
- receiving a request to forward the multiple-part electronic message to a recipient;
- in response to receiving the request, generating second message data to the recipient that defines the multiple-part electronic message; and
- providing the second message data to the second client.

18. A method of asynchronously dynamically updating information of a multiple-part electronic message, the method comprising the computer-implemented steps of:

- generating first message data, wherein the first message data defines at least a first message portion having a dynamic content region and one or more selection regions for one or more other message portions of a multiple-part electronic message having a plurality of message portions;
- providing the first message data to a first client;
- receiving one or more asynchronous updates to the dynamic content region;
- generating second message data that defines the first message portion, the dynamic content region including the one or more updates, and the one or more selection regions; and
- providing the second message data to a second client.

19. A method as recited in Claim 18, further comprising the steps of:

- receiving a selection of a second portion of the electronic message selected from among the other message portions;
- retrieving then-current dynamic content for a second dynamic content region of the second portion of the electronic message;
- generating third message data that defines the second portion of the electronic message and that includes the then-current dynamic content for the second dynamic content region;

· providing the third message data to the client.

20. A method for processing data at a user interface comprising the computer-implemented steps of:

receiving a request to display an electronic message;

in response to receiving the request to display an electronic message, requesting a first portion of an electronic message;

receiving first message data; and

processing the first message data to cause the first portion of the electronic message to be displayed on the user interface.

21. A method as recited in Claim 20, further comprising the computer-implemented steps of:

receiving a request to display a second portion of the electronic message;

in response to the request to display a second portion of the electronic message, requesting the second portion of the electronic message;

receiving second message data; and

processing the second message data to cause the second portion of the electronic message to be displayed on the user interface.

22. A method as recited in Claim 20, further comprising the computer-implemented steps of:

receiving user interface object data; and

processing the user interface object data to cause a user interface object to be displayed on the user interface;

and wherein the step of receiving a request to display a second portion of the electronic mail message receiving second message data includes detecting user manipulation of the user interface object.

23. A method as recited in Claim 20, wherein the method further comprises the computer-implemented steps of:

receiving user interface object data; and

processing the user interface object data to cause a user interface object to be displayed in association with a second portion of the electronic mail message that is not displayed concurrently with the first portion of the electronic mail message.

24. A method as recited in Claim 20, wherein the first portion of the electronic mail message is displayed on a panel.

25. A data processing apparatus comprising:  
a memory device configured to store electronic message data;  
a processor communicatively coupled to the memory device; and  
one or more sequences of instructions in the memory device which, when executed by the processor, cause the processor to carry out the steps of:  
generating first message data, wherein the first message data defines at least a first message portion and one or more selection regions for one or more other message portions of a multiple-part electronic message having a plurality of message portions;  
providing the first message data to a client;  
receiving from the client a request for a second portion of the electronic message selected from among the other message portions;  
generating second message data which, when processed at a user interface of the client, causes the client to display the second portion of the electronic message; and  
providing the second message data to the client.

26. An apparatus for processing a request to display an electronic message, comprising:  
means for generating first message data, wherein the first message data defines at least a first message portion and one or more selection regions for one or more other message portions of a multiple-part electronic message having a plurality of message portions;  
means for providing the first message data to a client;  
means for receiving from the client a request for a second portion of the electronic message selected from among the other message portions;  
means for generating second message data which, when processed at a user interface of the client, causes the client to display the second portion of the electronic message; and



means for providing the second message data to the client.

27. A computer-readable medium comprising one or more sequences of instructions for processing a request to display an electronic message, which instructions, when executed by one or more processors, cause the one or more processors to carry out the steps of:

- generating first message data, wherein the first message data defines at least a first message portion and one or more selection regions for one or more other message portions of a multiple-part electronic message having a plurality of message portions;
- providing the first message data to a client;
- receiving from the client a request for a second portion of the electronic message selected from among the other message portions;
- generating second message data which, when processed at a user interface of the client, causes the client to display the second portion of the electronic message; and
- providing the second message data to the client.

28. A method for associating related electronic messages in computer storage, the method comprising the computer-implemented steps of:

- creating and storing a first transportable application;
- receiving user input requesting creation of a link from the first transportable application to another transportable application;
- receiving user input that selects a second transportable application from among a plurality of previously created transportable applications; and
- creating and storing a link from the first transportable application to the second transportable application.

29. A method as recited in claim 28, wherein the step of creating and storing a link comprises the steps of:

- creating and storing an asynchronously dynamically updated list of references to other transportable applications in association with the first transportable application;
- creating and storing a reference to the second transportable application in the list of references.

30. A method as recited in claim 28, wherein the step of creating and storing a link comprises the steps of:

creating and storing an asynchronously dynamically updated List building block in association with the first transportable application;

creating and storing a reference to the second transportable application in the list building block.

31. A method as recited in Claim 28, wherein the step of receiving user input that selects a second transportable application comprises the steps of receiving user input that copies a URL of the second transportable application and receiving user input that pastes the URL into the first transportable application in a region associated with the list.

32. A method as recited in Claim 28, wherein the step of receiving user input that selects a second transportable application comprises the steps of receiving user input that drags a representation of the second transportable application into the first transportable application in a region associated with the list.

33. A method as recited in Claim 28, further comprising the steps of applying one or more access controls to the link, wherein the access controls specify that one or more users or groups may not access the second transportable application using the link.

34. A method of associating related electronic messages in computer storage, the method comprising the computer-implemented steps of:  
creating and storing a first transportable application;  
automatically creating and storing a second transportable application as a result of a workflow process or event associated with the first transportable application; and  
creating and storing a link from the first transportable application to the second transportable application.

35. A method as recited in claim 34, wherein the step of creating and storing a link comprises the steps of:

creating and storing an asynchronously dynamically updated list of references to other transportable applications in association with the first transportable application;

creating and storing a reference to the second transportable application in the list of references.

36. A method as recited in claim 34, wherein the step of creating and storing a link comprises the steps of:

creating and storing an asynchronously dynamically updated List building block in association with the first transportable application;

creating and storing a reference to the second transportable application in the list building block.

37. A method as recited in Claim 34, further comprising the steps of prompting a user associated with the first transportable application to confirm whether to link the first transportable application to the second transportable application; and

carrying out the step of creating and storing a link only in response to receiving user input that confirms that the first transportable application should link to the second transportable application.

38. A method as recited in Claim 34, further comprising the steps of applying one or more access controls to the link, wherein the access controls specify that one or more users or groups may not access the second transportable application using the link.

39. A method for associating related electronic messages in computer storage, the method comprising the computer-implemented steps of:

creating and storing a first transportable application;

creating and storing a link from the first transportable application to a second transportable application;

determining that a programmatic object associated with the first transportable application is new, updated or deleted;

in response thereto, modifying the link in accordance with the new, updated or deleted object.

40. A method as recited in Claim 39, wherein the step of modifying the link in response to an updated object comprises the steps of identifying all other transportable applications that are linked to the first transportable application and that reference the updated object, and modifying all references to the updated object.

41. A method as recited in Claim 39, wherein the object comprises a content element of the transportable application, and further comprising the steps of searching the content element for one or more recognizable object references, and creating one or more links relating to the recognizable object references in a list of automatically generated links.

42. A method as recited in Claim 41, wherein the object reference comprises an e-mail address or user name, and wherein the step of creating links relating to the object references comprises creating a mail link in the list which, when selected by a user, generates an e-mail message to the address or user name.

43. A method as recited in Claim 41, wherein the object reference comprises a Uniform Resource Locator, and wherein the step of creating links relating to the object references comprises creating a URL link in the list which, when selected by a user, generates a display of a hypertext document identified by the URL.

44. A method as recited in Claim 41, wherein the object reference comprises a title of a third transportable application, and wherein the step of creating links relating to the object references comprises creating a link in the list to the third transportable application.

45. A method of associating related electronic messages in computer storage, the method comprising the computer-implemented steps of:  
receiving user input associated with completing a task in a first transportable application;

generating a list of one or more other transportable applications that are likely to follow the first transportable application in a workflow or business process associated with the first transportable application, based on relevance of the other transportable applications to a context of the first transportable application;

requesting user input that specifies whether to link one or more of the other transportable applications to the first transportable application; and

creating and storing one or more links from the first transportable application to one or more of the other transportable applications.

46. A method of displaying a message that contains an embedded HTML document, comprising the computer-implemented steps of:

receiving a transportable application, which comprises an embedded HTML document, in an e-mail client application;

displaying the embedded HTML document in a graphical window of the e-mail client application;

displaying one or more graphical navigation buttons in association with the graphical window;

receiving user input that selects one or more of the graphical navigation buttons; and

displaying one or more other HTML documents in the graphical window in response to the user input.

47. A method as recited in Claim 46, wherein each of the graphical navigation buttons is associated with client-executable computer program code, and wherein the step of displaying one or more other HTML documents comprises the step of executing one or more instructions of the computer program code that are associated with one of the selected graphical navigation buttons that is selected by the user input.

48. A method as recited in Claim 46, wherein each of the graphical navigation buttons is associated with client-executable JavaScript code, and wherein the step of displaying one or more other HTML documents comprises the step of executing a portion of the JavaScript that is associated with one of the selected graphical navigation buttons that is selected by the user input.

49. A method of associating a plurality of sets of related electronic messages in computer storage, the method comprising the computer-implemented steps of:

- creating and storing a first set of a plurality of linked transportable applications;
- creating and storing a second set of a plurality of linked transportable applications;
- designating a first transportable application among the first set as a home transportable application for the first set;
- designating a second transportable application among the second set as a home transportable application for the second set;
- creating and storing, in association with the home transportable application of the first set, a next link that identifies the home transportable application of the second set;
- creating and storing, in association with the home transportable application of the second set, a previous link that identifies the home transportable application of the first set.

50. A method as recited in Claim 49, wherein the first set comprises a first message web, the second set comprises a second message web, and the links among the first message web and the second message web associate the first message web with the second message web in a message web ring.

51. A method as recited in Claim 49, further comprising the steps of:  
creating and storing a third set of a plurality of linked transportable applications having a third home transportable application;  
modifying the next link and the previous link of the first set and second set such that the third set of transportable application is logically inserted between the first set and the second set.

52. A method of generating a list of recipients for a first message that is linked to a second message, comprising the computer-implemented steps of:  
creating and storing a first transportable application that is linked to a second transportable application;  
creating a first recipient list in association with the first transportable application, wherein the first recipient list identifies one or more users or groups to whom the first transportable application is directed;  
creating a second recipient list in association with the second transportable application, wherein the second recipient list comprises at least one reference to the first recipient list;  
automatically resolving the at least one reference into a second list of one or more users or groups to whom the second transportable application is directed.

53. A method as recited in Claim 52, wherein the second recipient list further comprises at least one expression that identifies one or more users or groups to add or delete from the referenced first recipient list, and further comprising the steps of automatically determining a second list of one or more users or groups to whom the second transportable application is directed by resolving the at least one reference and applying the at least one expression.

54. A method of propagating data from a first message to a second message that is linked to the first message, comprising the computer-implemented steps of:  
creating and storing a first transportable application that is linked to a second transportable application;  
creating and storing a reference, in a first data field of the first transportable application, to a second data field of the second transportable application;  
determining that the first data field of the second transportable application is changed;  
automatically creating and storing the second data field of the second transportable application in the first data field of the first transportable application.

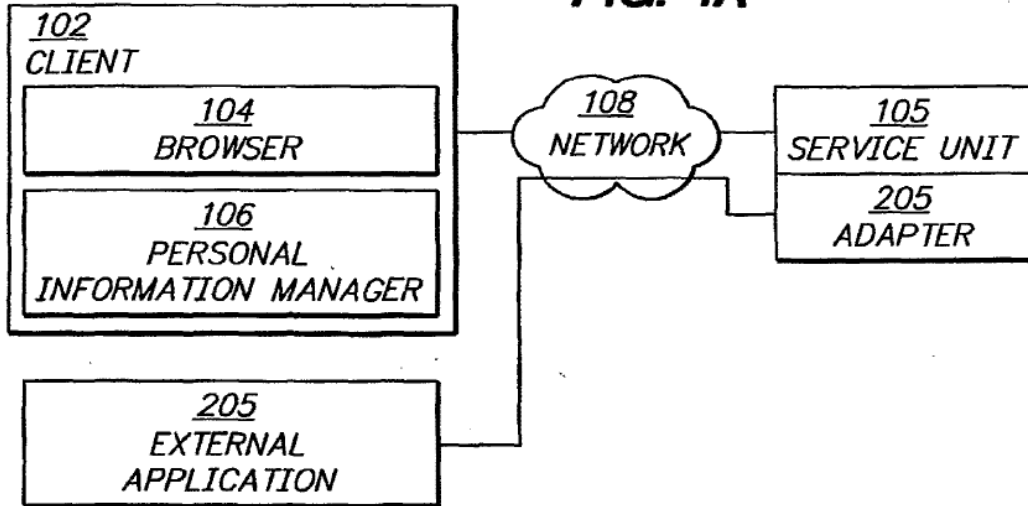
55. A method as recited in Claim 54, wherein the steps of determining and automatically creating comprise the steps of:  
determining whether the first data field of the first transportable application comprises a link to a second data field;  
verifying that the second data field comprises up-to-date data;  
retrieving data from the second data field;  
storing the retrieved data in the first data field.

56. A method as recited in Claim 55, further comprising the steps of:  
determining whether any other transportable applications are subscribed to the first data field;  
carrying out the step of automatically creating and storing the second data field only for each transportable application that is subscribed to the first data field.

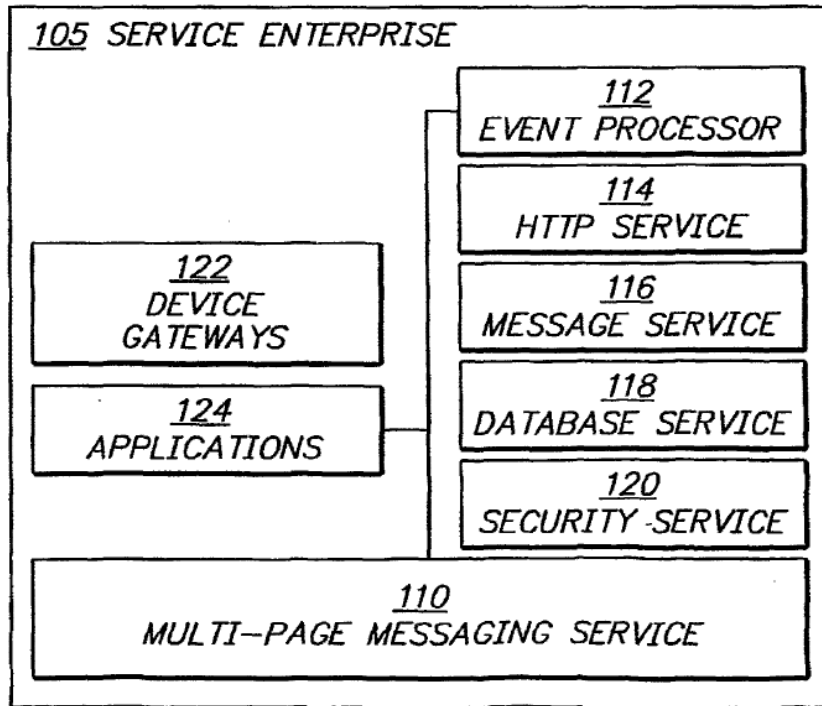


1/70

**FIG. 1A**

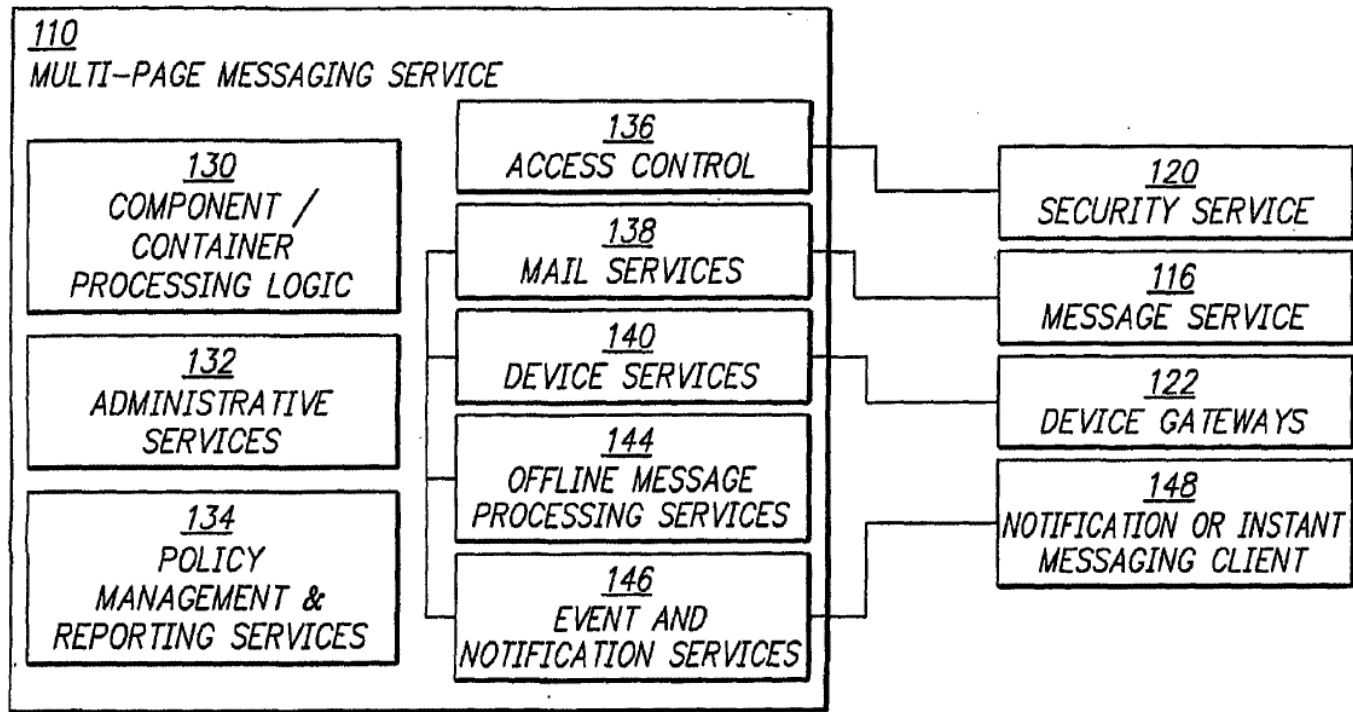


**FIG. 1B**



SUBSTITUTE SHEET (RULE 26)

FIG. 1C



SUBSTITUTE SHEET (RULE 26)

2/70

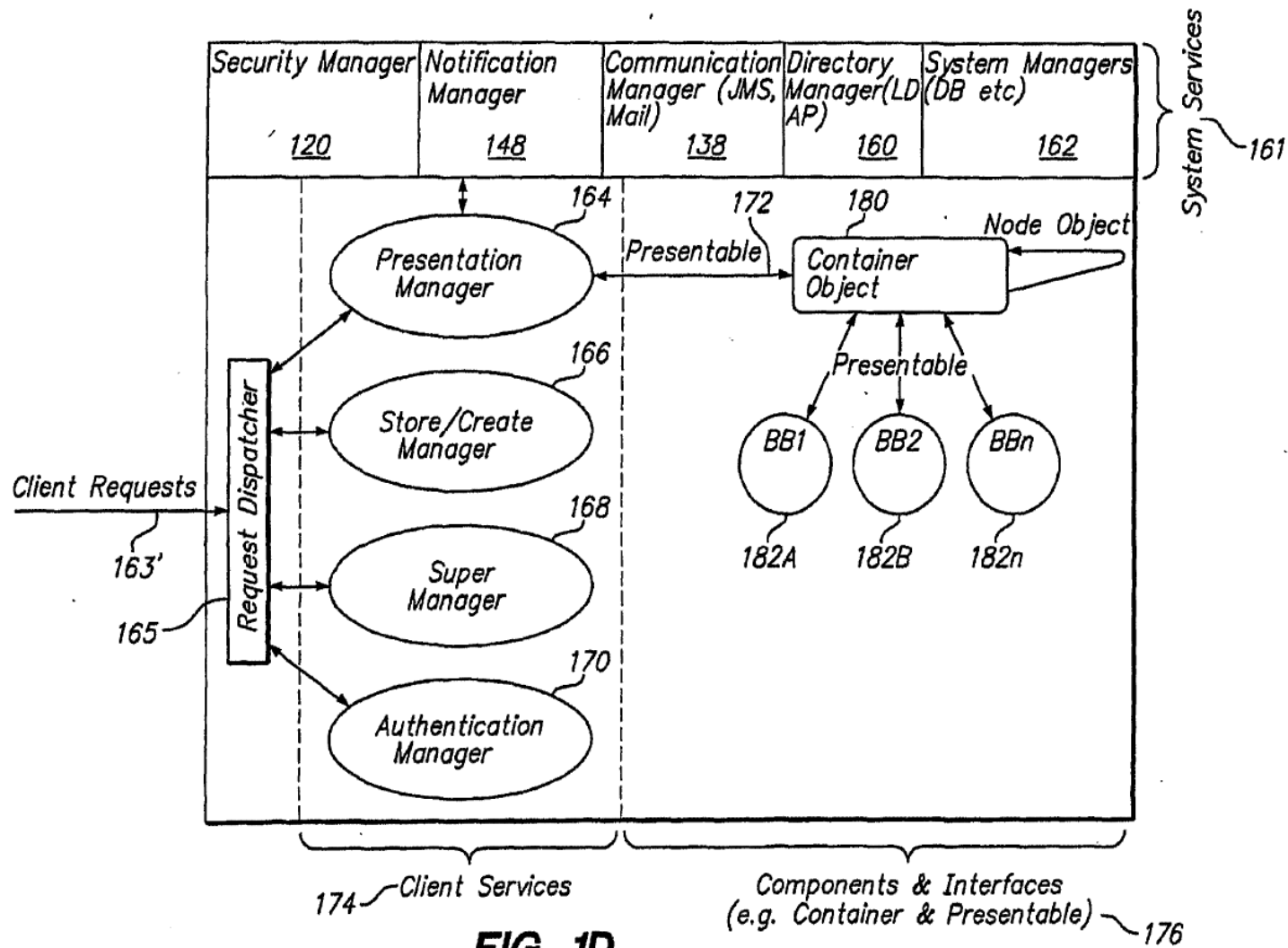
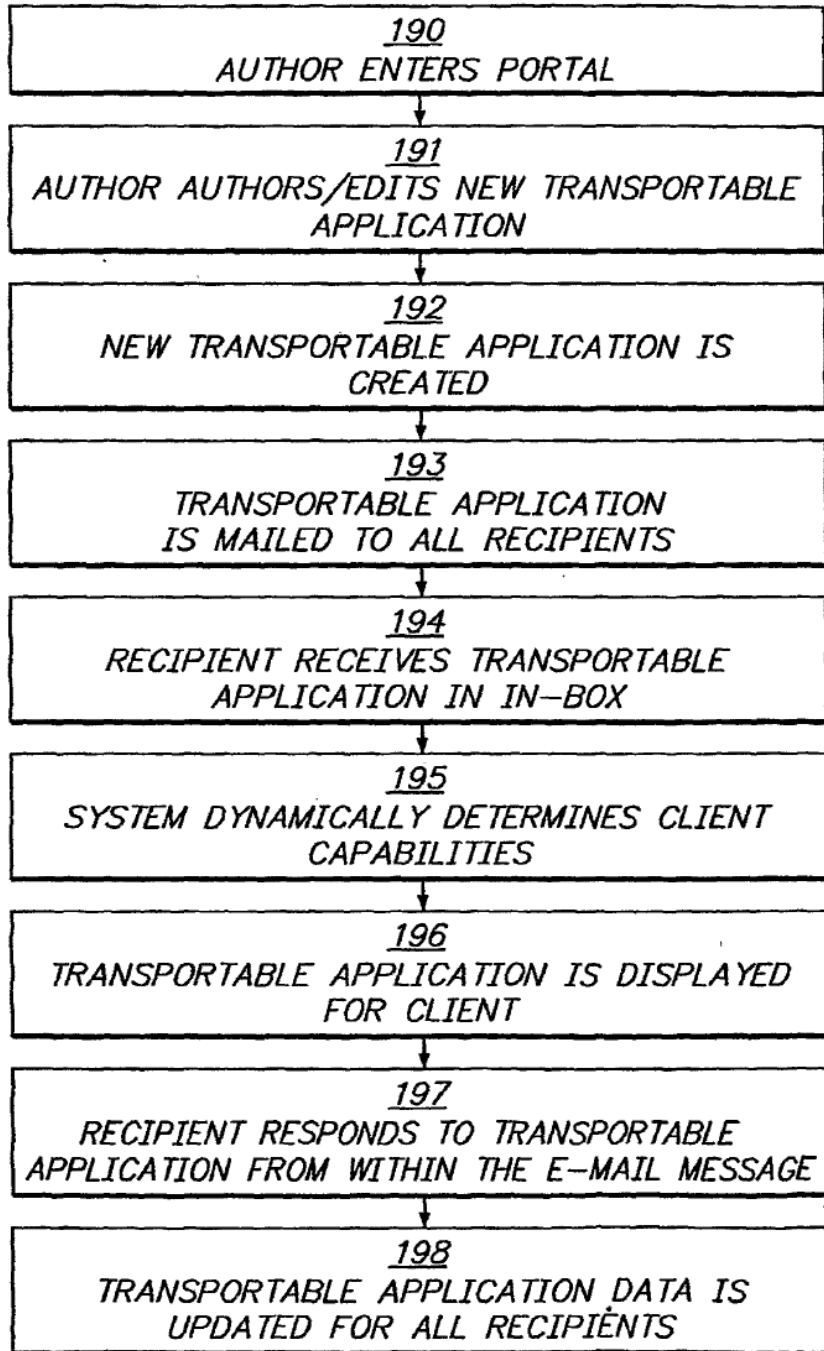


FIG. 1D

3/70

4/70

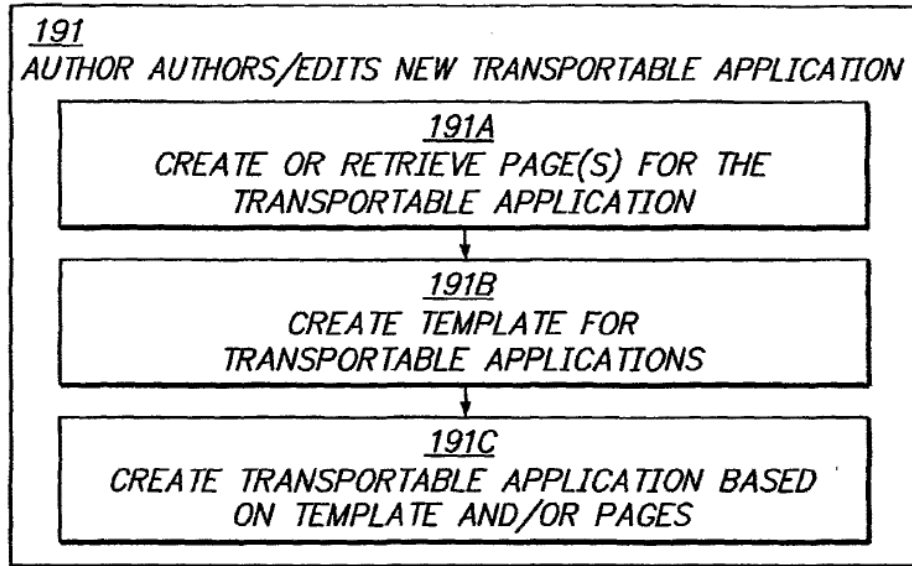
**FIG. 1E**



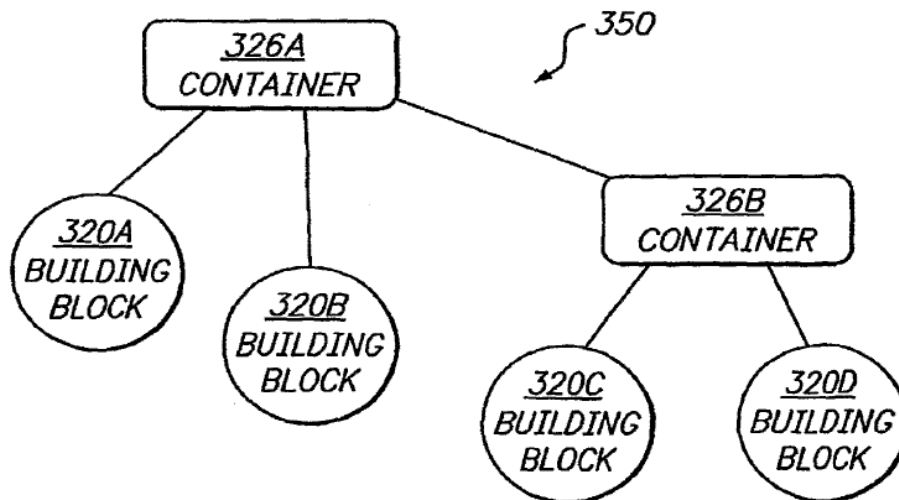
SUBSTITUTE SHEET (RULE 26)

5/70

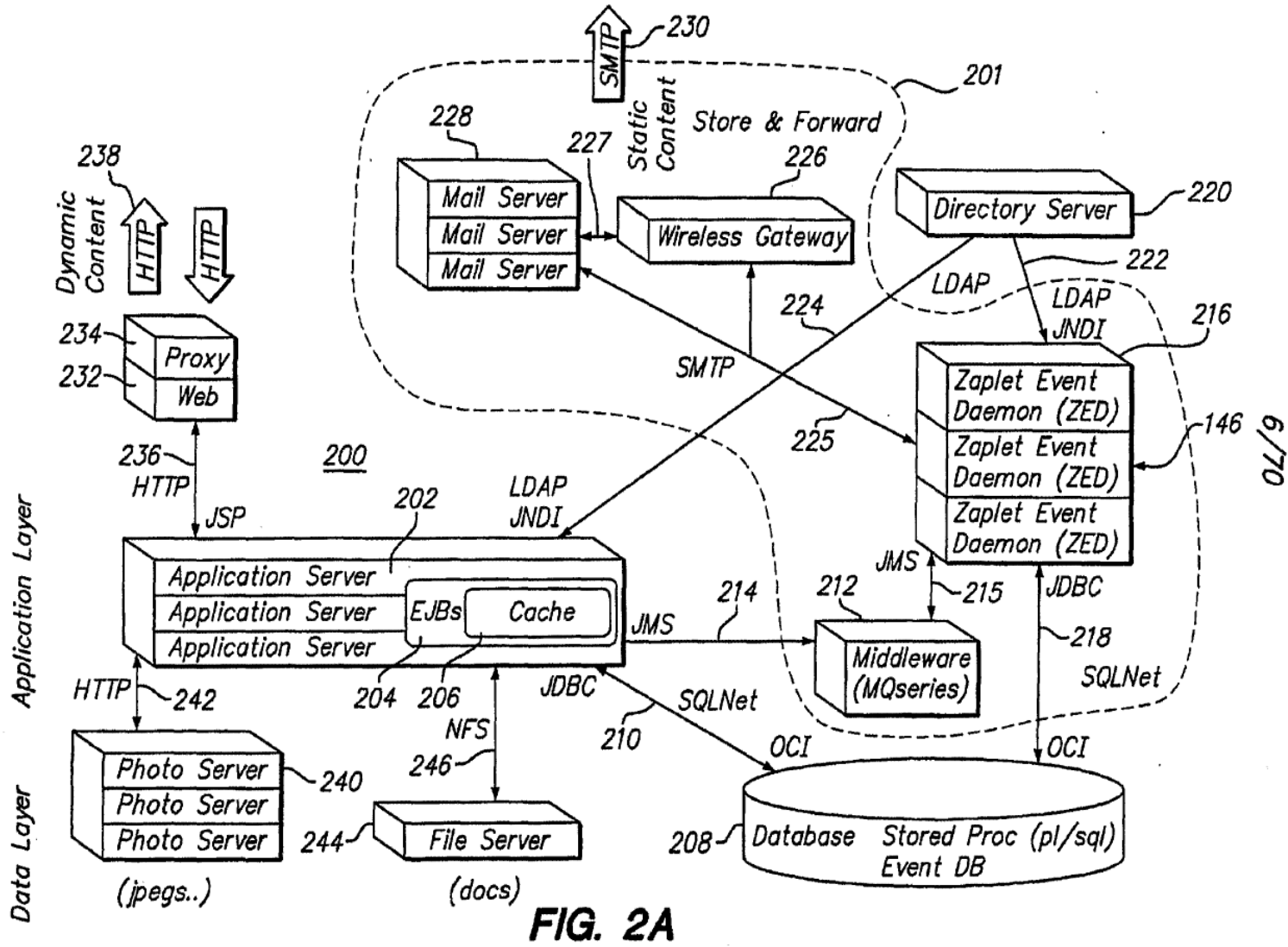
**FIG. 1F**



**FIG. 3B**



SUBSTITUTE SHEET (RULE 26)



**FIG. 2A**

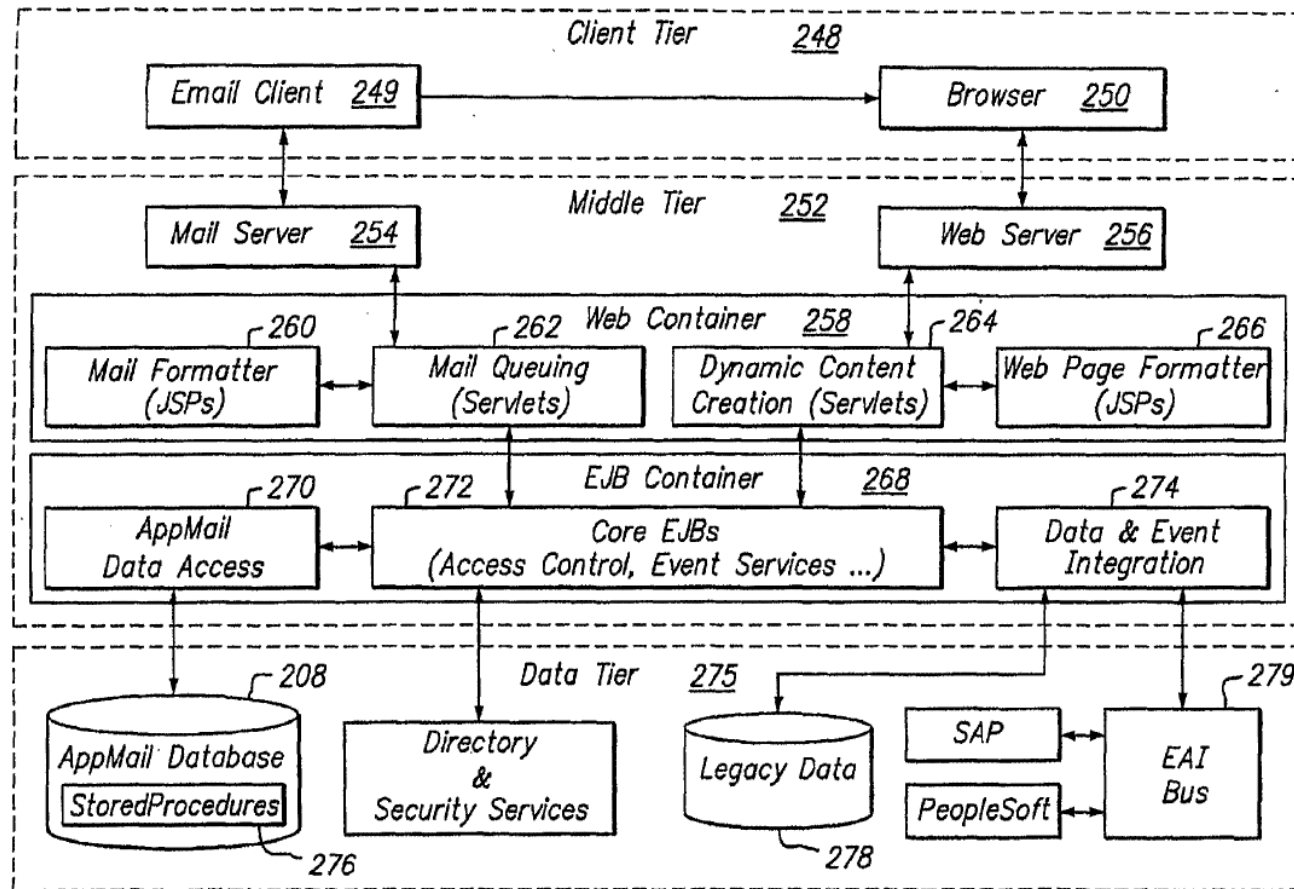


FIG. 2B

7/70

SUBSTITUTE SHEET (RULE 26)

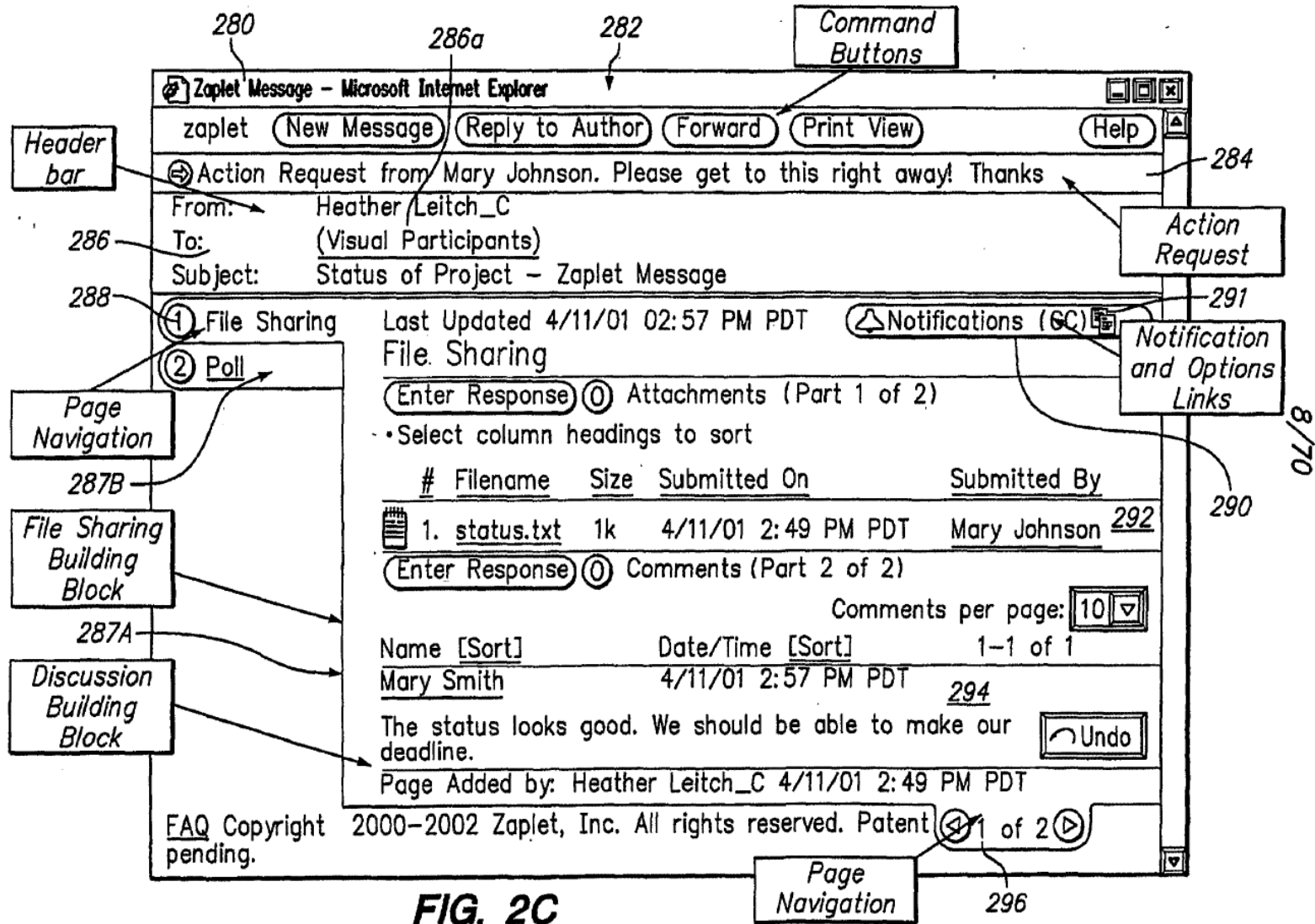


FIG. 2C



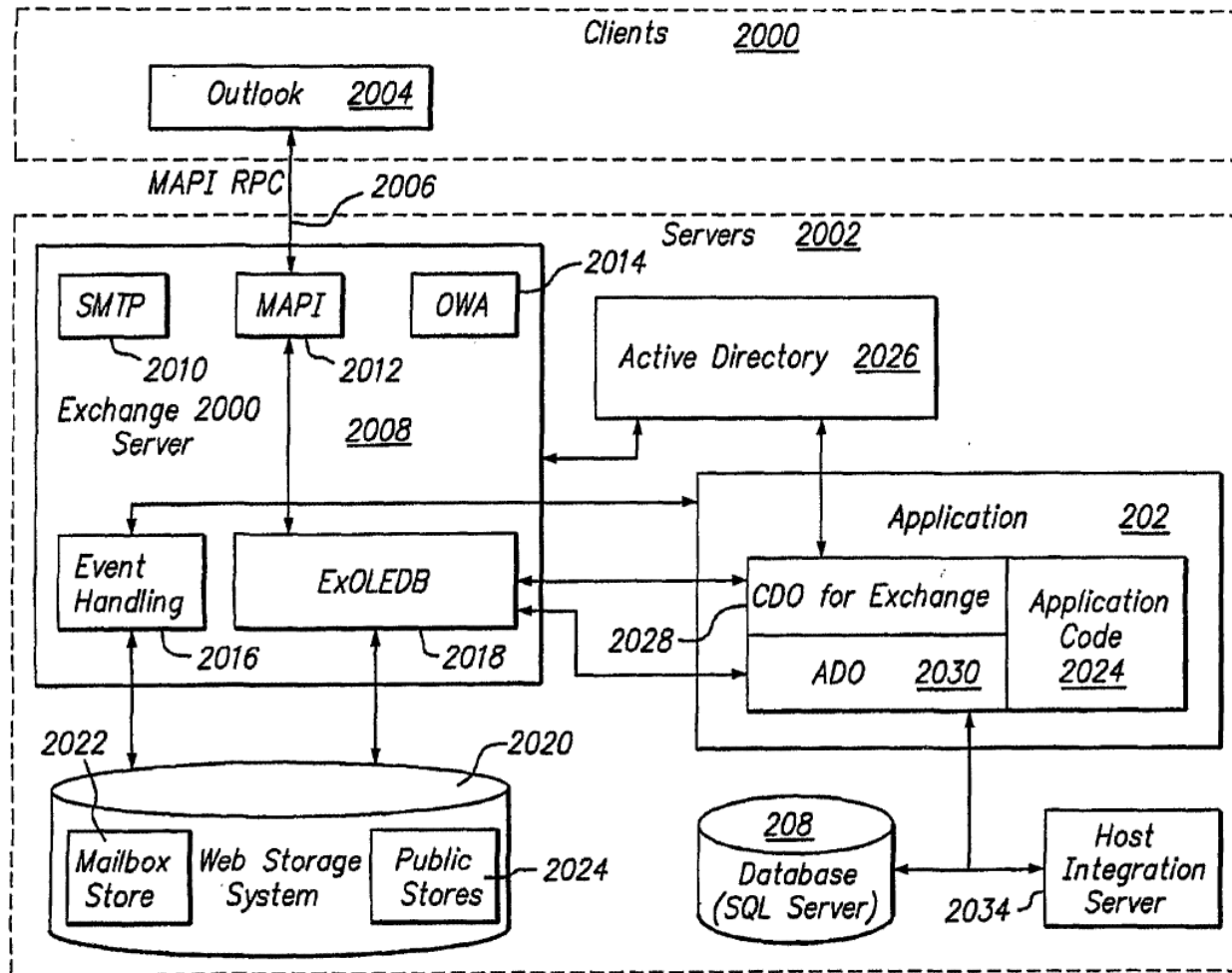


FIG. 2D

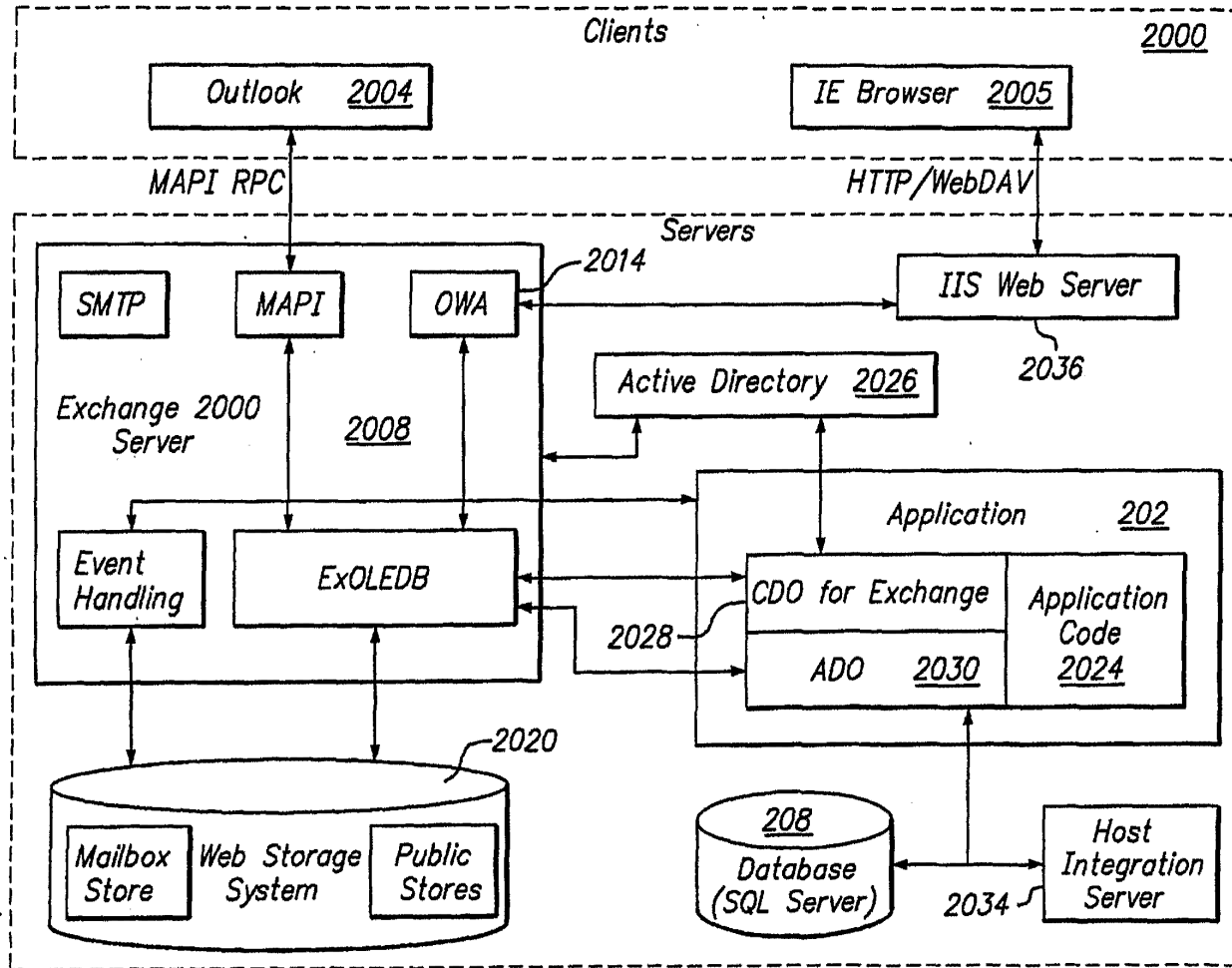


FIG. 2E

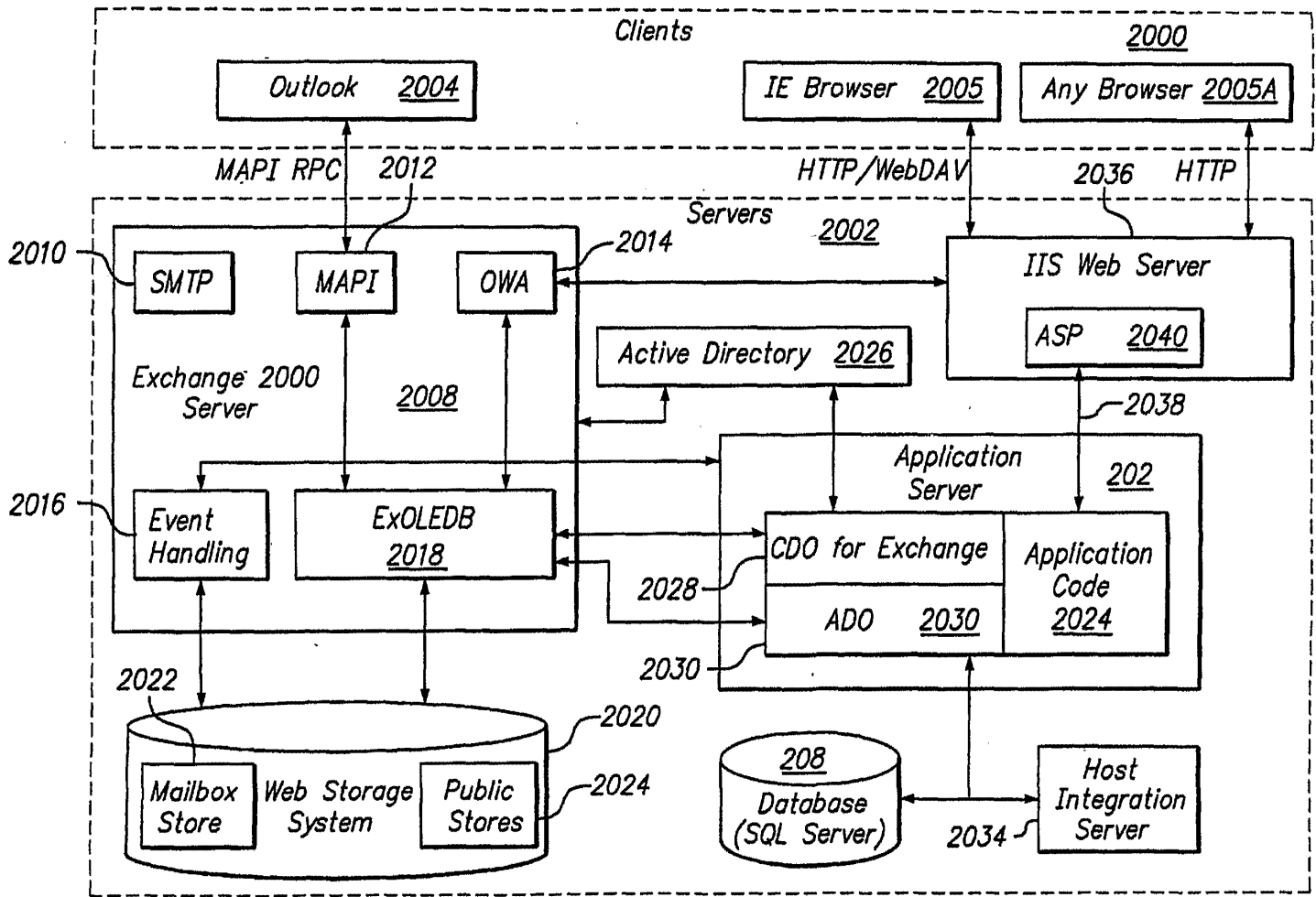
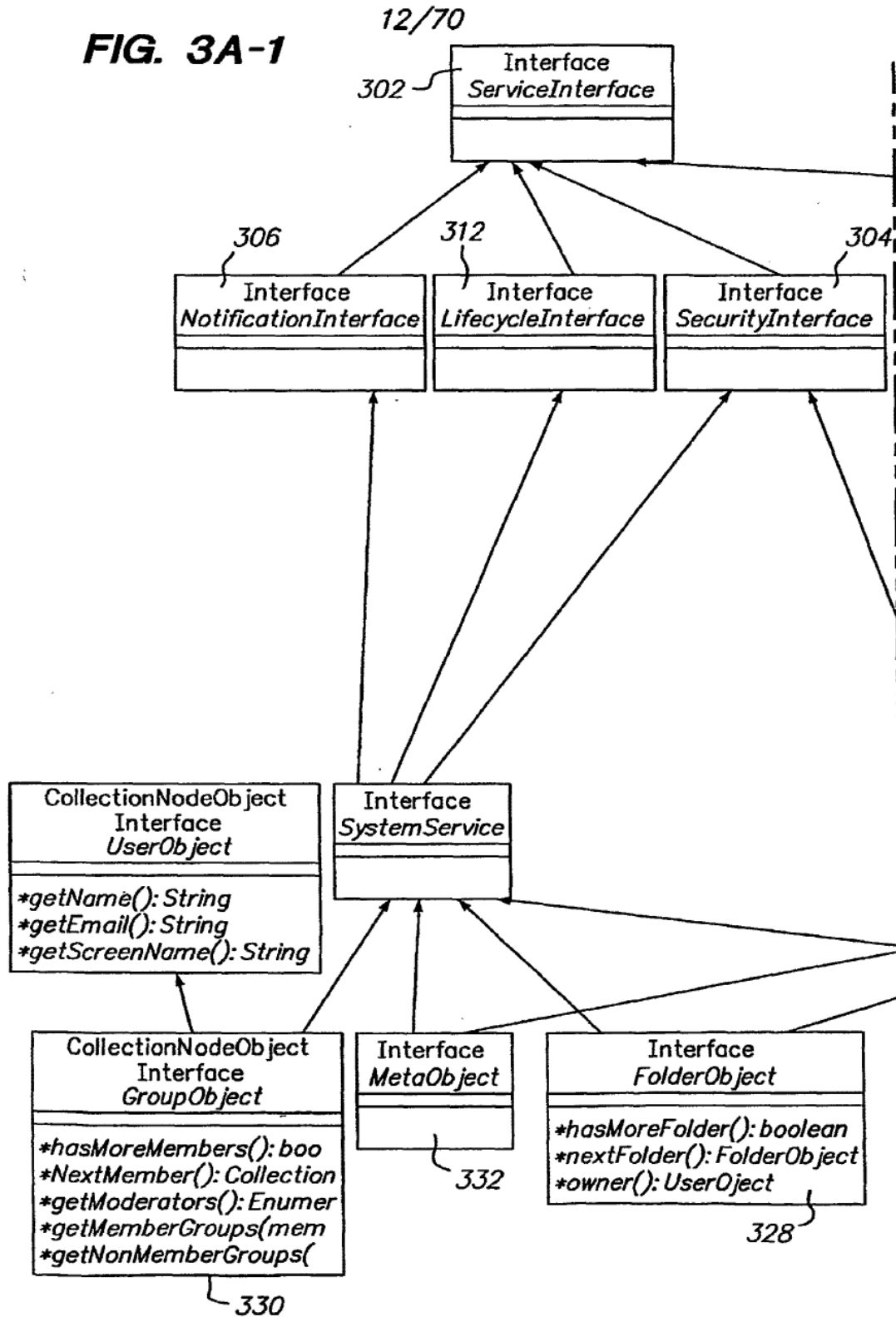


FIG. 2F

11/70

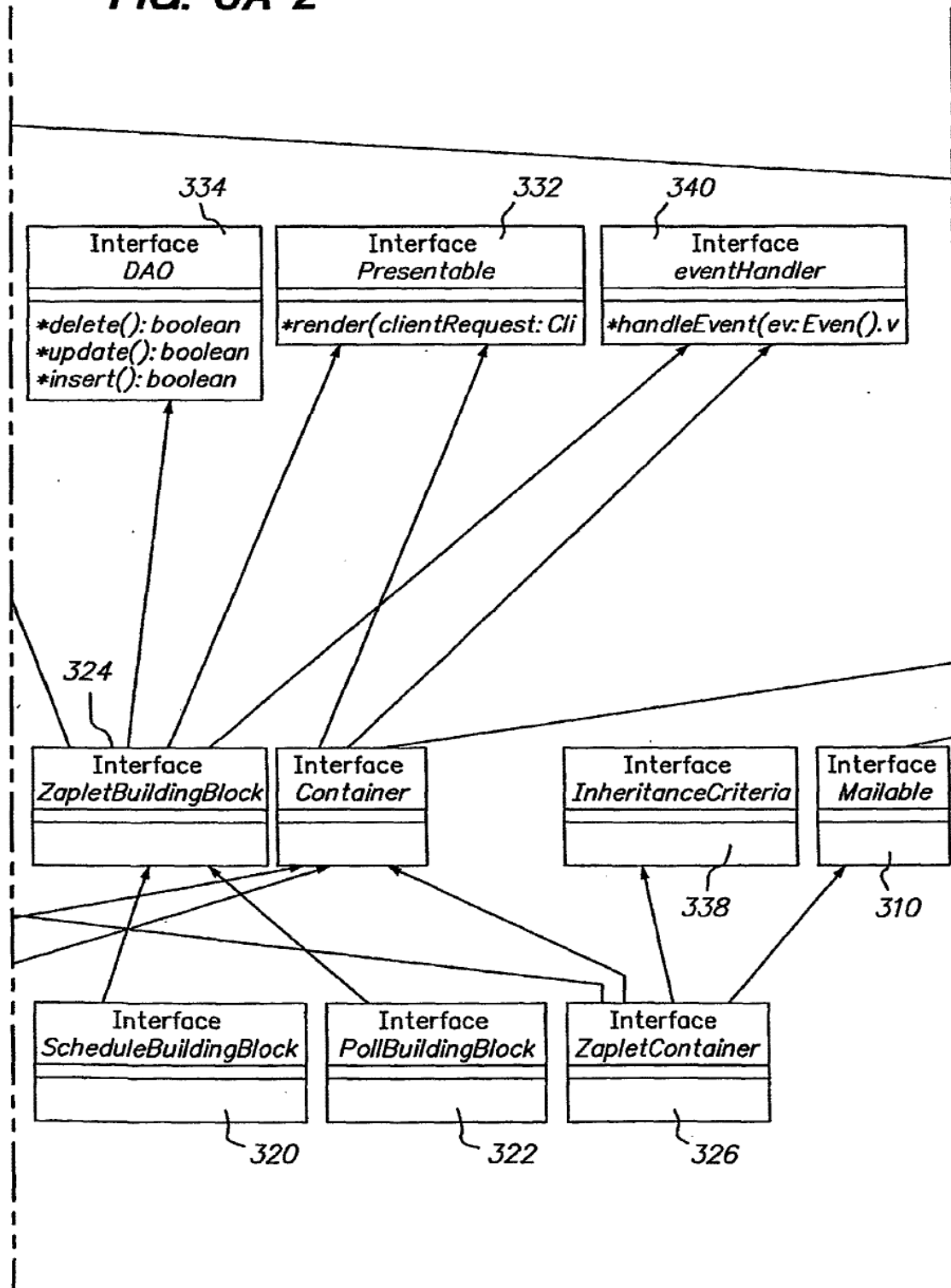
**FIG. 3A-1**



SUBSTITUTE SHEET (RULE 26)

13/70

FIG. 3A-2



SUBSTITUTE SHEET (RULE 26)

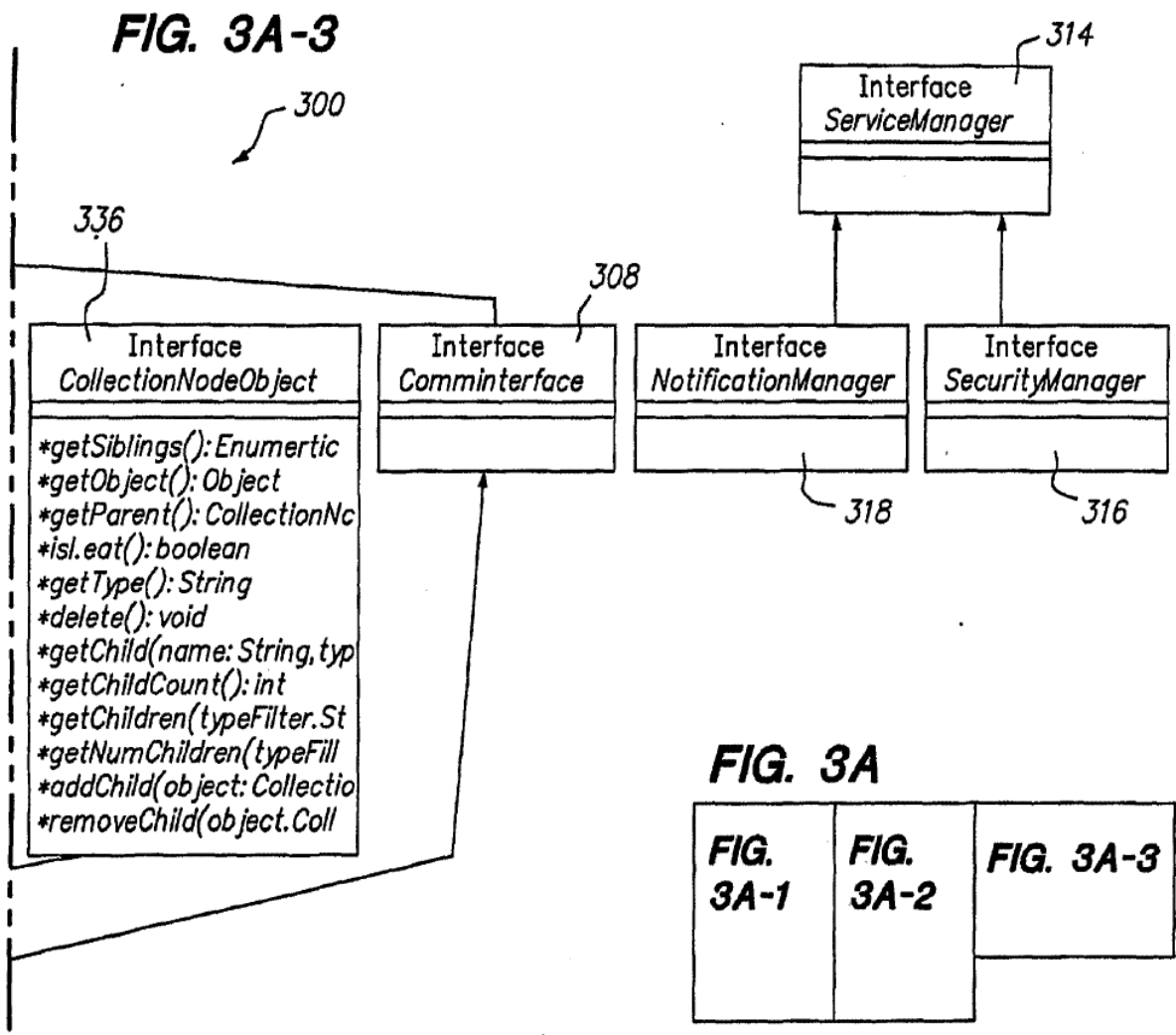
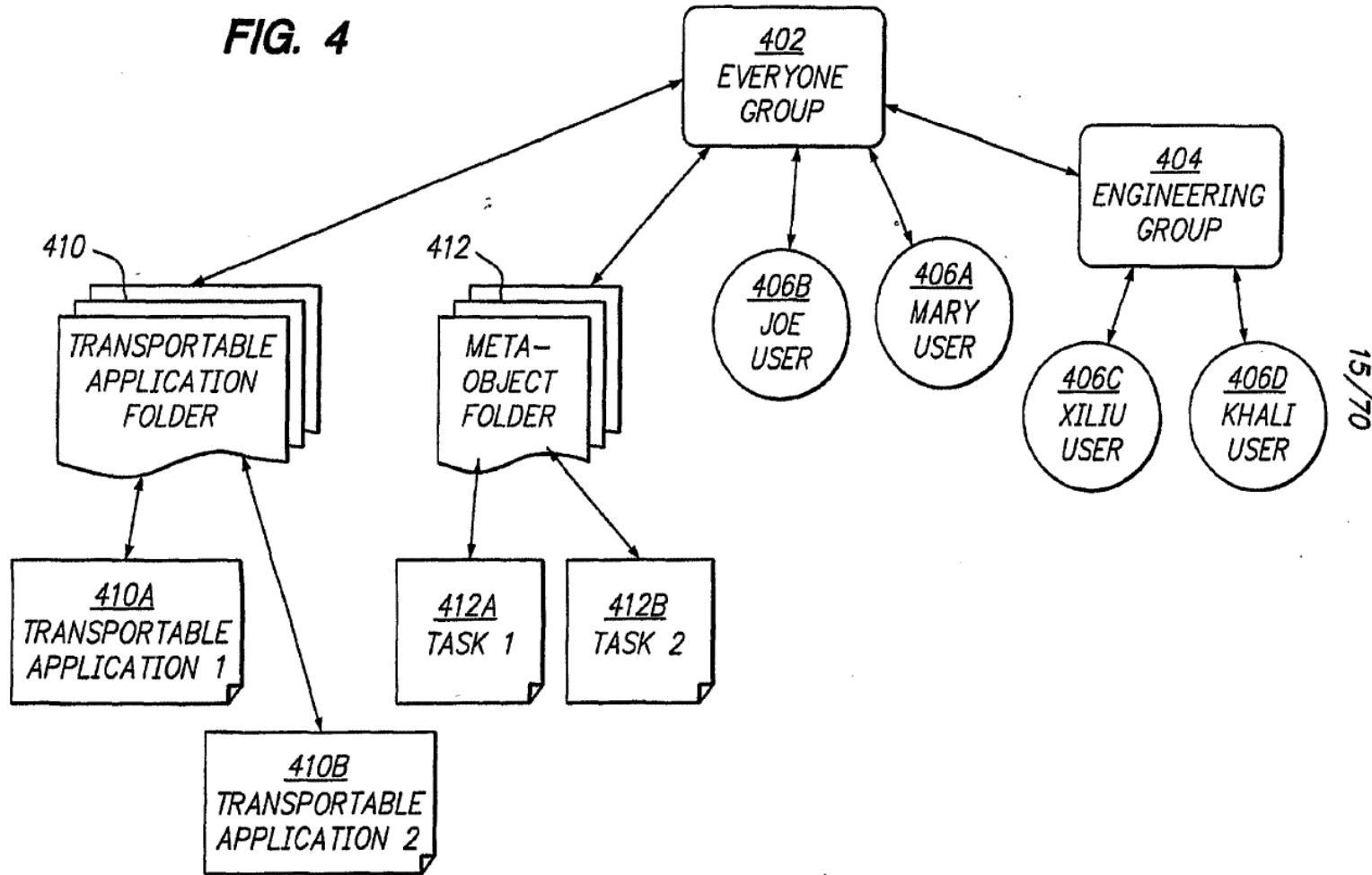
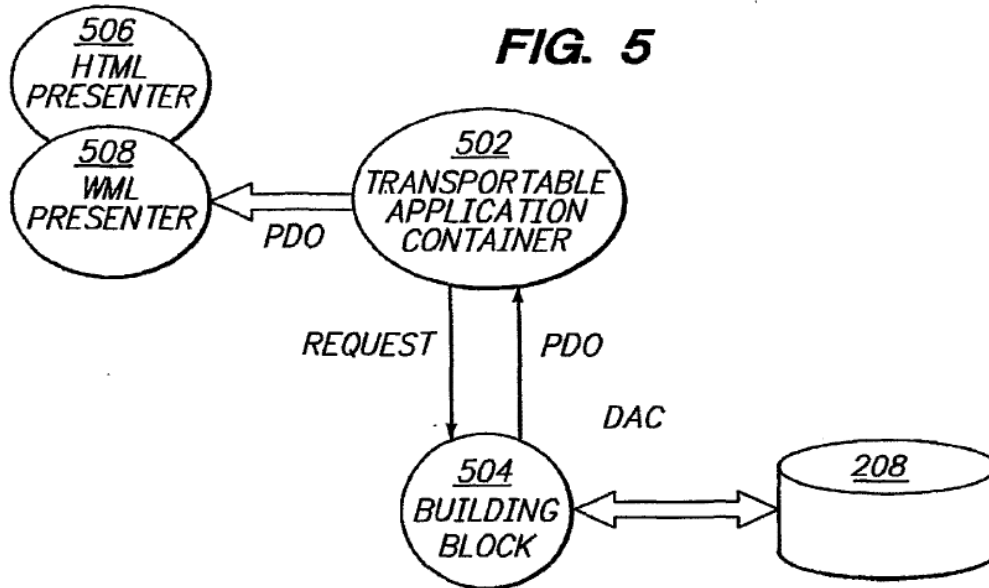


FIG. 4

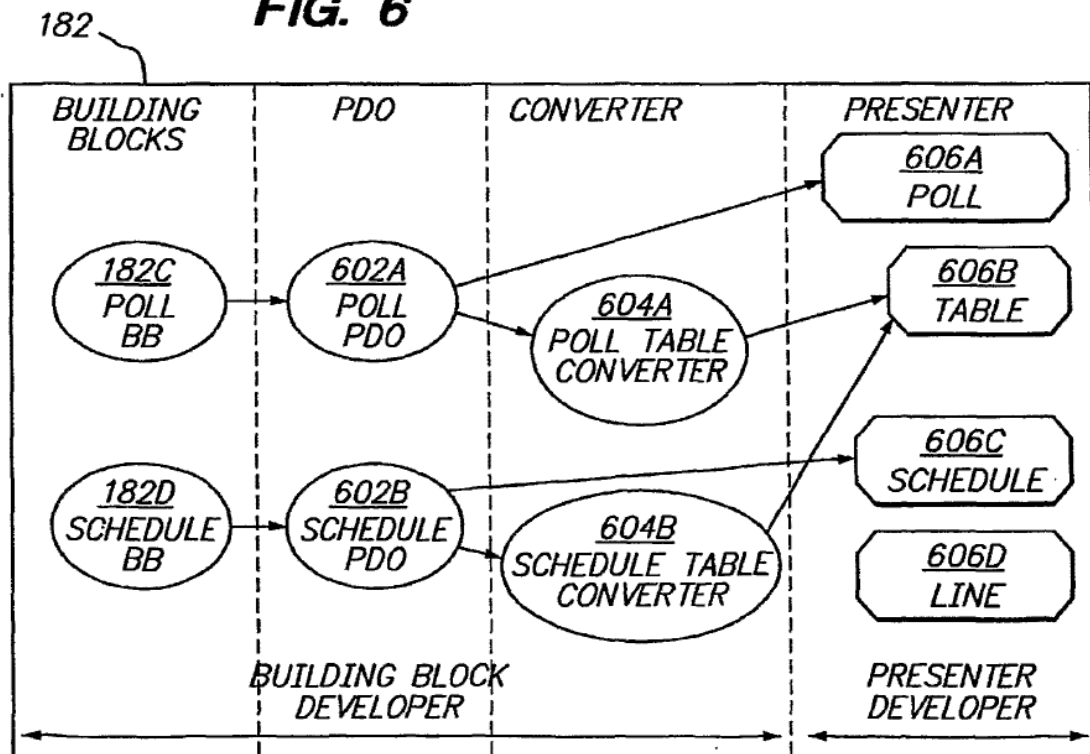


16/70

**FIG. 5**



**FIG. 6**



SUBSTITUTE SHEET (RULE 26)



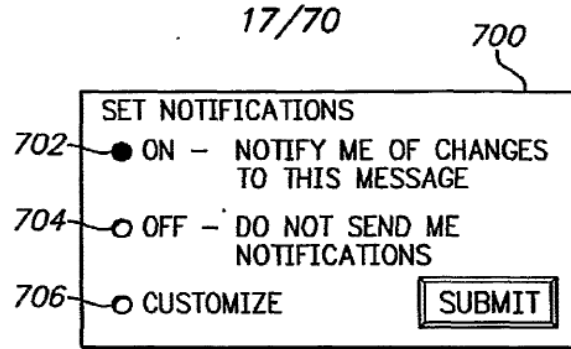


FIG. 7A

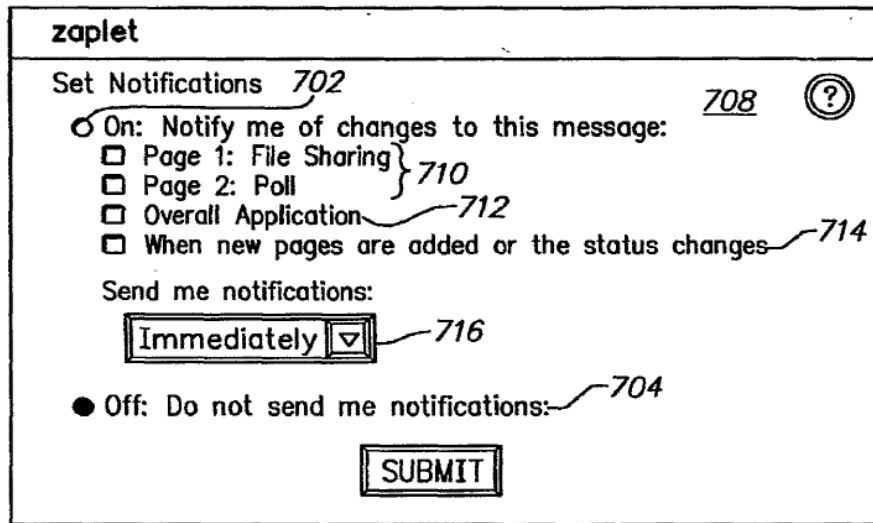


FIG. 7B

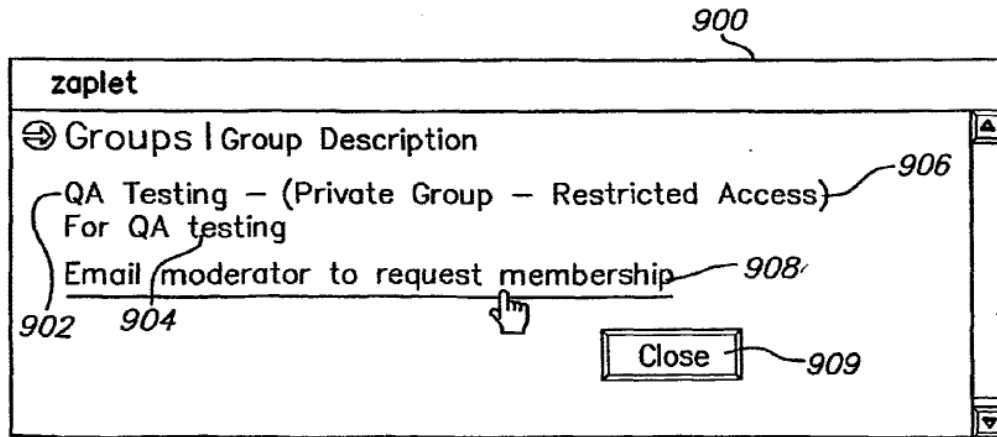
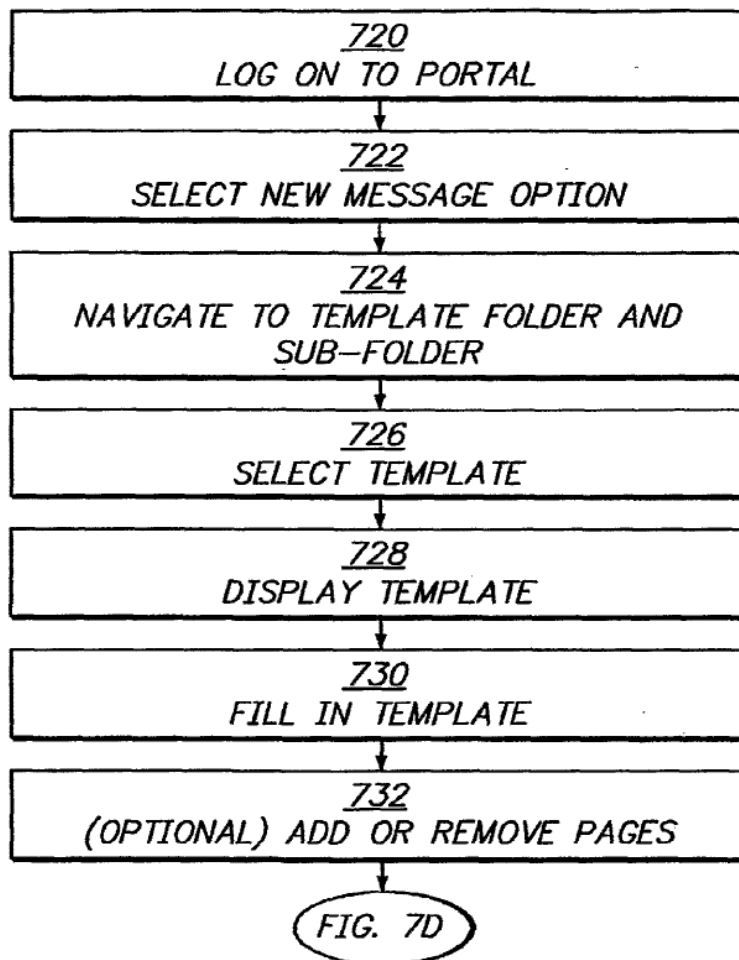


FIG. 9A

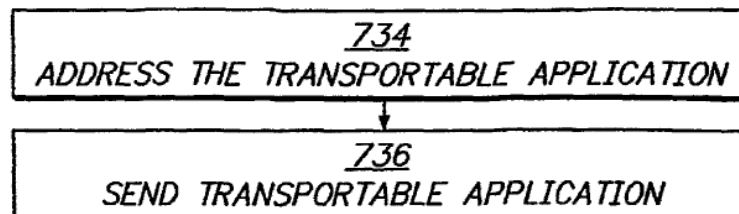
SUBSTITUTE SHEET (RULE 26)

18/70

**FIG. 7C**



**FIG. 7D**



SUBSTITUTE SHEET (RULE 26)

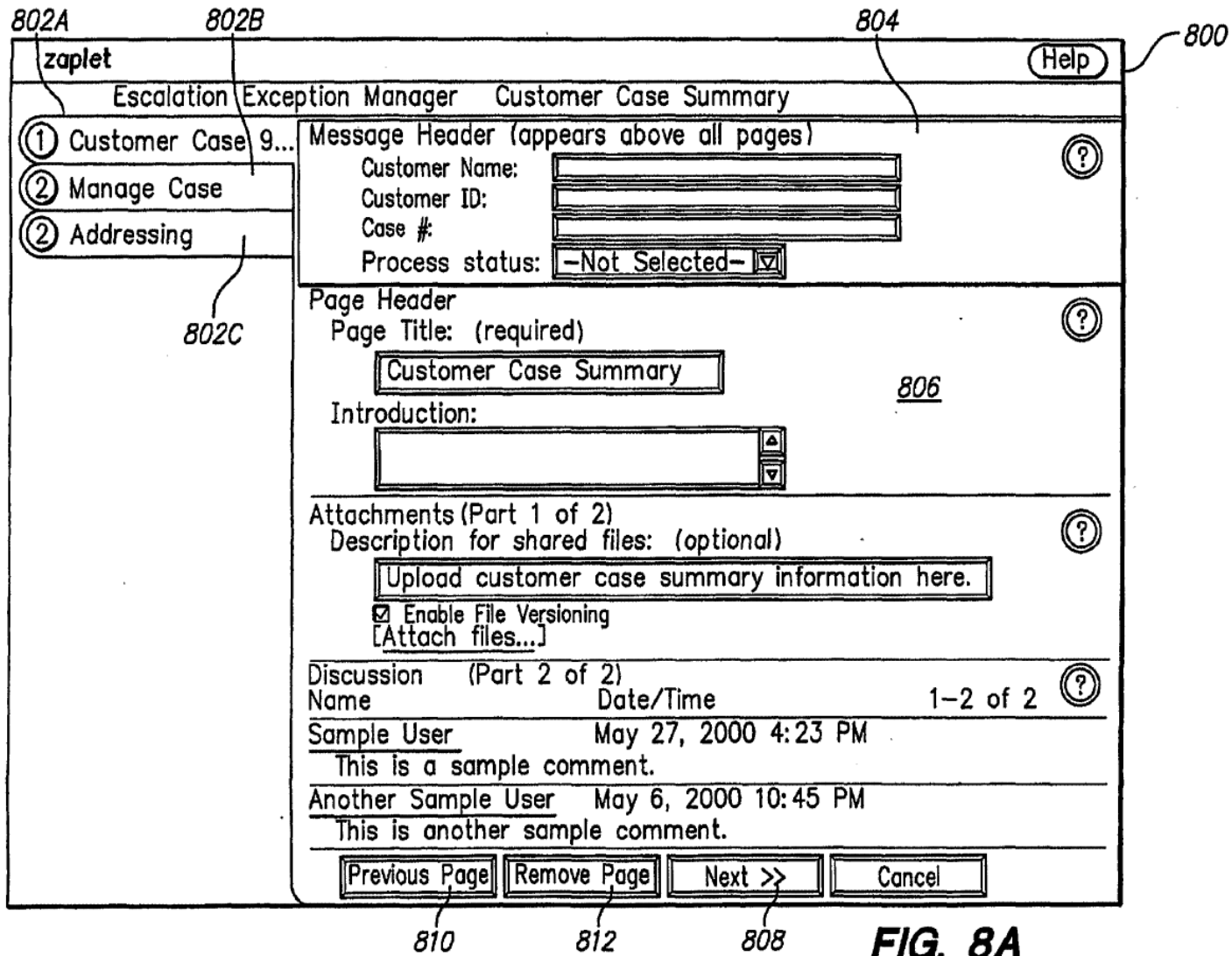


FIG. 8A

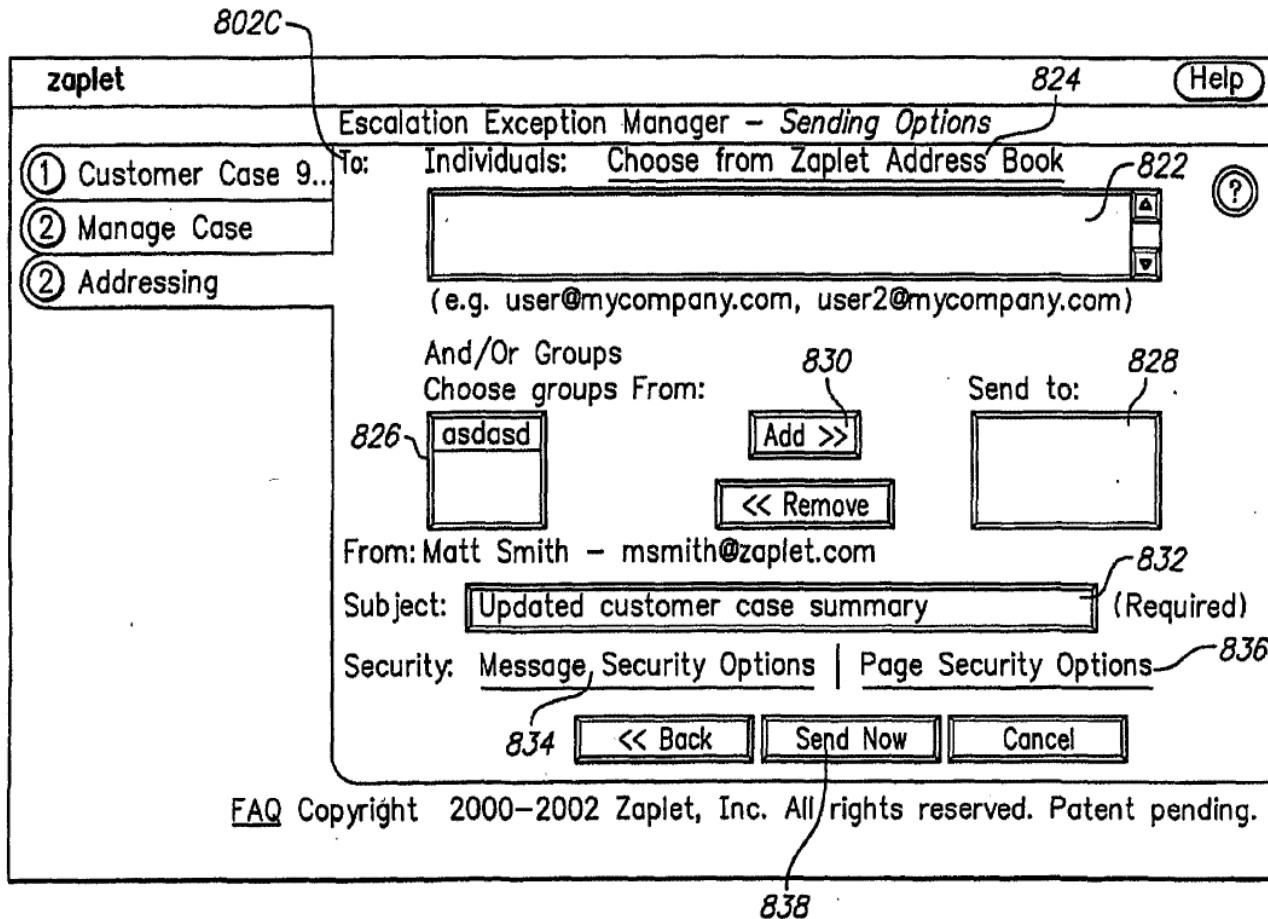


FIG. 8B

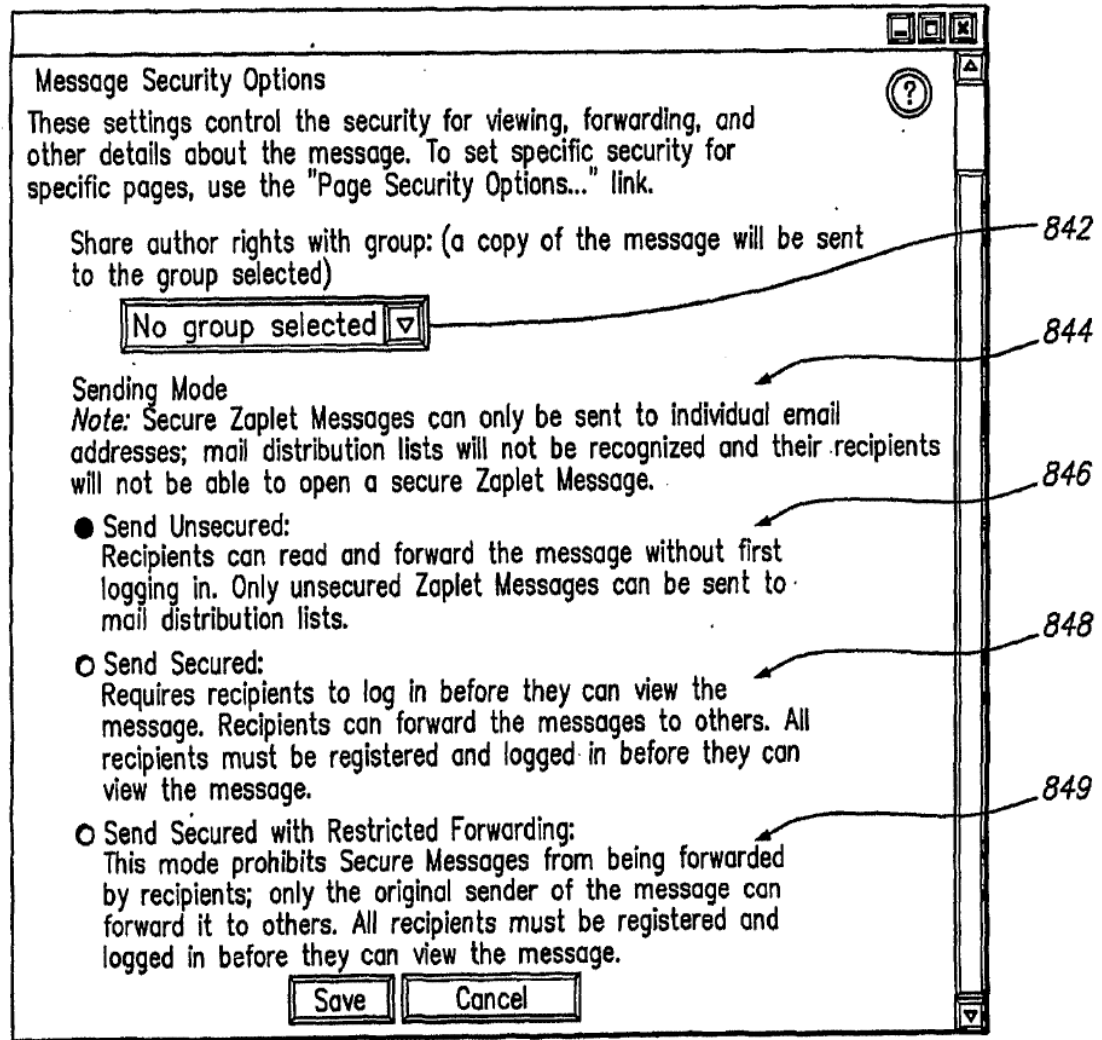


FIG. 8C

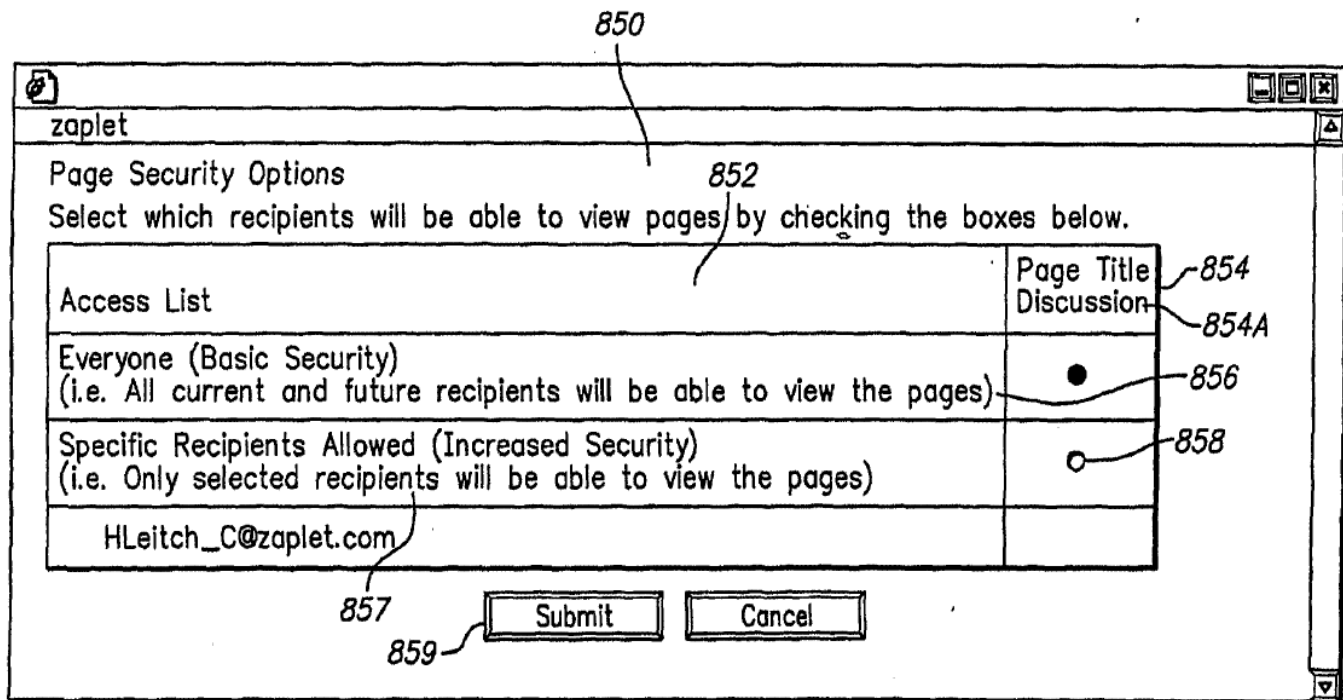


FIG. 8D

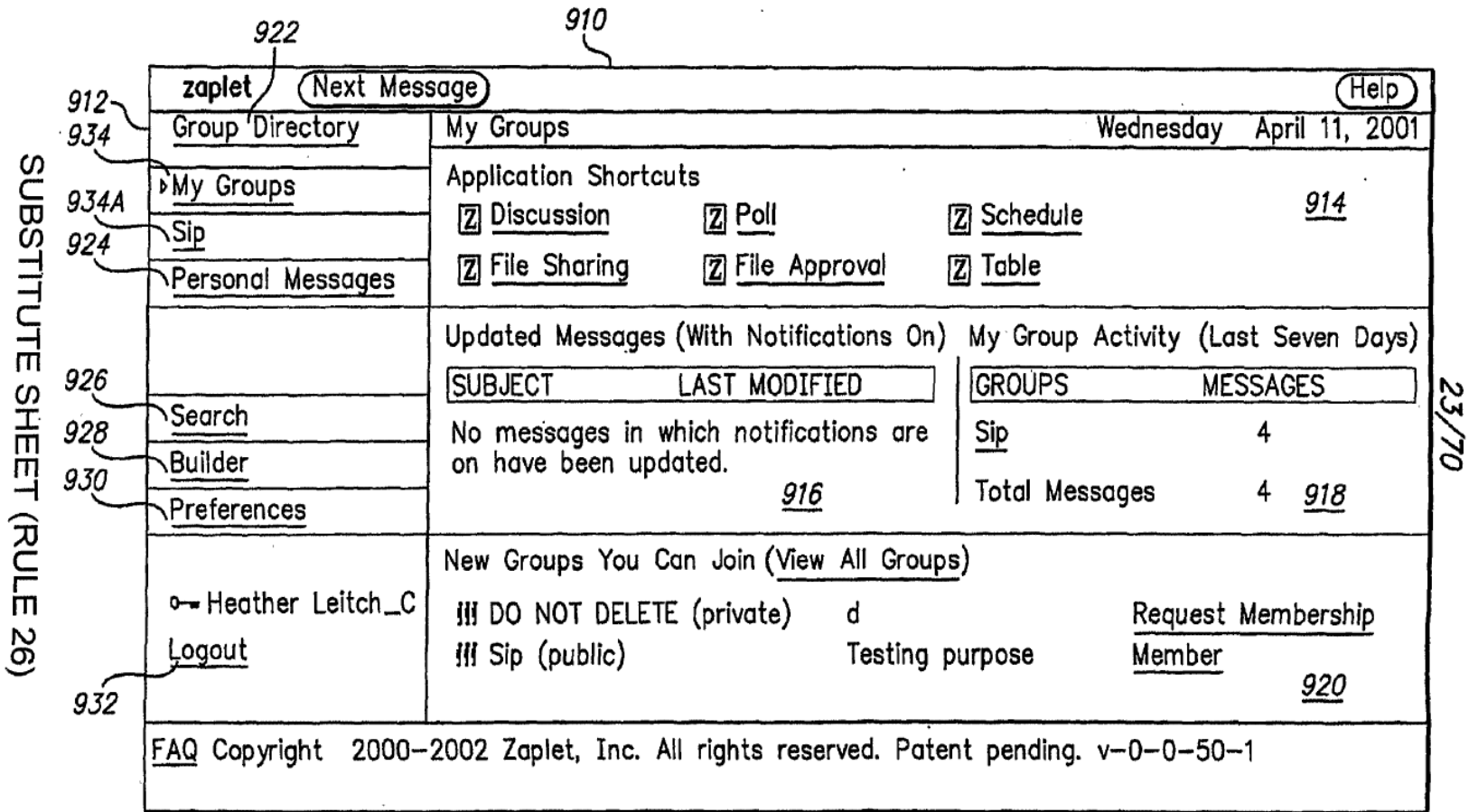


FIG. 9B

24/70

FIG. 9C

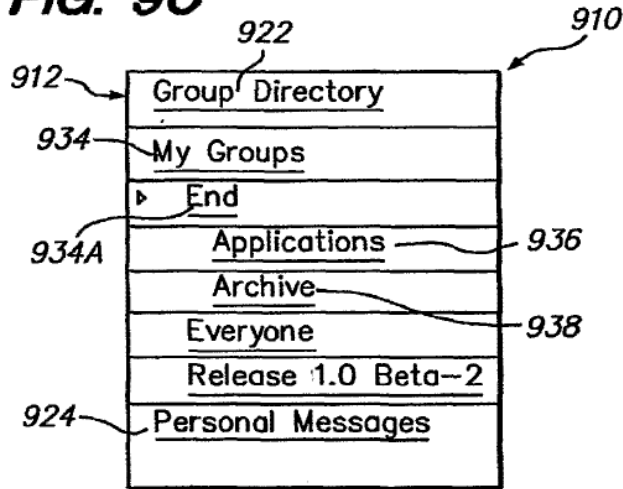
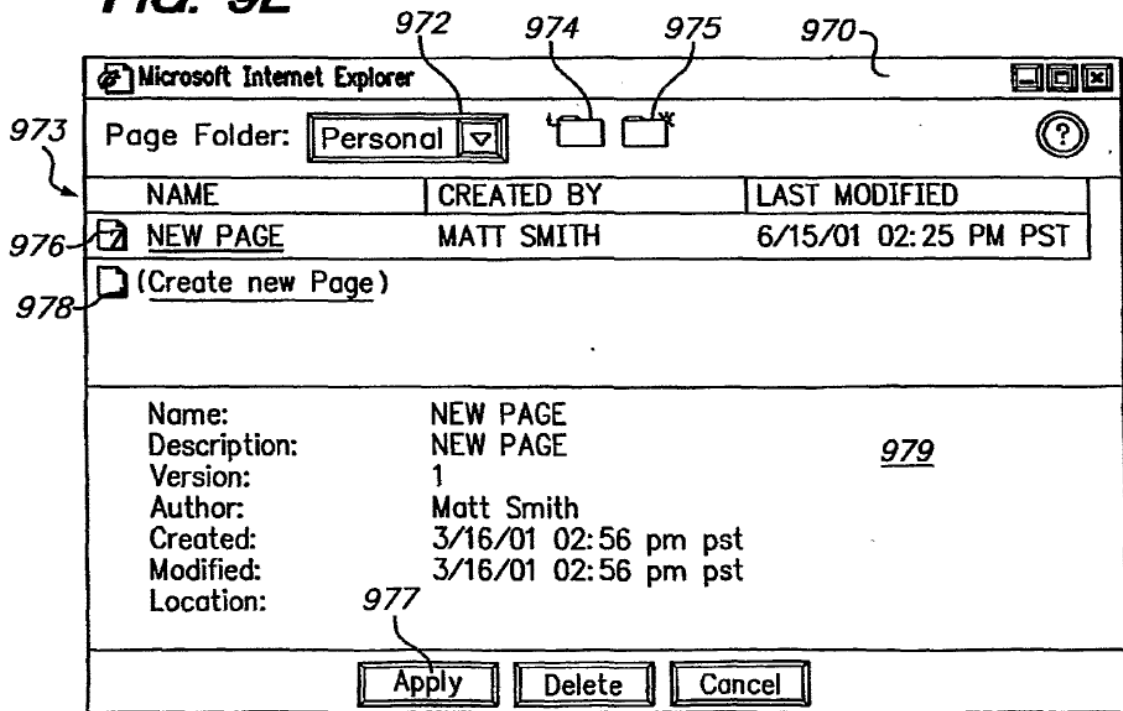


FIG. 9E



SUBSTITUTE SHEET (RULE 26)



25/70

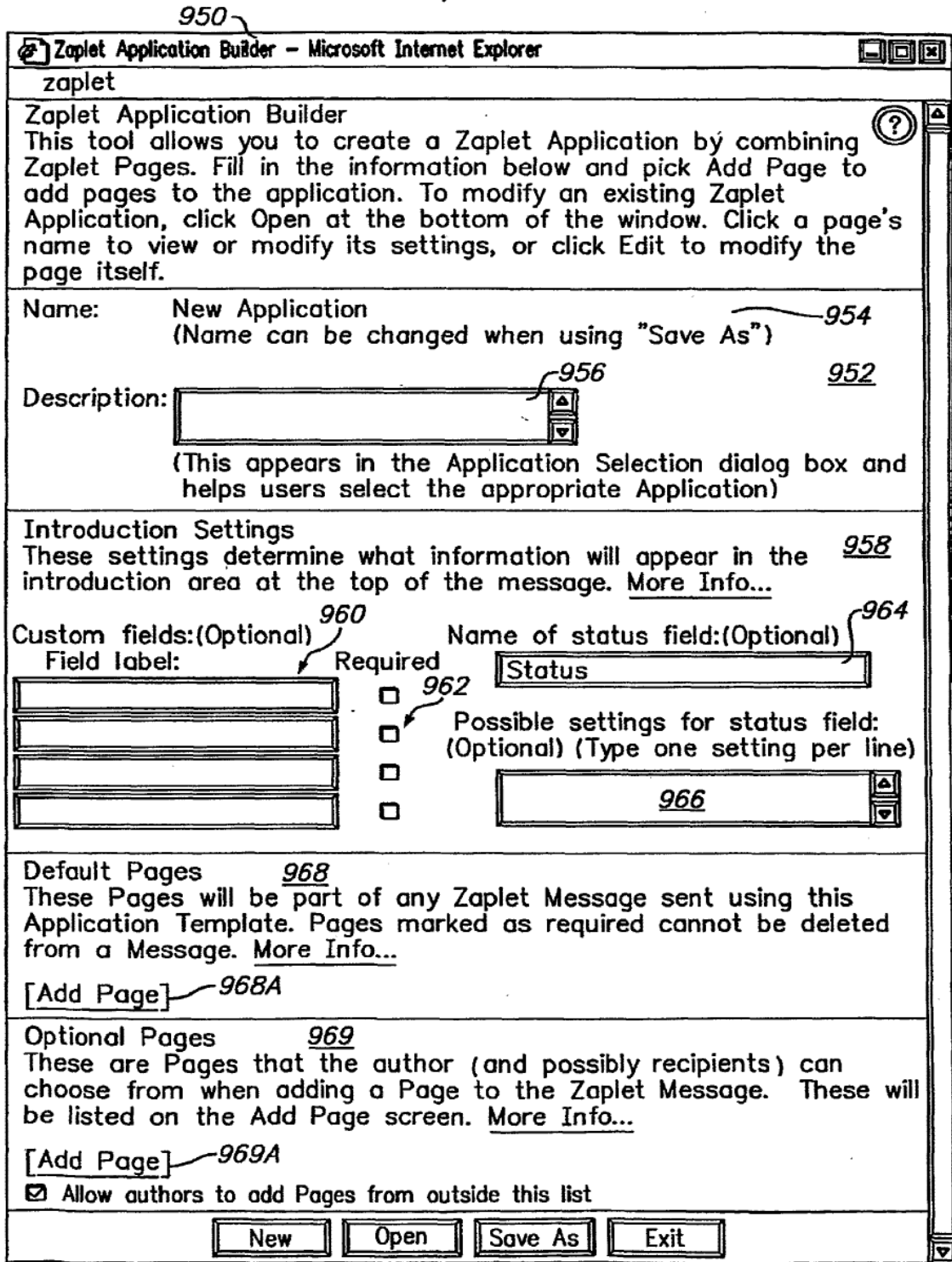


FIG. 9D

SUBSTITUTE SHEET (RULE 26)

FIG. 10A

26/70

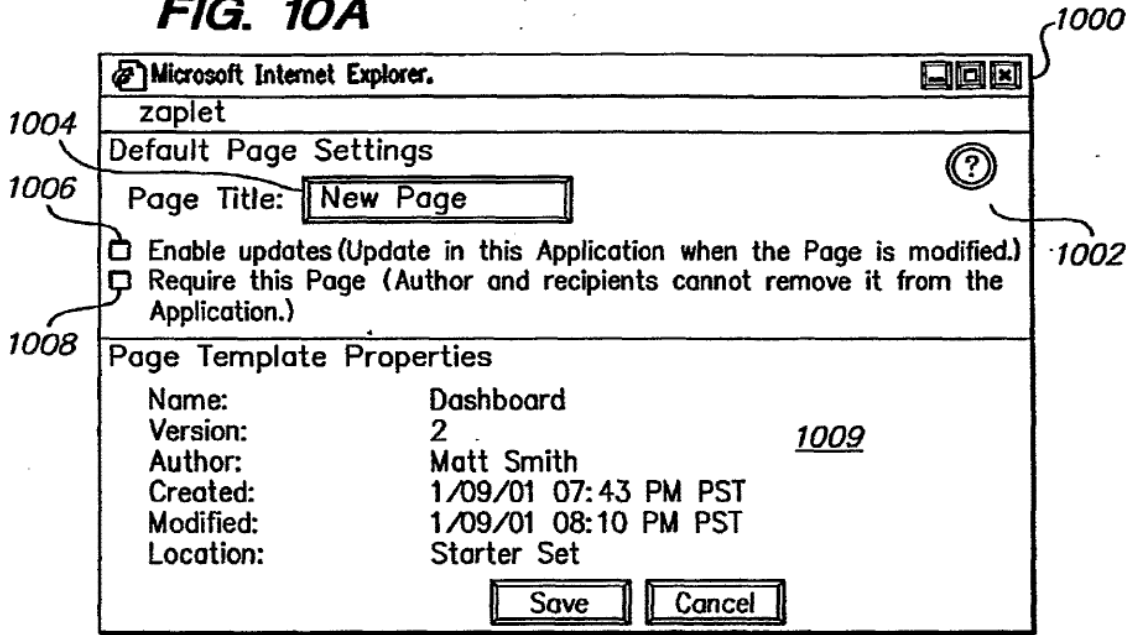
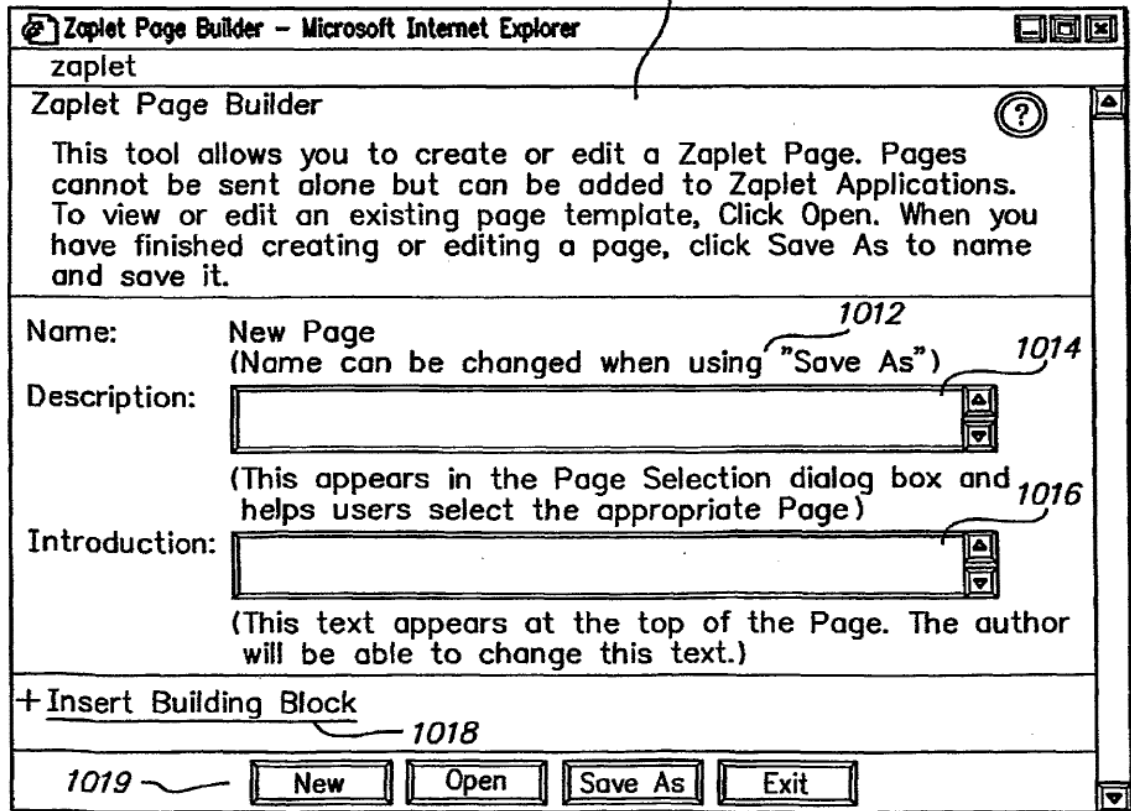


FIG. 10B

1010



SUBSTITUTE SHEET (RULE 26)

27/70

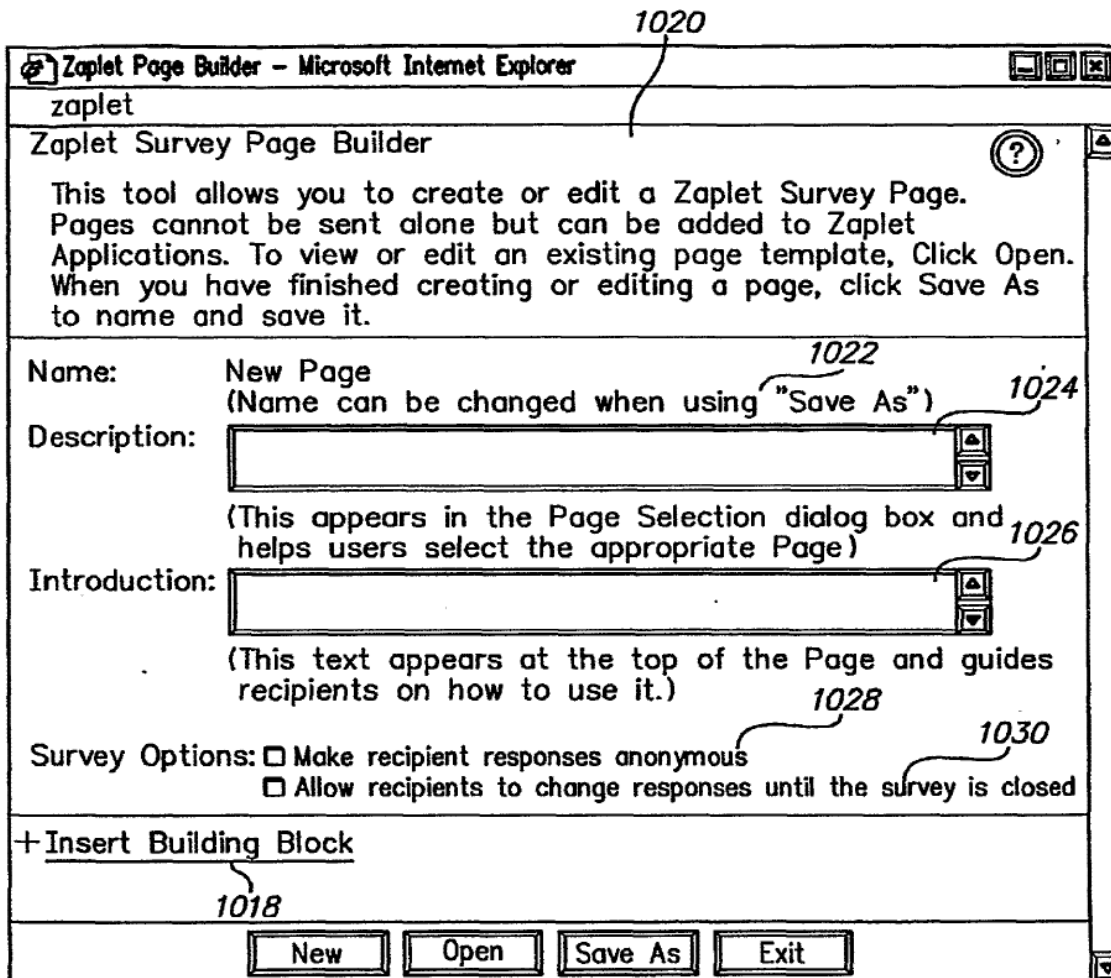


FIG. 10C

SUBSTITUTE SHEET (RULE 26)

28/70

1100

1102

1104

1106

1110

1112

1114

1116

	Approval
Sample User	Yes
Another User	No
Yet Another User	Yes

FIG. 11A

1120

1122

1124

1126

1128

Name	Date/Time	1-2 of 2
Sample User This is a sample comment.	Mar 27, 2000 4:25 PM	
Another User This is another sample comment.	Mar 28, 2000 10:48 PM	

FIG. 11B

1130

1132

1134

1136

FIG. 11C

SUBSTITUTE SHEET (RULE 26)

29/70

1140

The screenshot shows a form titled 'Image' with a header bar containing 'Move Up', 'Move Down', and 'Delete' buttons. The main content area includes a 'No Image Attached' message, an 'Attach Image...' button, an 'Image Name:(required)' text box, and an 'Image Description:(required)' text box with a scroll bar.

FIG. 11D

1150

The screenshot shows a form titled 'Image Gallery' with a header bar containing 'Move Up', 'Move Down', and 'Delete' buttons. The main content area is titled 'Image Worksheet' and includes 'Add Image' and 'Add Multiple Images' buttons, an 'Image Gallery Name' text box, a message about uploading images, and a 'Space available: 30 Images Page: 1 of 1' status. A checkbox labeled 'Allow participant contributions' is at the bottom.

FIG. 11E

1160

The screenshot shows a form titled 'Information Fields' with a header bar containing 'Move Up', 'Move Down', and 'Delete' buttons. The main content area has two columns: 'Field Names' and 'Field Values', each with three text boxes. An 'Add Row' button is located at the bottom left.

FIG. 11F

SUBSTITUTE SHEET (RULE 26)

30/70

1172 1170

⊞ Inline Document ⬆ Move Up ⬇ Move Down ✕ Delete

NOTE: Accepts only HTML (.htm and .html) files.  
[Upload file...] (required)

**FIG. 11G**

1170

⊞ Interactive Web Page ⬆ Move Up ⬇ Move Down ✕ Delete

Instructions  
Please enter the address of the Web page you would like to display and interact with  
in your Zaplet message: (required)

http://

(e.g., http:// www.zaplet.com/)

1182 1180

**FIG. 11H**

SUBSTITUTE SHEET (RULE 26)

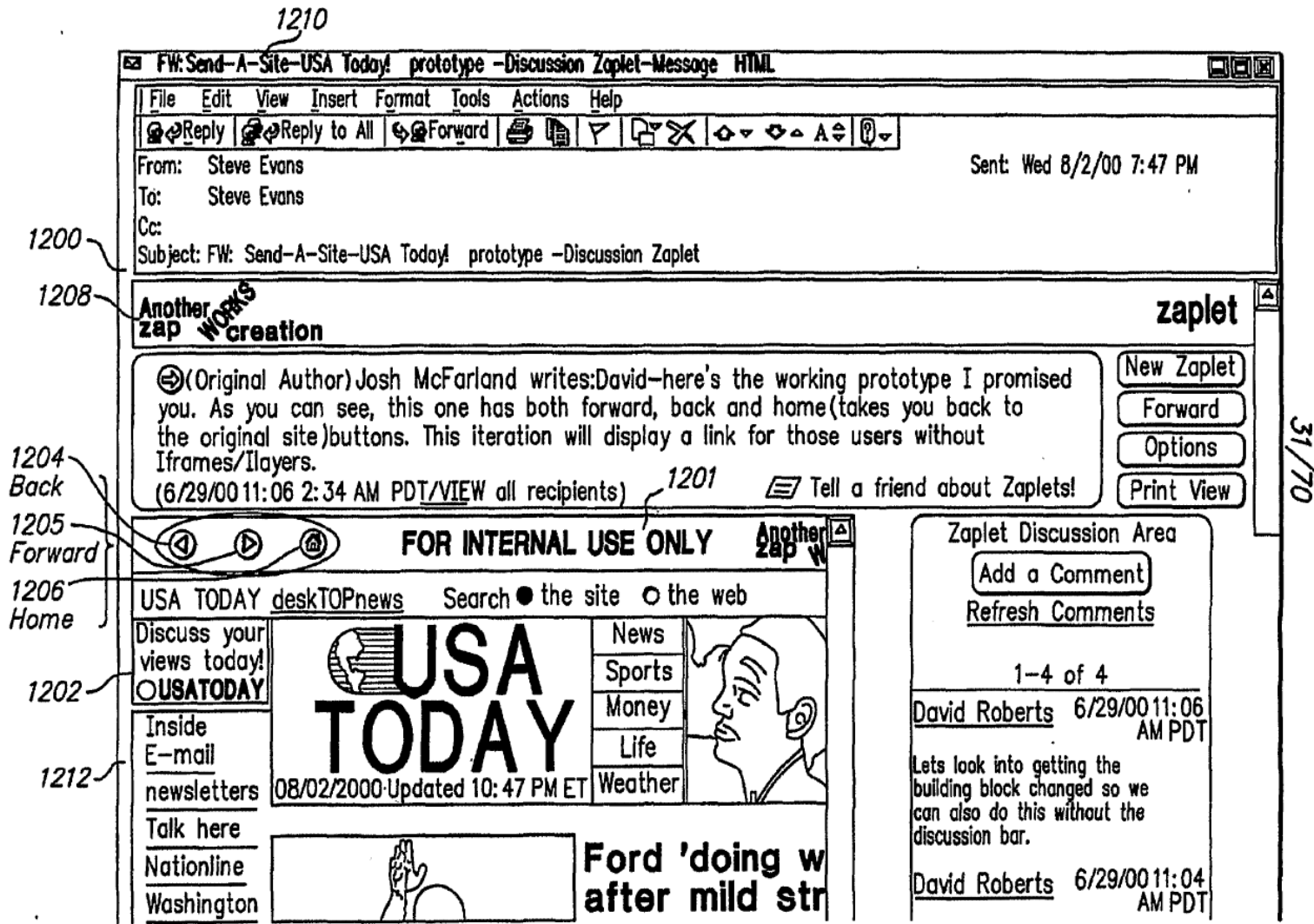


FIG. 12

WO 02/21413

PCT/US01/42041

32/70

Invitation Move Up Move Down Delete

1302 [Select Invitation Style...] (required)

1304 Event Title: (required) 1300

1306 Brief Description: (required)

1308 Event Details: (required)

1310 Date: (required)  
(e.g., Monday, 10/12/99)

1312 Duration: (required)  
(e.g., 7am-5pm)

1314 Location: (required)

1316 Address: (required)

1318 RSVP By: (required)  
(e.g., Monday, 10/12/99)

FIG. 13A

1330 Advanced Options zaplet

1332 Recipients can vote for?:  
Only 1 choice

1334  Include "Other" answer choice and let participants write-in.?

1336 Vote options?:  
 Allow participants to change their vote  
 Allow participants to vote anonymously

1338 Poll results available to participants?:  
 Always  After voting  After poll closes  Never  
Note: Author can always see results

Save Cancel

FIG. 13C

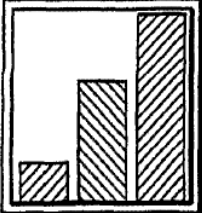
SUBSTITUTE SHEET (RULE 26)



1320

1322 Question for recipients to answer:(required) Advanced Options

(e.g., What is your opinion?)

1326   Pie Chart  Bar Chart

Include linked comment section  
(Shows participant votes and comments together)

1324 Enter up to 12 different answer choices for poll: (1 required)

<input type="text" value="A"/>	<input type="text" value="G"/>
<input type="text" value="B"/>	<input type="text" value="H"/>
<input type="text" value="C"/>	<input type="text" value="I"/>
<input type="text" value="D"/>	<input type="text" value="J"/>
<input type="text" value="E"/>	<input type="text" value="K"/>
<input type="text" value="F"/>	<input type="text" value="L"/>

(e.g., 1.Yes 2.No 3.Let's decide later)

FIG. 13B

Schedule

 ↑ Move Up   ↓ Move Down   ✕ Delete

Schedule type: (required)

Structured  
 (For each time slot, you specify the exact date, time and duration.)

Free-form  
 (For each time slot, you can type anything you wish.)

Proposed Location:

		Date	Start Time	Duration
Ideal Choice (required)	<input type="checkbox"/>	□/□/□	□:□ am	1 hour
1st Alternate	<input type="checkbox"/>	□/□/□	□:□ am	1 hour
2nd Alternate	<input type="checkbox"/>	□/□/□	□:□ am	1 hour
3rd Alternate	<input type="checkbox"/>	□/□/□	□:□ am	1 hour
4th Alternate	<input type="checkbox"/>	□/□/□	□:□ am	1 hour

Options:

Allow recipients to suggest new dates & times

Note: The total number of time slots, entered by author or recipient, is limited to five.

**FIG. 13D**

⊞ Table

 ⬆ Move Up   ⬇ Move Down   ✕ Delete

Instructions:  
Name your table. Then either click Import to populate the the table with a file or enter column names and data directly into the table below. Recipients can add/edit rows and data.

Table name:

(required)    Import

	Insert Rows/Cols	Delete Selected Row/Col	Properties of Selected Col		
	A ○	B ○	C ○	D ○	E ○
	<Column Name>	<Column Name>	<Column Name>	<Column Name>	<Column Name>
○ 1					

**FIG. 13E**

35/70

1422

1432

1434

1426

1420

1424

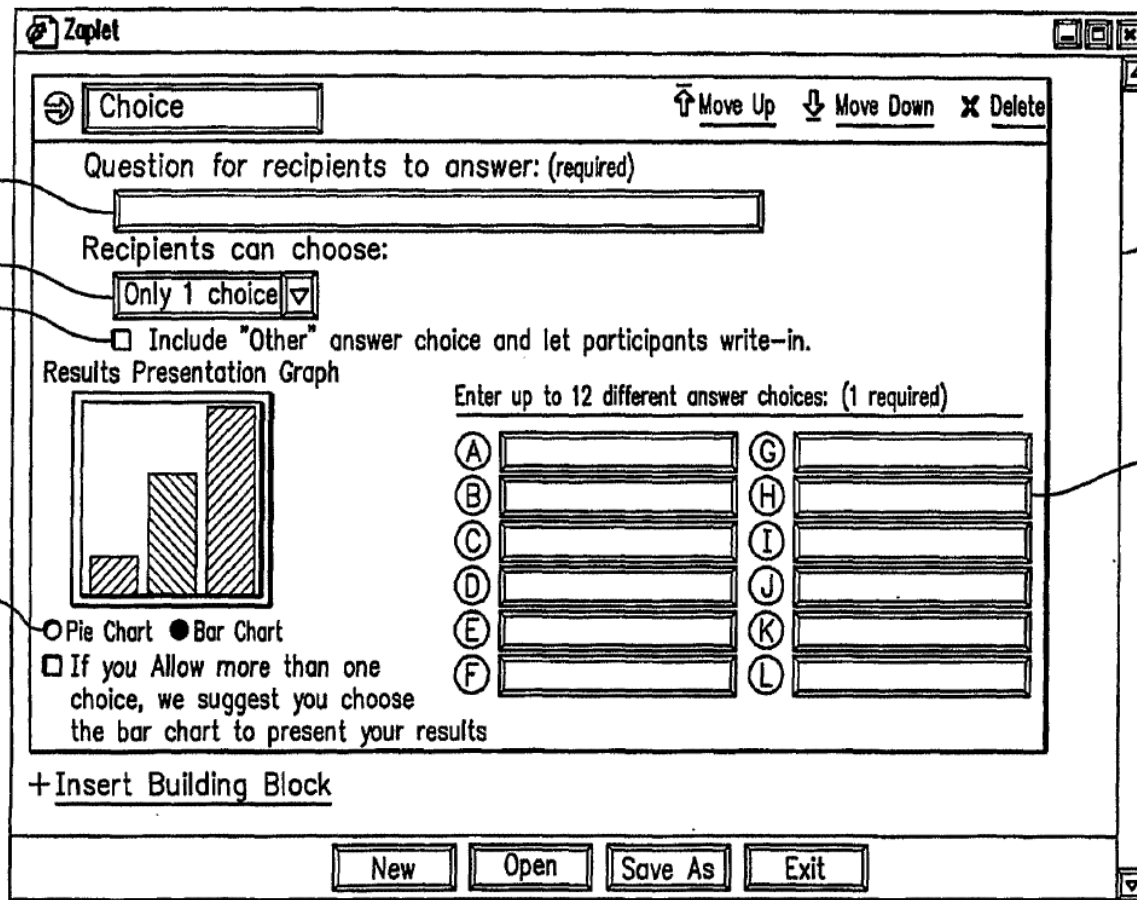


FIG. 14A

+Insert Building Block

Free Text Response ↑ Move Up ↓ Move Down ✕ Delete

1432 Question for recipients to answer:

(e.g., What is your opinion about...?)

1434 Recipients will see:

- Single-line text box
- Multi-line comment area

+Insert Building Block

**FIG. 14B**

38/70

**Ratings** Move Up Move Down Delete

Instructions for recipients:  
 1442  
(e.g., Please rate the candidate in the following areas...)

Rating Scale:  
Minimum: 1  
Maximum:  1444  
 Show N/A option 1446 1440

Enter rating labels: (optional)

1.   
2.   
3.   
4.   
5.  } 1448  
(e.g., Poor, Average, Excellent)

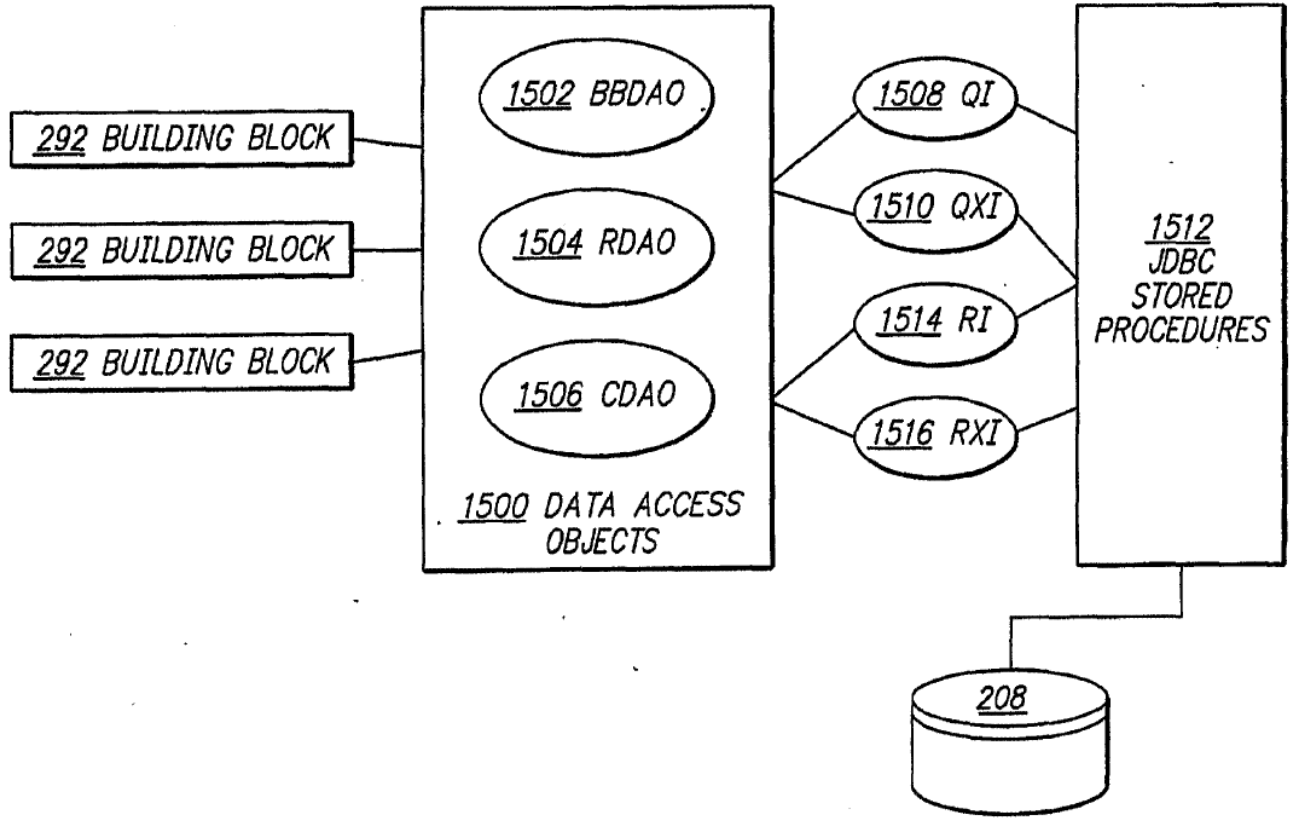
Questions or items to rate: (1 required) 1452  
[Add 5 Entries Remove 5 Entries] 1450

1.   
2.   
3.   
4.   
5.   
6.   
7.   
8.   
9.   
10.  } 1450  
(e.g., 1.Communications Skills 2.Technical Skills 3.Overall)

**FIG. 14C**

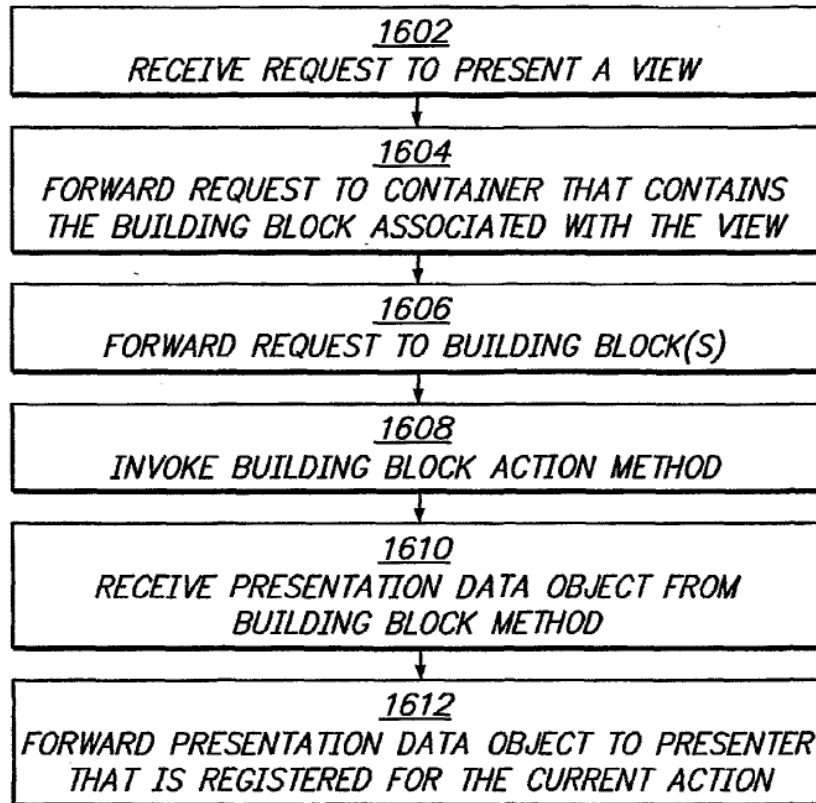
SUBSTITUTE SHEET (RULE 26)

FIG. 15



39/70

40/70

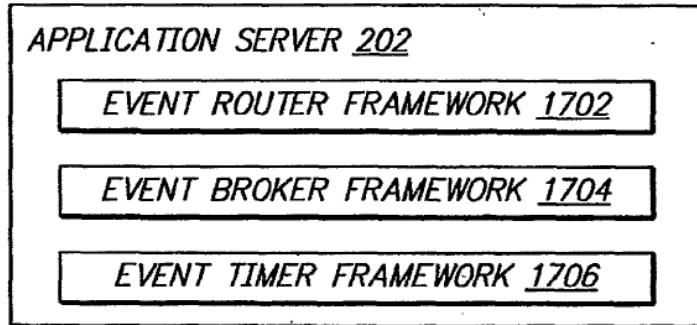
**FIG. 16**

SUBSTITUTE SHEET (RULE 26)

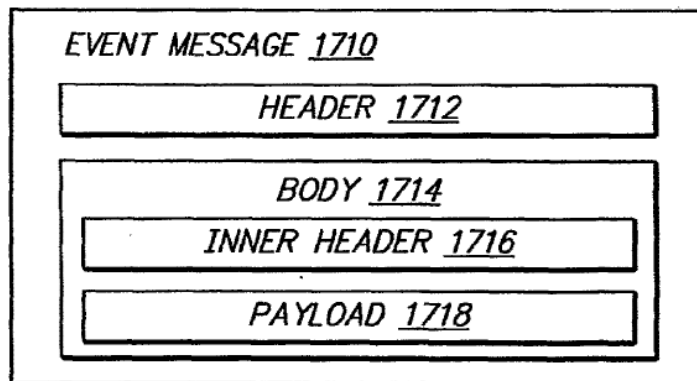


41/70

**FIG. 17A**

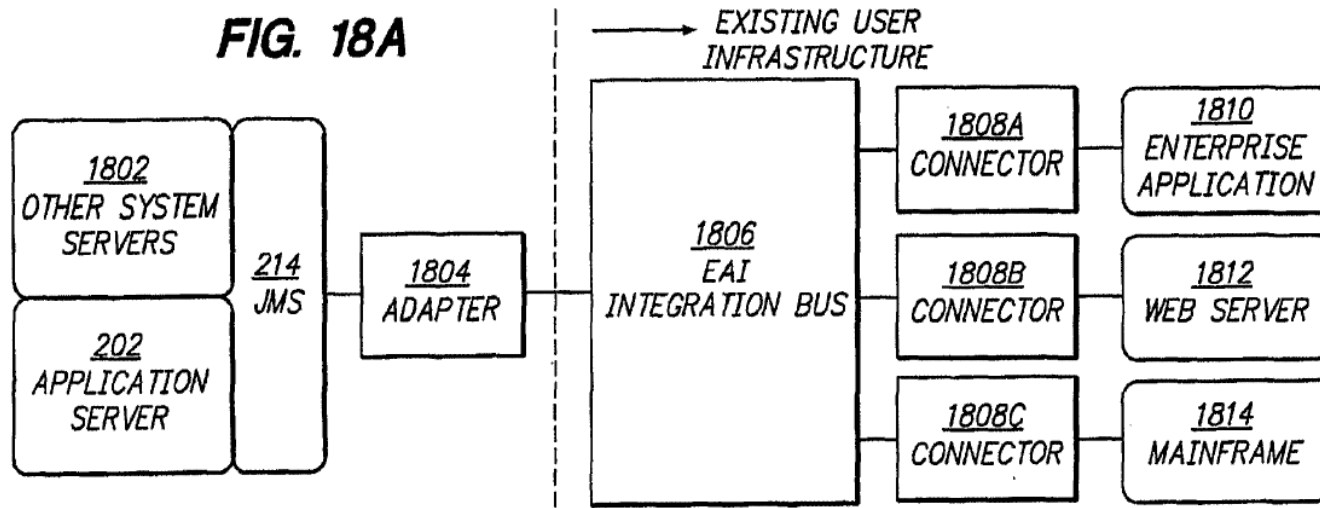


**FIG. 17B**



SUBSTITUTE SHEET (RULE 26)

**FIG. 18A**



**FIG. 18B**

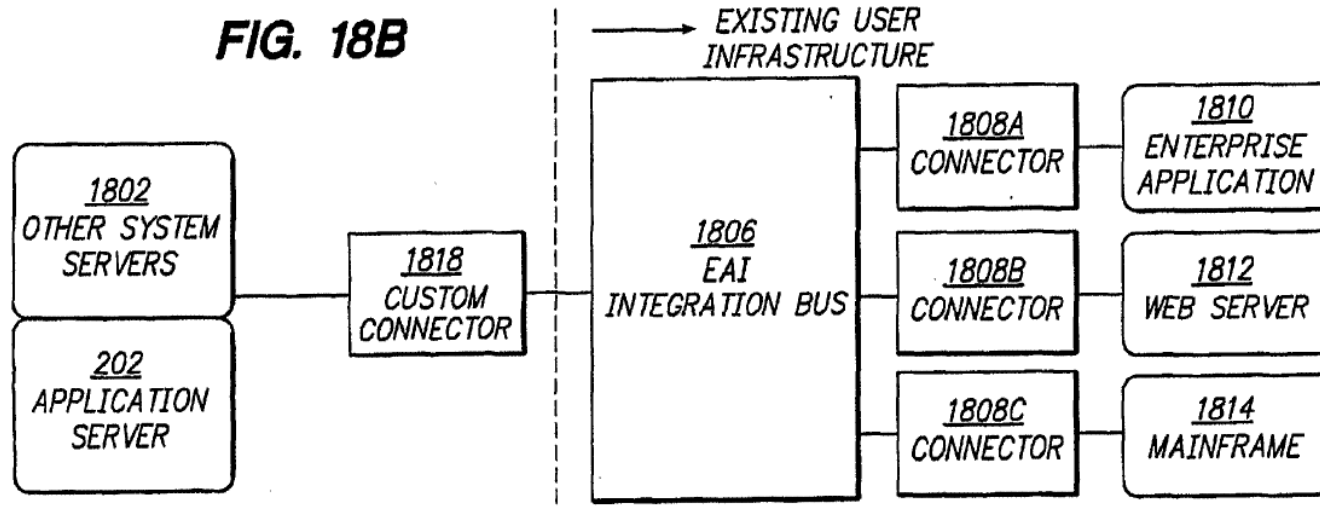
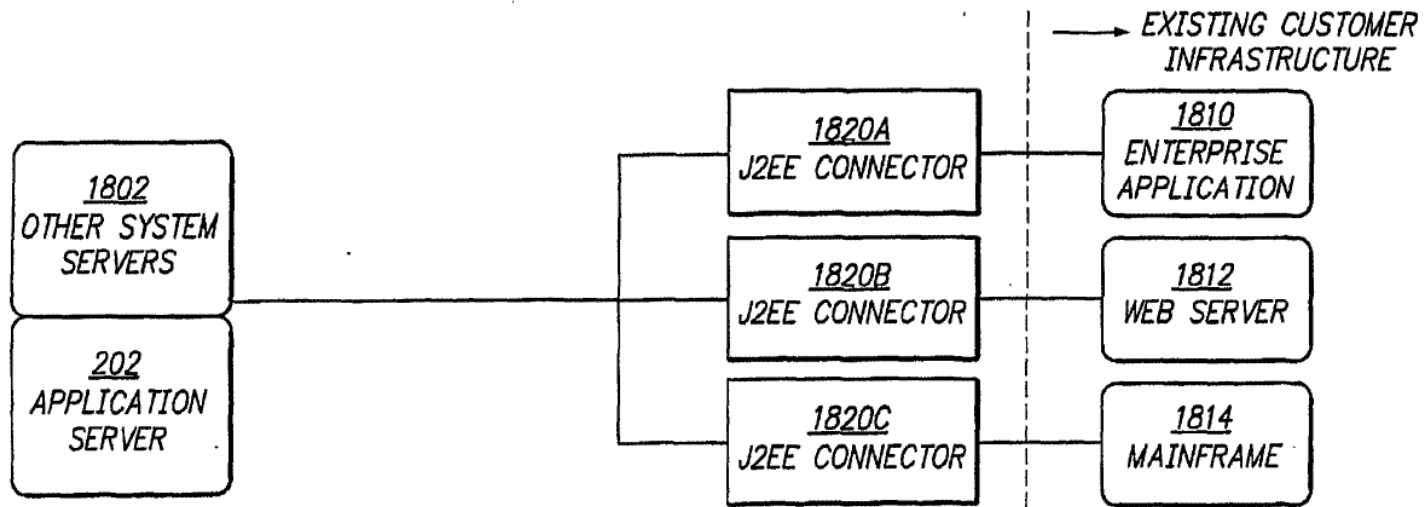
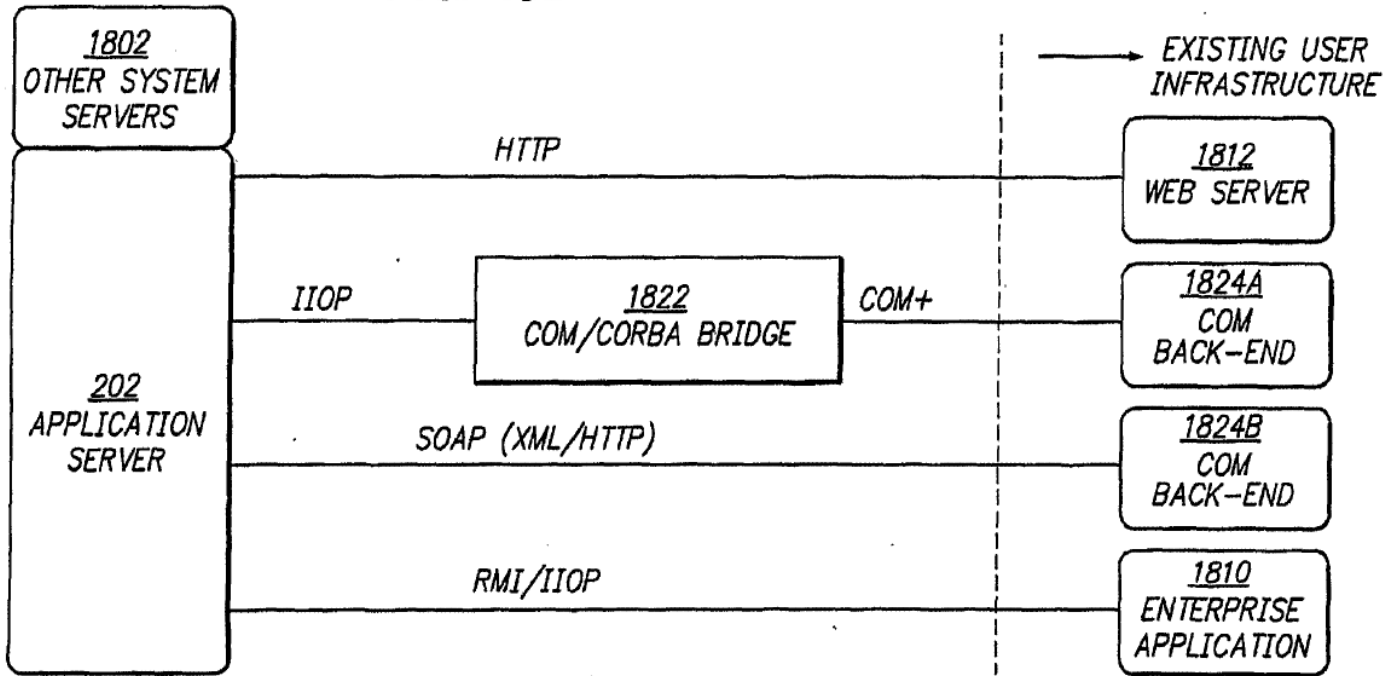


FIG. 18C



43/70

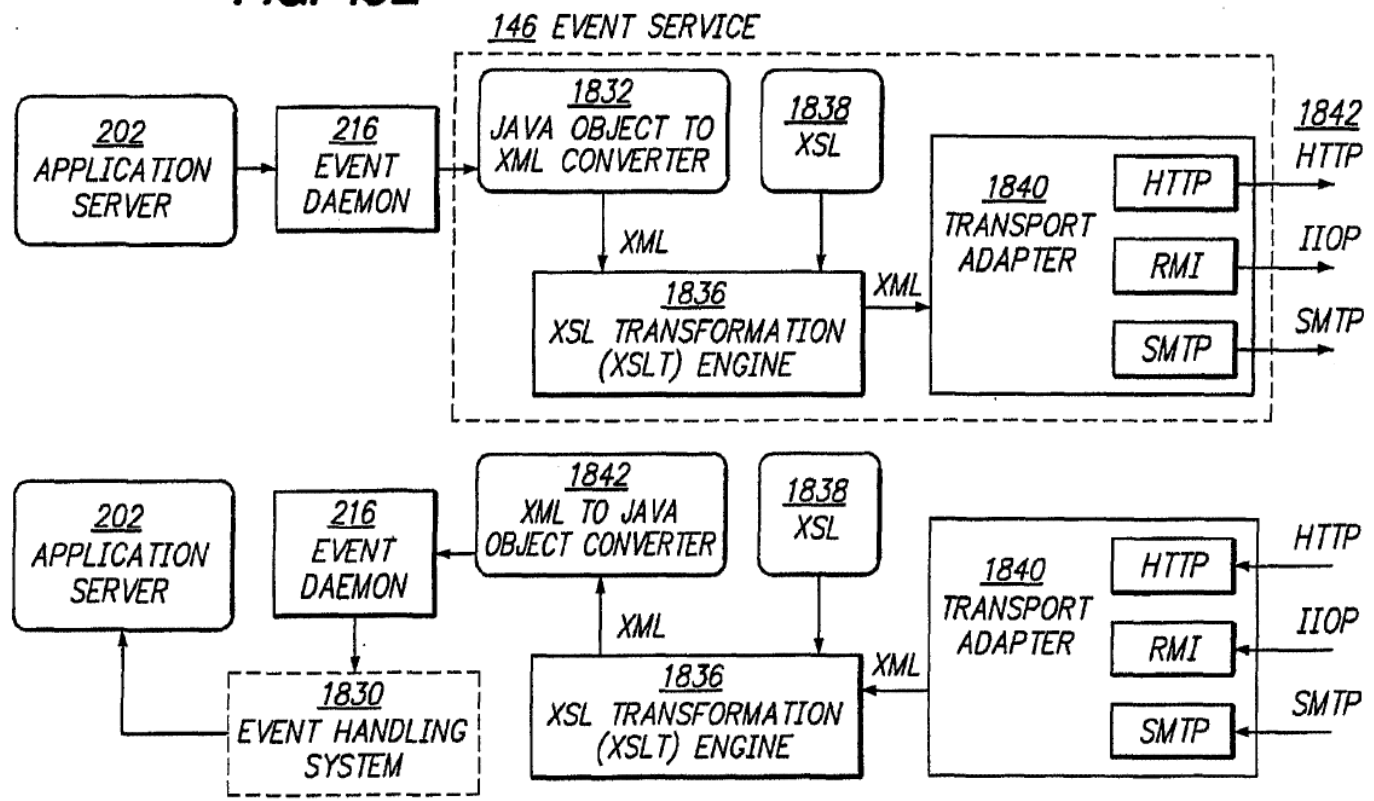
FIG. 18D



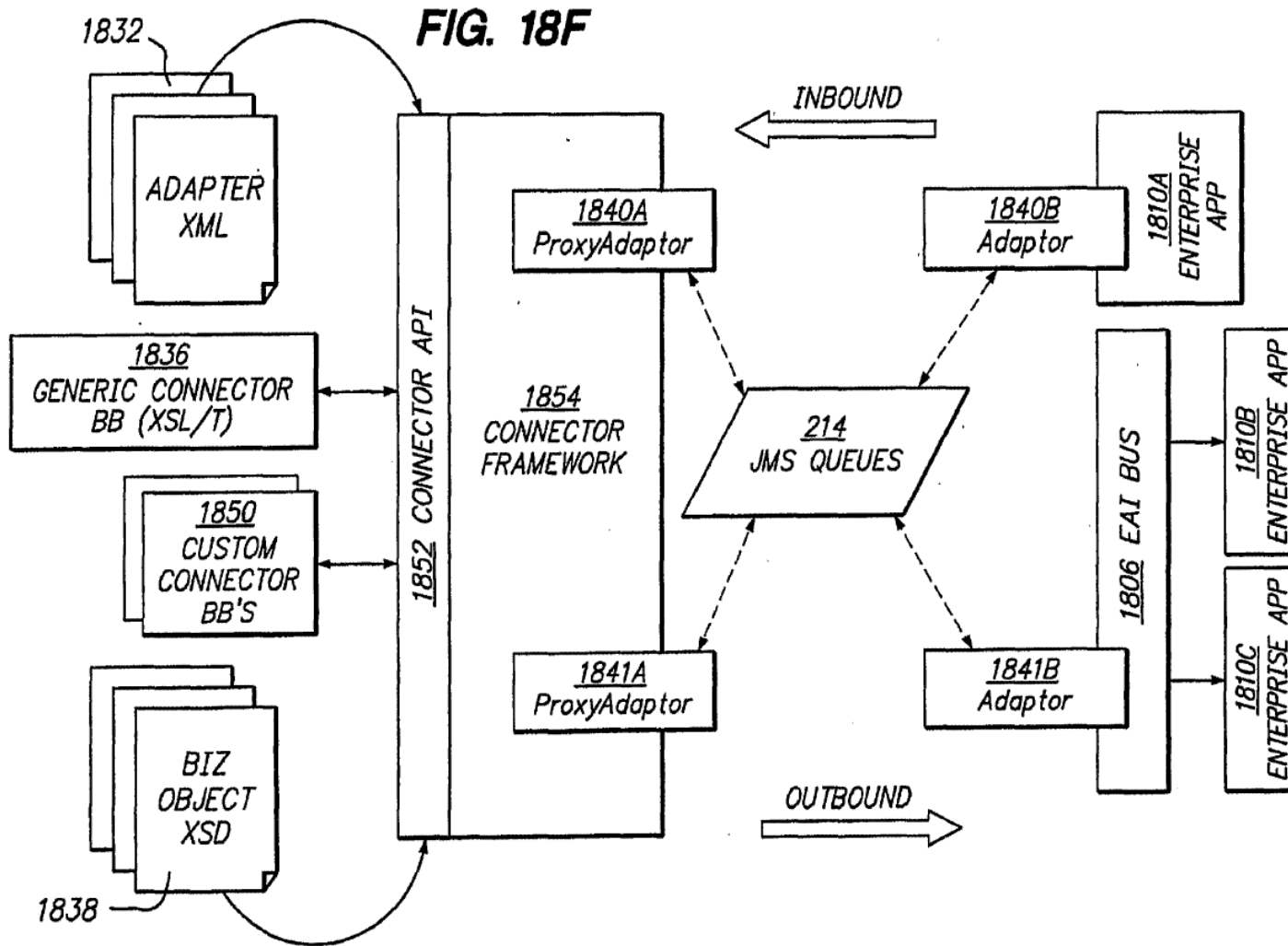
SUBSTITUTE SHEET (RULE 26)

44/70

FIG. 18E



45/70



46/70

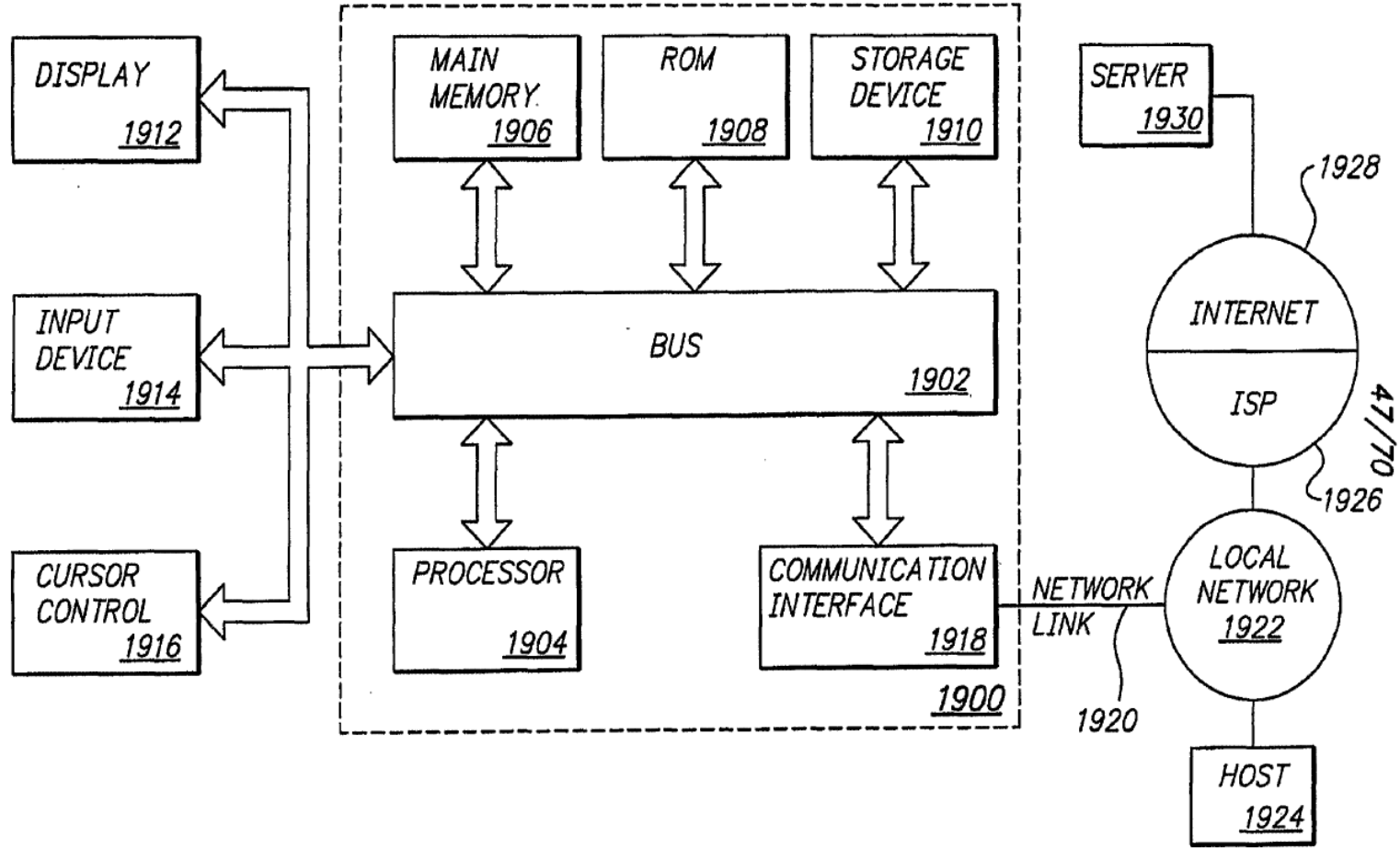
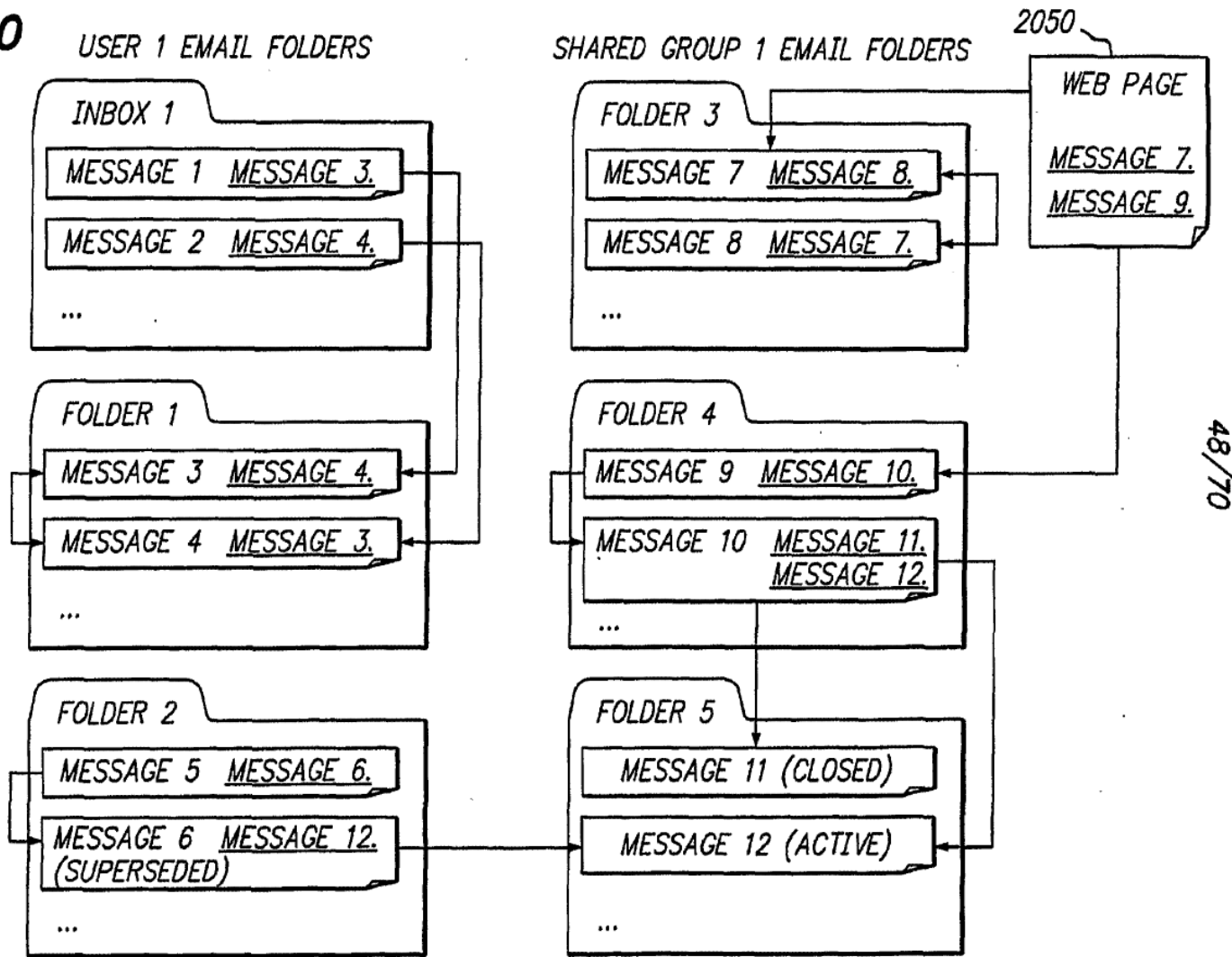


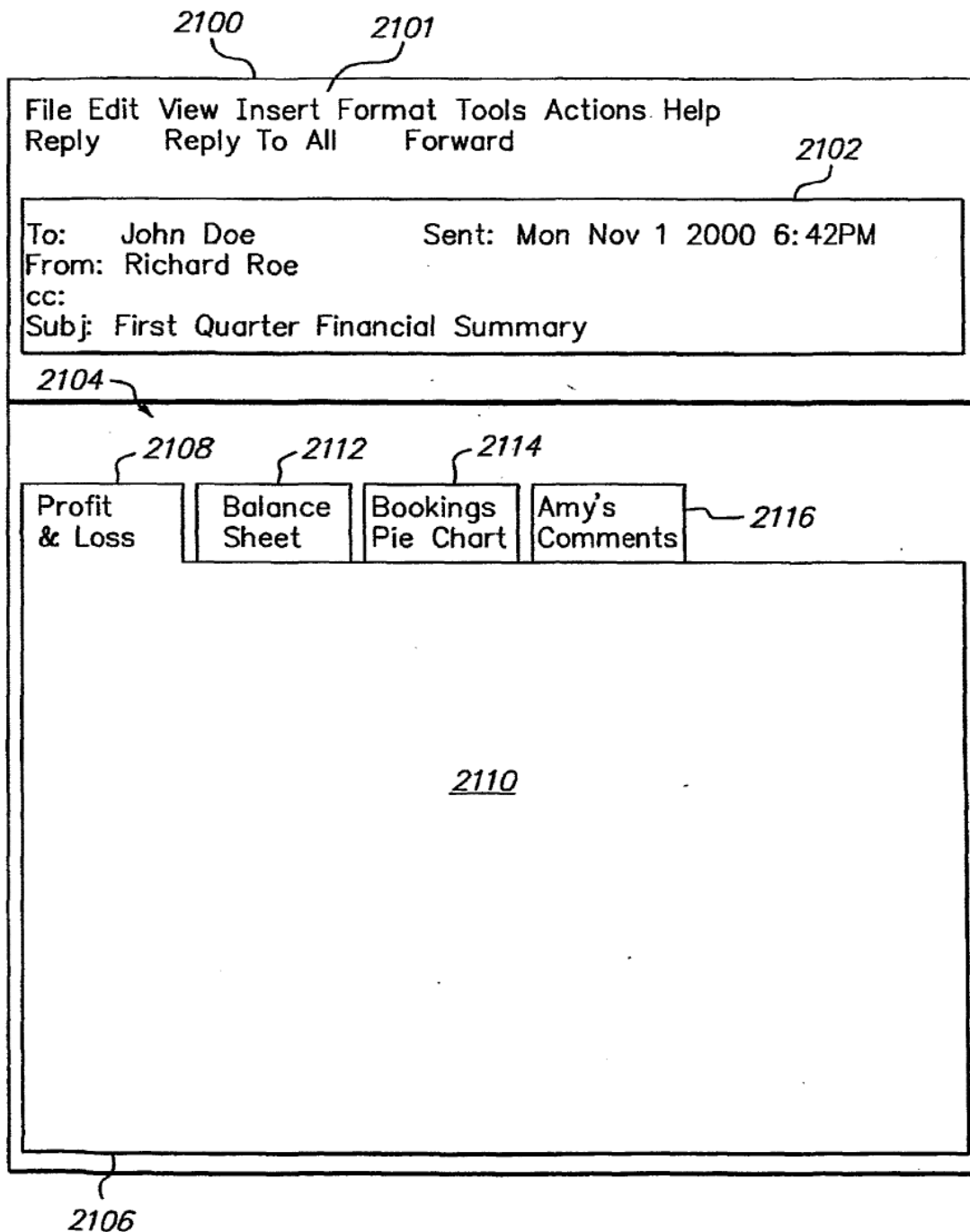
FIG. 19

FIG. 20





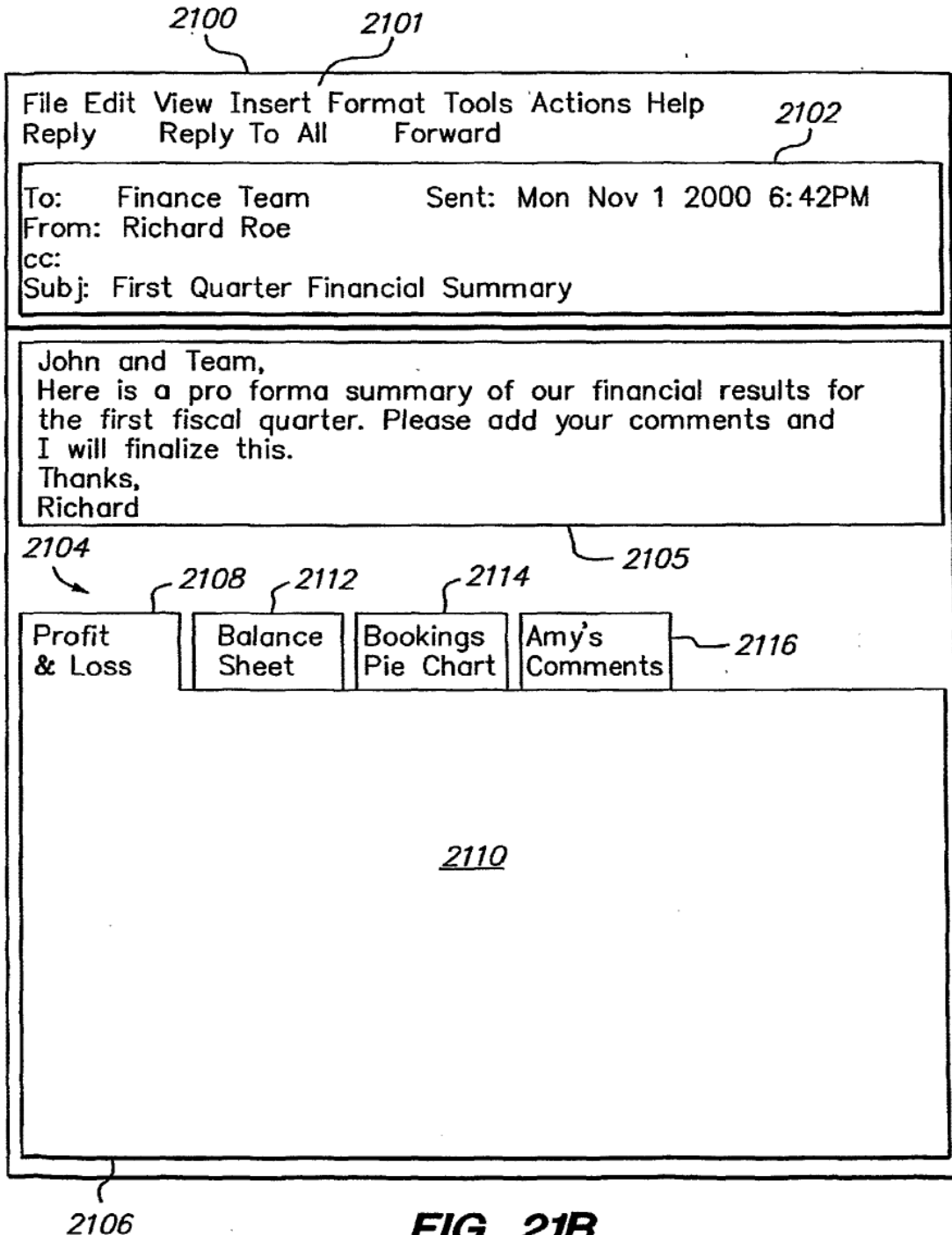
49/70



**FIG. 21A**

SUBSTITUTE SHEET (RULE 26)

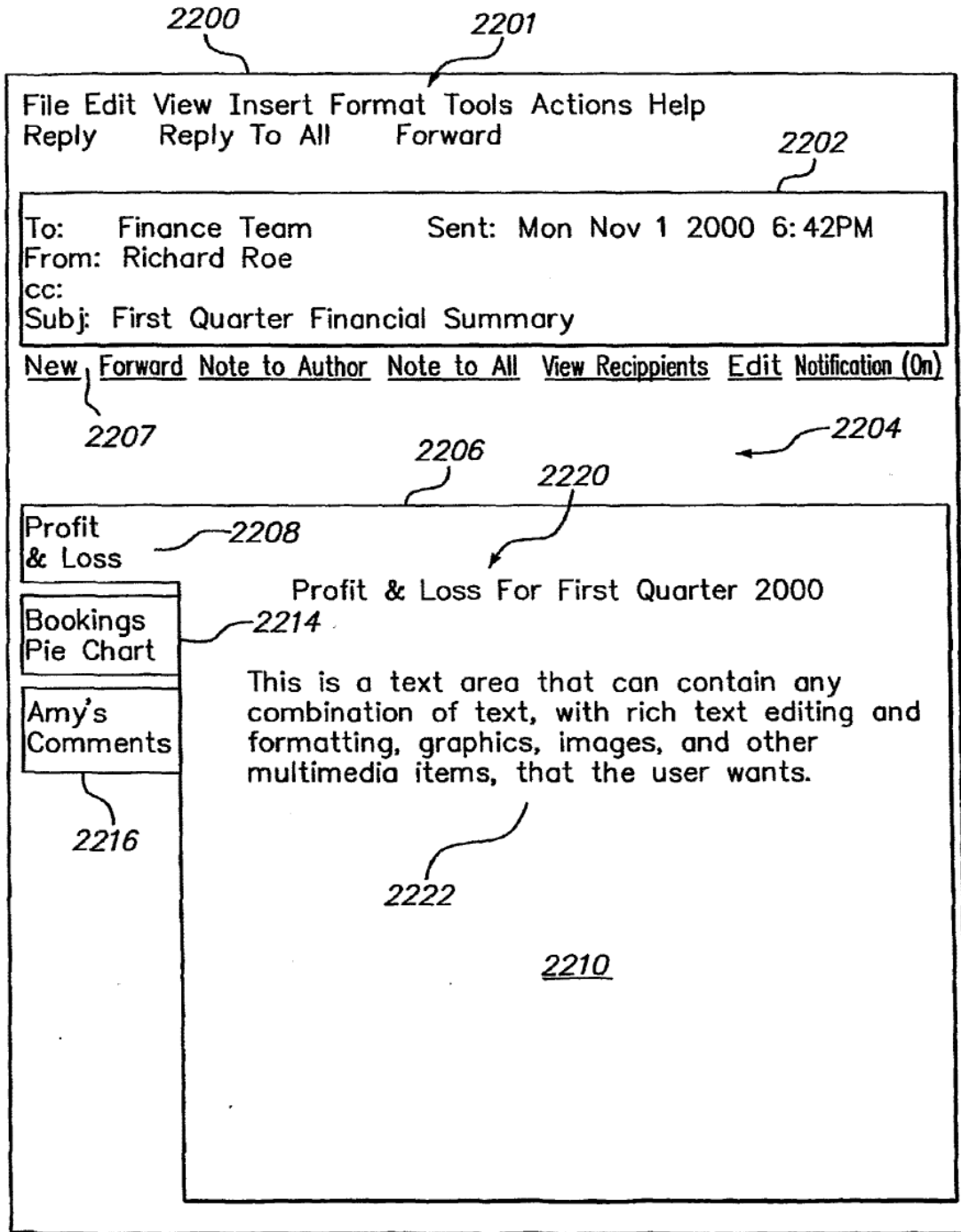
50/70



**FIG. 21B**

SUBSTITUTE SHEET (RULE 26)

51/70



**FIG. 22A**

SUBSTITUTE SHEET (RULE 26)

52/70

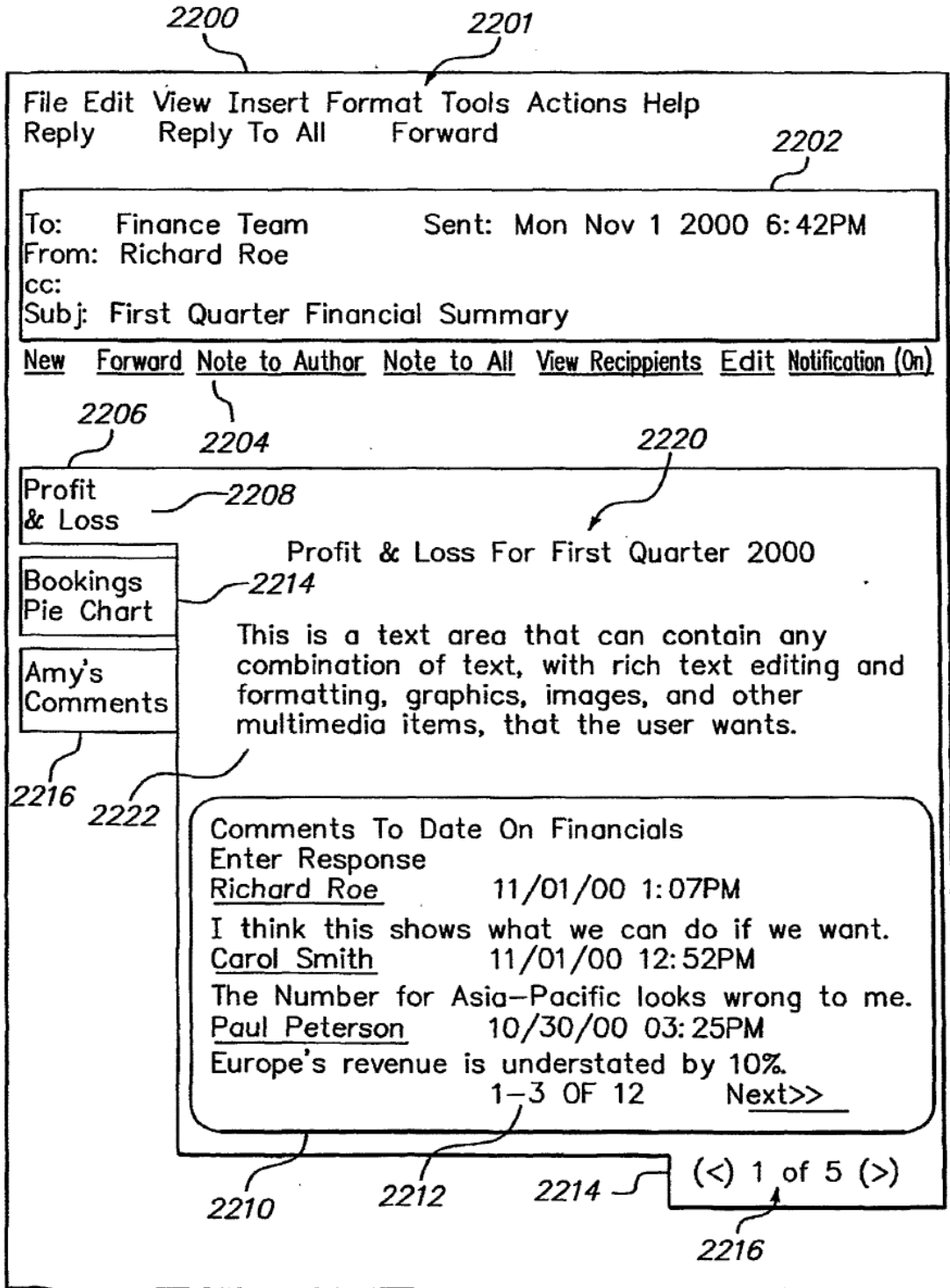
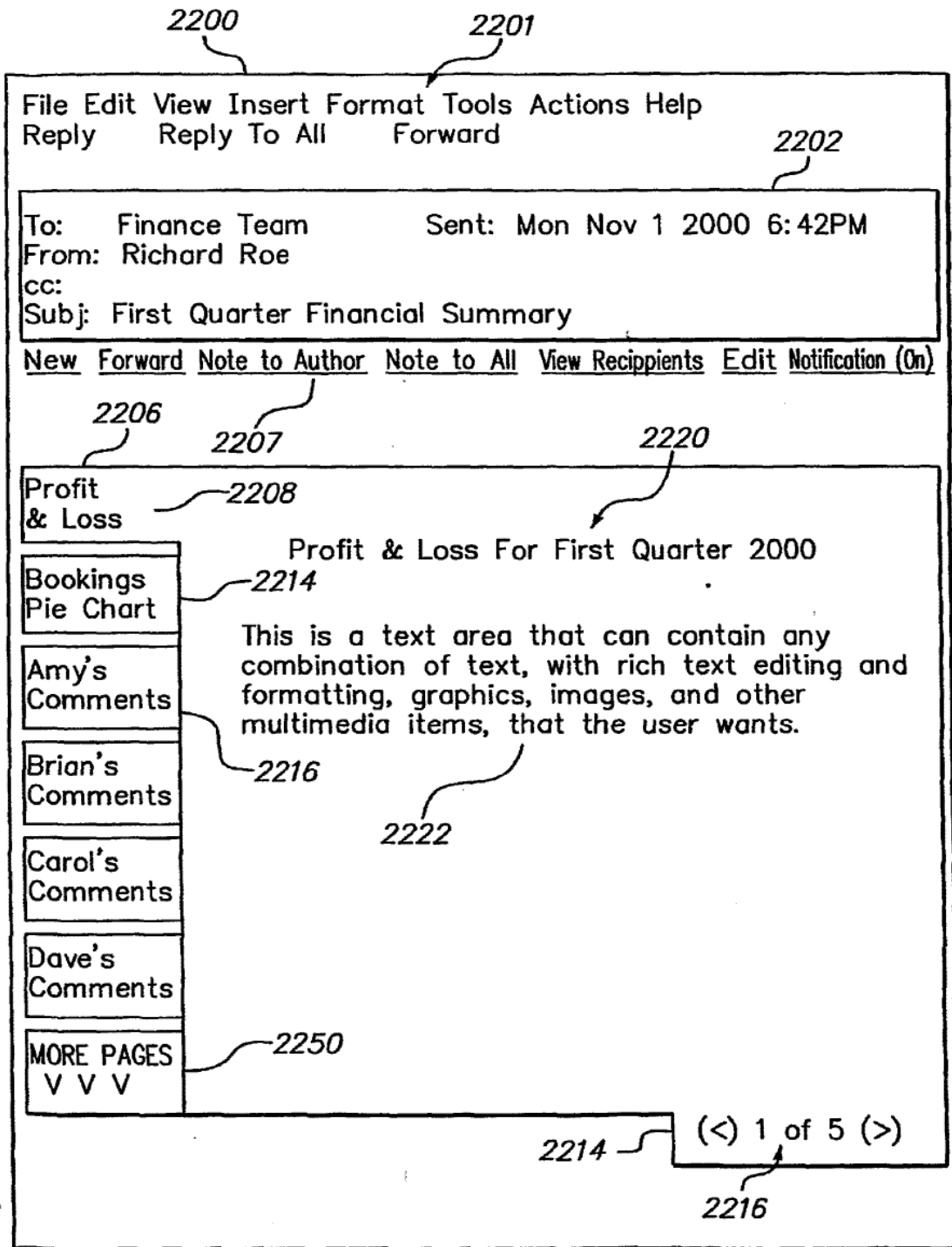


FIG. 22B

SUBSTITUTE SHEET (RULE 26)

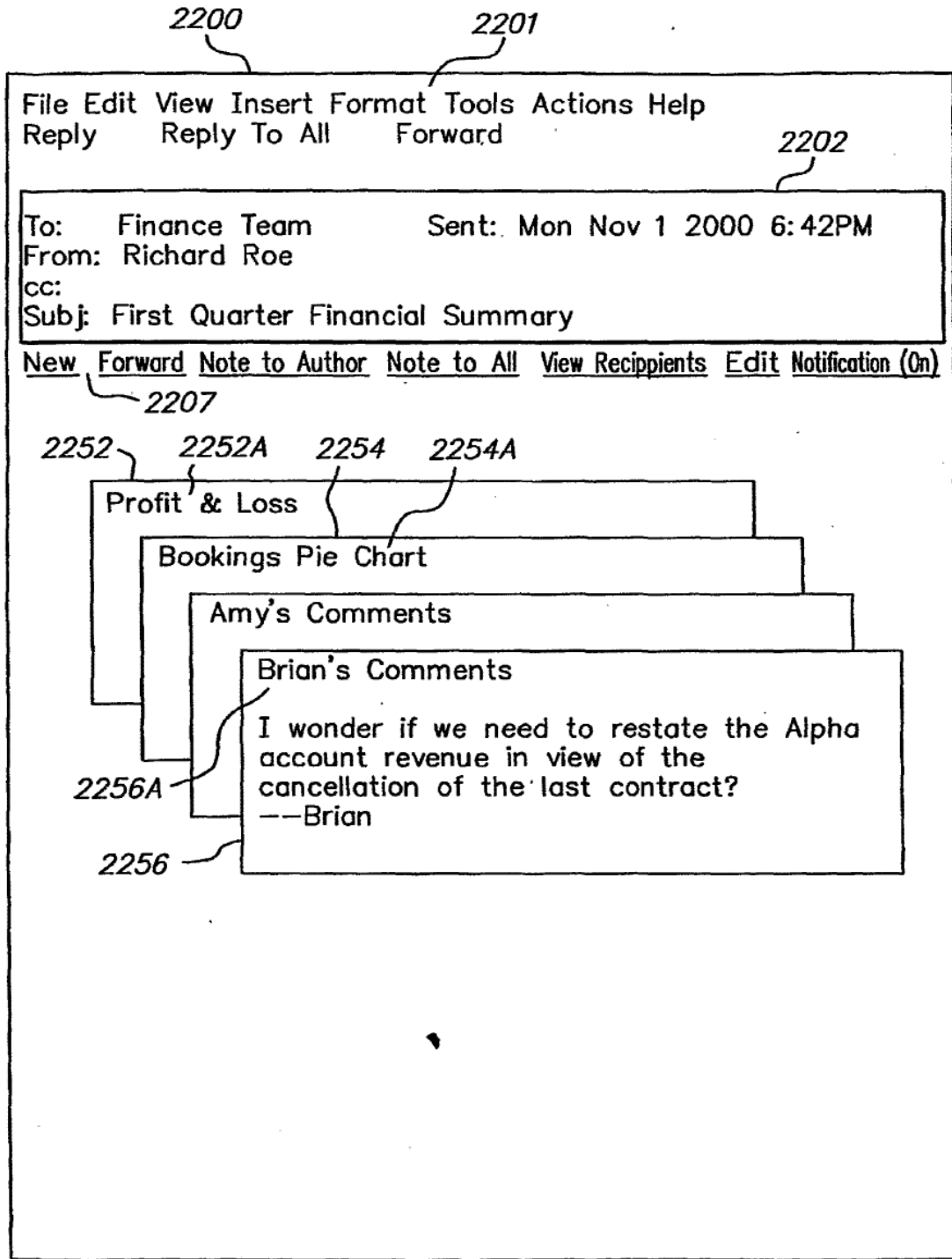
53/70



**FIG. 22C**

SUBSTITUTE SHEET (RULE 26)

54/70

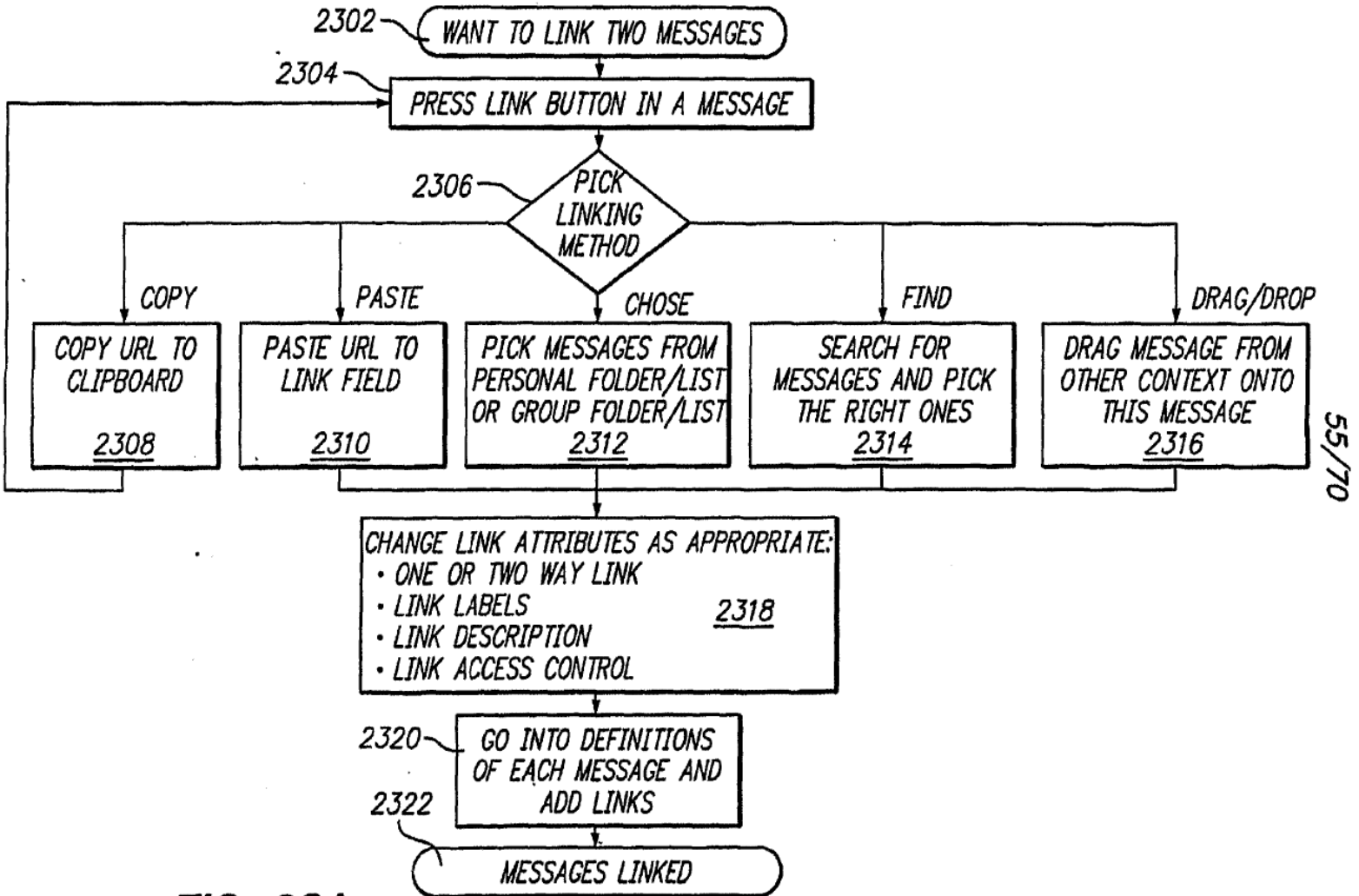


2204

**FIG. 22D**

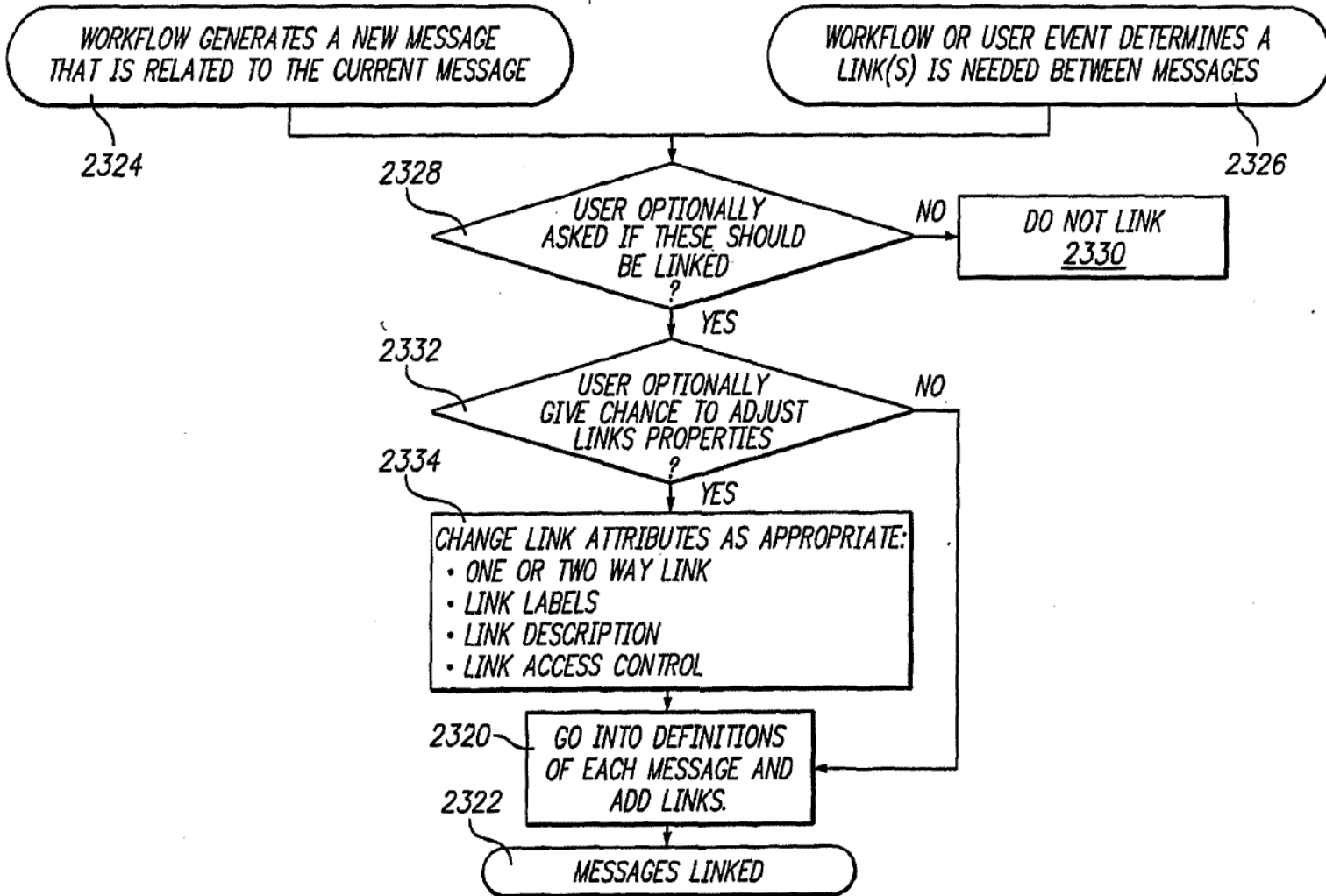
SUBSTITUTE SHEET (RULE 26)

SUBSTITUTE SHEET (RULE 26)



55/70

FIG. 23A



**FIG. 23B**

SUBSTITUTE SHEET (RULE 26)



SUBSTITUTE SHEET (RULE 26)

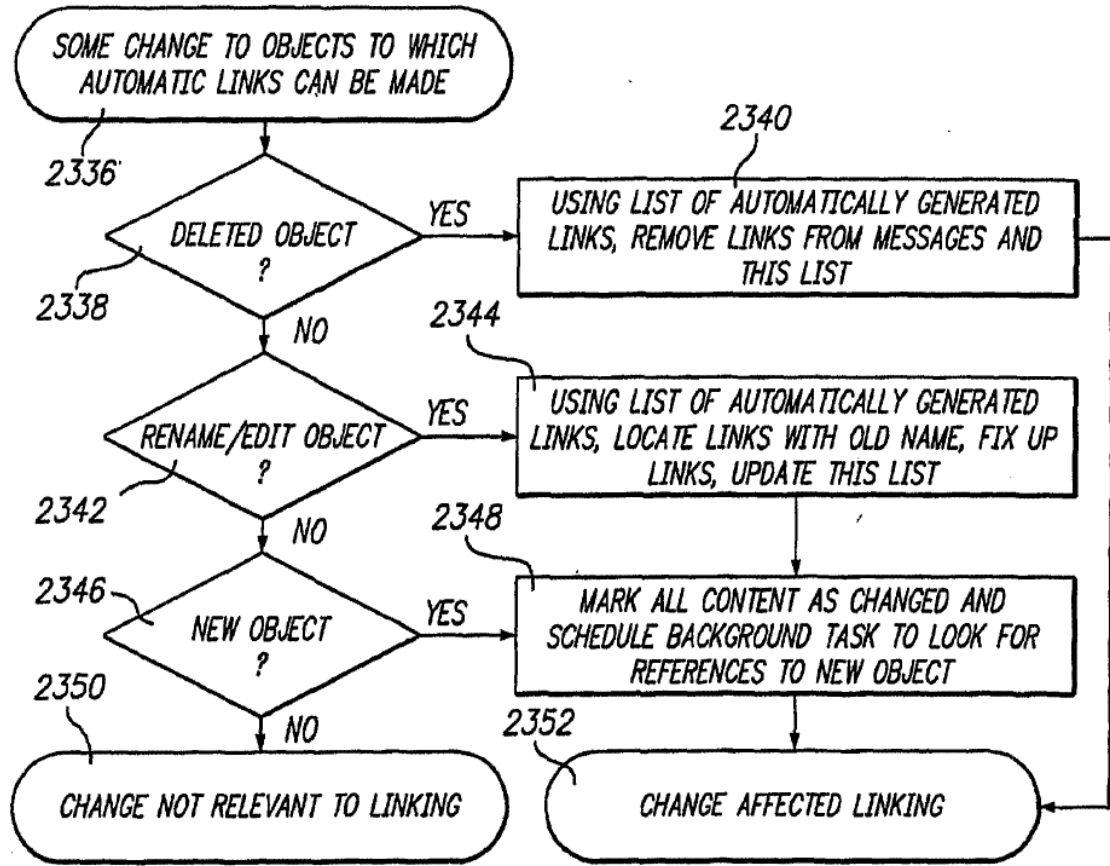


FIG. 23C

58/70

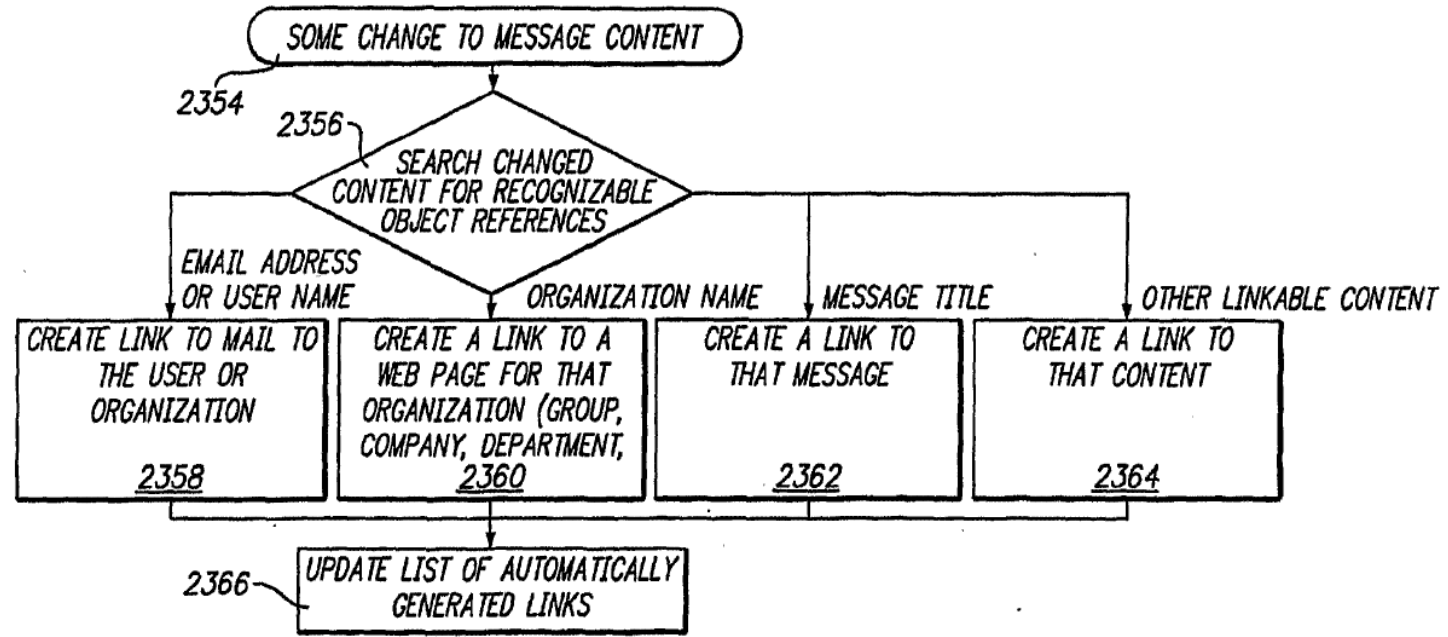


FIG. 23D

SUBSTITUTE SHEET (RULE 26)

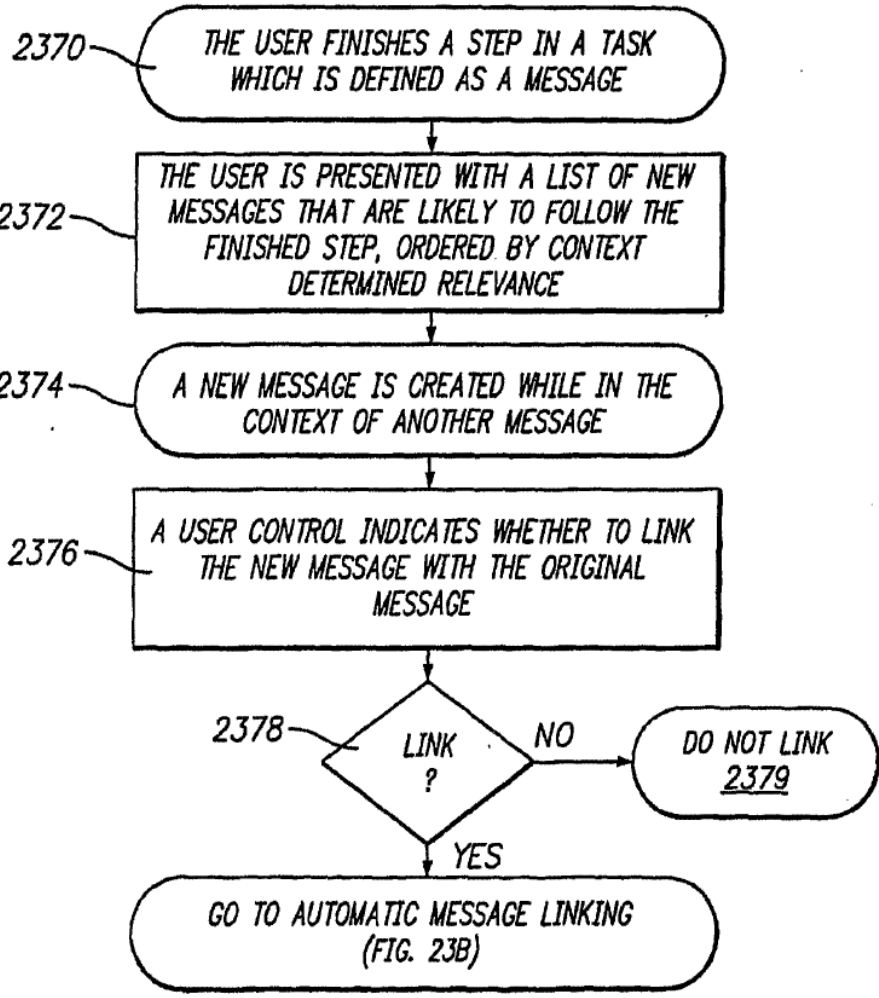


FIG. 23E

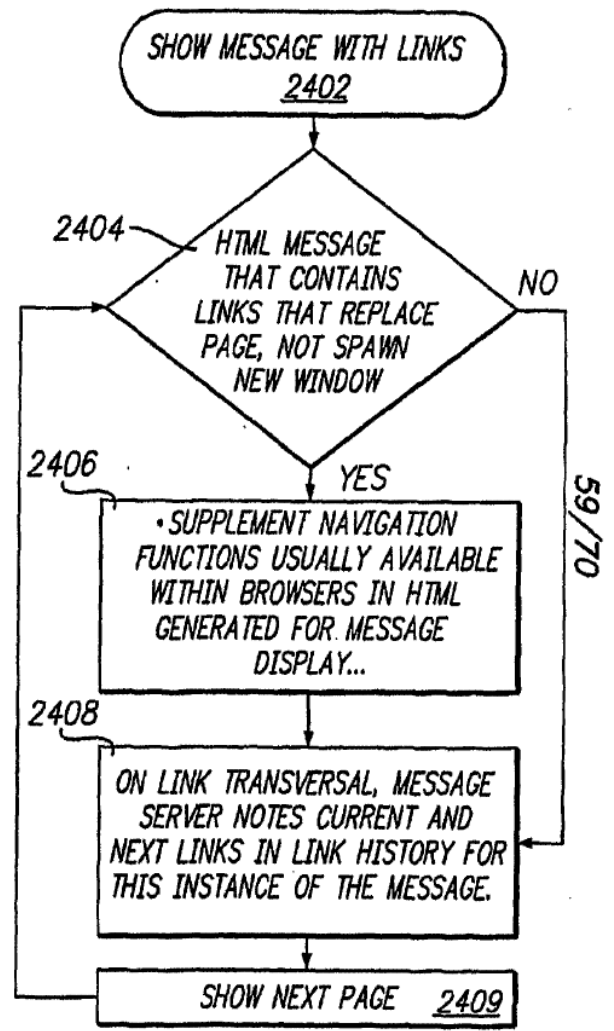
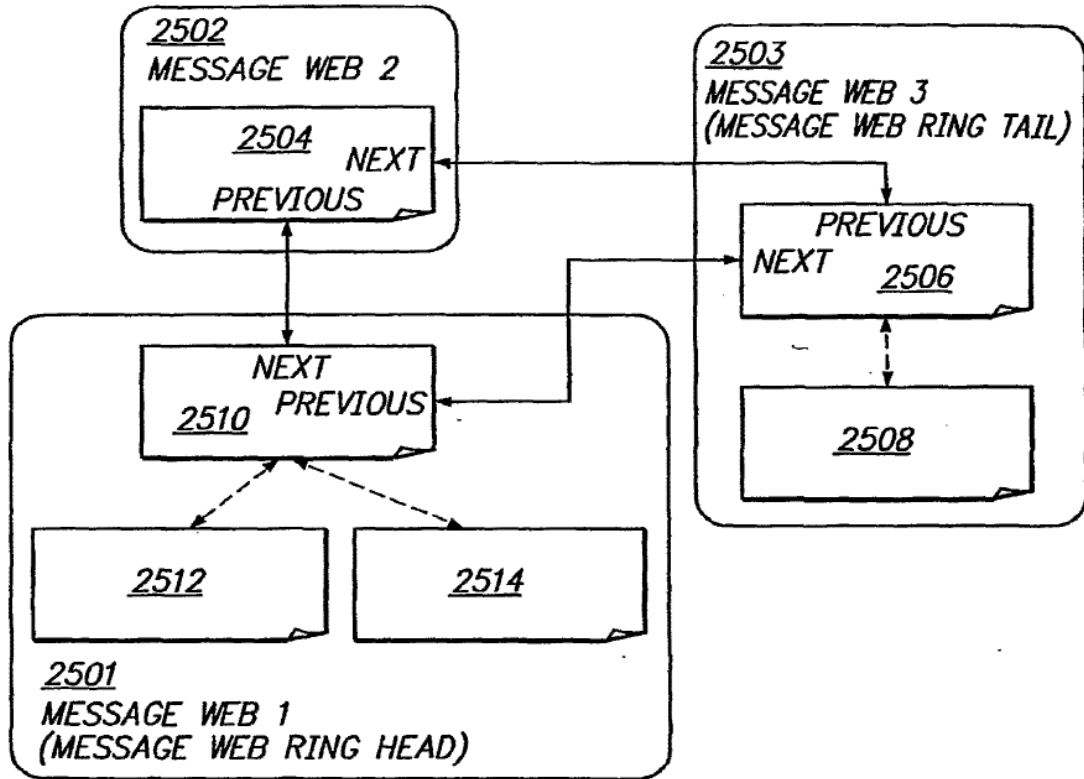


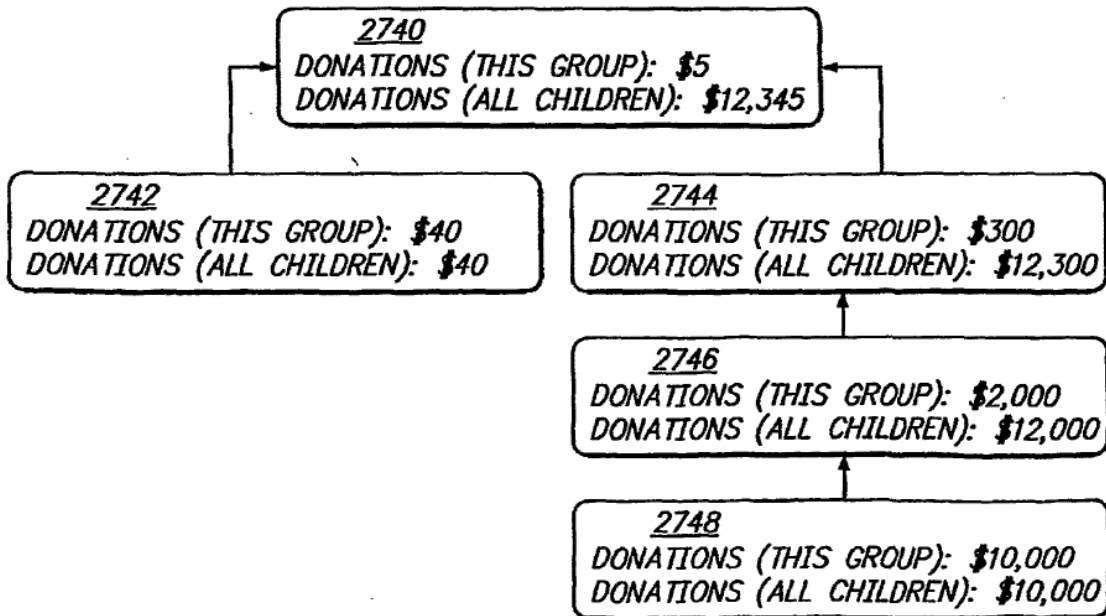
FIG. 24

**FIG. 25A**

60/70



**FIG. 27C**



SUBSTITUTE SHEET (RULE 26)

FIG. 25B

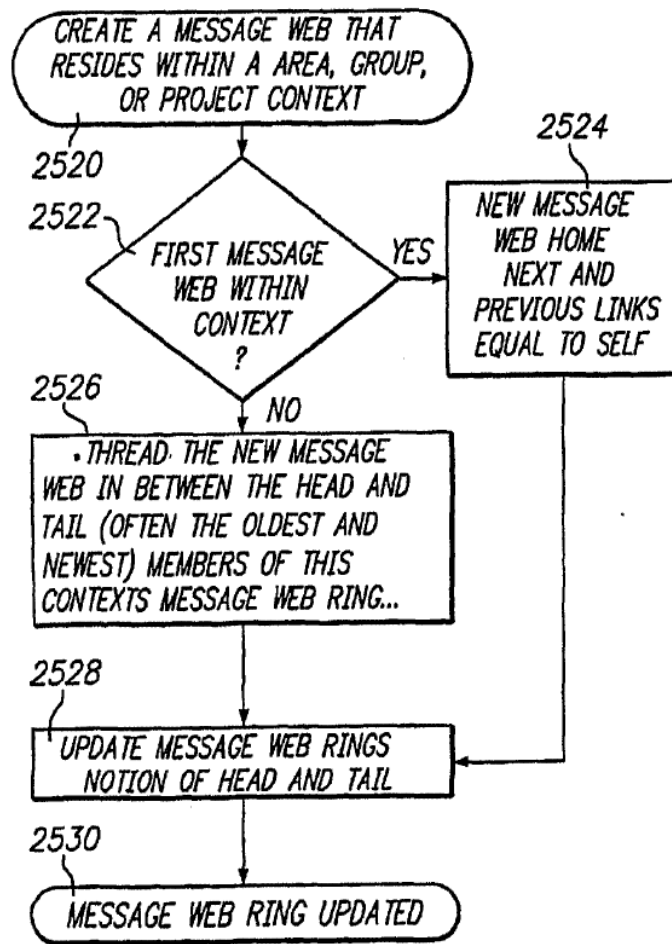
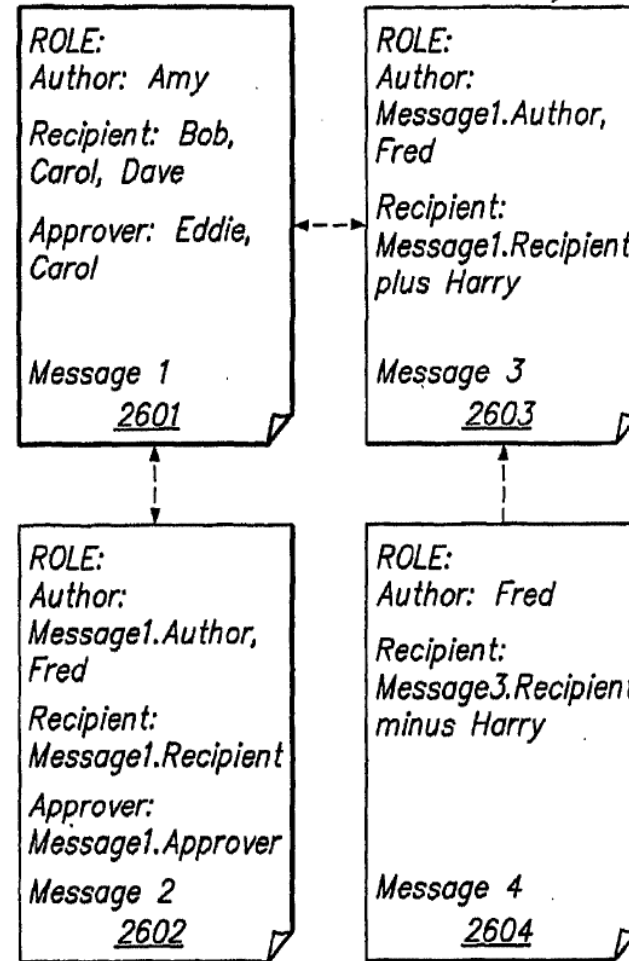
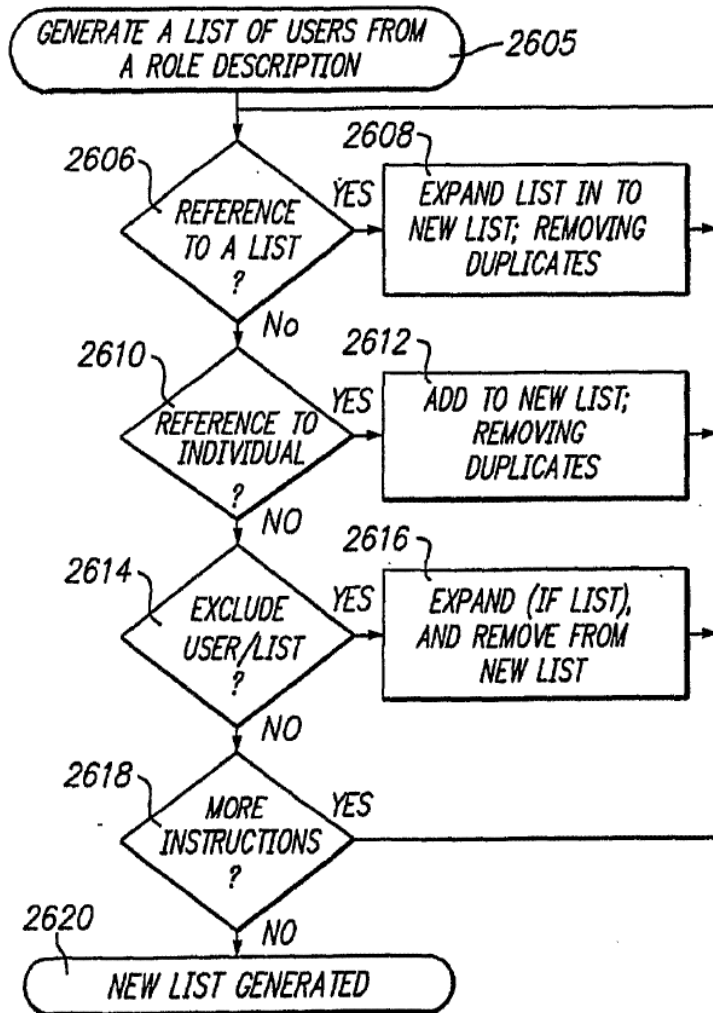


FIG. 26A



**FIG. 26B**



**FIG. 27A**

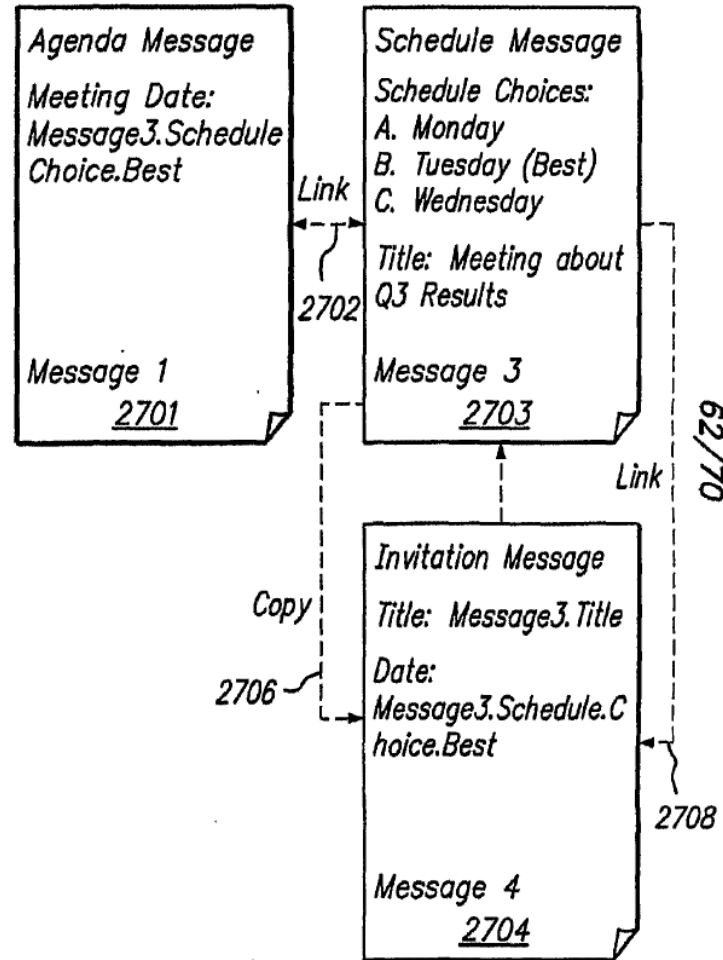
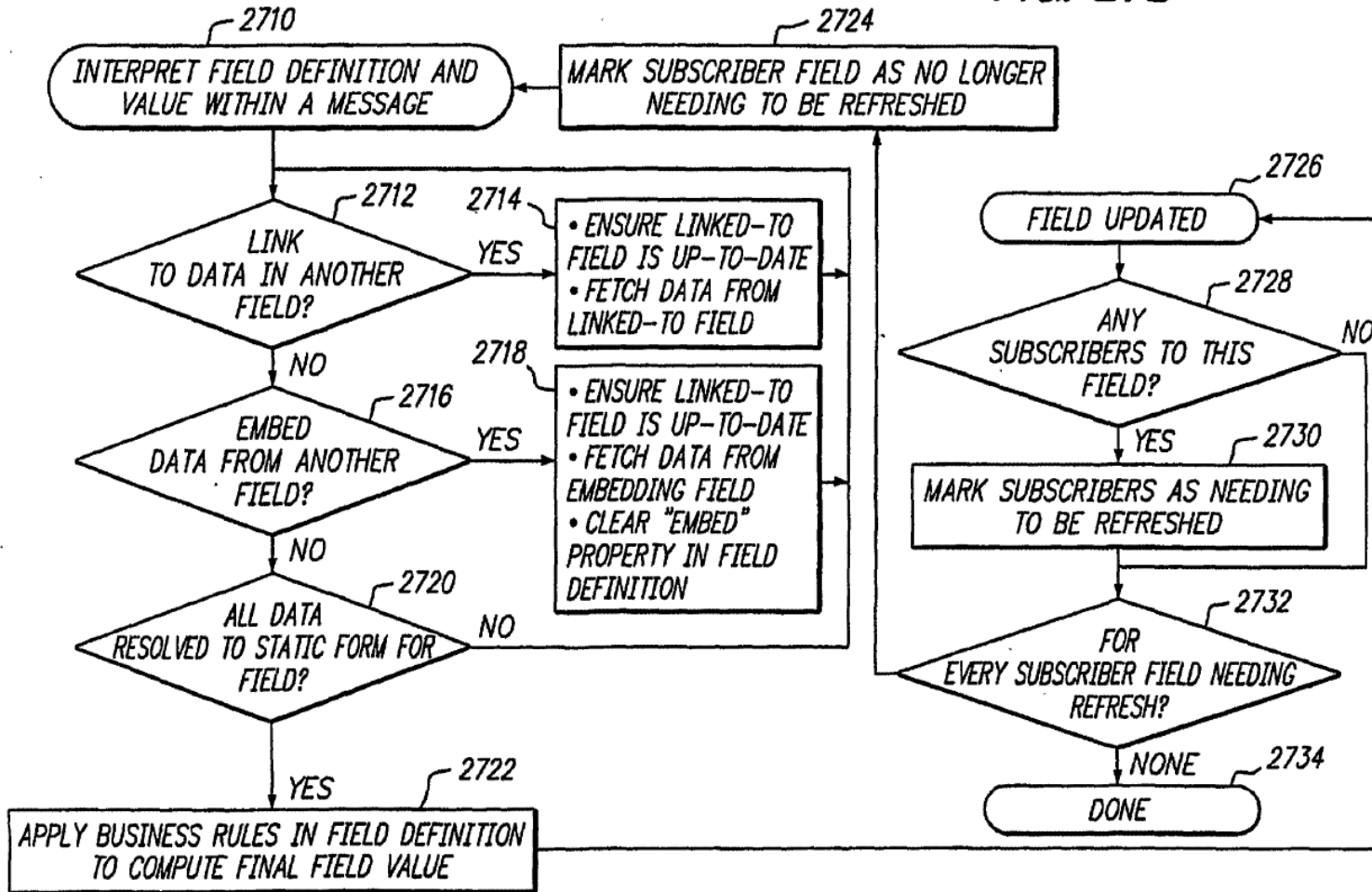


FIG. 27B



63/70

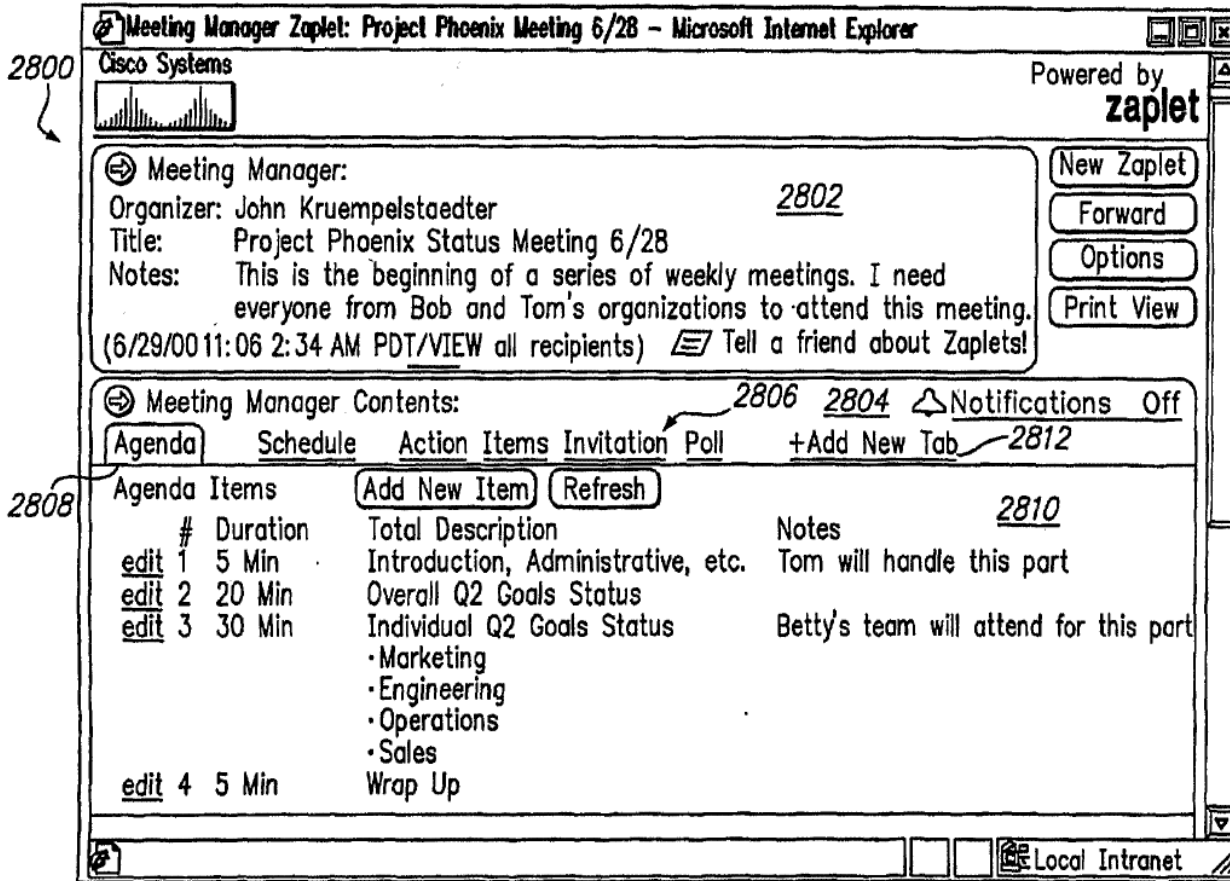


FIG. 28A



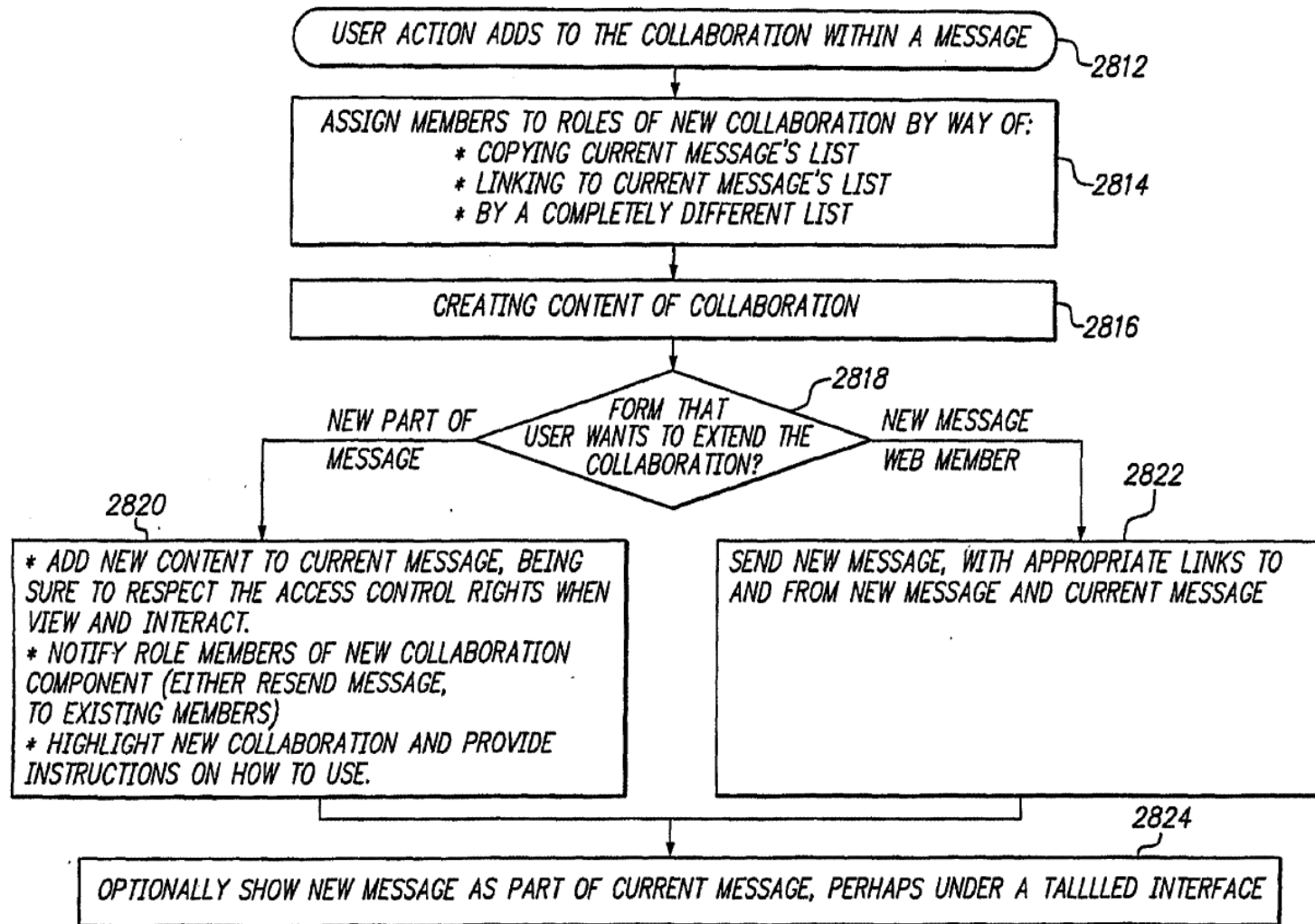


FIG. 28B

SUBSTITUTE SHEET (RULE 26)

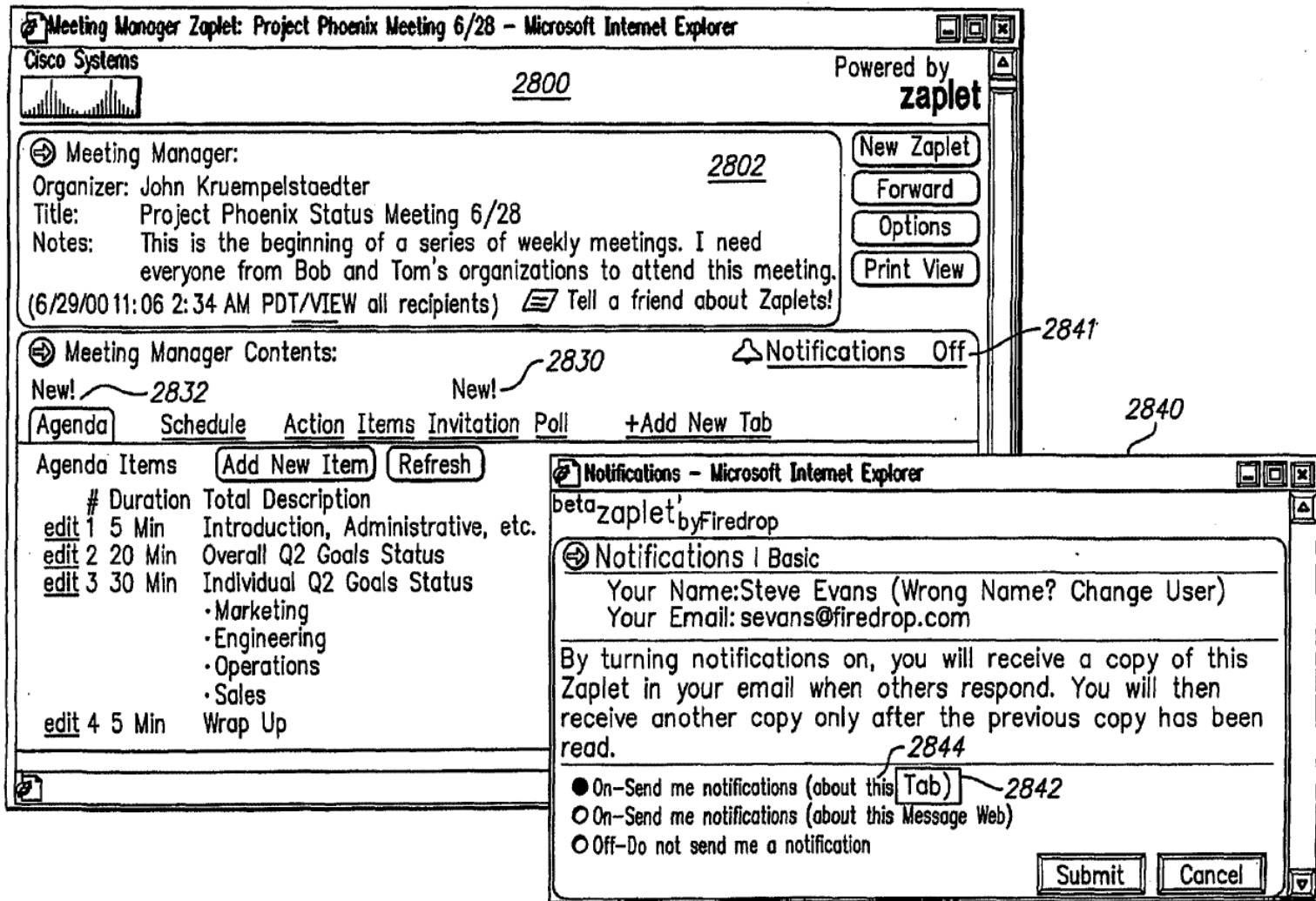


FIG. 28C

66/70

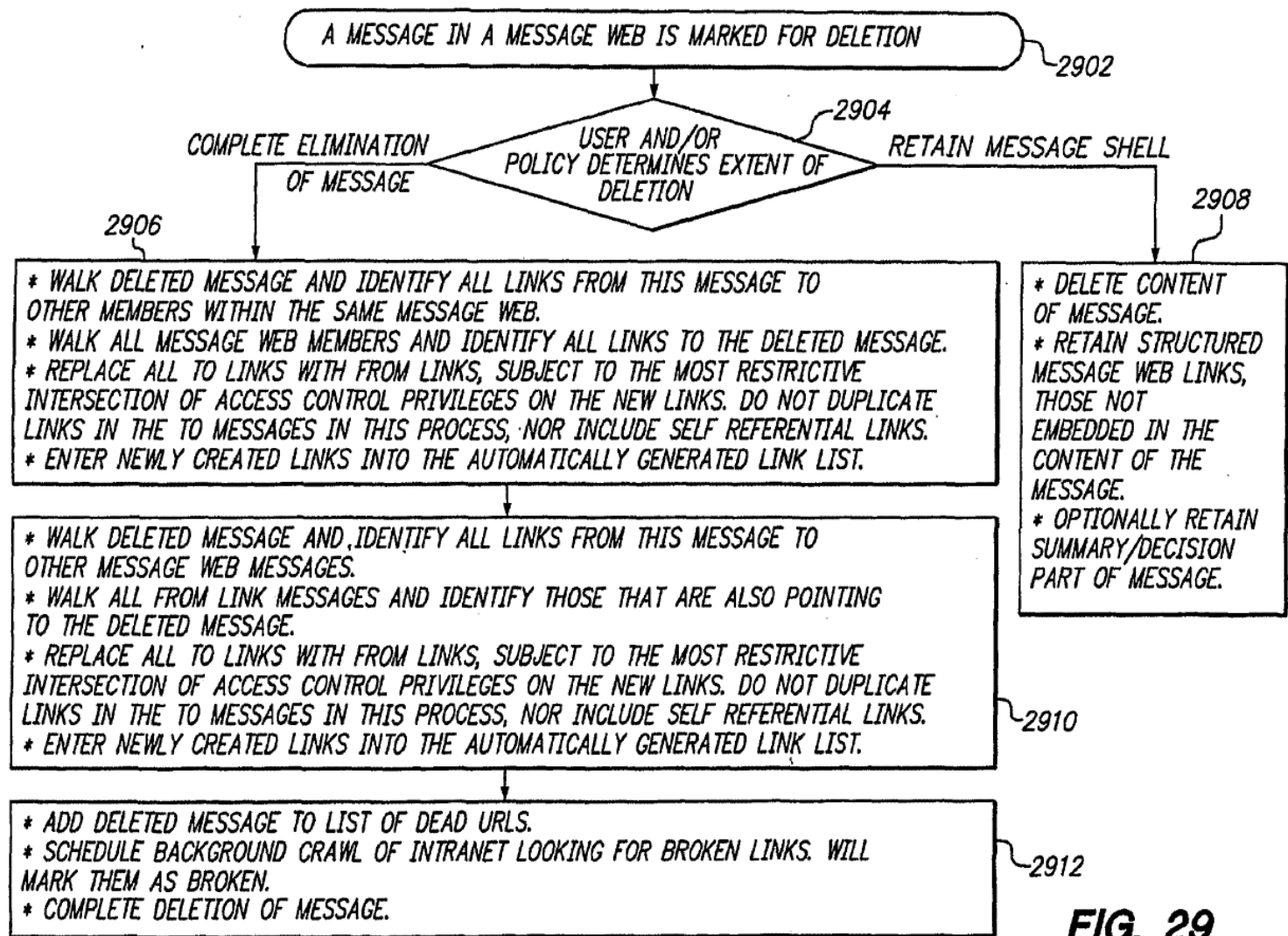


FIG. 29

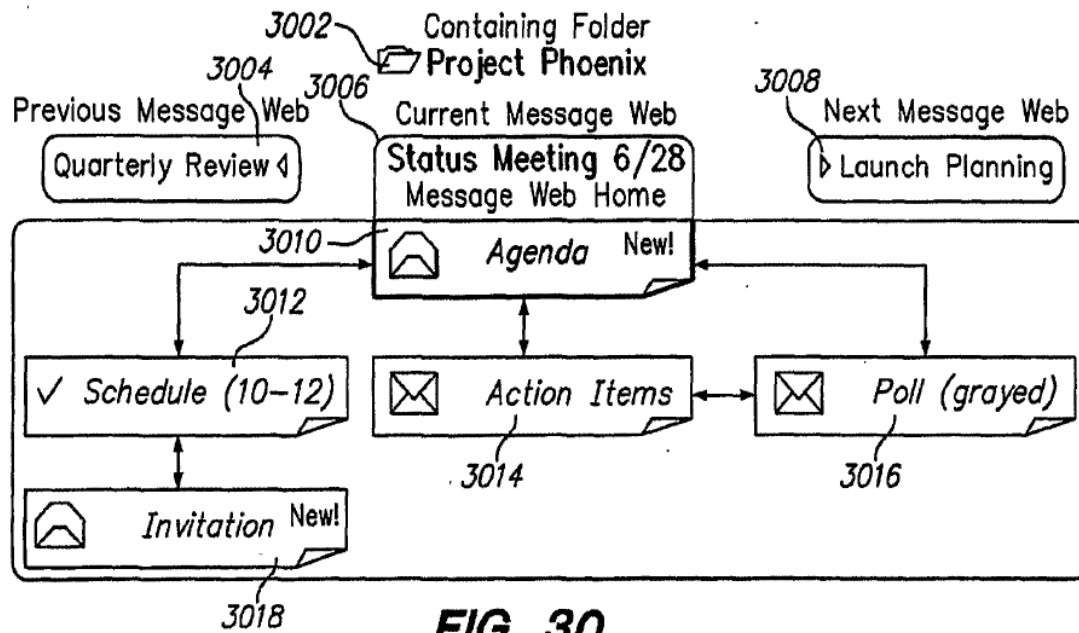
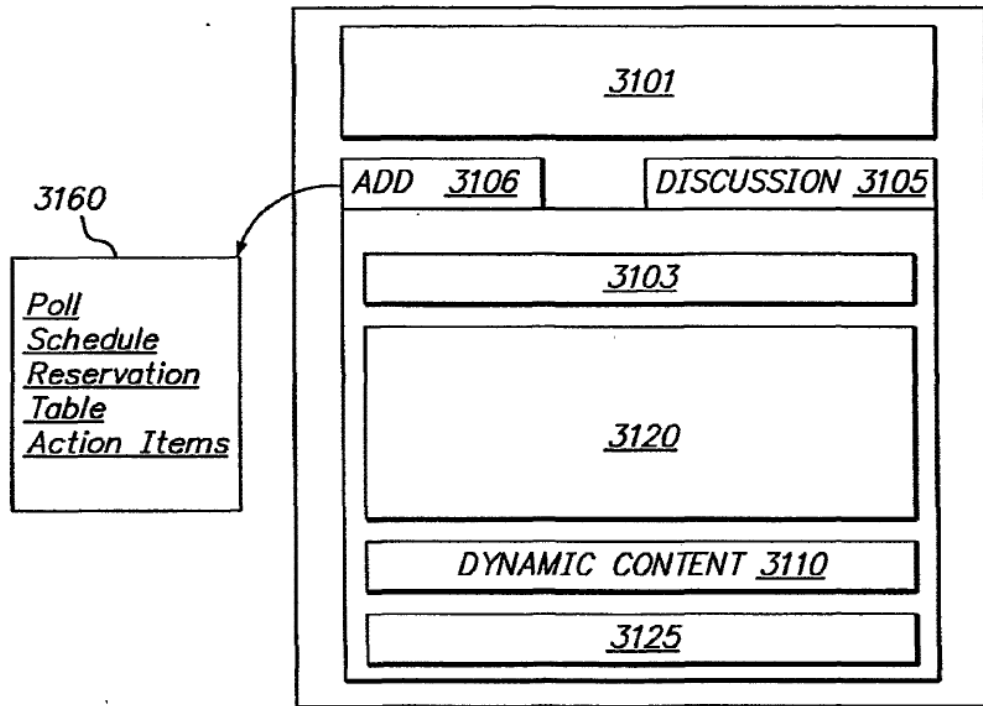


FIG. 30

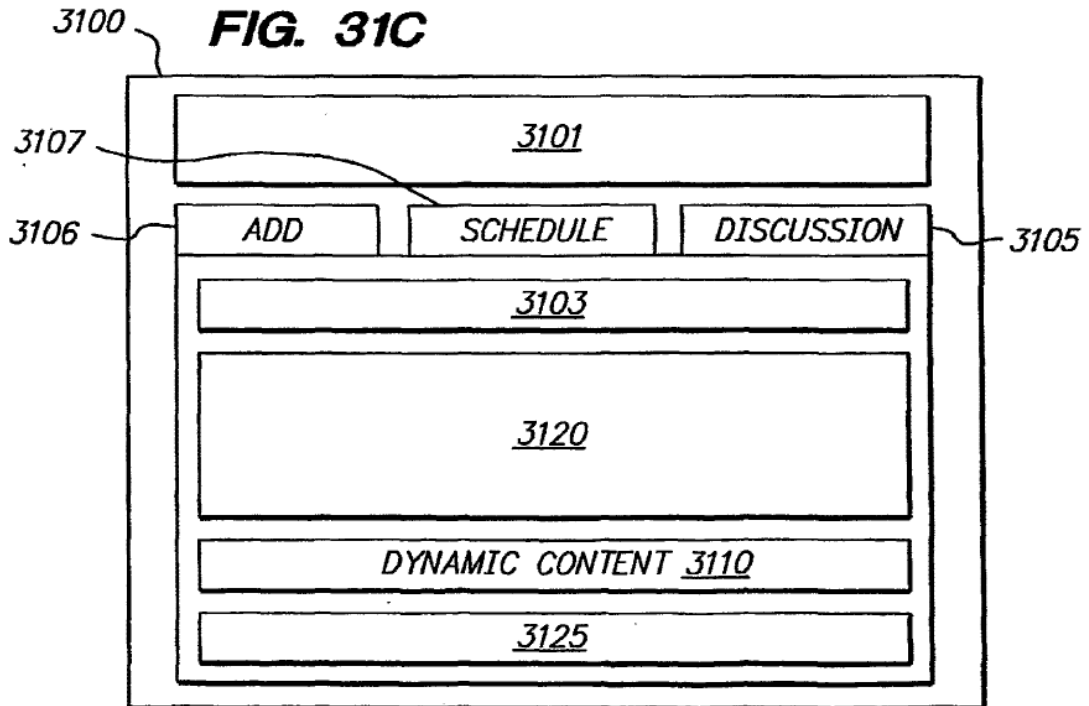
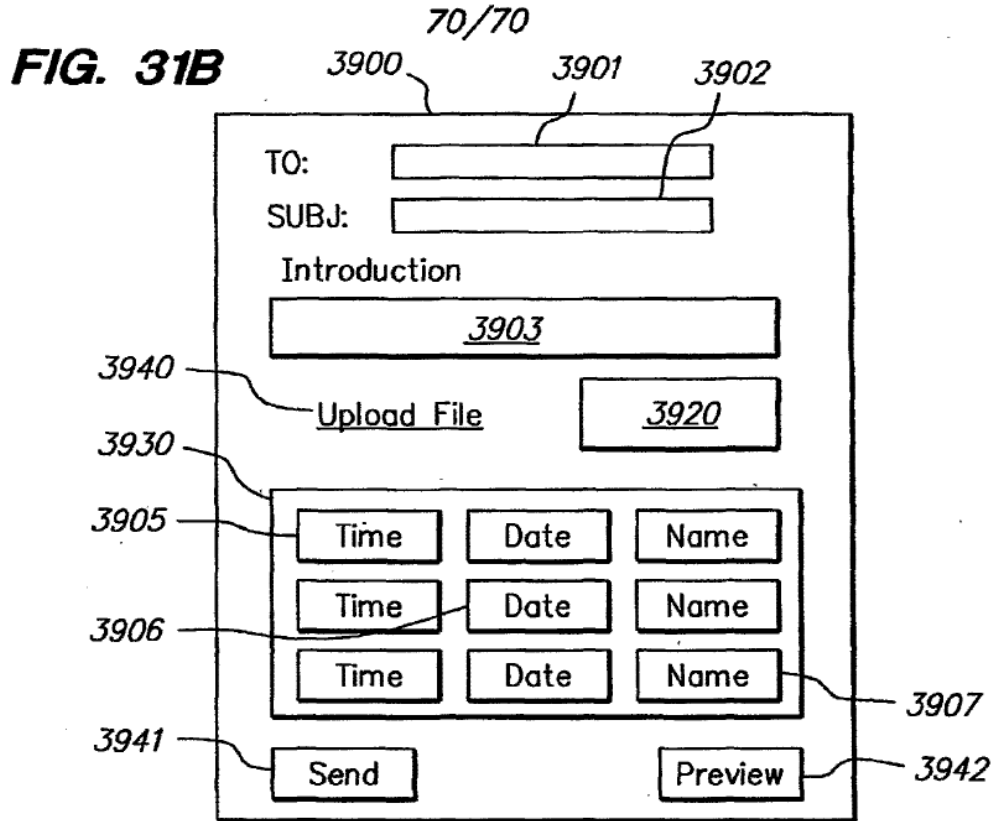
68/70

69/70

**FIG. 31A**



SUBSTITUTE SHEET (RULE 26)



SUBSTITUTE SHEET (RULE 26)

## Electronic Acknowledgement Receipt

<b>EFS ID:</b>	10916443
<b>Application Number:</b>	13111734
<b>International Application Number:</b>	
<b>Confirmation Number:</b>	6081
<b>Title of Invention:</b>	Handheld Electronic Device and Associated Method Providing Time Data in a Messaging Environment
<b>First Named Inventor/Applicant Name:</b>	Gerhard D. Klassen
<b>Customer Number:</b>	91704
<b>Filer:</b>	Brett Joseph Slaney/Judith Martin
<b>Filer Authorized By:</b>	Brett Joseph Slaney
<b>Attorney Docket Number:</b>	70314/00568
<b>Receipt Date:</b>	09-SEP-2011
<b>Filing Date:</b>	19-MAY-2011
<b>Time Stamp:</b>	16:10:21
<b>Application Type:</b>	Utility under 35 USC 111(a)

### Payment information:

Submitted with Payment	no
------------------------	----

### File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1		11144-US-CNT3_IDS.pdf	318087 <small>47af5ef41298218c40269339813adec2cae19ea4</small>	yes	3

Multipart Description/PDF files in .zip description					
Document Description			Start	End	
Transmittal Letter			1	2	
Information Disclosure Statement (IDS) Form (SB08)			3	3	
<b>Warnings:</b>					
<b>Information:</b>					
2	Foreign Reference	11144-US-CNT3_Foreign_Ref1.pdf	12665346	no	240
			77818810be5e001ff0d9d3c18e1787baddf391e1		
<b>Warnings:</b>					
<b>Information:</b>					
<b>Total Files Size (in bytes):</b>			12983433		
<p><b>This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.</b></p> <p><b><u>New Applications Under 35 U.S.C. 111</u></b>  If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.</p> <p><b><u>National Stage of an International Application under 35 U.S.C. 371</u></b>  If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.</p> <p><b><u>New International Application Filed with the USPTO as a Receiving Office</u></b>  If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.</p>					



**IN THE UNITED STATES PATENT & TRADEMARK OFFICE**

Appl. No.: **13/111,734**  
Applicant: **KLASSEN, Gerhard D. et al.**  
Filed: **May 19, 2011**  
Title: **Handheld Electronic Device and Associated Method Providing Time Data in a Messaging Environment**  
Art Unit: **2457**  
Examiner: **LAI, Michael C.**  
Docket No.: **70314/00568**

Mail Stop Amendment  
U.S. Patent & Trademark Office  
Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

Dear Sir:

**FIRST SUPPLEMENTAL**  
**INFORMATION DISCLOSURE STATEMENT**

Pursuant to the duty to disclose under 37 CFR §1.56, Applicant submits herewith a Form PTO/SB/08 listing references of which the Applicant is aware and which are brought to the attention of the Examiner. In accordance with 37 CFR §1.98(a)(2), a copy of each foreign patent document and non-patent reference document listed in the enclosed Form PTO/SB/08 is submitted herewith.

Pursuant to 35 USC §120, this application relies on the earlier filing date(s) of the following prior application(s):

<u>Serial Number</u>	<u>Filing Date</u>
10/944,925	September 20, 2004

The filing of this IDS shall not be construed as a representation that a search has been made, an admission that the information cited is, or is considered to be, material for patentability, or that no other material information exists. This filing shall not be construed as an admission against interest in any matter.

This IDS is being submitted pursuant to 37 CFR 1.97(c) prior to the issuance of a final action or a Notice of Allowance. Applicant hereby certifies that each item of information contained in the present Information Disclosure Statement was cited in a communication from a foreign Patent


Application No. 13/111,734

Office in a counterpart foreign application not more than 3 months prior to the filing of the present Statement. Accordingly, no fee is believed to be due for consideration of the documents submitted herewith.

Applicant respectfully requests consideration of the items listed and requests the Examiner to return a copy of the attached Form PTO/SB/08 after being marked as being considered by the Examiner.

Respectfully submitted,

Date: Sep. 9/11

  
Brett J. Slaney  
Registration No. 58,772  
Agent for Applicant

**BLAKE, CASSELS & GRAYDON LLP**  
199 Bay Street  
Suite 2800, Commerce Court West  
Toronto, Ontario, M5L 1A9  
Canada

Tel 416-863-2518  
Fax 416-863-2653

BSL/jm

(✓) encl.



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

Table with 4 columns: APPLICATION NUMBER (13/111,734), FILING OR 371(C) DATE (05/19/2011), FIRST NAMED APPLICANT (Gerhard D. Klassen), ATTY. DOCKET NO./TITLE (70314/00568)

CONFIRMATION NO. 6081

PUBLICATION NOTICE



91704
Blake, Cassels & Graydon LLP
199 BAY STREET , SUITE 4000
COMMERCE COURT WEST
TORONTO, ON M5L 1A9
CANADA

Title: Handheld Electronic Device and Associated Method Providing Time Data in a Messaging Environment

Publication No. US-2011-0216072-A1

Publication Date: 09/08/2011

NOTICE OF PUBLICATION OF APPLICATION

The above-identified application will be electronically published as a patent application publication pursuant to 37 CFR 1.211, et seq. The patent application publication number and publication date are set forth above.

The publication may be accessed through the USPTO's publically available Searchable Databases via the Internet at www.uspto.gov. The direct link to access the publication is currently http://www.uspto.gov/patft/.

The publication process established by the Office does not provide for mailing a copy of the publication to applicant. A copy of the publication may be obtained from the Office upon payment of the appropriate fee set forth in 37 CFR 1.19(a)(1). Orders for copies of patent application publications are handled by the USPTO's Office of Public Records. The Office of Public Records can be reached by telephone at (703) 308-9726 or (800) 972-6382, by facsimile at (703) 305-8759, by mail addressed to the United States Patent and Trademark Office, Office of Public Records, Alexandria, VA 22313-1450 or via the Internet.

In addition, information on the status of the application, including the mailing date of Office actions and the dates of receipt of correspondence filed in the Office, may also be accessed via the Internet through the Patent Electronic Business Center at www.uspto.gov using the public side of the Patent Application Information and Retrieval (PAIR) system. The direct link to access this status information is currently http://pair.uspto.gov/. Prior to publication, such status information is confidential and may only be obtained by applicant using the private side of PAIR.

Further assistance in electronically accessing the publication, or about PAIR, is available by calling the Patent Electronic Business Center at 1-866-217-9197.

Office of Data Management, Application Assistance Unit (571) 272-4000, or (571) 272-4200, or 1-888-786-0101



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

Table with 5 columns: APPLICATION NO., FILING DATE, FIRST NAMED INVENTOR, ATTORNEY DOCKET NO., CONFIRMATION NO.
13/111,734 05/19/2011 Gerhard D. Klassen 70314/00568 6081

91704 7590 08/09/2011
Blake, Cassels & Graydon LLP
199 BAY STREET, SUITE 4000
COMMERCE COURT WEST
TORONTO, ON M5L 1A9
CANADA

Table with 1 column: EXAMINER

LAI, MICHAEL C

Table with 2 columns: ART UNIT, PAPER NUMBER

2457

Table with 2 columns: NOTIFICATION DATE, DELIVERY MODE

08/09/2011

ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

rimpatent@blakes.com
brett.slaney@blakes.com
portfolioprossecution@rim.com



### DETAILED ACTION

1. This office action is responsive to communication filed on 5/19/2011. Claims 1-24 have been examined.

#### *Priority*

2. This application is a continuation of U.S. Patent Application No. 10/944,925 filed on September 20, 2004, now Patent No. 7,970,849, which claims the benefit of U.S. Provisional Application No. 60/504,379, filed on September 19, 2003.

#### *Specification*

3. The disclosure is objected to because of the following informalities: reference to US Patent No. 7,970,849 should be provided in the "Cross Reference to Related Applications" section.

#### *Claim Rejections - 35 USC § 101*

4. 35 U.S.C. 101 reads as follows:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

5. Claims 17-24 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

Claims 17-24 recite the limitation of "computer readable medium" in line 1.

The broadest reasonable interpretation of a claim drawn to a computer readable medium typically covers forms of non-transitory tangible media and transitory propagating signals *per se* in view of the ordinary and customary meaning of computer readable media, **particularly when the specification is silent**. See

MPEP 2111.01. These claims are rejected under 35 U.S.C. § 101 as covering non-statutory subject matter. *See In re Nuijten*, 500 F.3d 1346, 1356-57 (Fed. Cir. 2007) (transitory embodiments are not directed to statutory subject matter) and *Interim Examination Instructions for Evaluating Subject Matter Eligibility Under 35 U.S.C. § 101*, Aug. 24, 2009; p. 2.

Suggestion: narrow the claims to cover only statutory embodiments to avoid a rejection under 35 U.S.C. § 101 by adding the limitation “non-transitory” to the claims (e.g., non-transitory computer readable storage medium).

***Claim Rejections - 35 USC § 102***

6. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

7. Claims 1-5, 9-13, and 17-21 are rejected under 35 U.S.C. 102(e) as being anticipated by Appelman et al. (US 7,181,497 B1, hereinafter Appelman).

Regarding claim 1, Appelman discloses a method of displaying an instant message conversation on an electronic device [see at least Fig. 7, User 1; Fig. 16, the user], the instant message conversation comprising a plurality of instant messages exchanged between the electronic device and a second electronic device [see at least Fig. 7, User 2; Fig. 16, mroe1934], the method comprising:

receiving a plurality of incoming instant messages from the second electronic device, each incoming instant message having an incoming textual portion [see at least Figs. 16, 18, "F>" (from) messages; col. 9, lines 35-48];

transmitting a plurality of outgoing instant messages to the second electronic device, each outgoing instant message having an outgoing textual portion [see at least Figs. 16, 18, "T>" (to) messages; col. 9, lines 23-34];

associating each instant message with a corresponding time stamp [see at least Figs. 16, 18; col. 9, lines 23-48];

displaying the incoming textual portion of at least one of the incoming instant messages along with a respective time stamp, the incoming textual portion of each displayed incoming instant message being horizontally aligned at a same first horizontal position [see at least Figs. 16, 18, "F>" (from) messages; col. 9, lines 35-48]; and

displaying the outgoing textual portion of at least one of the outgoing instant messages along with a respective time stamp, the outgoing textual portion of each displayed outgoing instant message being horizontally aligned at a same second horizontal position, the second horizontal position being different from the first horizontal position [see at least Figs. 16, 18, "T>" (to) messages; col. 9, lines 23-34].



Regarding claim 2, Appelman further discloses displaying an incoming symbol with each displayed incoming instant message [see at least Figs. 16, 18, “F>” (from) is the incoming symbol; col. 9, lines 35-48].

Regarding claim 3, Appelman further discloses displaying an outgoing symbol with each displayed outgoing instant message [see at least Figs. 16, 18, “T>” (to) is the outgoing symbol; col. 9, lines 23-34].

Regarding claim 4, Appelman further discloses wherein at least one first time stamp is displayed adjacent to its corresponding incoming instant message [see at least Figs. 16, 18, “13:20:27”; col. 9, lines 35-48].

Regarding claim 5, Appelman further discloses wherein at least one second time stamp is displayed adjacent to its corresponding outgoing instant message [see at least Figs. 16, 18, “13:20:05”; col. 9, lines 23-34].

Regarding claim 9, Appelman discloses an electronic device for displaying an instant message conversation, the instant message conversation comprising a plurality of instant messages exchanged between the electronic device [see at least Fig. 7, User 1; Fig. 16, the user] and a second electronic device [see at least Fig. 7, User 2; Fig. 16, mroe1934], the electronic device comprising:

- a display [Fig. 1, 107, and Fig. 6];
- a memory [Fig. 1, 109]; and
- a processor [Fig. 1, 121] electronically coupled with the display and the memory, the processor configured to:

receive a plurality of incoming instant messages from the second electronic device, each incoming instant message having an incoming textual portion [see at least Figs. 16, 18, "F>" (from) messages; col. 9, lines 35-48];

transmit a plurality of outgoing instant messages to the second electronic device, each outgoing instant message having an outgoing textual portion [see at least Figs. 16, 18, "T>" (to) messages; col. 9, lines 23-34];

associate each instant message with a corresponding time stamp [see at least Figs. 16, 18; col. 9, lines 23-48];

display the incoming textual portion of at least one of the incoming instant messages along with a respective time stamp, the incoming textual portion of each displayed incoming instant message being horizontally aligned at a same first horizontal position [see at least Figs. 16, 18, "F>" (from) messages; col. 9, lines 35-48]; and

display the outgoing textual portion of at least one of the outgoing instant messages along with a respective time stamp, the outgoing textual portion of each displayed outgoing instant message being horizontally aligned at a same second horizontal position, the second horizontal position being different from the first horizontal position [see at least Figs. 16, 18, "T>" (to) messages; col. 9, lines 23-34].

Regarding claim 10, Appelman further discloses wherein the processor is further configured to display an incoming symbol with each displayed incoming

instant message [see at least Figs. 16, 18, “F>” (from) is the incoming symbol; col. 9, lines 35-48].

Regarding claim 11, Appelman further discloses wherein the processor is further configured to display an outgoing symbol with each displayed outgoing instant message [see at least Figs. 16, 18, “T>” (to) is the outgoing symbol; col. 9, lines 23-34].

Regarding claim 12, Appelman further discloses wherein at least one first time stamp is displayed adjacent to its corresponding incoming instant message [see at least Figs. 16, 18, “13:20:27”; col. 9, lines 35-48].

Regarding claim 13, Appelman further discloses wherein at least one second time stamp is displayed adjacent to its corresponding outgoing instant message [see at least Figs. 16, 18, “13:20:05”; col. 9, lines 23-34].

Regarding claim 17, Appelman discloses a computer readable medium comprising computer executable instructions embedded thereon for execution by a processor of an electronic device for displaying an instant message conversation upon a display of the electronic device, the instant message conversation comprising a plurality of instant messages exchanged between the electronic device [see at least Fig. 7, User 1; Fig. 16, the user] and a second electronic device [see at least Fig. 7, User 2; Fig. 16, mroe1934], such that when executed, the processor:

receives a plurality of incoming instant messages from the second electronic device, each incoming instant message having an incoming textual portion [see at least Figs. 16, 18, "F>" (from) messages; col. 9, lines 35-48];

transmits a plurality of outgoing instant messages to the second electronic device, each outgoing instant message having an outgoing textual portion [see at least Figs. 16, 18, "T>" (to) messages; col. 9, lines 23-34];

associates each instant message with a corresponding time stamp [see at least Figs. 16, 18; col. 9, lines 23-48];

displays the incoming textual portion of at least one of the incoming instant messages along with a respective time stamp, the incoming textual portion of each displayed incoming instant message being horizontally aligned at a same first horizontal position [see at least Figs. 16, 18, "F>" (from) messages; col. 9, lines 35-48]; and

displays the outgoing textual portion of at least one of the outgoing instant messages along with a respective time stamp, the outgoing textual portion of each displayed outgoing instant message being horizontally aligned at a same second horizontal position, the second horizontal position being different from the first horizontal position [see at least Figs. 16, 18, "T>" (to) messages; col. 9, lines 23-34].

Regarding claim 18, Appelman further discloses computer instructions such that when executed cause the processor to display an incoming symbol with

each displayed incoming instant message [see at least Figs. 16, 18, “F>” (from) is the incoming symbol; col. 9, lines 35-48].

Regarding claim 19, Appelman further discloses computer instructions such that when executed cause the processor to display an outgoing symbol with each displayed outgoing instant message [see at least Figs. 16, 18, “T>” (to) is the outgoing symbol; col. 9, lines 23-34].

Regarding claim 20, Appelman further discloses wherein at least one first time stamp is displayed adjacent to its corresponding incoming instant message [see at least Figs. 16, 18, “13:20:27”; col. 9, lines 35-48].

Regarding claim 21, Appelman further discloses wherein at least one second time stamp is displayed adjacent to its corresponding outgoing instant message [see at least Figs. 16, 18, “13:20:05”; col. 9, lines 23-34].

***Claim Rejections - 35 USC § 103***

8. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

9. Claims 6-8, 14-16, and 22-24 are rejected under 35 U.S.C. 103(a) as being unpatentable over Appelman et al. (US 7,181,497 B1, hereinafter Appelman) as applied to claim 1, and in view of Yamada (US 6,889,063 B2, hereinafter Yamada).

Regarding claim 6, Appelman discloses the method of claim 1 but is silent about detecting an interruption in the instant message conversation; and refraining from displaying a time stamp associated with a next one of an incoming instant message and an outgoing instant message if the interruption is less than a predetermined duration of time. However, Appelman teaches displaying time stamp on messages [see at least claim 1] and Yamada teaches displaying an in-absence incoming call message on a display if the user of the cellular phone does not answer an incoming call [read as detecting an interruption, see the abstract]. Yamada further discloses starting a timer with a predetermined period of time at the same time the in-absence incoming call message is displayed. When the timer does not expire, no alert tone/vibration/light is outputted to alert the user to the in-absence incoming call [see Figure 2 and col. 2 line 7 through col. 3 line 22, only when the timer expires, an alert tone/vibration/light is outputted to alert the user to the in-absence incoming call]. Thus it would have been obvious to a person with ordinary skill in the art at the time the invention was made to incorporate Yamada's teaching into Appelman's method in order to alert the user only when the timer expires before next message by refraining from displaying a time stamp associated with a next

one of an incoming instant message and an outgoing instant message if the interruption is less than a predetermined duration of time [see the abstract].

Regarding claim 7, Appelman discloses the method of claim 1 but is silent about refraining from displaying at least one of the corresponding time stamps. However, Appelman teaches displaying time stamp on messages [see at least Fig. 12, item 624] and Yamada teaches displaying an in-absence incoming call message on a display if the user of the cellular phone does not answer an incoming call [read as detecting an interruption, see the abstract]. Yamada further discloses starting a timer with a predetermined period of time at the same time the in-absence incoming call message is displayed. When the timer does not expire, no alert tone/vibration/light is outputted to alert the user to the in-absence incoming call [see Figure 2 and col. 2 line 7 through col. 3 line 22, only when the timer expires, an alert tone/vibration/light is outputted to alert the user to the in-absence incoming call]. Thus it would have been obvious to a person with ordinary skill in the art at the time the invention was made to incorporate Yamada's teaching into Appelman's method in order to alert the user only when the timer expires before next message by refraining from displaying at least one of the corresponding time stamps [see the abstract].

Regarding claim 8, Appelman and Yamada disclose the method of claim 7, including wherein the refraining is performed if an amount of time that has lapsed between the at least one corresponding time stamp and a previous corresponding time stamp is less than a predetermined duration of time

[Yamada, see Figure 2 and col. 2 line 7 through col. 3 line 22, only when the timer expires, an alert tone/vibration/light is outputted to alert the user to the in-absence incoming call]. See claim 6 for motivation.

Regarding claim 14, Appelman discloses the electronic device of claim 9 but is silent about wherein the processor is further configured to: detect an interruption in the instant message conversation; and refrain from displaying a time stamp associated with a next one of an incoming instant message and an outgoing instant message if the interruption is less than a predetermined duration of time. However, Appelman teaches displaying time stamp on messages [see at least Fig. 12, item 624] and Yamada teaches displaying an in-absence incoming call message on a display if the user of the cellular phone does not answer an incoming call [read as detecting an interruption, see the abstract]. Yamada further discloses starting a timer with a predetermined period of time at the same time the in-absence incoming call message is displayed. When the timer does not expire, no alert tone/vibration/light is outputted to alert the user to the in-absence incoming call [see Figure 2 and col. 2 line 7 through col. 3 line 22, only when the timer expires, an alert tone/vibration/light is outputted to alert the user to the in-absence incoming call]. Thus it would have been obvious to a person with ordinary skill in the art at the time the invention was made to incorporate Yamada's teaching into Appelman's electronic device in order to alert the user only when the timer expires before next message by refraining from displaying a time stamp associated with a next one of an incoming instant



message and an outgoing instant message if the interruption is less than a predetermined duration of time [see the abstract].

Regarding claim 15, Appelman discloses the electronic device of claim 9 but is silent about wherein the processor is further configured to refrain from displaying at least one of the corresponding time stamps. However, Appelman teaches displaying time stamp on messages [see at least Fig. 12, item 624] and Yamada teaches displaying an in-absence incoming call message on a display if the user of the cellular phone does not answer an incoming call [read as detecting an interruption, see the abstract]. Yamada further discloses starting a timer with a predetermined period of time at the same time the in-absence incoming call message is displayed. When the timer does not expire, no alert tone/vibration/light is outputted to alert the user to the in-absence incoming call [see Figure 2 and col. 2 line 7 through col. 3 line 22, only when the timer expires, an alert tone/vibration/light is outputted to alert the user to the in-absence incoming call]. Thus it would have been obvious to a person with ordinary skill in the art at the time the invention was made to incorporate Yamada's teaching into Appelman's electronic device in order to alert the user only when the timer expires before next message by refraining from displaying at least one of the corresponding time stamps [see the abstract].

Regarding claim 16, Appelman and Yamada disclose the electronic device of claim 15, including wherein the processor refrains from displaying the at least one of the corresponding time stamps if an amount of time that has lapsed

between the at least one corresponding time stamp and a previous corresponding time stamp is less than a predetermined duration of time [Yamada, see Figure 2 and col. 2 line 7 through col. 3 line 22, only when the timer expires, an alert tone/vibration/light is outputted to alert the user to the in-absence incoming call]. See claim 14 for motivation.

Regarding claim 22, Appelman discloses the computer readable medium of claim 17 but is silent about computer instructions such that when executed cause the processor to: detect an interruption in the instant message conversation; and refrain from displaying a time stamp associated with a next one of an incoming instant message and an outgoing instant message if the interruption is less than a predetermined duration of time. However, Appelman teaches displaying time stamp on messages [see at least Fig. 12, item 624] and Yamada teaches displaying an in-absence incoming call message on a display if the user of the cellular phone does not answer an incoming call [read as detecting an interruption, see the abstract]. Yamada further discloses starting a timer with a predetermined period of time at the same time the in-absence incoming call message is displayed. When the timer does not expire, no alert tone/vibration/light is outputted to alert the user to the in-absence incoming call [see Figure 2 and col. 2 line 7 through col. 3 line 22, only when the timer expires, an alert tone/vibration/light is outputted to alert the user to the in-absence incoming call]. Thus it would have been obvious to a person with ordinary skill in the art at the time the invention was made to incorporate Yamada's teaching into

Appelman's computer readable medium in order to alert the user only when the timer expires before next message by refraining from displaying a time stamp associated with a next one of an incoming instant message and an outgoing instant message if the interruption is less than a predetermined duration of time [see the abstract].

Regarding claim 23, Appelman discloses the computer readable medium of claim 17 but is silent about comprising computer instructions such that when executed cause the processor to refrain from displaying at least one of the corresponding time stamps. However, Appelman teaches displaying time stamp on messages [see at least Fig. 12, item 624] and Yamada teaches displaying an in-absence incoming call message on a display if the user of the cellular phone does not answer an incoming call [read as detecting an interruption, see the abstract]. Yamada further discloses starting a timer with a predetermined period of time at the same time the in-absence incoming call message is displayed. When the timer does not expire, no alert tone/vibration/light is outputted to alert the user to the in-absence incoming call [see Figure 2 and col. 2 line 7 through col. 3 line 22, only when the timer expires, an alert tone/vibration/light is outputted to alert the user to the in-absence incoming call]. Thus it would have been obvious to a person with ordinary skill in the art at the time the invention was made to incorporate Yamada's teaching into Appelman's computer readable medium in order to alert the user only when the timer expires before next

message by refraining from displaying at least one of the corresponding time stamps [see the abstract].

Regarding claim 24, Appelman and Yamada disclose the computer readable medium of claim 23, including wherein the computer instructions that when executed cause the processor to refrain from displaying at least one of the corresponding time stamps are executed if an amount of time that has lapsed between the at least one corresponding time stamp and a previous corresponding time stamp is less than a 5 predetermined duration of time [Yamada, see Figure 2 and col. 2 line 7 through col. 3 line 22, only when the timer expires, an alert tone/vibration/light is outputted to alert the user to the in-absence incoming call]. See claim 22 for motivation.

### ***Conclusion***

**Examiner's Note:** Examiner has cited particular columns and line numbers in the references applied to the claims above for the convenience of the applicant. Although the specified citations are representative of the teachings of the art and are applied to specific limitations within the individual claim, other passages and figures may apply as well. It is respectfully requested from the applicant in preparing responses, to fully consider the references in entirety as potentially teaching all or part of the claimed invention, as well as the context of the passage as taught by the prior art or disclosed by the Examiner. In the case of amending the claimed invention, Applicant is respectfully requested to indicate the portion(s) of the specification which dictate(s) the structure relied on for proper

interpretation and also to verify and ascertain the metes and bounds of the claimed invention.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Michael C. Lai whose telephone number is (571) 270-3236. The examiner can normally be reached on M-F 8:30 - 5:00 EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Ario Etienne can be reached on (571) 272-4001. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

Michael C. Lai  
Art Unit 2457  
Phone: (571) 270-3236

Application/Control Number: 13/111,734  
Art Unit: 2457

Page 18

Fax: (571) 270-4236

/MICHAEL C LAI/

Examiner, Art Unit 2457

<b>Notice of References Cited</b>	Application/Control No. 13/111,734	Applicant(s)/Patent Under Reexamination KLASSEN ET AL.	
	Examiner MICHAEL C. LAI	Art Unit 2457	Page 1 of 1

**U.S. PATENT DOCUMENTS**

*	Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
*	A US-5,896,567 A	04-1999	Ogushi, Masuo	455/421
*	B US-2004/0205775 A1	10-2004	Heikes et al.	719/318
*	C US-6,889,063 B2	05-2005	Yamada, Hironori	455/567
*	D US-6,940,407 B2	09-2005	Miranda-Knapp et al.	340/572.1
*	E US-7,181,497 B1	02-2007	Appelman et al.	709/206
	F US-			
	G US-			
	H US-			
	I US-			
	J US-			
	K US-			
	L US-			
	M US-			

**FOREIGN PATENT DOCUMENTS**

*	Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N				
	O				
	P				
	Q				
	R				
	S				
	T				

**NON-PATENT DOCUMENTS**

*	Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)				
	U				
	V				
	W				
	X				

\*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)  
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.


<b><i>Index of Claims</i></b>  	<b>Application/Control No.</b>  13111734	<b>Applicant(s)/Patent Under Reexamination</b>  KLASSEN ET AL.
	<b>Examiner</b>  MICHAEL C LAI	<b>Art Unit</b>  2457

✓	<b>Rejected</b>	-	<b>Cancelled</b>	N	<b>Non-Elected</b>	A	<b>Appeal</b>
=	<b>Allowed</b>	÷	<b>Restricted</b>	I	<b>Interference</b>	O	<b>Objected</b>

Claims renumbered in the same order as presented by applicant
  CPA
  T.D.
  R.1.47

CLAIM		DATE									
Final	Original	07/28/2011									
	1	✓									
	2	✓									
	3	✓									
	4	✓									
	5	✓									
	6	✓									
	7	✓									
	8	✓									
	9	✓									
	10	✓									
	11	✓									
	12	✓									
	13	✓									
	14	✓									
	15	✓									
	16	✓									
	17	✓									
	18	✓									
	19	✓									
	20	✓									
	21	✓									
	22	✓									
	23	✓									
	24	✓									



<b>Search Notes</b>  	<b>Application/Control No.</b>  13111734	<b>Applicant(s)/Patent Under Reexamination</b>  KLASSEN ET AL.
	<b>Examiner</b>  MICHAEL C LAI	<b>Art Unit</b>  2457

SEARCHED			
Class	Subclass	Date	Examiner
Inventor search		7/25/11	Lai
709	206, 207	7/26/11	Lai

SEARCH NOTES		
Search Notes	Date	Examiner
EAST text search	7/26/11	Lai

INTERFERENCE SEARCH			
Class	Subclass	Date	Examiner

--	--


**UNITED STATES PATENT AND TRADEMARK OFFICE**

UNITED STATES DEPARTMENT OF COMMERCE  
**United States Patent and Trademark Office**  
 Address: COMMISSIONER FOR PATENTS  
 P.O. Box 1450  
 Alexandria, Virginia 22313-1450  
 www.uspto.gov

**BIB DATA SHEET**
**CONFIRMATION NO. 6081**

SERIAL NUMBER	FILING or 371(c) DATE	CLASS	GROUP ART UNIT	ATTORNEY DOCKET NO.		
13/111,734	05/19/2011	709	2457	70314/00568		
<b>RULE</b>						
<b>APPLICANTS</b> Gerhard D. Klassen, Waterloo, CANADA; Christopher R. Wormald, Kitchener, CANADA; Lawrence E. Kuhl, Waterloo, CANADA;						
<b>** CONTINUING DATA *****</b> This application is a CON of 10/944,925 09/20/2004 PAT 7,970,849 which claims benefit of 60/504,379 09/19/2003						
<b>** FOREIGN APPLICATIONS *****</b>						
<b>** IF REQUIRED, FOREIGN FILING LICENSE GRANTED **</b> 06/01/2011						
Foreign Priority claimed <input type="checkbox"/> Yes <input checked="" type="checkbox"/> No		STATE OR COUNTRY		SHEETS DRAWINGS	TOTAL CLAIMS	INDEPENDENT CLAIMS
35 USC 119(a-d) conditions met <input type="checkbox"/> Yes <input checked="" type="checkbox"/> No		CANADA		7	24	3
Verified and Acknowledged		/MICHAEL C LAI/ Examiner's Signature		mcl Initials		
<b>ADDRESS</b> Blake, Cassels & Graydon LLP 199 BAY STREET , SUITE 4000 COMMERCE COURT WEST TORONTO, ON M5L 1A9 CANADA						
<b>TITLE</b> Handheld Electronic Device and Associated Method Providing Time Data in a Messaging Environment						
<b>FILING FEE RECEIVED</b> 1298	FEES: Authority has been given in Paper No. _____ to charge/credit DEPOSIT ACCOUNT No. _____ for following:				<input type="checkbox"/> All Fees <input type="checkbox"/> 1.16 Fees (Filing) <input type="checkbox"/> 1.17 Fees (Processing Ext. of time) <input type="checkbox"/> 1.18 Fees (Issue) <input type="checkbox"/> Other _____ <input type="checkbox"/> Credit	

## EAST Search History

## EAST Search History (Prior Art)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
S1	11346	709/206.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	AND	OFF	2011/07/26 11:57
S2	2237	709/207.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	AND	OFF	2011/07/26 11:58
S3	1028	S1 S2	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	AND	OFF	2011/07/26 11:58
S4	13	"7181497"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	AND	OFF	2011/07/26 12:00
S5	137	"6539421"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	AND	OFF	2011/07/26 12:17
S6	4	"5896567"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	AND	OFF	2011/07/26 15:02
S7	492201	(without or no or interrupt\$4 or discontin\$4 or disconnect\$4) with (communication or conversation or messag \$4)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	AND	OFF	2011/07/26 15:55
S8	300348	time near2 (stamp or indicat\$4)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	AND	OFF	2011/07/26 15:56

S9	1409	S7 with S8	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	AND	OFF	2011/07/26 15:56
S10	27935	(display\$4 or output \$4) near4 time near2 (stamp or indicat\$4)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	AND	OFF	2011/07/26 15:58
S11	56	S9 same S10	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	AND	OFF	2011/07/26 15:58
S12	11346	709/206.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	AND	OFF	2011/07/26 15:58
S13	2237	709/207.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	AND	OFF	2011/07/26 15:58
S14	12555	S12 or S13	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	AND	OFF	2011/07/26 15:58
S15	3	S11 S14	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	AND	OFF	2011/07/26 15:58
S16	25	S11 @ad < "20030919"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	AND	OFF	2011/07/26 16:15
S17	26	"7007085"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	AND	OFF	2011/07/26 16:36
S18	3	"6889063"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	AND	OFF	2011/07/26 18:40

S19	125	(without or refrain\$4 or stop\$4) adj2 (display \$4 or output\$4) near4 time near2 (stamp or indicat\$4)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	AND	OFF	2011/07/26 22:27
S20	492201	(without or no or interrupt\$4 or discontinu\$4 or disconnect\$4) with (communication or conversation or messag \$4)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	AND	OFF	2011/07/26 22:27
S21	12	S19 S20	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	AND	OFF	2011/07/26 22:27
S22	19	(without or refrain\$4 or stop\$4) adj2 (display \$4 or output\$4) adj4 time adj2 (stamp or indicat\$4)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	AND	OFF	2011/07/26 22:36
S23	19	S22 not S21	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	AND	OFF	2011/07/26 22:37
S24	13	S23 @ad < "20030919"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	AND	OFF	2011/07/26 22:37
S25	66	S19 @ad < "20030919"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	AND	OFF	2011/07/26 22:49
S26	53	S25 not S24	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	AND	OFF	2011/07/26 22:49
S27	13	"6940407"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	AND	OFF	2011/07/27 10:36

S28	3	"20040205775"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	AND	OFF	2011/07/27 10:58
S29	4	"5896567"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	AND	OFF	2011/07/27 11:00

**EAST Search History (I nterference)**

< This search history is empty >

**7/ 28/ 2011 2:07:14 PM**

**C:\ Documents and Settings\ mlai\ My Documents\ EAST\ Workspaces\ 13111734.wsp**

Substitute for form 1449/PTO		<b>INFORMATION DISCLOSURE STATEMENT BY APPLICANT</b>		<i>(Use as many sheets as necessary)</i>		<b>Complete if Known</b>	
						Application Number	13/111,734
		Filing Date	May 19, 2011				
		First named Inventor	KLASSEN, Gerhard D.				
		Art Unit	2629				
		Examiner Name	Not yet assigned				
		Attorney Docket Number	70314/00568				
Sheet	1	of	3				

U.S. PATENT DOCUMENTS						
Examiner Initials*	Cite No. <sup>1</sup>	Document Number		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		Number-Kind Code <sup>2</sup> (if known)				
		US-2002/0075303	A1	06-20-2002	THOMPSON et al.	
		US-2002/0087649	A1	07-04-2002	HORVITZ	
		US-2003/0060240	A1	03-27-2003	GRAHAM et al.	
		US-2003/0104841	A1	06-05-2003	YAMAMOTO	
		US-2004/0137967	A1	07-15-2004	BODLEY et al.	
		US-2004/0228531	A1	11-18-2004	FERNANDEZ et al.	
		US-6,301,609	B1	10-09-2001	ARAVAMUDAN et al.	
		US-6,590,529	B2	07-08-2003	SCHWOEGLER et al.	
		US-6,889,063	B2	05-03-2005	YAMADA	
		US-7,043,530	B2	05-09-2006	ISAACS et al.	
		US-7,099,700	B2	08-29-2006	HWANG et al.	
		US-7,111,044	B2	09-19-2006	LEE	
		US-7,181,497	B1	02-20-2007	APPELMAN et al.	

FOREIGN PATENT DOCUMENTS								
Examiner Initials*	Cite No. <sup>1</sup>	Foreign Patent Document			Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T <sup>6</sup>
		Country Code <sup>3</sup>	Number <sup>4</sup>	Kind-Code <sup>5</sup> (if known)				
		WO	2004/064362	A1	07-29-2004	GN NETCOM A/S		
		WO	02/65250	A2	08-22-2002	INVERTIX CORPORATION		
		WO	01/30091	A1	04-26-2001	MOTOROLA, INC.		
		GB	2384150	A	07-16-2003	NEC CORPORATION		
		GB	2350746	A	12-06-2000	NEC CORPORATION		
		EP	1176840	A1	01-30-2002	MICROSOFT CORPORATION		
		EP	0743762	A2	11-20-1996	NEC CORPORATION		
		JP	200311145	A	12-26-1190	MATSUSHITA ELECTRIC WORKS LTD		

Examiner Signature	/Michael Lai/	Date Considered	07/25/2011
--------------------	---------------	-----------------	------------

**EXAMINER:** Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. <sup>1</sup> Applicants' unique citation designation number (optional). <sup>2</sup> See Kinds Codes of USPTO Patent Documents at [www.uspto.gov](http://www.uspto.gov) or MPEP 901.04. <sup>3</sup> Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). <sup>4</sup> For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. <sup>5</sup> Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST. 16 if possible. <sup>6</sup> Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.







Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO <b>INFORMATION DISCLOSURE STATEMENT BY APPLICANT</b> <i>(Use as many sheets as necessary)</i>				<b>Complete if Known</b>		
				Application Number	13/111,734	Filing Date
Sheet		1	of	3	First named Inventor	KLASSEN, Gerhard D.
					Art Unit	2629
					Examiner Name	Not yet assigned
					Attorney Docket Number	70314/00568

U.S. PATENT DOCUMENTS						
Examiner Initials*	Cite No. <sup>1</sup>	Document Number		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		Number-Kind Code <sup>2</sup> (if known)				
		US-2002/0075303	A1	06-20-2002	THOMPSON et al.	
		US-2002/0087649	A1	07-04-2002	HORVITZ	
		US-2003/0060240	A1	03-27-2003	GRAHAM et al.	
		US-2003/0104841	A1	06-05-2003	YAMAMOTO	
		US-2004/0137967	A1	07-15-2004	BODLEY et al.	
		US-2004/0228531	A1	11-18-2004	FERNANDEZ et al.	
		US-6,301,609	B1	10-09-2001	ARAVAMUDAN et al.	
		US-6,590,529	B2	07-08-2003	SCHWOEGLER et al.	
		US-6,889,063	B2	05-03-2005	YAMADA	
		US-7,043,530	B2	05-09-2006	ISAACS et al.	
		US-7,099,700	B2	08-29-2006	HWANG et al.	
		US-7,111,044	B2	09-19-2006	LEE	
		US-7,181,497	B1	02-20-2007	APPELMAN et al.	

FOREIGN PATENT DOCUMENTS								
Examiner Initials*	Cite No. <sup>1</sup>	Foreign Patent Document			Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T <sup>6</sup>
		Country Code <sup>3</sup>	Number <sup>4</sup>	Kind-Code <sup>5</sup> (if known)				
		WO	2004/064362	A1	07-29-2004	GN NETCOM A/S		
		WO	02/65250	A2	08-22-2002	INVERTIX CORPORATION		
		WO	01/30091	A1	04-26-2001	MOTOROLA, INC.		
		GB	2384150	A	07-16-2003	NEC CORPORATION		
		GB	2350746	A	12-06-2000	NEC CORPORATION		
		EP	1176840	A1	01-30-2002	MICROSOFT CORPORATION		
		EP	0743762	A2	11-20-1996	NEC CORPORATION		
		JP	200311145	A	12-26-1190	MATSUSHITA ELECTRIC WORKS LTD		

Examiner Signature		Date Considered	
--------------------	--	-----------------	--

**EXAMINER:** Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. <sup>1</sup> Applicants' unique citation designation number (optional). <sup>2</sup> See Kinds Codes of USPTO Patent Documents at [www.uspto.gov](http://www.uspto.gov) or MPEP 901.04. <sup>3</sup> Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). <sup>4</sup> For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. <sup>5</sup> Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST. 16 if possible. <sup>6</sup> Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO  <b>INFORMATION DISCLOSURE STATEMENT BY APPLICANT</b>  (Use as many sheets as necessary)				<b>Complete if Known</b>	
				Application Number	13/111,734
				Filing Date	May 19, 2011
				First named Inventor	KLASSEN, Gerhard D.
				Art Unit	2629
				Examiner Name	Not yet assigned
				Attorney Docket Number	70314/00568
Sheet	2	of	3		

U.S. PATENT DOCUMENTS					
Examiner Initials*	Cite No. <sup>1</sup>	Document Number	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		Number-Kind Code <sup>2</sup> (if known)			
		7,236,472 B2	06-26-2007	LAZARIDIS et al.	
		7,305,441 B2	12-04-2007	MATHEWSON II et al.	

FOREIGN PATENT DOCUMENTS								
Examiner Initials*	Cite No. <sup>1</sup>	Foreign Patent Document			Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T <sup>6</sup>
		Country Code <sup>3</sup>	Number <sup>4</sup>	Kind-Code <sup>5</sup> (if known)				

Examiner Signature	Date Considered
--------------------	-----------------

**EXAMINER:** Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. <sup>1</sup> Applicants' unique citation designation number (optional). <sup>2</sup> See Kinds Codes of USPTO Patent Documents at [www.uspto.gov](http://www.uspto.gov) or MPEP 901.04. <sup>3</sup> Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). <sup>4</sup> For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. <sup>5</sup> Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST. 16 if possible. <sup>6</sup> Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. **DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

*If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.*



(19) World Intellectual Property  
Organization  
International Bureau



(43) International Publication Date  
29 July 2004 (29.07.2004)

PCT

(10) International Publication Number  
**WO 2004/064362 A1**

(51) International Patent Classification<sup>7</sup>: **H04M 1/05**, 1/60

(21) International Application Number:  
PCT/DK2004/000012

(22) International Filing Date: 13 January 2004 (13.01.2004)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:  
10/346,396 15 January 2003 (15.01.2003) US

(71) Applicant (for all designated States except US): **GN NET-COM A/S** [DK/DK]; Metalbuen 66, DK-2750 Ballerup (DK).

(72) Inventor; and

(75) Inventor/Applicant (for US only): **BODLEY, Martin, R.** [US/US]; 52 Easy Street, Sudbury, MA 01776 (US).

(74) Agent: **LARSEN & BIRKEHOLM A/S**; Skandinavisk Patentbureau, Banegårdspladsen 1, P.O. Box 362, DK-1570 København V (DK).

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM,

AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

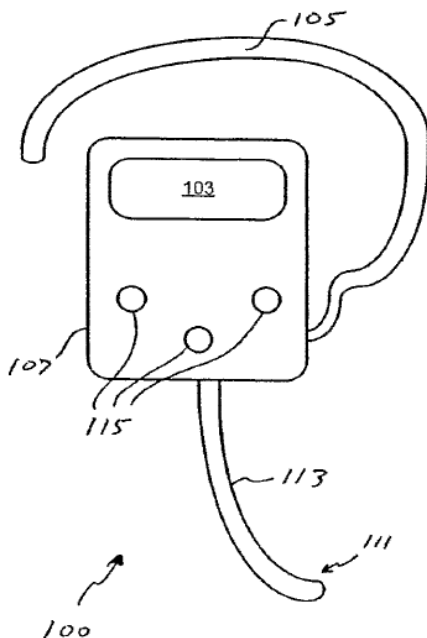
(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

**Published:**

- with international search report
- before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments

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

(54) Title: WIRELESS HEADSET WITH INTEGRATED DISPLAY



(57) Abstract: A wireless headset with an integral display is provided, the headset capable of communicating via a wireless network to a cellular telephone, cellular telephone adaptor, land-line telephone, land-line telephone adaptor, computer, personal digital assistant, or other device capable of communicating via the wireless network. The wireless headset of the invention includes an input transducer (e.g., a microphone), an output transducer (e.g., a speaker), a wireless networking subsystem and a controller/controller interface. The headset may also include means for attaching the headset to the user in order to allow hands-free operation. The integral display, fabricated using any of a variety of suitable technologies, allows headset and system information to be displayed (e.g., battery levels, signal levels, call status, caller identification, incoming call alert, current time, current date, elapsed use time, etc.). The integral display can also be used to aid headset/system configuration (e.g., headset volume, voice dialing, ring mode, roaming mode, etc.). The integral display can also provide added functionality to the headset (e.g., phone lists, text messages, calendar functions, appointment and/or task lists, etc.).

WO 2004/064362 A1

## WIRELESS HEADSET WITH INTEGRATED DISPLAY

### BACKGROUND OF THE INVENTION

5 The need for hands-free communication devices began soon after telephones were first invented. For example, early telephone operators used headsets that included a speaker and a microphone that could be wired or patched into a switchboard, the headset allowing hands-free operation. In the decades that followed, other methods of allowing hands-free operation emerged, ranging from telephone handset cradles that allowed the user to comfortably cradle the headset between the user's head and shoulder, to speaker phones that allowed absolute hands-free operation as long as the user was within the allowable range of the phone's speaker and microphone. More recently, short range wireless telephones have provided people with the freedom to roam 'unconnected' within a limited range of a base unit (e.g., around the home or office) while the advent of cellular telephones and networks have substantially increased this freedom, allowing people to roam untethered over both national and international regions.

20 While great progress has been made in the development of communication systems, only recently with the advent of Bluetooth and similar open specification technologies have developers been given the freedom to design short range, wireless devices that can connect to a variety of different networks and systems while offering worldwide compatibility. One type of device resulting from the development of these technologies is a wireless headset that can connect to any similarly enabled device or system. When used with a cellular phone that is enabled/adapted for use with one of these technologies (e.g., Bluetooth), the user of such a headset is able to talk freely, unencumbered by wires or cables, while taking advantage of the many benefits of a cellular phone. Unfortunately these headsets tend to be difficult to program and configure and offer the user

very limited functionality. Typically the headset is only provided with volume controls, an LED status indicator, and a simple multifunction button that may only allow the user to answer and end a call.

- 5 Accordingly, what is needed in the art is a wireless headset with expanded functionality. The present invention provides such a headset.

### SUMMARY OF THE INVENTION

- 10 The present invention provides a wireless headset with an integral display, the headset capable of communicating via a wireless network to a cellular telephone, cellular telephone adaptor, land-line telephone, land-line telephone adaptor, computer, personal digital assistant, or other device capable of communicating via the wireless network. The wireless headset
- 15 of the invention includes an input transducer (e.g., a microphone), an output transducer (e.g., a speaker), a wireless networking subsystem and a controller/controller interface. Preferably the headset also includes means for attaching the headset to the user in order to allow hands-free operation. The integral display, fabricated using any of a variety of suitable
- 20 technologies, allows headset and system information to be displayed (e.g., battery levels, signal levels, call status, caller identification, incoming call alert, current time, current date, elapsed use time, etc.). The integral display can also be used to aid headset/system configuration (e.g., headset volume, voice dialing, ring mode, roaming mode, etc.). The integral display
- 25 can also provide added functionality to the headset (e.g., phone lists, text messages, calendar functions, appointment and/or task lists, etc.).

- A further understanding of the nature and advantages of the present invention may be realized by reference to the remaining portions of the
- 30 specification and the drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a frontal view of a generic headset that includes a display in accordance with the invention;

5

Fig. 2 is a back view of the generic headset shown in Fig. 1;

Fig. 3 illustrates a headset with an integral boom member that includes a display in accordance with the invention;

10

Fig. 4 is a frontal view of a headset with a foldable boom member and a display in accordance with the invention, the foldable boom member located in the folded position;

15

Fig. 5 is a back view of the headset shown in Fig. 4;

Fig. 6 is a frontal view of the headset of Fig. 4 with the foldable boom member located in the un-folded position;

20

Fig. 7 is a back view of the headset shown in Fig. 6;

Fig. 8 is a block diagram of the electronics of a preferred embodiment of a headset according to the invention; and

25

Fig. 9 illustrates the block diagram of the electronics of another preferred embodiment of the invention.

DESCRIPTION OF THE SPECIFIC EMBODIMENTS

30

Figs. 1 and 2 illustrate the front (Fig. 1) and back (Fig. 2) surfaces of a generic headset 100 that includes the display means 103 of the present



invention. Although headset 100 can be used to communicate with any peripheral electronic device in which hands-free operation is desired, preferably headset 100 is used with a device that enables both voice and hearing data transfer (e.g., a computer with voice synthesis and voice recognition capabilities), still more preferably headset 100 is used with a communications system, and still more preferably headset 100 is used with a cellular telephone.

Although headset 100 may use any of a variety of means of attaching to a user (e.g., headband, earpiece, etc.) and thus allowing hands-free operation of the headset, preferably headset 100 includes an earpiece 105 (e.g., an ear hook) that is coupled to the user's ear, or a portion thereof. More preferably, earpiece 105 fits around a portion of the user's ear. Earpiece 105 is coupled to body 107 of the headset, body 107 housing the various electronic components necessary to provide functionality to headset 100. Attached to the back surface of body 107 is an output transducer 109 (e.g., a speaker). Also coupled to body 107 is an input transducer 111 (e.g., a microphone), which, in this embodiment, is located at the end of a boom member 113.

It will be understood that the present invention is not limited to use with a headset of the design shown in Figs. 1 and 2. For example, it can be used with a headset such as that illustrated in Fig. 3 in which the shape of body portion 301 provides an integral boom 303 to which input transducer 111 is attached (note that input transducer 111 is located on the other side of boom 303 and therefore is not visible in this figure). Alternately, the invention can be used with a headset such as that shown in Figs. 4-7. Figs. 4 and 5 show front and back views, respectively, of a headset 400 that includes a folding boom member 501, member 501 shown in its folded position in Figs. 4 and 5. Figs. 6 and 7 show front and back views,

respectively, of headset 400 with boom member 501 in its unfolded position.

5 Display 103 can be used to provide the user with various types of information depending upon the design goals for the specific headset. The design goals are primarily based on the expected headset use, desired headset functionality, display size, display resolution, allowable headset power drain, and the processing capabilities of the headset on-board processor (also referred to as a controller or an interface controller).  
10 Examples of information that can be displayed on display 103 are headset battery level, headset signal level, call status (e.g., dialing, in use, etc.), caller identification, current time/date, and timer information such as elapsed time associated with a particular call. Display 103 can also be configured to flash or otherwise visually indicate an incoming call.  
15 Assuming that the information is provided to the headset processor, display 103 can also display the battery level of the base unit (e.g., cell phone, dongle, etc.) and the signal strength of the base unit (e.g., cell phone). Additionally, display 103 can be used during headset configuration (e.g., to adjust or set the ring mode, ring loudness, headset volume, display mode,  
20 voice dialing, time, date, etc.) or to provide additional headset functionality (e.g., phone lists, text messages, dialing using virtual numeric keypad or phone list dialing). Additionally, display 103 can be touch sensitive, thus providing a touch screen for data input (e.g., phone numbers into a phone list, dialing using a virtual numeric keypad).

25 In addition to providing visual feedback to the user for a variety of headset functions as noted above (e.g., incoming calls, time/date/timer information, caller ID, battery and signal strength, etc.), display 103 can dramatically simplify the process of configuring or otherwise modifying the functionality  
30 of the headset. As opposed to the user (or technician) relying on an instruction manual and either flashes from an on-board light emitting diode

(LED) or beeps from an on-board sound processor, the user/technician can be provided with on-board written instructions and textual and/or graphical prompts. Thus, for example, the user or technician can easily navigate through a menu system presented on display 103 simply by using one or more input means (e.g., keys, buttons, switches, etc.) preferably located on the outside of the headset body. Preferably the input means includes at least one function key and a pair of volume keys (e.g., "+" and "-"), thus simplifying menu navigation. Examples of input means include keys/buttons 115 on headset 100, keys/buttons 305 on headset 300 and keys/buttons 403 on headset 400.

Fig. 8 is a high-level block diagram of the electronics of a preferred embodiment of a headset according to the invention. As shown, system 800 includes a wireless networking module 801 that provides short distance (e.g., on the order of 30 feet) wireless communications between the headset (e.g., a headset such as those shown in Figs. 1-7) and a correspondingly enabled peripheral electronic device 802. Preferably peripheral electronic device 802 is a cellular telephone that communicates data (e.g., voice communications) via a cellular network 803 to other devices 804. As cellular telephones and cellular telephone networks are well known in the art, further description will not be provided herein. Short distance wireless networking module 801 includes a transceiver 805 and can utilize any of a variety of networking technologies and protocols, as long as the selected system provides suitable networking capabilities between system 800 of the headset and device 802. Examples of suitable technologies and standards include Bluetooth and IEEE802.11. As such technologies and standards are well known in the art (see, for example, the specifications found at [www.bluetooth.com](http://www.bluetooth.com), [www.standards.ieee.org/getieee802/802.11.html](http://www.standards.ieee.org/getieee802/802.11.html) and [www.grouper.ieee.org/groups/802/11/](http://www.grouper.ieee.org/groups/802/11/), all of which are incorporated herein by reference), further description will not be provided herein. Module 801,

which is coupled to an appropriate antenna 806, controls the communication of signals between output transducer 109 and input transducer 111 of the headset and device 802.

- 5        Although device 802 is preferably a cellular telephone, the present invention can be used equally well with other types of network enabled devices (e.g., cellular telephone adaptors, land-line telephone, land-line telephone adaptor, computers, personal digital assistants or PDAs, etc.).
- 10       System 800 includes at least one controller (e.g., processor, micro-controller, application specific integrated circuit or ASIC, etc.) and controller interface (for purposes of illustration, shown as a single module 807). Controller/controller interface 807 may be either within networking module 801, within system 800 but separate from module 801 (as shown), or within
- 15       both. Controller/controller interface 807 can be used to program and/or modify module 801's programming as well as program and/or modify the functionality of the headset. Preferably one or more keys or switches or other input means 809 are coupled to controller/controller interface 807, thus providing a straightforward means of configuring the system and thus
- 20       the headset. An interface port 811 (e.g., a serial port or universal serial bus) may also be coupled to controller/controller interface 807, thus allowing the system and/or the graphical user interface (i.e., GUI) presented on display 103 to be configured via an external device such as a computer.
- 25       In accordance with the invention, display 103 is coupled to controller/controller interface 807. As such, display 103 can be used in conjunction with input means 809 to configure the headset in general, and module 801 in particular. For example, display 103 can be used to graphically display ring modes, ring volume, headset volume, headset
- 30       information, signal strength, battery status, charge level, etc. Display 103 can also be used to display information received from device 802, for

example caller identification, text messages, calendaring functions, personal phone book information, appointment and/or task lists, etc.

5 System 800 also includes a power sub-system 812, typically coupled to a charger port 813, power sub-system 812 providing power for the headset electronics.

10 Fig. 9 illustrates the high-level block diagram of the electronics of another preferred embodiment of the invention. As shown, system 900 includes a wireless networking module 901 which utilizes Bluetooth technology to provide communications between the headset and a Bluetooth enabled device 902. Preferably device 902 is a cellular telephone that communicates data (e.g., voice communications) via a cellular network 903 to other devices 904. Module 901 includes a transceiver 905 coupled to an  
15 appropriate antenna 907. Module 901 may also include a coder/decoder (i.e., CODEC) 909 for encoding and decoding signals that are to be communicated between output transducer 109 and input transducer 111 of the headset and device 902. Module 901 also includes the various circuits and processors to control transceiver 905 (e.g., link manager 911 which carries control information and baseband processor 913).  
20

In this embodiment, controller/controller interface 807 is located within module 901 and input means 809 is comprised of a pair of volume keys/buttons 915 and a function key/button 917. Volume keys/buttons 915,  
25 function key/button 917 and controller/controller interface 807, preferably in combination with display 103, are used to program and/or modify module 901's programming and control the interface of the headset with the network module. Examples of functions that are preferably controlled in this manner include the volume of the received and/or transmitted signals,  
30 ring volume, status indicators, headset status, network status, and roaming capabilities. The use of display 103 in conjunction with the input means

and the controller interface allow graphical and/or textual display of these functions. For example, bar charts can be used to display network signal strength, headset power, headset volume and ring volume while textual messages are used to communicate ring mode, status, headset information, etc. Additional input means 809, such as a switch 919 coupled to an extendable boom member, can be used in conjunction with controller/controller interface 807 to control headset power. A switch 921, for example coupled to a folding earhook and used in conjunction with controller/controller interface 807, can be used to control the headset status.

As in the previous embodiment, display 103 can also be used to display information either received from device 902 or stored within memory resident within the headset (not shown). Examples of such information include caller ID, text messages, calendars, phone book information, etc.

Display 103 preferably uses liquid crystal display (LCD) technology, although other types of technology can be used. For example, display 103 can use light emitting polymers (LEP), electroluminescent (EL) or active matrix electroluminescent (AMEL) technology, organic thin film transistors (organic TFT), active matrix organic light emitting diodes (AMOLED), amorphous silicon integrated displays (ASID), pliable display technology (PDT) or any other display technology that can provide a suitable resolution in the desired display size.

As will be understood by those familiar with the art, the present invention may be embodied in other specific forms without departing from the spirit or essential characteristics thereof. Accordingly, the disclosures and descriptions herein are intended to be illustrative, but not limiting, of the scope of the invention which is set forth in the following claims.

WHAT IS CLAIMED IS:

1. A wireless headset comprising:  
a housing;  
5 an input transducer coupled to said housing;  
an output transducer coupled to said housing;  
a wireless networking module adapted to transmit first signals via a short  
distance wireless network to a peripheral electronic device and to receive  
second signals via said short distance wireless network from said  
10 peripheral electronic device, wherein said first signals are initially received  
by said input transducer, and wherein said second signals are output by  
said output transducer after receipt by said wireless networking module;  
and  
a display means coupled to said housing, said display means capable of  
15 displaying text.
2. The wireless headset of claim 1, wherein said input transducer is a  
microphone.
- 20 3. The wireless headset of claim 1, wherein said output transducer is a  
speaker.
4. The wireless headset of claim 1, wherein said peripheral electronic  
device forwards said first signals via a long distance communication  
25 network and wherein said second signals are transmitted to said peripheral  
electronic device via said long distance communication network.
5. The wireless headset of claim 4, wherein said long distance  
communication network is a cellular telephone network.  
30

6. The wireless headset of claim 1, wherein said peripheral electronic device is a cellular telephone.
7. The wireless headset of claim 1, wherein said wireless networking module is a Bluetooth enabled networking module and said peripheral electronic device is a Bluetooth enabled cellular telephone.
8. The wireless headset of claim 1, wherein said wireless networking module is a Bluetooth enabled networking module, said peripheral electronic device is a cellular telephone, and wherein said wireless headset system further comprises a Bluetooth enabled adaptor, said adaptor coupled to said peripheral electronic device and providing Bluetooth enablement to said peripheral electronic device.
9. The wireless headset of claim 1, further comprising a boom member with a base end and a distal end, said base end coupled to said housing and said input transducer located proximate said distal end.
10. The wireless headset of claim 1, wherein said display means is capable of displaying headset status information.
11. The wireless headset of claim 10, wherein said headset status information includes at least one of headset battery level, headset signal level, call status, caller identification, time, elapsed time, and date.
12. The wireless headset of claim 1, wherein said display means is capable of indicating an incoming call.
13. The wireless headset of claim 1, wherein said display means is capable of displaying peripheral electronic device status information.



14. The wireless headset of claim 13, wherein said peripheral electronic device status information includes at least one of peripheral electronic device battery level and peripheral electronic device signal level.

5 15. The wireless headset of claim 1, wherein said display means is capable of displaying configuration feedback information.

16. The wireless headset of claim 13, wherein said configuration feedback information includes at least one of text configuration instructions, graphical configuration feedback, ring mode, ring volume, headset volume, display mode, voice dialing, time and date.

17. The wireless headset of claim 1, wherein said display means is capable of displaying a phone list.

15

18. The wireless headset of claim 1, wherein said display means is capable of displaying a text message.

19. The wireless headset of claim 1, wherein said display means is a touch sensitive display.

20

20. The wireless headset of claim 1, wherein said display means is selected from the group of display means consisting of liquid crystal displays, light emitting polymer displays, electroluminescent displays, active matrix electroluminescent displays, organic thin film transistor displays, active matrix organic light emitting diode displays, amorphous silicon integrated displays, and pliable display technology displays.

25

21. The wireless headset of claim 1, further comprising at least one controller and a controller interface.

30

22. The wireless headset of claim 1, further comprising means for attaching said wireless headset to a user.

5

23. The wireless headset of claim 22, wherein said attaching means is an earpiece.

24. A wireless headset comprising:

a housing;

10

a wireless networking module adapted to transmit first signals via a short distance wireless network to a peripheral electronic device and to receive second signals via said short distance wireless network from said peripheral electronic device;

15

a microphone coupled to said housing and coupled to said wireless networking module, wherein said first signals are initially received by said microphone;

a speaker coupled to said housing and coupled to said wireless networking module, wherein said second signals are output by said speaker after receipt by said wireless networking module; and  
a liquid crystal display capable of displaying text.

20

25. A wireless headset comprising:

a housing;

25

a Bluetooth enabled wireless networking module adapted to transmit first signals via a short distance wireless network to a Bluetooth enabled cellular telephone and to receive second signals via said short distance wireless network from said Bluetooth enabled cellular telephone wherein said Bluetooth enabled cellular telephone forwards said first signals via a cellular telephone network and wherein said second signals are transmitted to said Bluetooth enabled cellular telephone via said cellular telephone network;

a microphone coupled to said housing and coupled to said Bluetooth enabled wireless networking module, wherein said first signals are initially received by said microphone;

a speaker coupled to said housing and coupled to said Bluetooth enabled wireless networking module, wherein said second signals are output by said speaker after receipt by said Bluetooth enabled wireless networking module;

a controller and a controller interface coupled to said Bluetooth enabled wireless networking module; and

a display coupled to said controller interface, said display capable of displaying text.

26. The wireless headset of claim 25, said Bluetooth enabled cellular telephone further comprising a Bluetooth adaptor.

27. The wireless headset of claim 25, wherein said display is capable of displaying at least one of headset status information, Bluetooth enabled cellular telephone status information, incoming call status, configuration feedback information, phone list information, and text messages.

28. The wireless headset of claim 25, wherein said display is selected from the group of displays consisting of liquid crystal displays, light emitting polymer displays, electroluminescent displays, active matrix electroluminescent displays, organic thin film transistor displays, active matrix organic light emitting diode displays, amorphous silicon integrated displays, and pliable display technology displays.

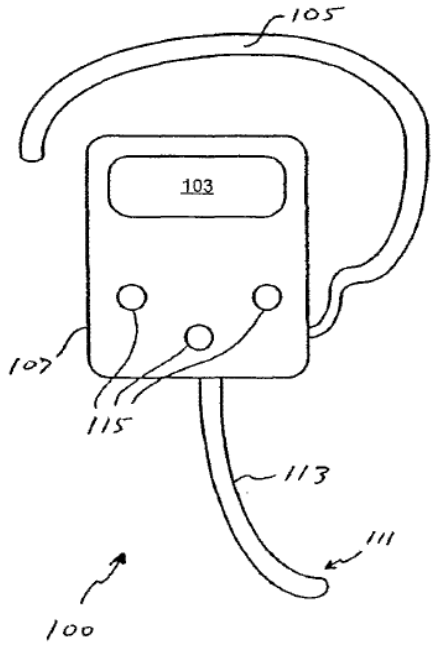


FIG. 1

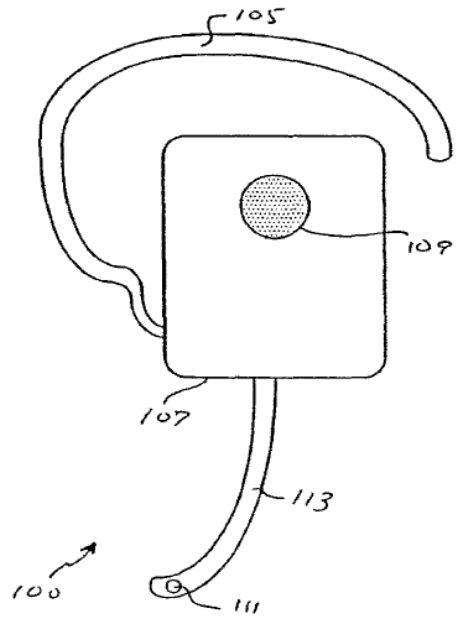


FIG. 2

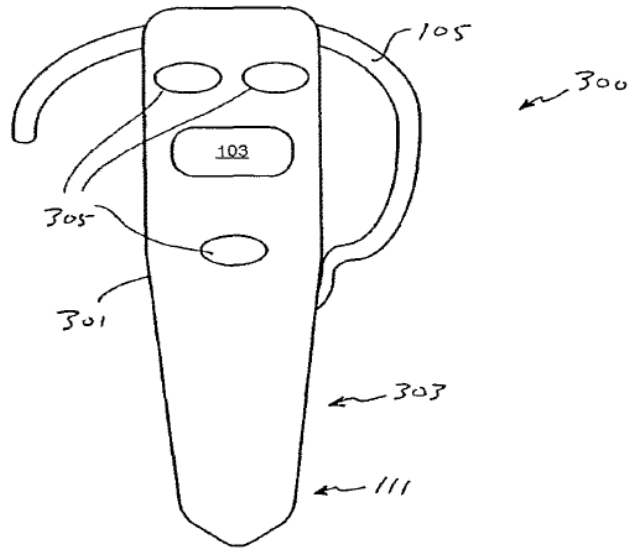


FIG. 3

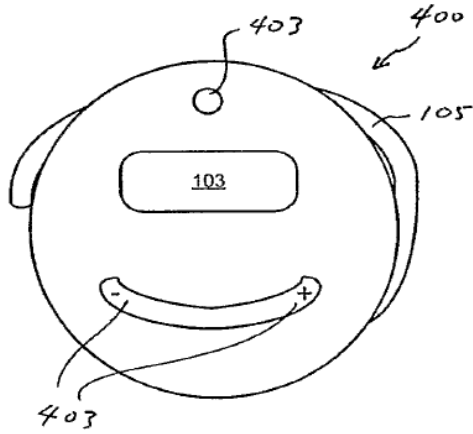


FIG. 4

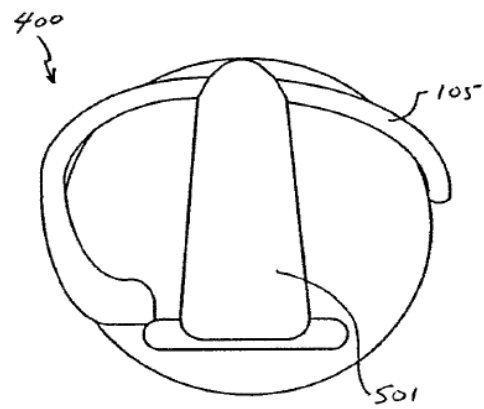


FIG. 5

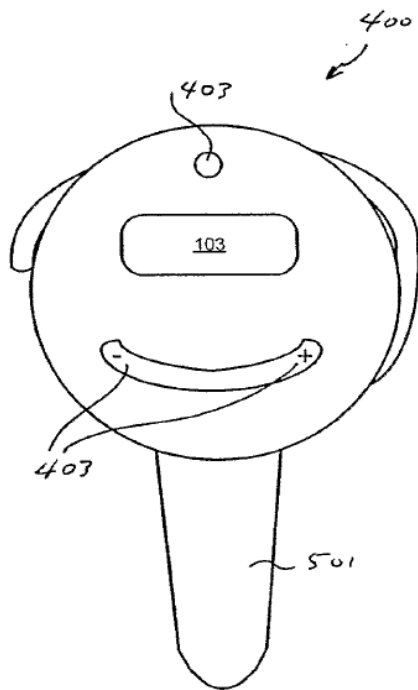


FIG. 6

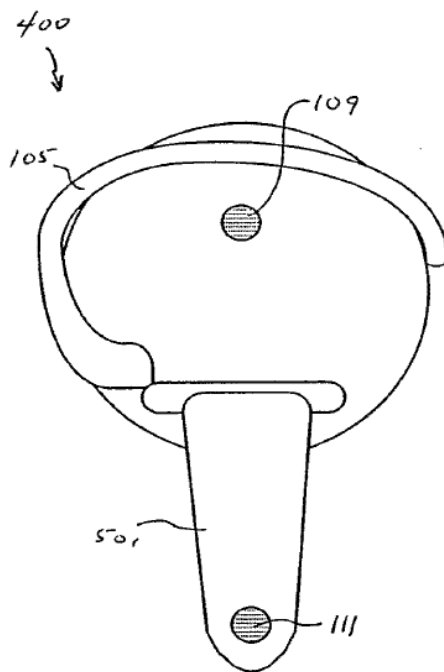


FIG. 7

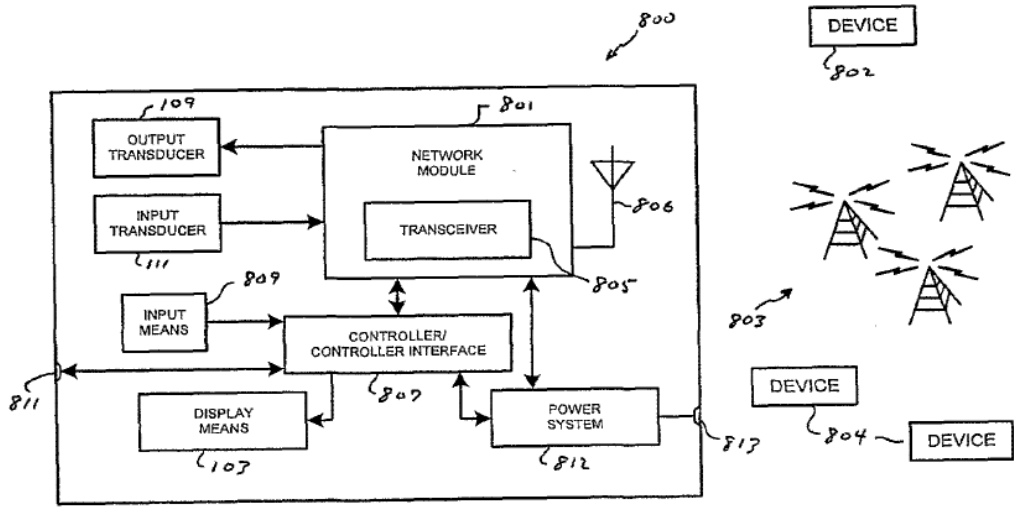


FIG. 8

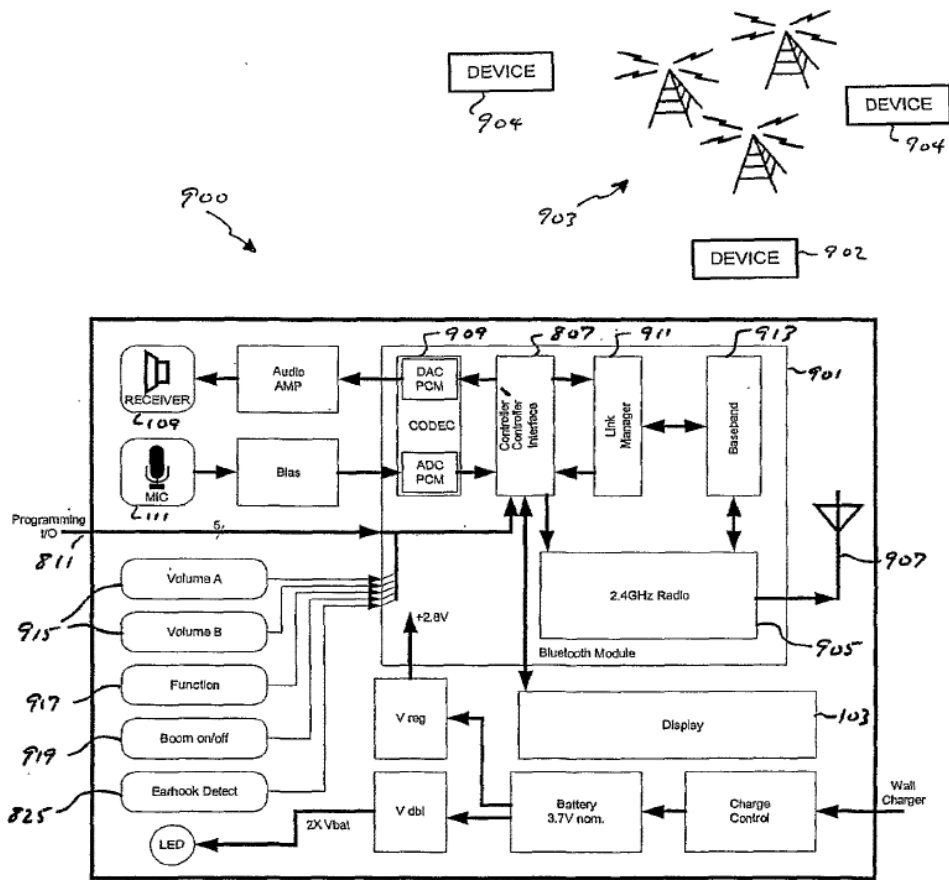


FIG. 9



# INTERNATIONAL SEARCH REPORT

International Application No  
PCT/DK2004/000012

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT		
Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
P,X	WO 03/056790 A (GOH KOON YEAP) 10 July 2003 (2003-07-10) abstract page 16, line 15 -page 18, line 14; figure 4 page 14, line 23 -page 15, line 7 -----	1-9, 24-27
A	US 2002/090099 A1 (HWANG SUNG-GUL) 11 July 2002 (2002-07-11) abstract; figures 1,2 -----	1,24,25



# INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No  
PCT/DK2004/000012

Patent document cited in search report		Publication date		Patent family member(s)	Publication date
DE 10106072	A	14-08-2002	DE	10106072 A1	14-08-2002
US 5991637	A	23-11-1999	US	6510325 B1	21-01-2003
WO 03056790	A	10-07-2003	WO	03056790 A1	10-07-2003
US 2002090099	A1	11-07-2002	WO	02054711 A2	11-07-2002

(19) World Intellectual Property Organization  
International Bureau



(43) International Publication Date  
22 August 2002 (22.08.2002)

PCT

(10) International Publication Number  
WO 02/065250 A2

- (51) International Patent Classification<sup>7</sup>: **G06F**
- (21) International Application Number: PCT/US02/04533
- (22) International Filing Date: 15 February 2002 (15.02.2002)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:
 

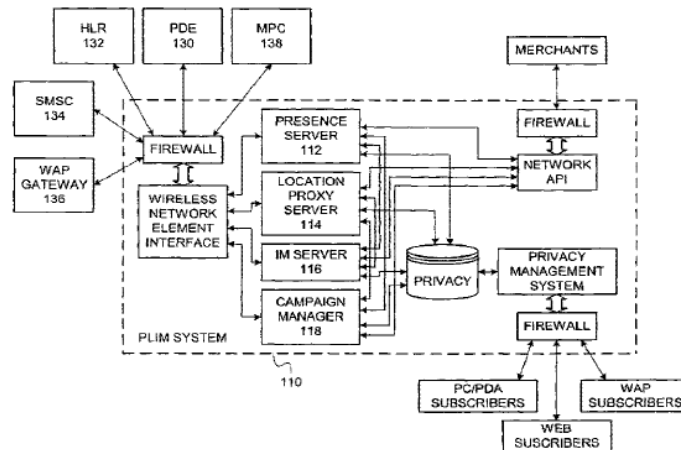
60/268,977	15 February 2001 (15.02.2001)	US
09/810,114	16 March 2001 (16.03.2001)	US
- (71) Applicant: **INVERTIX CORPORATION** [US/US];  
5285 Shawnee Road, Suite 401, Alexandria, VA 22312 (US).
- (72) Inventors: **MCDOWELL, Mark**; 5008 John Ticer Drive, Alexandria, VA 22304 (US). **KHALIL, Joseph**; 3305 Windham Circle, Alexandria, VA 22302 (US). **ZWEIFACH, Steven**; 9018 Greylock Street, Alexandria, VA 22308 (US). **STEAD, Graham**; 4639 5th Street South, Arlington, VA 22304 (US). **LEJEUNE, David, Jr.**; 13285 Coppermill Drive, Herndon, VA 20171 (US).

- (74) Agents: **ROBERTS, Jon, L.** et al.; Roberts Abokhari & Mardula, LLC, Suite 1000, 11800 Sunrise Valley Drive, Reston, VA 20191 (US).
- (81) Designated States (*national*): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZM, ZW.
- (84) Designated States (*regional*): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

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

[Continued on next page]

(54) Title: USE OF PRESENCE AND LOCATION INFORMATION CONCERNING WIRELESS SUBSCRIBERS FOR INSTANT MESSAGING AND MOBILE COMMERCE



(57) Abstract: Presence determination, location determination, instant messaging, and mobile commerce are integrated into a functionally seamless system, which may be implemented as an added component of a wireless provider's network. Alternatively, the integrated system enables instant messaging and mobile commerce as a centralized gateway attached to the networks of a large number of wireless providers. The gateway facilitates a business model that advances beyond today's practices, in which individual wireless carriers enter into bi-lateral agreements with specific Internet content providers. The functionally integrated gateway disclosed empowers Internet services that require real time information about wireless subscribers in order to conduct m-commerce or offer advanced messaging services. Optimization of a wireless network is also facilitated by taking network performance measurements, without using a special drive test team, via devices that are regularly using the network during standard network operation.



WO 02/065250 A2



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

**USE OF PRESENCE AND LOCATION INFORMATION  
CONCERNING WIRELESS SUBSCRIBERS FOR INSTANT  
MESSAGING AND MOBILE COMMERCE**

**INTRODUCTION**

[0001] The present invention relates generally to the field of wireless communications. More particularly, the present invention relates to enablement of instant messaging and location-based mobile commerce across Internet and diverse wireless network infrastructures.

**BACKGROUND OF THE INVENTION**

[0002] There are two major technical fields that have shown explosive growth over the past few years: the first is wireless communications and the second is use of data services, particularly the Internet. The growth of wireless communications has been astounding. Twenty years ago, there was virtually no use of wireless communications devices such as cellular phones. In contrast, the market penetration for wireless devices in the U.S. in 1999 was 32 percent. The current forecast is that 80 percent of the U.S. population will be wireless subscribers by 2008. Likewise, current and expected adoption of data services is phenomenal. Interestingly, wireless communications and data services are beginning to converge.

[0003] An example of this convergence is found in Instant Messaging (or "IM"). Originally an Internet-based text communication technology, IM will soon be integrated with wireless networks. It remains to be seen how smoothly this integration will proceed. Presence detection is an important element of any IM solution because an essential aspect of the IM technology is the detection of whether the members of each IM user's buddy list are present on the network. Although presence detection was fairly straightforward in the Internet environment, when the various wireless networks are to be integrated into the IM phenomenon presence is no longer so easy to establish comprehensively.

[0004] Thus, what is needed is an infrastructure technology to enable mobile IM services and provide for effective mobile buddy lists.

[0005] Another potential benefit of the integration of wireless networks with fixed IP networks, such as the Internet, is mobile commerce (also referred to as m-commerce).

Thus far, mobile commerce has been severely limited. To date, mobile commerce has typically been subscriber-initiated, with the subscriber using a handset to locate a product or service. This approach is consistent with E-911 implementations, where the subscriber initiates a call that requires location information. However, for mobile commerce to be broadly successful this paradigm needs to be inverted. This inversion occurs because wireless devices (telephone handsets, personal digital assistants, etc.) are not suitable for “window shopping.” Merchants should have the ability to initiate promotions – on a permission-oriented basis – just as they do with other media.

[0006] In the next three years, the number of m-commerce providers is expected to grow from almost zero to more than 18,000 worldwide. In addition, traditional retailers will also seek to engage mobile customers. The current model of bi-lateral agreements cannot scale to meet the demands of m-commerce, messaging, and traditional retail. A centralized gateway, where subscriber information can be sold (on a permission-oriented basis) to firms that require such information, would be an advantageous advance

[0007] Thus, what is needed is a centralized gateway where subscriber information can be sold, on a permission-oriented basis, to commercial firms.

[0008] The availability of location information concerning the wireless handsets is important to the enablement of mobile commerce. Although handset location information is not strictly required for mobile commerce to occur, it certainly facilitates the establishment of an effective m-commerce campaign.

[0009] Wireless carriers worldwide are preparing to offer location-based services to their subscribers. At the heart of these services is the Position Determining Equipment (PDE) which determines the location of a wireless device. The available PDE solutions employ several distinct methods of location determination: triangulation of RF signals among base stations; RF fingerprinting; and, embedded GPS in the wireless device. Regardless of the method employed, the PDE's most critical interface is to the Mobile Positioning Center (MPC), which routes emergency 911 voice calls and their associated location information to the local Public Safety Access Point (PSAP). The PDE also has an interface to a Location Proxy Server (LPS), which makes location information available to non-emergency third parties outside the wireless network.

[0010] There is a trend in the industry to combine the MPC and the LPS into a single platform: both systems route voice calls with embedded location information to third parties. However, there are good reasons that this conventional trend should be reversed and the MPC and LPS should be decoupled. Whereas the MPC performs the proven, stable function of routing emergency calls to the PSAP, the LPS is expected to evolve rapidly to accommodate the massive demands of Internet-based businesses and services. Perhaps most importantly, the LPS must accommodate merchant-initiated transactions, which should become a significant aspect of mobile commerce but cannot be provided by voice call-driven MPC technology. In the same way that Home Location Registers (HLRs) have been decoupled from MSCs – allowing “intelligence” to be decoupled from switching fabric – the LPS should be decoupled from the MPC and allowed to evolve into a highly intelligent engine responsible for making wireless Internet access relevant, personal, and timely.

[0011] A number of companies (e.g., FolloWap, OpenWave, SignalSoft, CTMotion, Air2Web, AirFlash, Ericsson’s “Oz,” InfoSpace, WindWire, OpenGrid, Aether Systems, 724 Solutions, MessageVine, Lucent, Nortel, Nokia, Quickdot, Xypoint, Cellpoint, just to name a few) currently promise technologies that will provide some form of IM or m-commerce solutions for wireless handsets to communicate with Internet-connected users. However, none of these companies have been able to develop a system that integrates presence determination, location determination, Instant Messaging, and mobile commerce.

[0012] Thus, what is needed is a an infrastructure technology that allows the integration of presence determination, location determination, Instant Messaging, and mobile commerce.

#### **SUMMARY OF THE INVENTION**

[0013] Accordingly, one aspect of the present invention is the integration of presence determination, location determination, Instant Messaging, and mobile commerce into a functionally seamless system. This integrated Presence, Location, Instant messaging, and Mobile commerce (or “PLIM”) system may be implemented as an added component of a wireless provider’s network.

[0014] Alternatively, a further aspect of the invention is the integration of presence determination, location determination, Instant Messaging, and mobile commerce as a

centralized gateway that may be attached to the networks of a large number of wireless providers.

[0015] The gateway arrangement according to this aspect of the invention facilitates a business model that represents a step forward from today's practices, in which individual wireless carriers are entering into bi-lateral agreements with specific Internet content providers. The PLIM gateway generates revenues from Internet services that require real-time information about wireless subscribers in order to conduct m-commerce or offer advanced messaging services. The gateway may then share the revenue generated through the sale of subscriber information with the participating wireless carriers that host the subscribers.

[0016] The PLIM gateway makes wireless subscriber presence, location, and profile information available on a 100% permission-oriented basis to Internet services. The PLIM gateway obtains subscriber information through direct electronic connections into wireless carrier networks. These connections are non-intrusive and pose no risks to the wireless carriers. Raw data collected from carriers is formatted and cached inside the PLIM gateway, and made available to registered Internet services through an electronic Internet-based interface.

[0017] Another aspect of the present invention is the enabling of true merchant initiated mobile commerce. This is made possible due to the integration of location information with a facility to manage marketing campaigns and a subscriber privacy management database. This ensures that the mobile commerce is conducted so that merchants obtain efficient marketing service and subscribers are subject only to marketing that they have expressly consented to.

[0018] Additionally, another aspect of the invention is the implementation of network optimization and performance measurement features to enable network operators to measure the performance and increase the efficiency of their networks.

[0019] It is an object of the present invention to provide integration, for one or more wireless networks, of presence information, location information, Instant Messaging, and mobile commerce.

[0020] It is another object of the present invention to provide integration of Instant Messaging and mobile commerce as a central gateway servicing the needs of multiple wireless networks.

[0021] It is yet another object of the present invention to provide integration of Instant Messaging and mobile commerce as a dedicated system servicing the needs of only a single wireless network.

[0022] It is still another object of the present invention to enable merchant initiated mobile commerce by integrating information about wireless subscribers' location, presence, and privacy choices.

[0023] Additional objects and advantages of the present invention will be apparent in the following detailed description read in conjunction with the accompanying drawing figures.

#### **BRIEF DESCRIPTION OF THE DRAWINGS**

[0024] Fig. 1 illustrates a block diagram view of the architecture of a PLIM system according to the present invention.

[0025] Fig. 2 illustrates the flow of signals via the PLIM system of the present invention when subscriber-initiated location-sensitive Web browsing is practiced.

[0026] Fig. 3 illustrates the flow of signals via the PLIM system of the present invention when merchant-initiated mobile commerce is practiced.

[0027] Fig. 4 illustrates an initial registration process for effecting subscriber provisioning.

[0028] Fig. 5 illustrates a preference selection process for effecting subscriber provisioning.

[0029] Fig. 6 illustrates an updating process for maintaining subscriber provisioning.

[0030] Fig. 7 illustrates a carrier specific PLIM server implementation.

[0031] Fig. 8 illustrates a PLIM system implemented as a centralized gateway.

[0032] Figs. 9 and 10 illustrate a system for network optimization and performance measurement, according to a further embodiment of the present invention.



### DETAILED DESCRIPTION OF THE INVENTION

[0033] One embodiment of the present invention is a computing platform that facilitates communications for wireless subscribers of a wireless network. The computing platform includes a presence module that maintains data concerning network presence of the wireless subscribers, as well as a location proxy module that maintains location data concerning physical location of the wireless subscribers. It also includes an instant messaging module connected to provide instant messaging service for the wireless subscribers utilizing the data concerning network presence. To enable mobile commerce, the computing platform further includes a campaign manager module connected to provide commercial message transmission to one or more of the wireless subscribers selected based on the data concerning network presence and the data concerning physical location.

[0034] Another aspect of the computing platform is a privacy database containing records of data permission settings corresponding to individual ones of the wireless subscribers. The data permission settings of the privacy database are used as a further basis on which the wireless subscribers are selected to be provided commercial message transmission.

[0035] Another aspect of the computing platform is that the presence module also maintains data concerning network presence of non-wireless instant messaging subscribers.

[0036] Another embodiment of the present invention is a network gateway for collecting presence information and location information concerning wireless subscribers of plural wireless networks, and for facilitating instant messaging and mobile commerce. The network gateway includes a presence module that maintains data concerning network presence of the wireless subscribers, as well as a location proxy module that maintains location data concerning physical location of the wireless subscribers. It also includes an instant messaging module connected to provide instant messaging service for the wireless subscribers utilizing the data concerning network presence. To enable mobile commerce, the network gateway further includes a campaign manager module connected to provide commercial message transmission to one or more of the wireless subscribers selected based on the data concerning network presence and the data concerning physical location.

[0037] The present invention also encompasses process embodiments for the conduct of mobile commerce. One process embodiment includes receiving a request from an approved merchant for current location information concerning a mobile subscriber, and verifying via a privacy database that the subscriber has given permission for the merchant to access the requested information. The process further includes obtaining current location information concerning the mobile subscriber from position determining equipment associated with a wireless network, and providing the subscriber's current location information to the merchant. By this process, the merchant is free to transmit to the subscriber personalized content based on the subscriber's current location information.

[0038] The integrated Presence Location Instant messaging and Mobile commerce (PLIM) system according to the present invention provides two salient advantages to a wireless carrier. One is the enabling of wireless instant messaging with "mobile buddy list" capability. The second is enabling of merchant-initiated mobile commerce. These applications hold great promise for immediate and untapped sources of revenue for wireless carriers.

[0039] IM is a revenue generating service. Instant messaging is a popular – perhaps indispensable – Internet service that wireless subscribers will pay to receive on their mobile devices. Even for carriers who do not bill separately for the IM service, the additional message traffic and airtime represent significant sources of new revenue on their existing infrastructure.

[0040] Enablement of mobile commerce (or "m commerce") is also a revenue generator. A PLIM system according to the present invention makes subscriber presence, location, and interest information available to merchants who desire to initiate transactions with wireless subscribers. Carriers can generate multiple streams of revenue from the sale of presence and location information, as well as activate lucrative mobile commerce agreements with strategic partners.

[0041] In addition, the ability to offer the PLIM as a gateway, independent of a single network, allows several important advantages. Most importantly, it provides a virtual "one stop shop" so that merchants, customers, and network operators can establish a business relationship with a single entity that provides a seamless interface.

[0042] A PLIM system according to the present invention offers significant additional advantages to wireless carriers in terms of network optimization and performance measurement. Through its Presence Server, the platform allows other network elements to function more efficiently. For example, by indicating that a subscriber's phone is OFF, the PLIM system eliminates unnecessary and resource-consuming SMS delivery re-attempts. Similarly, the PLIM system can signal the Positioning Determining Equipment when a subscriber has registered in a different market, allowing the equipment to rapidly "re-locate" the subscriber.

[0043] The present invention ensures that mobile commerce is conducted in such a way that subscriber privacy is not compromised. The PLIM system provides for a 100% opt-in service. Subscriber information is firewalled inside the carrier's network, or inside a centralized gateway, depending on implementation options. Subscriber information is not provided to any third party without explicit permission. Subscribers have the ability to establish and change their permissions and preferences easily and frequently using both PC and wireless interfaces.

[0044] Referring to **Fig. 1**, a block diagram of the architecture of a PLIM system **110** according to the present invention is illustrated. One element of a system according to the present invention is the Presence Server **112**. The Presence Server **112** determines whether a mobile device is ON or OFF in real-time. The Presence Server **112** inter-operates with system databases to allow sophisticated presence management.

[0045] Another element of a system according to the present invention is the Location Proxy Server (LPS) **114**. The LPS **114** makes subscriber location, as determined by third party equipment (e.g., PDE), available to merchants and other external entities under controlled conditions. An additional element of a system according to the present invention is the IM Server **116**. The IM Server **116** allows the wireless networks to send and receive instant messages from common IM platforms.

[0046] Another element of a system according to the present invention is the mobile commerce Campaign Manager **118**. The Campaign Manager **118** allows wireless carriers to automatically deliver targeted messages and e-coupons on behalf of mobile commerce merchants.

- [0047] Each of these elements **112**, **114**, **116**, **118** may be implemented on separate servers, but need not be to practice the present invention. Conceptually, the Presence Server **112**, the Location Proxy Server **114**, the IM Server **116**, and the Campaign Manager **118** may be implemented as software modules that may execute on separate physical machines, or on a single physical machine, at a common location, or remotely from one another, depending on operational convenience. In other words, these elements of the present invention are not dependent on the specifics of hardware implementation to provide the functions that make the present invention useful.
- [0048] The integrated Presence Server **112**, according to the present invention, determines the network presence of a wireless subscriber or IM user. The Presence Server **112** determines if a mobile phone or other mobile device is ON or OFF. It also determines if an Internet-based IM user is ONLINE or OFFLINE. The Presence Server **112** makes network presence information available between wireless networks and the Internet.
- [0049] The simplest use of the Presence Server **112** is the instant messaging buddy list. Because the Presence Server **112** is able to determine if a wireless device is ON or OFF, traditional IM buddy lists can be extended to indicate whether “mobile buddies” are ON or OFF. Likewise, WAP applications and embedded software in the handset can indicate whether buddies are ONLINE at their PCs or at their wireless devices.
- [0050] The buddy list is an integral part of the IM experience – it allows one to send messages to buddies who are online and therefore able to receive them instantly. But the buddy list is becoming a valuable application in its own right. It is useful to know if buddies, co-workers, staff, and others are ONLINE or have their phones ON. Particularly in mobile applications, merely knowing that a device is ON or OFF has intrinsic value.
- [0051] The Presence Server **112** according to the present invention does much more than power the buddy list. It enables “presence management” in a world where there are many ways to be online, and individuals may carry several different wireless devices. Presence management allows subscribers to direct calls, messages, and data traffic to particular devices. Presence management extends well beyond ON or OFF information: subscribers may wish to indicate “ON-busy” or “ON-meeting” or “ON-emergency only” or any number of other personal settings. Subscribers may even desire for their presence

information to be presented differently to different outside parties, for example “ON-busy” for co-workers but “ON-available” for spouse or supervisor. Moreover, Presence management preferences may change at different times of the day, and on different days of the week.

- [0052] Presence information is also useful internally for the wireless network to streamline operations. One example involves the Short Message Service Center (SMSC) 134. The SMSC 134 does not know if a mobile device is ON or OFF, and therefore must employ a complex, multi-day message delivery and re-delivery algorithm to ensure that messages are delivered successfully. Using the PLIM system 110 according to the present invention, the SMSC 134 can query the Presence Server 112 before attempting to send a message, eliminating inefficient retry attempts.
- [0053] Another example involves the Position Determining Equipment (PDE) 130 that many wireless carriers are expecting to deploy in the near future. Network-based PDE devices need to know the market where the subscriber is operating before they can locate the subscriber. If a subscriber moves from one market to another (for example, during a business trip), the PDE does not know where to begin searching for the subscriber. Using the present invention, the PDE can query the Presence Server to obtain market presence information, and then rapidly locate the subscriber.
- [0054] The Presence Server communicates with other components of the PLIM system platform as well as external network elements in the wireless network.
- [0055] Within the PLIM system platform, the Presence Server communicates with the Instant Messaging (IM) Server, the Campaign Manager, the Privacy Database, and the Network API components. The IM Server queries the Presence Server to determine if a subscriber’s phone is ON or OFF so that accurate “mobile buddy list” information is available for the IM clients. The Campaign Manager queries the Presence Server to know if a particular subscriber’s phone is ON or OFF before attempting to send a targeted mobile commerce message. The Presence Server queries the Privacy Database to ensure that an external entity requesting subscriber presence information is authorized to receive the information. Presence information is made available to approved external entities via “push” or “pull” through the Network API.