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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
14 May 2010 (14.05.2010)

PCT

(10) International Publication Number
WO 2010/053699 A2

(51) International Patent Classification:
G06F 9/06 (2006.01) *G06F 11/07* (2006.01)

(21) International Application Number:
PCT/US2009/061515

(22) International Filing Date:
21 October 2009 (21.10.2009)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
12/291,164 5 November 2008 (05.11.2008) US

(71) Applicant *(for all designated States except US)*: **VULCAN TECHNOLOGIES LLC** [US/US]; 505 Fifth Ave., South, Suite 900, Seattle, Washington 98104 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **MALLECK, Mark, Allen** [US/US]; 7315 - 116th Ave. SE, Newcastle, Washington 98056 (US). **JONES, Christopher, W.** [US/US]; 3255 NE 100th Street, Seattle, Washington 98125 (US). **BROCKWAY, Kevin, S.** [US/US]; 11755 - 24th Ave., NE, Seattle, Washington 98125 (US). **GRUBBS, Thomas, Robert** [US/US]; 3637 - 248th Place SE, Issaquah, Washington 98029 (US). **ALLEN, Paul, G.** [US/US]; 505 Fifth Ave., South, Suite 900, Seattle, Washington 98104 (US). **STACIE, Michael, Owen** [US/US]; 13315 NE 136th Place, Kirkland, Washington 98034 (US).

(74) Agent: **BERGSTROM, Robert, W.**; Olympic Patent Works PLLC, P.O. Box 4277, Seattle, Washington 98194-0277 (US).

[Continued on next page]

(54) Title: LIGHTWEIGHT APPLICATION-LEVEL RUNTIME STATE SAVE-AND-RESTORE UTILITY

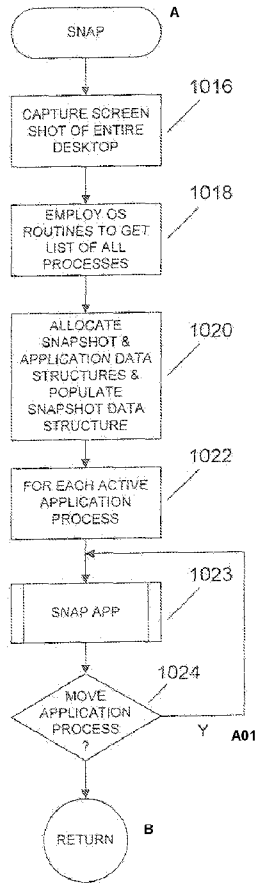


Figure 10B

(57) Abstract: Embodiments of the present invention are directed to computer-management tools and utilities that allow computer users to save and restore current application-level runtime states of computational environments created by users on computer systems. Certain embodiments of the present invention are particularly directed to time-and-memory efficient state-saving and state-restoring tools and utilities that allow a user to save snapshots of the user's application-level runtime environment, to manage and review stored snapshots, and to subsequently restore the application-level runtime state of a computer system from a previously saved snapshot.

WO 2010/053699 A2

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, 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, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to the identity of the inventor (Rule 4.17(i))*

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

LIGHTWEIGHT APPLICATION-LEVEL RUNTIME STATE SAVE-AND-RESTORE UTILITY

TECHNICAL FIELD

The present invention is related to tools and utilities for managing and maintaining computer systems and, in particular, to a tool that allows the user of a computer system to store the application-level runtime state of the computer system at particular points in time and to subsequently restore the computer system to a previously stored application-level runtime state.

BACKGROUND OF THE INVENTION

During the past 50 years, the computational bandwidths and data-storage capacities of computer systems have increased exponentially. Currently available personal computers, which can be purchased for under $1,000, provide greater computational bandwidth and data-storage capacity than the largest supercomputers of two generations past. The rapid improvement in the capabilities of computer systems has provided the basis for significant, continuous improvements in application programs, management tools, operating systems, and a host of other software-implemented functionalities provided to computer users by various programs that execute within computer systems. While, 40 years ago, computer users routinely loaded programs coded in punched cards for stand-alone execution within primitive execution environments, even a casual personal-computer user can today launch tens of powerful, concurrently-executing application programs that interact with one another as well as with applications programs executing on remote computer systems and with remote data-storage systems.

Modern operating systems provide a variety of powerful computer-system management tools, as do certain application programs. The vast increase in functionality, computing bandwidth, and data-search capacity have led to increasingly complex patterns of computer usage. Even a casual computer user may easily create very complex computational states as a result of launching multiple application programs and interacting with them using multiple associated windows and graphical user interfaces. Management of computer systems in such complex contexts is not

adequately addressed by many currently-available tools and utilities. For this reason, researchers and developers, operating-system and application-program developers, computer vendors, software vendors, and ultimately users of computer systems continue to seek useful and powerful management tools to assist computer users in managing the complex runtime states that users may create during the course of launching and interacting with multiple application programs.

SUMMARY OF THE INVENTION

Embodiments of the present invention are directed to computer-management tools and utilities that allow computer users to save and restore current application-level runtime states of computational environments created by users on computer systems. Certain embodiments of the present invention are particularly directed to time-and-memory efficient state-saving and state-restoring tools and utilities that allow a user to save snapshots of the user's application-level runtime environment, to manage and review stored snapshots, and to subsequently restore the application-level runtime state of a computer system from a previously saved snapshot.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 illustrates a personal computer and application-level runtime state.

Figure 2 illustrates an application-level-runtime-state snapshot operation.

Figure 3 illustrates an application-level-runtime-state restore operation.

Figure 4 illustrates the various different software programs and modules that may be resident within a computer system at any given point in time.

Figure 5 illustrates the memory of a computer system.

Figure 6 illustrates certain of the data structures employed by an operating system to provide a software execution environment.

Figure 7 shows a simplified diagram of the major internal electronic components of a computer system.

Figures 8-9 illustrate the information stored, and data structures used to store the information, for each currently executing application program when an application-level-runtime-state snapshot is taken according to embodiments of the present invention.

Figures 10A-C provide control-flow diagrams for the application-level-runtime-state snapshot operation that represents one embodiment of the present invention.

Figures 11A-D illustrate the restore-snapshot operation that represents one embodiment of the present invention.

Figures 12A-M illustrate aspects of the graphical user interface for an application-level-runtime-state save-and-restore utility that represents one embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

Embodiments of the present invention are directed to lightweight application-level-runtime-state save-and-restore management tools and utilities. Figure 1 illustrates a personal computer and application-level runtime state. The personal computer includes a device enclosure 102 and display monitor 104, as well as various input/output ("I/O") controls, such as keyboards, mice, and other I/O devices, not shown in Figure 1. From the perspective of a user, the current application-level runtime state of a computer system can be thought of as the number, sizes, positions, and overall organization of currently open application windows and other graphically represented objects and entities displayed on the display device 104, as well as a list of the application programs currently executing on the computer system. For example, in Figure 1, a hypothetical user has launched, and is interacting with, a business-application program, associated with two windows 106 and 108, an email program associated with one window 110, an Internet browser associated with a browser window 112, and a word processing program associated with a displayed document 114. Additional windows and applications, including window 116,

currently reside within a set of background objects associated with application programs with which the user is not currently interacting. From the user's standpoint, all of the currently running application programs, and associated windows, including windows that may have been minimized and are displayed as small icons, comprise the application-level runtime state of the computer system.

For many computer users, the runtime state of the computer system is quite dynamic, and changes throughout each period of time during which the computer users interact with heir computer systems. For example, a computer user may edit a text document using a text editor, save the text document, open another blank document and cut and paste text from an Internet browser to the blank document, launch a spreadsheet or business application, save the text document and close the text editor, and continue to launch, close, and interact with various additional, different application programs. A user may discontinue interacting with the computer system in order to perform other tasks, and may then return to the computer system at a later time to resume interacting with the various application programs, expecting the application-level state of the computer system to be virtually identical to the application-level state at the time the user left the computer to carry out the other tasks. Alternatively, a user may interact with some number of application programs up until a point in time at which the user launches a different set of applications and begins interacting with those applications while the applications with which the user was previously interacting are suspended, and remain dormant in the background, while the user proceeds to interact with the newly launched application programs.

Although a user expects to be able to return to a previous application-level state by, for example, returning to the computer system which the user temporarily abandoned in order to attend to other affairs, or by closing currently displayed application windows so that the application windows with which the user was previously interacting emerge from the background, there are many cases in which application-level runtime state is lost. Examples include power failures, inadvertent shutting down of applications and/or the operating system, application-program failures, and many other types of events. As another example, a user may be

engaged in a series of complex tasks at a workplace and need to leave work and resume working at home on a different computer system with a generally much different application-level runtime state than that of the workplace computer system.

It would be enormously useful for a user to be able to save the current application-level runtime state of a computer system or, in other words, take a snapshot of the current application-level runtime state of the computer system so that, in the event that the current application-level state is partially or completely lost, due to any of the various events discussed above, a user could restore the application-level runtime state of the computer system to the application-level state previously captured in the snapshot. The concept of saving and restoring data states is well known in a variety of different contexts, including database management systems and text editors. Many database management systems allow the current data state of a database to be saved, as a snapshot, to allow the database to be subsequently restored to that data state in the event of intervening database failures or database corruptions. Similarly, certain text editors allow for incremental saving of the current state of a document that is being edited. These save and restore operations are directed to data, rather than to the current runtime state of database or text-editor applications. Certain operating systems allow a computer-system configuration to be saved, and the computer subsequently returned to that configuration by a restore operation. However, operating-system configuration store and configuration restore operations do not store and recover the current runtime environment, including current state of executing application programs, displayed windows, open files, and other resources associated with executing application programs, but are instead essentially data-state save-and-restore utilities.

Figure 2 illustrates an application-level-runtime-state snapshot operation. In Figure 2, a horizontal timeline 202 represents the time during which a user interacts with a computer system. At various points in time 204-207, the user desires to save the current runtime state of the computer system, represented in Figure 2 by the current information displayed by the display monitor 208-211 at each point in time 204-207, respectively. A user may wish to save a snapshot of the runtime state of the computer system frequently, during the course of interacting with a

computer system, by manual application of the snapshot or state-save operation, or may alternatively wish to configure automatic snapshot operations so that, in the event of any of various different types of failures or other events, the user can easily return to a previous application-level runtime state by selecting the appropriate snapshot and executing a restore operation on that snapshot.

Figure 3 illustrates an application-level-runtime-state restore operation. In Figure 3, a series of snapshots are represented by points in time 302-308 and corresponding thumbnail images of the application-level runtime state of the computer system, as revealed by the information displayed by the display monitor 310-316 at those points in time. A user may select any particular snapshot, such as snapshot 310 taken at time 302, and invoke a restore operation in order to restore the application-level runtime state of the computer system 320 to the application-level runtime state captured in the snapshot.

As mentioned above, various applications programs and operating systems provide different types of data-state save-and-restore utilities, but there are currently no management tools that allow computer user to save the application-level runtime state of a computer system and restore a saved application-level runtime state at a later time. Although there are special runtime-state capture tools and utilities developed for computer-system debugging and analysis by software engineers, these tools and utilities are generally complex, have complex interfaces and functionality sets, are intended for exclusive use by computer professionals who understand the structure and operational characteristics of operating systems, and are generally extremely time inefficient and require storage of massive amounts of data. They are clearly not designed for convenient and frequent saving and restoration of application-level runtime states by general computer users.

Figures 4-7 illustrate aspects of the runtime state of a computer system. Figures 4-7 illustrate why saving of even a partial runtime state of a computer system can be a very time-inefficient and data-storage-space-inefficient operation. While a computer user, as discussed above with respect to Figure 1, considers the runtime state of a computer system to comprise the number of currently executing application programs and windows, open files, and other resources

associated with those application programs, the actual, complete state of a computer system is far more complex. It is, in fact, generally not possible to capture the complete state of a computer system at any instant in time, because the complete state of a computer system includes the current states of various electronic components that may not be accessible to state-capturing management tools and that, even were they accessible, have complex and rapidly changing states that cannot be instantaneously captured, particularly since their states are significantly perturbed by any attempt to capture the state of the computer system in which they reside. A close approximation to instantaneous complete-state capture is carried out by the hibernation feature provided by certain operating systems. Upon receiving a hibernation command, these systems attempt to reach a quiescent state with respect to hardware components, and save the memory-state of the computer system to a mass-storage device. When the system is brought back up, the memory state is restored, However, the hibernation feature does not represent a reasonable approach to application-level runtime-state snapshot-and-restore operations, being extremely inefficient in time and storage resources, applicable only to a single computer system, and often unreliable.

Figure 4 illustrates the various different software programs and modules that may be resident within a computer system at any given point in time. There may be many tens of application programs 402-405 concurrently executing within a computer system, each of which may, in turn, comprise a single process or many tens to hundreds of processes and threads. Certain types of application programs may be associated with backend routines 408 and 410 that continuously execute in order to carry out various tasks on behalf of application processes. For example, an email application that displays an email interface and provides for creation and editing of emails may be associated with an email backend process or processes that monitor communications interfaces for incoming email, queue emails for sending to remote computers, automatically retry failed attempts to send email, and carry out additional tasks on behalf of the email program. In addition, there may be a number of libraries containing executables that are common to multiple applications and application backends 412-414. The applications, application backend programs, and library routines all execute above an execution environment

created by an operating system 416, often a very large program with multiple processes and threads that concurrently execute to provide an interface to hardware components of the computer system. Operating systems, in turn, generally include lower-level routines, such as device drivers 418-424, that provide interfaces to various hardware components, including various types of controllers that control internal components and external peripheral components. One aspect of the full runtime state of a computer system is the set of all currently executing processes and threads associated with each of the currently executing application programs, application backend programs, operating system, and operating system components, including the values in various processor registers associated with the last point in time at which the process or thread executed.

Figure 5 illustrates the memory of a computer system. The memory 502 can be thought of as a very long list of memory units, such as words, each with a unique address, which form a large linear address space that is, in part, contained in physical electronic memory and, in part, stored on one or more mass-storage devices 504. Physical, electronic memory is generally much smaller than the virtual memory provided by the operating system of a computer system. Data is continuously shuffled from mass-storage devices to physical memory and from physical memory to mass-storage devices in order to simulate a large virtual-memory address space for both the operating system and for application programs. One aspect of the full runtime state of a computer system is all of the data currently resident in physical, electronic data, all data resident within mass-storage devices that support the virtual memory, and the current state of all files within the file system. On modern personal computers, the runtime state of the virtual memory of the computer system may comprise tens of gigabytes of data, or even greater amounts of data.

Figure 6 illustrates certain of the data structures employed by an operating system to provide a software execution environment. An operating system generally maintains a list of process-control blocks 602, each process-control block, such as process-control block 604, containing various fields that describe the current execution state of a process. As processes are executed, suspended, and queued to various wait queues, the state of the processes change, as reflected in the values of

fields within the process control block. Generally, an operating system may manage a number of different queues 606-608 that contain references to process-control blocks and represent queues of processes waiting for events or resources. An operating system maintains large, complicated hierarchical structures 610 to represent the open files and directories of a file system. Operating systems maintain many other, similar data structures, for managing the very complex and dynamic execution-environment state of the computer system, including complex data structures that reflect distribution of pending tasks in a multi-processor computer system. All of these structures, and the data contained within these data structures, are an important part of the runtime state of a computer system, in addition to the contents of memory, discussed with reference to Figure 5, and the identities and hierarchical relationships between all of the currently executing programs and routines, as discussed with reference to Figure 4.

Figure 7 shows a simplified diagram of the major internal electronic components of a computer system. These components include one or more processors 702, various types of electronic memory 704, a video controller 706, a number of additional device controllers 708-713, and a variety of different internal communications media 714-717 that interconnect the various internal components. The operation of the electronic components are generally expressed as state-transition diagrams, with the current state of an electronic component represented by the contents of various electronic memory registers, latches, and other state-storing subcomponents within the component. Ultimately, the current runtime state of a computer system includes the electronic states of all of these internal components.

As can be hopefully appreciated from the above discussion with reference to Figures 4-7, there is really no way that the full runtime state of a computer system can be captured at any given instant in time. To capture the run-state of a computer system would involved determining the states of both the electrical and software components of the computer system, but the information required to characterize the states of all of these components is voluminous, and the time required to determine and store this information far exceeds the time scales at which the internal states of the electrical and software components change. At best, a

partial runtime state might possibly be captured by suspending, as much as possible, operation of various components and storing a snapshot of the contents of registers and virtual memory. In fact, there are various state-capturing tools, including tools employed for debugging computer programs and operating systems, that attempt to capture the state of a computer system in this fashion. However, as discussed above, even for capturing a partial state in this manner, the snapshot often contains gigabytes of data, and requires minutes or hours for extraction, assembly, and storage. Obviously, runtime-state capture and storage at this level would not provide the type of state-save-and-restore tool described above with reference to Figures 1-3.

Embodiments of the present invention take a much different approach to runtime-state capture, encoding, and storage than the partial computer-system-state capture described in the preceding paragraph. First, the full, complete runtime state of a computer system is generally not needed in order to provide the application-level save-and-restore tool needed by a computer user. The user generally does not need to restore the exact, instantaneous runtime state of the computer system, but, instead, wishes to recreate the high-level, application-program environment visible to the user as the contents of the display monitor, shown in Figure 1. In other words, the user is chiefly concerned with saving a list of the application programs currently executing, at a particular point in time, along with the positions and content of open windows associated with those applications and other open resources associated with the applications, including open file-system files, sockets, and other such resources. An application-level runtime state can be captured in an extremely time-efficient and data-storage-efficient manner, as discussed in greater detail below. Furthermore, a user does not need, in general, a precise restoration of the states of all application programs at the time that the user executes a snapshot of the application-level runtime state of the computer system, but instead can often tolerate a rather large degree of imprecision in application-state restoration. As one example, if, at the time of a snapshot, the user is editing a text file using a text-editor application, with the cursor positioned on the fifth line of the 25th page of the text file, a restoration process that restores the application state approximately, so that instead of positioning the cursor on the fifth line of the 25th page of the document, the cursor is instead placed on the

first line of the first page of the document might nonetheless provide an adequate and useful application-level runtime-state, since the user can quite quickly and efficiently navigate to the proper line of the text file using application commands. In fact, restoration of a text editor so that the cursor is correctly positioned at the point to which the cursor is positioned at the time of the snapshot is relatively straightforward, but, in other cases, restoring the exact, precise state of an application from a stored snapshot may not be possible. For example, in those systems in which the snapshot operation does not save copies of the text files currently being edited by a text-editor application, the text file may be significantly altered, or even deleted, during the time that elapses between a snapshot operation and restoration of the runtime state of the computer system using the stored snapshot. In this case, the restoration process cannot restore the precise runtime state of the application program, since the underlying text file has either changed or is no longer available. Even in that case, however, restoring the application-level runtime state of the computer system to a best-possible extent may still provide enormous benefit to a computer user.

To summarize, embodiments of the present invention are directed to taking application-level runtime snapshots, rather than full runtime snapshots, of computer systems, and restoration operations that attempt to reproduce the application-level runtime state of each application listed in a snapshot as best as possible using available operating-system tools and application interfaces. In addition, as discussed in greater detail below, embodiments of the present invention store screenshots of the windows and interfaces displayed on the display device of a computer system at the time of a snapshot in order to facilitate a more precise restoration of the application-level runtime state of a computer system by a computer user using application interface commands and operating-system commands following the approximate restoration of an application-level runtime state from an application-level-runtime-state snapshot by embodiments of the present invention. By saving and restoring application-level runtime computer-system states, rather than full runtime states, and by relaxing the precision to which application-level states are restored, the application-level runtime-state save-and-restore procedures run quickly and consume comparatively very little data-storage space. In many embodiments of

the present invention, the level of precision of restoration and the amount of data stored for each snapshot can be controlled by user-specified parameters. In addition, the frequency of automated application-level-runtime-state snapshots may also be controlled by user input to a snapshot-control interface.

Figures 8-9 illustrate the information stored, and data structures used to store the information, for each currently executing application program when an application-level-runtime-state snapshot is taken according to embodiments of the present invention. As shown in Figure 8, for each currently executing application program, state information for that application program is stored in an application record 802. The application record references additional records 804 and 805, each of which describes an open window associated with the executing application, additional records 806 and 807, each of which describes an open file associated with an executing application, and, when possible, a list of commands 808 executed on behalf of a user by the executing application program since the application program was launched or since some save point or synchronization point last reached by the application program. Additional types of open resources associated with an application may be described by additional types of open-resource records. The application record 802 may store a name, executable filename, and file-system path, a version number, installation date and time, and a last-update date and time for the application, as shown in Figure 8, as well as many additional fields, including and ID, a plug-in ID, a reference to a thumbnail description and/or image, and many other types of data.

Each open-window record, such as open-window record 804, may store a window name, a reference to a file or object, a representation of which is currently displayed in the window, the *x,y* position and current size of the window on the display-monitor screen, the *z*-position of the window with respect to other windows displayed on the display monitor, the *x,y* position of a cursor associated with the window, and many other types of information useful in restoring a window by a restoration procedure.

Each open-file record, such as open-file record 806, may contain the name of a file, the file-system path of the file, a current active position within the file,

creation date and time of the file, last update date and time, an indication of the file type, and many other types of information that facilitate restoration of an open file associated with a restored application program.

The commands stored in a command record, such as command record 808, may assist the restoration process to restore the exact state of an executing application by re-issuing the commands, following launching of the application. For example, a text-editor application may be launched to an initial state, and then saved commands may be sequentially re-issued to the text-editor application to open particular documents and carry out those editing tasks that had occurred prior to the application-level-runtime-state snapshot. Of course, many considerations are involved in re-executing application commands. For example, when files are no longer available, commands directed to operations on unavailable files should not be issued, or, if issued, errors returned as a result of re-issuing the commands need to be handled in an appropriate fashion.

Figure 9 shows a data structure that stores a number of application-level-runtime-state snapshots. The highest-level data structure 902 comprises a list of snapshot records, such as snapshot record 904. Each snapshot record contains information about the snapshot, including a name, a date and time at which the snapshot was taken, a date and time at which the snapshot will expire, and other useful information, including a size, user name, user domain, alphanumeric description, machine name, machine domain, thumbnail graphic, an indication of whether or not the snapshot should be stored without possibility of automated deletion, to make room for subsequent snapshots, and other such types of information that may be useful in storing, restoring, and managing application-level-runtime-state snapshots. Each snapshot record, such as snapshot record 904, references a list of application records, such as application records 906 and 907 referenced by snapshot record 904. As discussed with reference to Figure 8, each application record, in turn, may reference various lists of records that describe open windows and other open resources associated with an application. Lists of application records may be linked together by internal references to form link lists, to facilitate allocation, deletion, and reference to the application records by the application-level-runtime-state save and

application-level-runtime-state restore procedures that represent embodiments of the present invention.

The detailed contents of snapshot records, application records, and open-resource records may vary from embodiment to embodiment. In certain heavier-weight embodiments of the present invention, the actual files and other entities and objects on which applications operate, or with which applications may be associated, may also be stored as part of a snapshot. For example, the text documents associated with an executing text-editor application may be saved and stored, by the snapshot procedure, so that the text-editor application may be precisely restored at a later time. In more lightweight embodiments of the present invention, the files and other objects and entities associated with executing applications may not be stored, so that, when these files, entities, and other objects are altered or deleted following a snapshot, the restoration process cannot precisely restore the applications associated with them.

Figures 10A-C provide control-flow diagrams for the application-level-runtime-state snapshot operation that represents one embodiment of the present invention. Figure 10A provides the highest-level control-flow diagram for the snapshot operation. In Figure 1002, the routine "save snapshot" displays a snapshot-saving interface to a user to collect information from the user for launching a snapshot operation. Once the interface is displayed, in step 1002, the "save snapshot" routine waits, in step 1004, for the user to enter a command through the save-snapshot interface. If the user chooses to close the interface, and not to undertake a snapshot operation, as determined in step 1006, then, in step 1008, the save-snapshot interface displayed in step 1002 is removed from the display and the routine finishes. Otherwise, if the user has entered a save-snapshot command, then, when all the fields of the save-snapshot interface contain valid information entered by the user, as determined in step 1010, the routine "snap" is called in step 1012 to execute the snapshot operation. If the user has entered incomplete or incorrect information, as determined in step 1010, then the routine "save snapshot" displays indications of the errors or missing information, in step 1014, and then returns to step 1004 to wait for proper input.

Of course, more complex embodiments of the present invention may involve more complex control flows, including very specific data entry and data-verification-related control flows. In the interest of brevity and generality, those details, straightforwardly implemented by software engineers, are omitted from the current figures and discussion.

Figure 10B provides a control-flow diagram for the routine "snap" called in step 1012 of Figure 10A. In step 1016, the routine "snap" captures a screenshot of the entire desktop displayed on the display device of a user's computer system. This screenshot may be used in a thumbnail graphical description of the snapshot and may also be used to facilitate the restoration process. In step 1018, the routine "snap" employs operating system routines and other available mechanisms to create a list of all currently executing application processes within the computer system. Moreover, when a particular application executes as multiple processes, the multiple processes are aggregated together into a group of processes associated with the application. Then, in step 1020, a snapshot record is allocated and application records are allocated for all of the processes. As many fields within the snapshot record as possible are initialized with values obtained from the snapshot interface, with other field values determined and entered as the snapshot proceeds. In the *for*-loop of steps 1022-1024, the routine "snap app" is called for each application in order to store sufficient information as part of the stored snapshot to allow for restoration of the current application-level runtime state of each application.

Figure 10C provides a control-flow diagram for the routine "snap app." In step 1030, the routine "snap app" uses available operating system routines and application-programming-interface ("API") routines to determine all of the open windows, open files, and other open resources associated with a particular executing application. In step 1032, the routine "snap app" uses operating system routines and API routines to obtain as much state information as possible for the executing application for which the routine "snap app" was called, in step 1023 of Figure 10B, and saves that information into the application record. Next, in step 1034, the routine "snap app" allocates window records and other resource records for describing all the open windows and other open resources associated with the application and then

enters references to these data structures into the application record. Then, in the *for*-loop of steps 1036-1040, the routine "snap app" considers each open window and other open resource and collects as much information about the state of each open window and other open resource as possible and saves that information into the corresponding open-window or other open-resource record. In the case of open windows, and in the case of other resources with graphical representations, as determined in step 1037, a screenshot of the graphical representation of the window or other object is taken and stored, and a reference to the screenshot is saved in the open-resource data structure.

Figures 11A-D illustrate the restore-snapshot operation that represents one embodiment of the present invention. Figure 11A provides a control-flow diagram for the highest-level routine "restore snapshot." In step 1102, a restore-snapshot graphical interface is displayed to a user. In step 1104, the routine "restore snapshot" waits for the user to enter a command via the displayed restore-snapshot interface. If the user wishes to close the interface without restoring a snapshot, as determined in step 1106, then the restore-snapshot interface is removed, in step 1108. Otherwise, if all the fields have been filled in with valid information, as determined in step 1110, then the routine "restore" is called in step 1112. Otherwise, the routine "restore snapshot" displays errors and indications of missing information, in step 1114, before returning to step 1104 to wait for the user to supply necessary information to launch a restore operation.

Figure 11B provides a control-flow diagram for the routine "restore," called in step 1112 of Figure 11A, that represents one embodiment of the present invention. In step 1116, the routine "restore" retrieves all of the stored data structures containing state information for a snapshot identified by the user for restoration. In the *for*-loop of steps 1118-1120, the routine "restore" calls the routine "restore app" for each application record in order to restore all applications that were executing at the time that the snapshot was made. In step 1122, the routine "restore" reports, by way of a graphical user interface or other means of communicating information to a user, any of the applications or application-associated windows or other resources that failed to be restored for any of various reasons. Using the above-mentioned example,

when text files are not saved as part of the snapshot process, it may be impossible to find the text file that was being edited by a text-editor application at the time the snapshot was made. In that case, the text editor is opened, but an indication is returned to the user that the text file that was being edited cannot be found.

Figure 11C provides a control-flow diagram for the routine "restore app" called in step 1119 of Figure 11B. In step 1124, the routine "restore app" locates the executable for the application. If the executable cannot be found, as determined in step 1126, then failure to find the application is noted, in step 1128 for subsequent reporting to the user, and the routine "restore app" returns. Otherwise, if the application is already running, as determined in step 1130, then, in certain embodiments of the present invention, the current state of the application is saved, as far as possible, via the application API and the application is closed, in step 1132, to prevent the restoration process from deleteriously impacting currently executing applications. In certain embodiments of the present invention, this step would be unnecessary, since the user would be required to close all currently executing applications before undertaking an application-level runtime-state-restoration process. In other cases, the restoration process itself closes all currently executing applications as a first step. In yet additional embodiments of the present invention, currently executing applications are allowed to remain, in the background, while the restoration process restores the applications that were executing at the time the snapshot was made. User-specified parameters may allow a user, in various embodiments of the present invention, to control when and if currently executing applications are closed prior to, or during, an application-level-runtime-state restoration. In step 1134, the application for which the routine "restore app" was called, in step 1119 of Figure 11B, is launched. If the launch is not successful, then failure to launch the application is noted, in step 1138, for subsequent reporting to the user and the routine "restore app" returns. Otherwise, in step 1140, the routine "restore app" restores the application, as far as possible, to the state at which the application was executing at the time of the snapshot. As discussed above, in certain cases, a list of commands that had been executed by the user up to the time of the snapshot is retrieved from the stored data structures and re-executed in order to bring the application back to the state at which

it was executing prior to the snapshot. In other cases, the application is launched, with responsibility for fine-granularity application-state restoration lying with the user. Intermediate levels of application-state restoration are also possible in alternative embodiments of the current invention. Then, in the *for*-loop of steps 1142-1144, all of the open resources associated with the application and described by open-resource records linked to the application record are restored via a call to the routine "restore resource" in step 1143.

Figure 11D provides a control-flow diagram for the routine "restore resource" called in step 1143 of Figure 11C. In step 1150, the routine "restore resource" locates the resource and determines how to open or restore the resource. If the resource is not found, or no method is determined by which the resource can be restored, then a failure is noted in step 1154 for subsequent reporting to the user. Otherwise, if the resource is not already open, as determined in step 1156, the resource is opened in step 1158. If the resource cannot be successively opened, as determined in step 1160, then the failure to open the resource is noted, in step 1162, and any other possible error-recovery mechanisms are invoked. Otherwise, in step 1164, the routine "restore resource" applies any of various resource-specific commands or operations that may be used to restore the resource to the state of the resource at the time of the snapshot. For example, the routine "restore resource" may seek for a location in a text file at which the executing application had positioned the text file prior to the snapshot. As another example, a window may be sized and positioned as it was at the time of the snapshot, and the contents displayed in the window at the time of the snapshot recovered as far as possible. In alternate embodiments of the present invention, when the resource is already open, as determined in step 1156, and when the resource is not shared with other applications that were executing at the time of the snapshot, the state of the resource may be saved and the resource closed, prior to again opening the resource in step 1158. These levels of complexity are implementation-specific and embodiment-specific, and require careful consideration of all possible side effects and potential conflicts concerning shared resourced and different versions of resources.

The application-level runtime-state save-and-restore utilities that represent embodiments of the present invention also generally include a number of routines that implement snapshot management functionality. This functionality allows a user to manage saved snapshots and locate particular snapshots for restoration, as well as to export snapshots to remote computers, archive snapshots, delete snapshots, and pin snapshots to prevent their automatic deletion. This functionality is similar to that of other data-management systems, and can be discerned from representations of the graphical user interface, discussed below. In an alternative embodiment, rather than pinning snapshots, an expiration date associated with a snapshot can be set to a particular value to prevent the snapshot from expiring, in which case a padlock icon may be used to visually indicate to a user that the snapshot is protected from deletion.

In alternative embodiments of the present invention, multiple, different snapshots may be opened together, in a single operation, or opened cumulatively, using a series of snapshot-open commands. In this case, the runtime-state save-and-restore utility ensures that no duplicative or redundant application invocations occur, and that only a single instance of each open window and open resource among the open windows and open resources specified by the multiple snapshots is reopened during a cumulative, multiple-snapshot restore operation. A multiple snapshot restore operation can result in a new computer-system state that represents a combination of states stored in the multiple snapshots. In similar fashion, snapshot-editing tools allow a user, in certain embodiments of the present invention, to edit snapshots in order to create application-level-runtime states that were not captured at a particular time, but are, instead, composite snapshots that combine multiple snapshots or that, in fact, when restored, result in new application-level-runtime states that were not saved in a snapshot operation. In certain embodiments of the present invention, the restore operation provides significant flexibility in the restoration process. A user may wish, for example, to restore only a subset of the open applications, open windows, and open resources saved in the snapshot, or may wish to manually launch certain of the applications and open resources, and then invoke automated application restoral for remaining applications or open resources. The

application-level-runtime state save-and-restore utility that represents an embodiment of the present invention may provide an additional interface to allow users to specify variations of how an application is restored. Furthermore, a snapshot can be saved, by one user, and restored by a second user into that user's computing domain. Snapshots can thus provide a means by which users can save and share application-level-runtime states among themselves. Application-level-runtime-state capture and storage in snapshots may represent computer-workspace states, allowing a user to invoke all applications and opening of associated resources for particular tasks using a single snapshot-restore command.

Figures 12A-L illustrate aspects of the graphical user interface for an application-level-runtime-state save-and-restore utility that represents one embodiment of the present invention. The graphical user interface ("GUI") shown in Figures 12A-L is part of an embodiment of the present invention referred to as the current environment restore tool ("CERT"). A small, low-display-footprint interface to CERT is shown in Figure 12A. This interface 1202 includes a title bar 1204, a snapshot list 1206 from which a user can select snapshots for restoration, a restore button 1208 for invoking a restore operation, a save-snapshot button 1209 for saving the current application-level-runtime-state of the user's computer, and a graphical indication 1210 of the types of applications for which execution state can be restored.

Figure 12B shows a larger GUI for the CERT embodiment of the present invention. The larger GUI provides a more detailed representation of available snapshots 1212, including snapshots taken on other machines that can be imported and restored on a local machine. Snapshots are also represented by graphical images 1213, including screenshots of each open window and of the entire desktop that are captured during the snapshot process. These thumbnail screenshots can be expanded to full-screen size, before and during restoration, to facilitate restoration of fine-granularity state within executing applications by a user. Screenshots also serve to facilitate a user's recollection of the application-level runtime state of the computer system at the time of the snapshot. For example, viewing a screenshot spreadsheet may facilitate a user's reconstruction of the exact state of a spreadsheet and spreadsheet operations that were anticipated by the user at

the time of the snapshot, to allow the user to recreate that state using various spreadsheet commands.

Figure 12C shows details of a CERT toolbar included with a full-size interface shown in Figure 12B. Figure 12D illustrates details of the available-snapshot panel (1212 in Figure 12B) included in the larger GUI. Figures 12E-G provide details of the thumbnail-based descriptions of snapshots (1213 in Figure 12B). Figure 12H illustrates details of the snapshot-preview pane (1214 in Figure 12B) of the full-size GUI. Figure 12I shows an expanded activity log invoked by means of the activity-log button (1216 in Figure 12C) of the CERT toolbar. The activity log indicates all of the save-and-restore operations carried out over a selectable period of time. Figure 12J shows an embodiment of the save-snapshot interface provided to a user to manually snapshot a current application-level runtime state of a computer system. Figure 12K shows an options interface that allows a user to tailor operation of the CERT embodiment of the present invention to meet the user's needs. Figure 12L shows a snapshot-restore interface provided to a user to allow a user to select a snapshot and restore the application-level runtime-state computer system encoded in that snapshot. Figure 12M shows an alternative implementation of the larger GUI for the CERT embodiment of the present invention shown in Figure 12B.

There are many different alternative embodiments of the present invention. As briefly discussed above, the application-level runtime-state save-and-restore tool of various embodiments of the present invention can be implemented to save only a very basic application-level state so that, upon restoration, a user is provided essentially with a set of launched applications and open windows associated with those applications. The most lightweight version of the present invention then requires a user to reconstruct details of the application-level runtime state through interactions with the launched applications. At the other extreme of embodiments of the present invention are embodiments which strive to store all possible available state information to facilitate precise restoration of the application states of all applications re-launched as part of the restoration process. This may involve storing redundant versions of files, objects, and other entities produced by, or accessed by,

executing application programs. In general, it is not feasible to expect that the precise application-level state of executing applications can be faithfully restored, since embodiments of the present invention do not attempt to capture and store content of a computer system's virtual memory, file system, and machine-level electronic state. However, for many purposes, restoration, even when approximate, of an application-level state provides enormous benefit to users. The snapshots serve essentially as electronic bookmarks to an entire, long-term interaction of a user with a computer system, allowing a user to return to application states at particular points in time during that interaction and recover a significant portion of the application state that existed at the time the snapshot was made. In many cases, a user may seek only to recover an application state related to a particular application state. In this case, a lightweight approach, relying on screen capture to refresh a user's recollection, may be not only most efficient, but actually most desirable from the user's standpoint. As one example, a user may recall having accessed and having begun to read a particular email at a point in time prior to a particular snapshot, and may wish to recover only information in that email that is available in a screenshot of the email window, while indifferent to all other activities that were ongoing at the time of the snapshot.

Although the present invention has been described in terms of particular embodiments, it is not intended that the invention be limited to these embodiments. Modifications will be apparent to those skilled in the art. For example, embodiments of the present invention may be implemented in many different ways by varying familiar programming and development parameters, including programming languages, operating-system platforms, modular organization of implementations, control structures used in implementations, data structures used both in implementing the store-and-restore facilities as well as used in storing application-level runtime states, and by varying many other of these types of parameters and approaches. Application-level runtime-state save-and-restore utilities may be implemented for use in any number of different types of computers and computational environments, including distributed computing environments in which snapshots taken on a first machine can be used to restore an application-level runtime state on a second machine.

The foregoing description, for purposes of explanation, used specific nomenclature to provide a thorough understanding of the invention. However, it will be apparent to one skilled in the art that the specific details are not required in order to practice the invention. The foregoing descriptions of specific embodiments of the present invention are presented for purpose of illustration and description. They are not intended to be exhaustive or to limit the invention to the precise forms disclosed. Many modifications and variations are possible in view of the above teachings. The embodiments are shown and described in order to best explain the principles of the invention and its practical applications, to thereby enable others skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the following claims and their equivalents:

CLAIMS

1. A application-level-runtime-state save-and-restore utility comprising:

a snapshot component, implemented as computer instructions stored on a computer-readable medium, that determines, encodes, and stores a snapshot of the current application-level runtime state of a computer system in a computer-readable medium; and

a restore component, implemented as computer instructions stored on a computer-readable medium, that restores the application-level runtime state of a computer system from a stored snapshot.

2. The application-level-runtime-state save-and-restore utility of claim 1 further comprising:

a snapshot-management component, implemented as computer instructions stored on a computer-readable medium, that provides an interface which provides for searching, displaying information about, deleting, editing, and organizing snapshots.

3. The application-level-runtime-state save-and-restore utility of claim 2 wherein the snapshot-management component displays screenshots and descriptions of each snapshot stored on a computer system or accessible to the computer system.

4. The application-level-runtime-state save-and-restore utility of claim 1 wherein the snapshot component determines the application-level runtime state of the computer system by:

executes operating-system calls to identify each currently executing application program; and

for each currently executing application program,

executes operating-system calls and/or application-program-interface calls to determine the current execution state of the application program.

5. The application-level-runtime-state save-and-restore utility of claim 4 wherein the current execution state of the application program includes:

open windows associated with the application program;

information displayed in the open windows associated with the application program; and

other open resources associated with the application program, including open file-system files.

6. The application-level-runtime-state save-and-restore utility of claim 5 wherein the current execution state of the application program further includes:

a list of application commands executed by the application program.

7. The application-level-runtime-state save-and-restore utility of claim 1 wherein the snapshot component encodes the application-level runtime state of the computer system by:

storing snapshot-related information in a snapshot data structure;

for each currently executing application program, storing information related to a current application-state in an application data structure; and

linking the one or more application data structures created for the currently executing application programs to the snapshot data structure.

8. The application-level-runtime-state save-and-restore utility of claim 7 wherein the snapshot-related information includes:

a snapshot name;

a date and time for snapshot creation;

a snapshot expiration time;

a snapshot size;

a name and domain of a user;

a computer name and computer domain;

a text description; and

a reference to a screen shot of the contents of the display monitor.

9. The application-level-runtime-state save-and-restore utility of claim 7 wherein the information related to a current application-state includes:

an application name;

an executable filename;

a file-system path;

a version number;

an ID;

a plug-in ID; and

a reference to a thumbnail description and/or image.

10. The application-level-runtime-state save-and-restore utility of claim 9 wherein the information related to a current application-state further includes:

an installation date and time; and

a last-update date and time.

11. The application-level-runtime-state save-and-restore utility of claim 7 wherein the snapshot component further encodes the application-level runtime state of the computer system by:

for each open resource associated with each application program, storing information related to the open resource in an open-resource data structure; and

linking the open-resource data structures representing open resources associated with an application program to the application data structure representing the state of the application program.

12. The application-level-runtime-state save-and-restore utility of claim 11 wherein open resources include:

open display windows;

open files;

open sockets; and

other operating-system-provided entities and objects.

13. The application-level-runtime-state save-and-restore utility of claim 12 wherein the information related to an open display window that is stored in an open-resource data structure includes:

a window name;

a reference to a file or object, a representation of which is currently displayed in the open display window;

an *x,y* position and current size of the open display window on the display-monitor screen;

a *z*-position of the open display window with respect to other windows displayed on the display monitor;

an *x,y* position of a cursor associated with the open display window; and

a reference to a screenshot of the open display window.

14. The application-level-runtime-state save-and-restore utility of claim 12 wherein the information related to an open file that is stored in an open-resource data structure includes:

a name of the file;

a file-system path;

a current active position within the file;

a creation date and time of the file;

a last update date and time; and

an indication of the file type.

15. The application-level-runtime-state save-and-restore utility of claim 1 wherein the restore component restores the application-level runtime state of a computer system from a stored snapshot by:

accessing a stored snapshot data structure according to information supplied through a restore-snapshot interface;

for each application data structure linked to the snapshot data structure,

using information in the application data structure to launch a corresponding application program, and

for each open-resource data structure linked to the application data structure,

using information in the open-resource data structure to open a resource and associate the opened resource with the launched application program.

16. The application-level-runtime-state save-and-restore utility of claim 15 further including reissuing application commands to an application program following launch of the application program.

17. The application-level-runtime-state save-and-restore utility of claim 15 wherein an application-level-runtime state encoded as a snapshot on a first computer is restored by the restore component of an instance of the application-level-runtime-state save-and-restore utility on a second computer.

18. The application-level-runtime-state save-and-restore utility of claim 17 wherein the restored application-level-runtime state on the second state is restored on behalf of the user who saved the snapshot.

19. The application-level-runtime-state save-and-restore utility of claim 17 wherein the restored application-level-runtime state on the second state is restored on behalf of a user different from the user who saved the snapshot.

20. The application-level-runtime-state save-and-restore utility of claim 1 wherein an application-level-runtime state can be restored incrementally and selectively by a user.

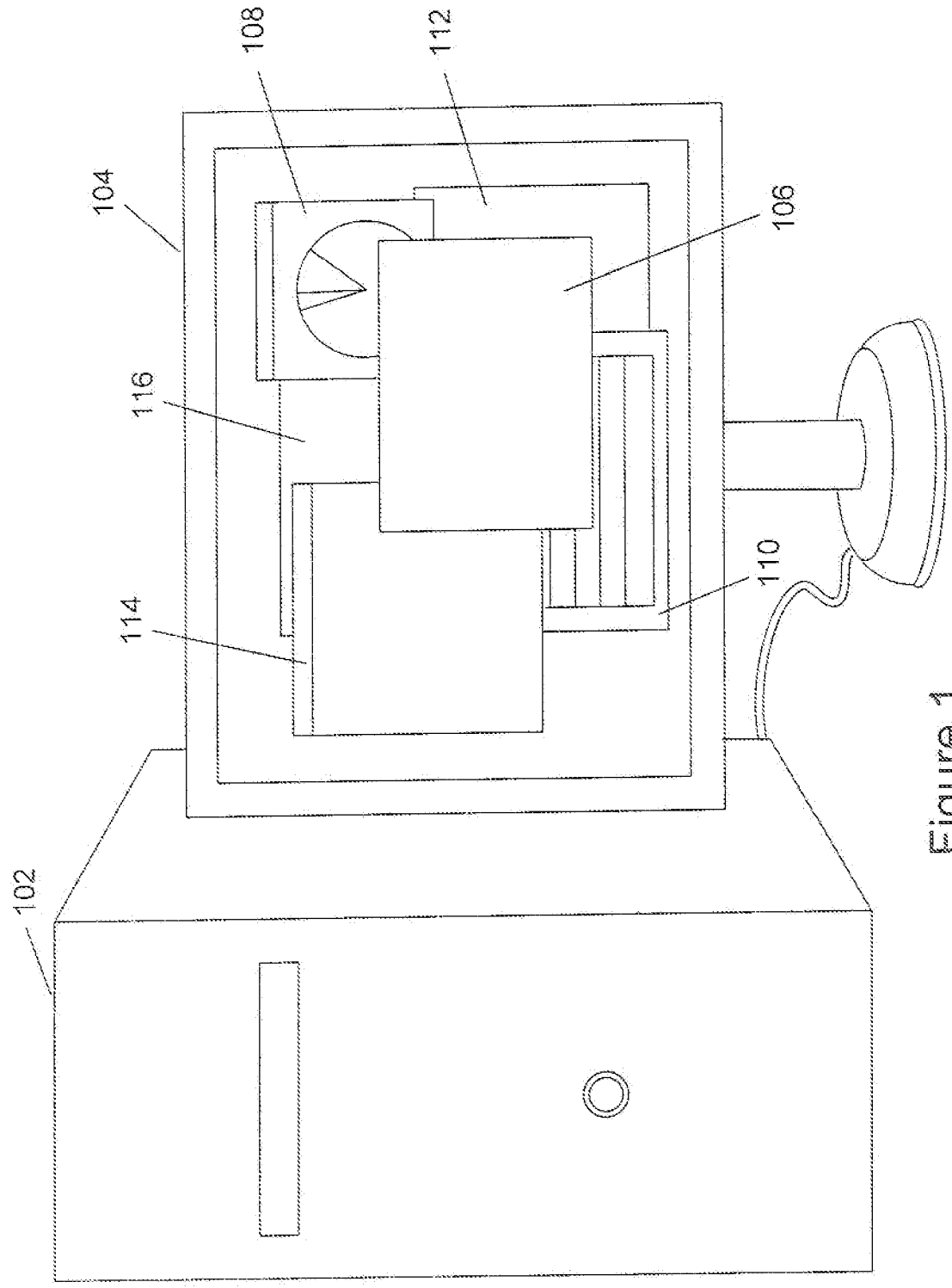
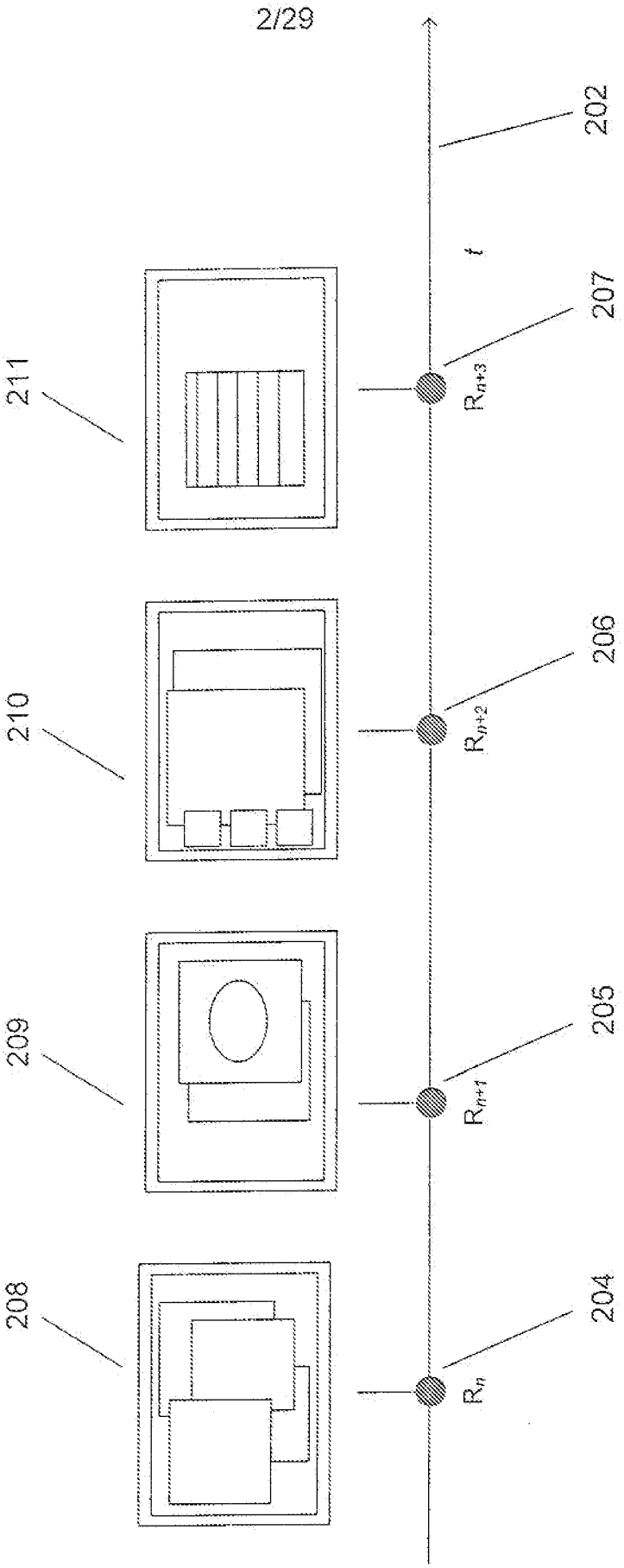


Figure 1


202
t
207
R_{n+3}
211
206
R_{n+2}
210
205
R_{n+1}
209
204
R_n
208

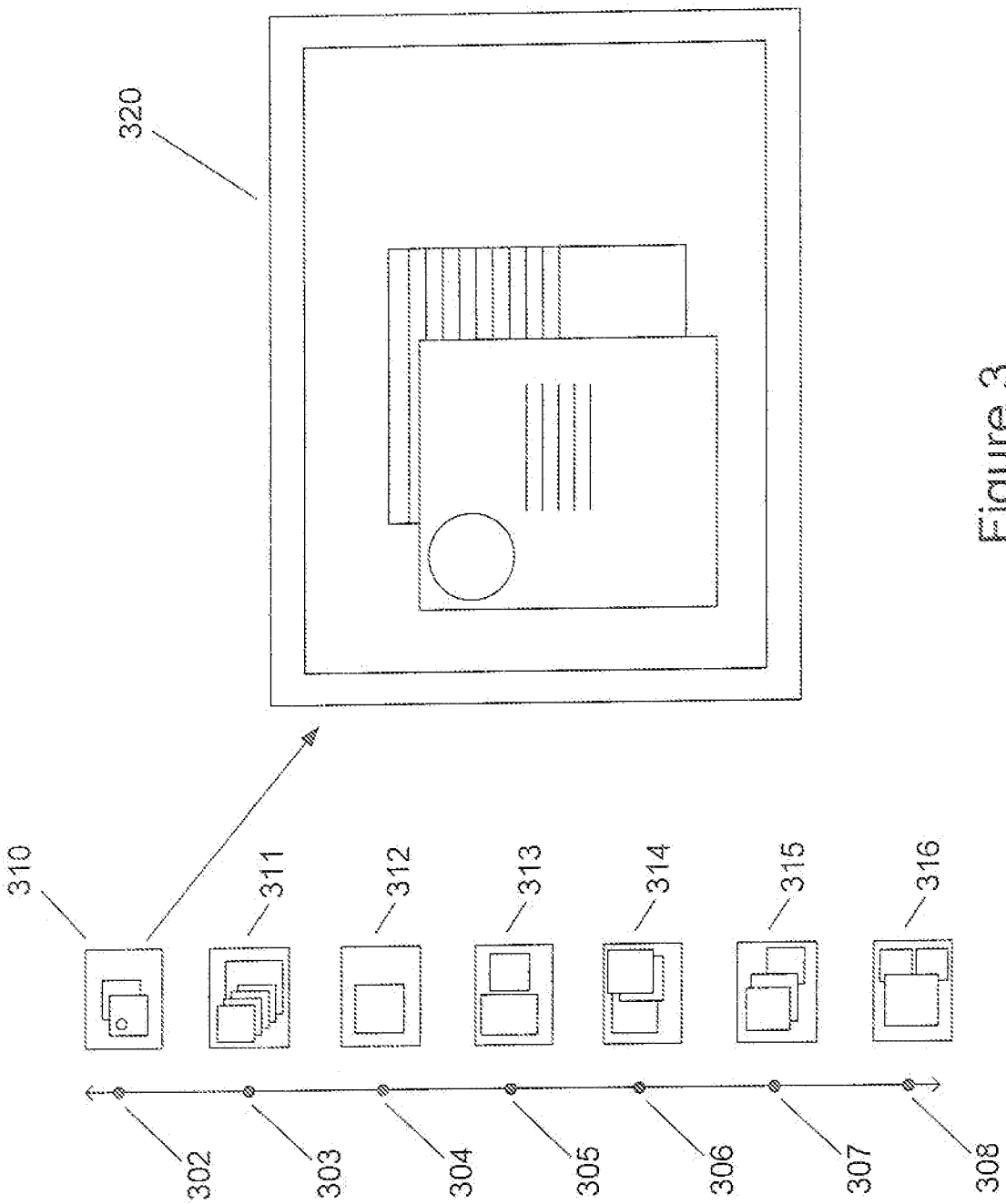


Figure 3

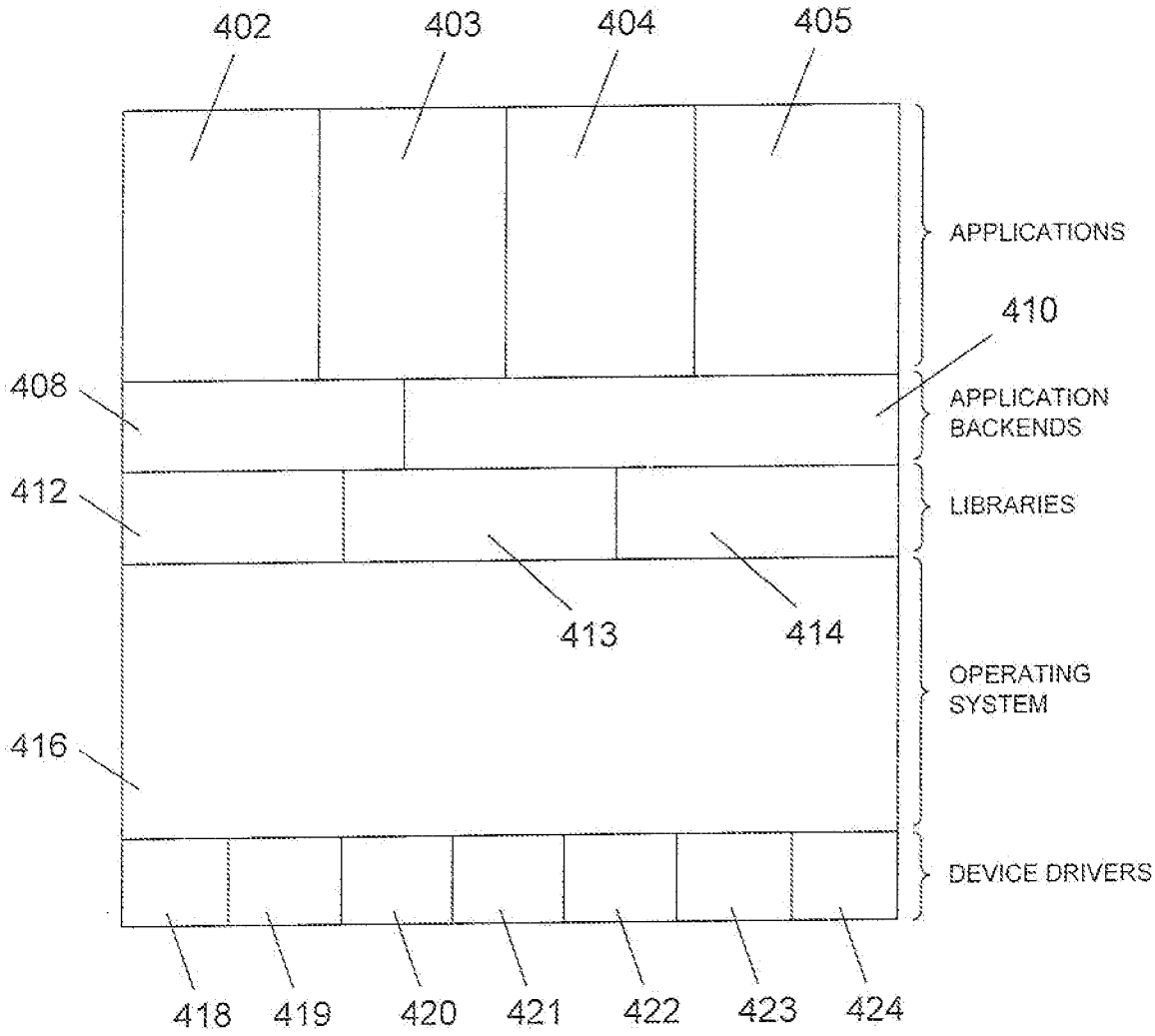

402
403
404
405
APPLICATIONS
410
408
APPLICATION BACKENDS
412
LIBRARIES
413
414
OPERATING SYSTEM
416
DEVICE DRIVERS
418
419
420
421
422
423
424

Figure 4

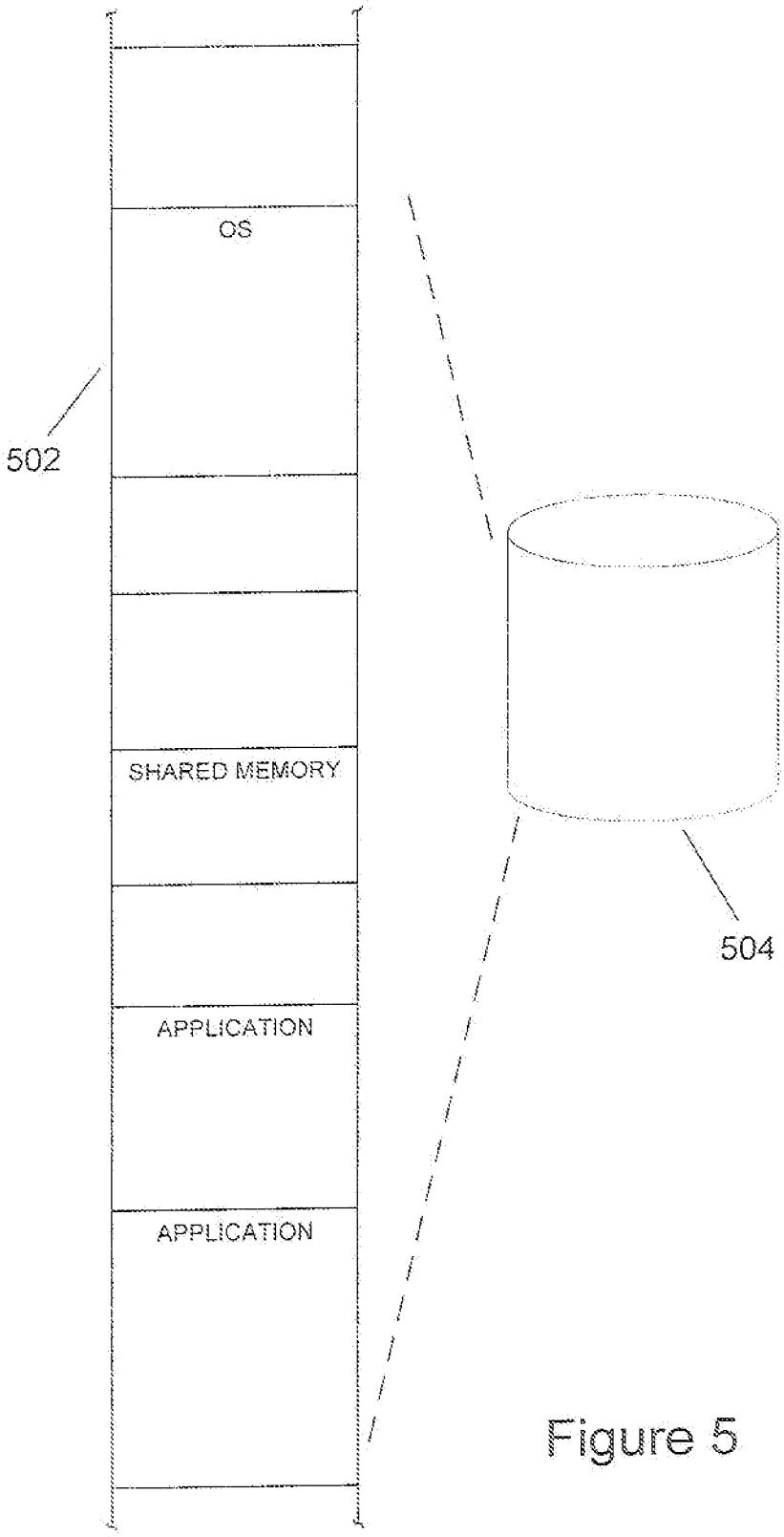


Figure 5

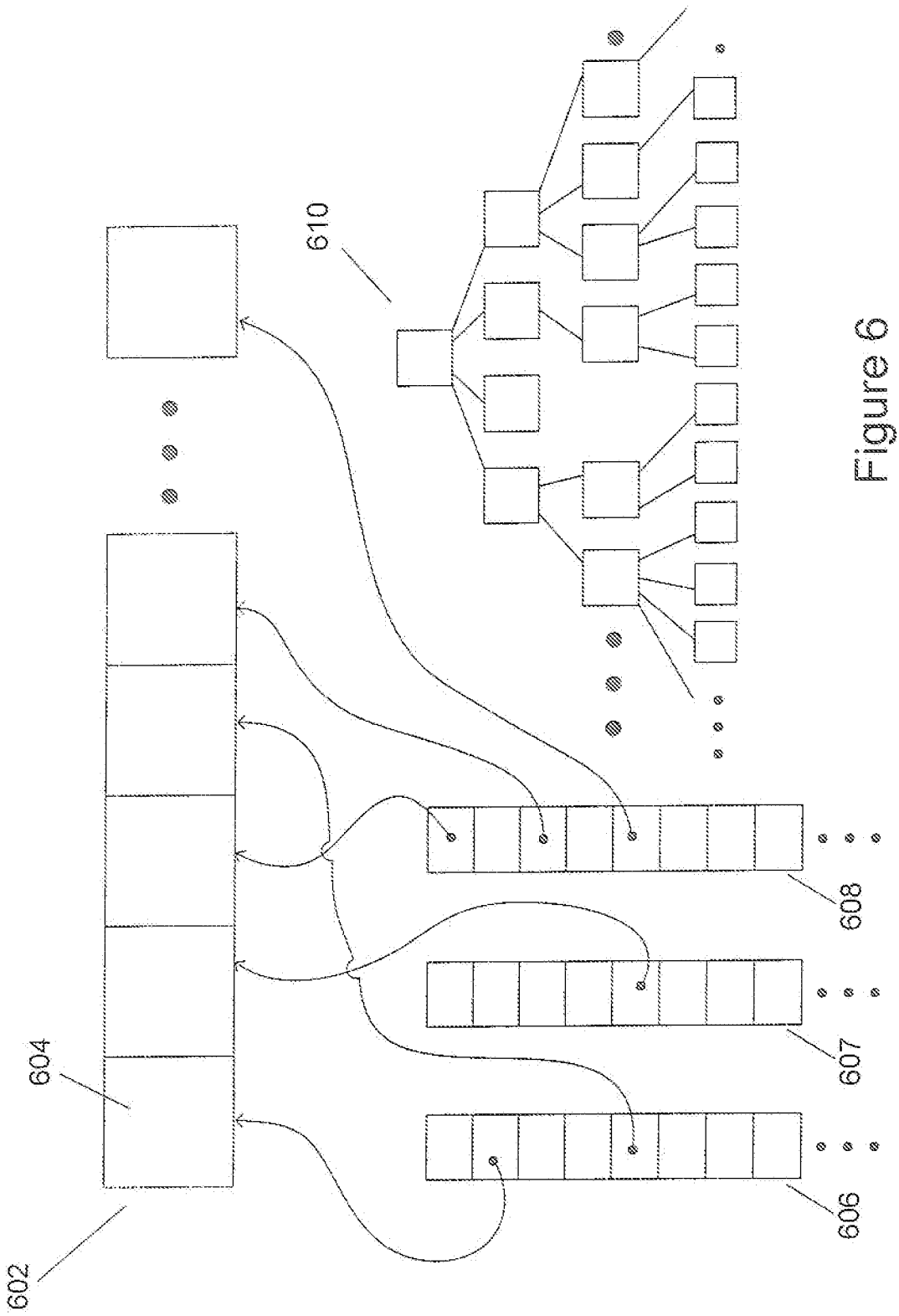


Figure 6

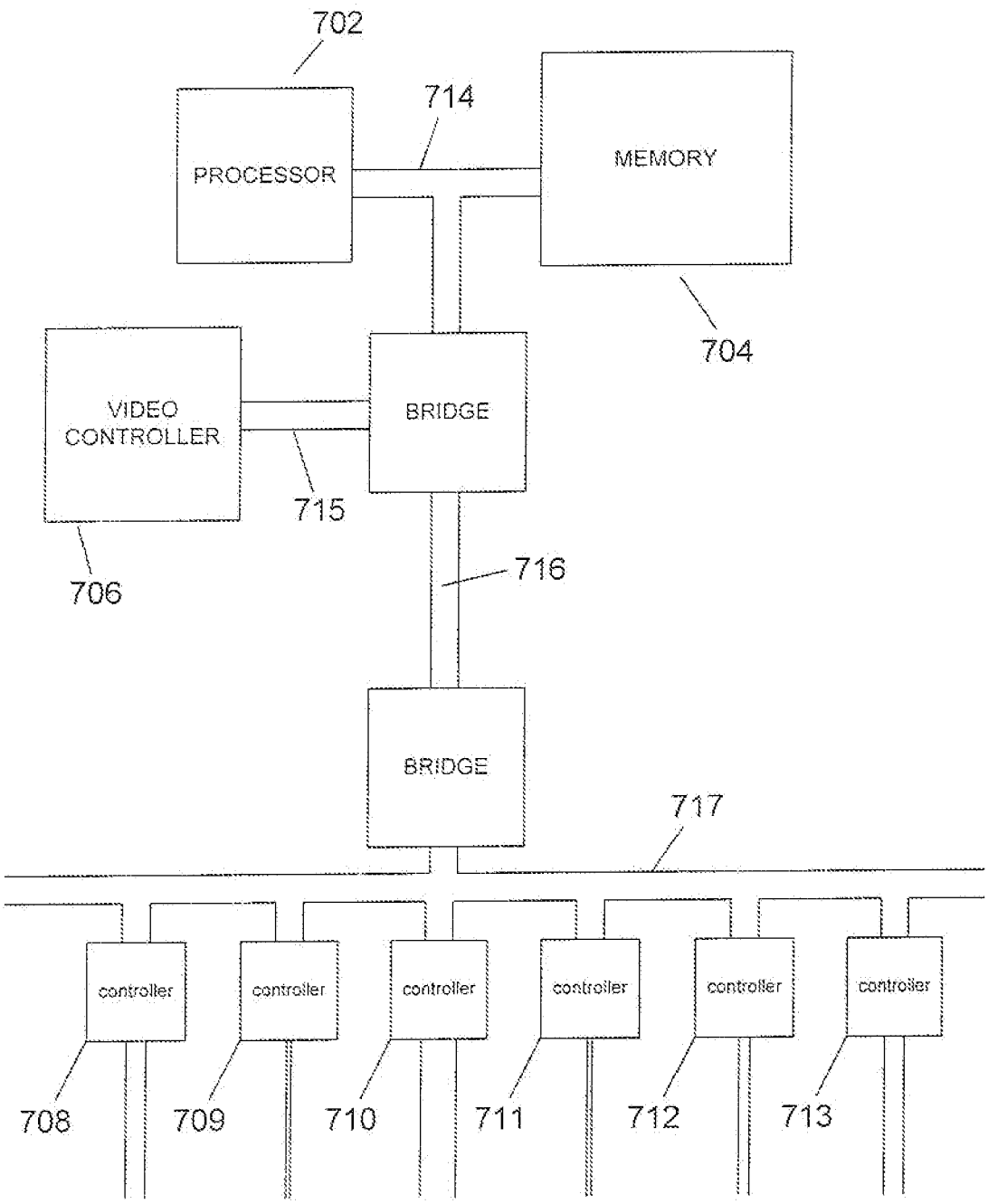

702
PROCESSOR
714
MEMORY
704
VIDEO CONTROLLER
BRIDGE
715
706
716
BRIDGE
717
controller
controller
controller
controller
controller
controller
708
709
710
711
712
713

Figure 7

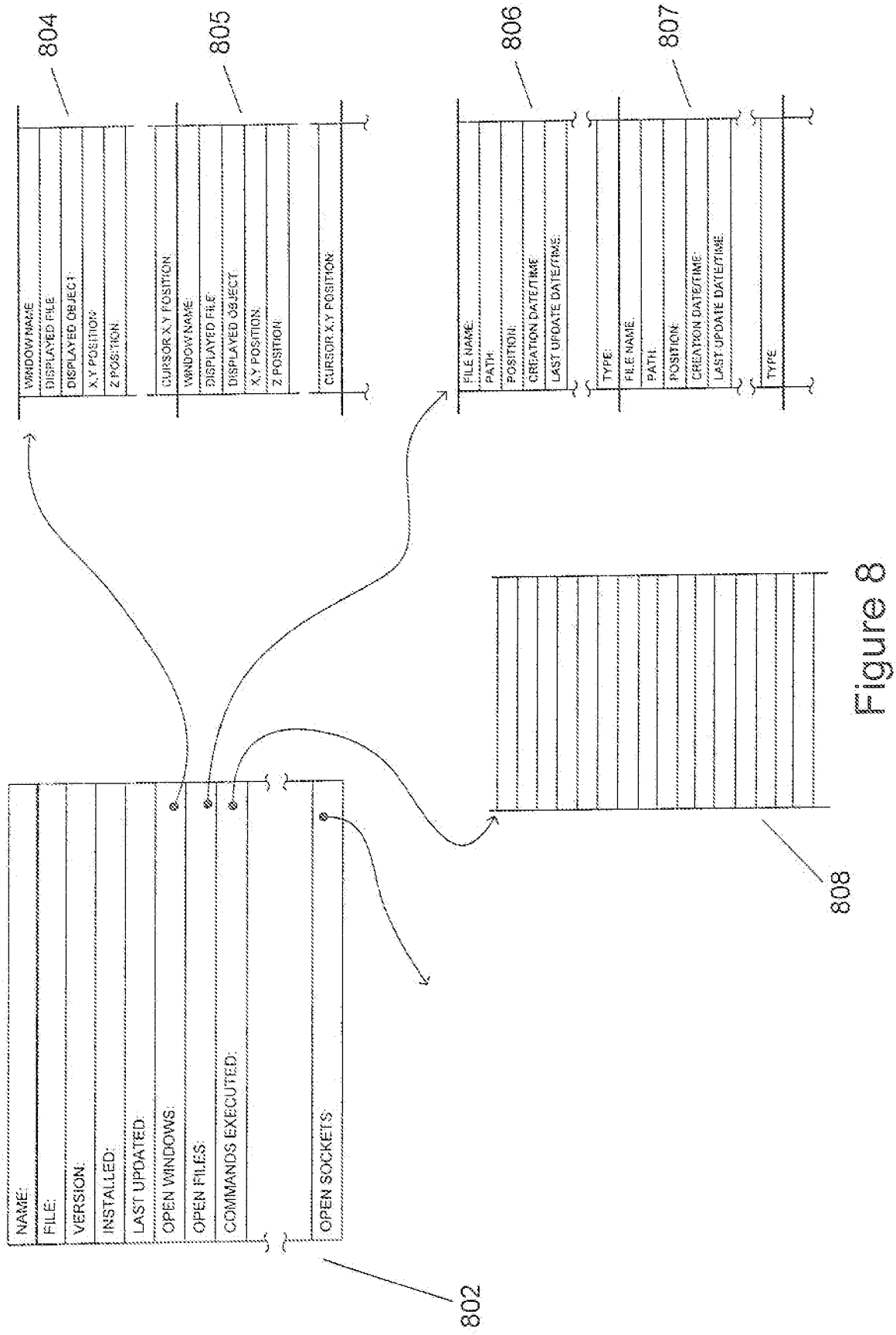

804
805
806
807
NAME:
FILE:
VERSION:
INSTALLED:
LAST UPDATED:
OPEN WINDOWS:
OPEN FILES:
COMMANDS EXECUTED:
OPEN SOCKETS:
802
WINDOW NAME:
DISPLAYED FILE
DISPLAYED OBJECT
X,Y POSITION:
Z POSITION:
CURSOR X,Y POSITION:
WINDOW NAME:
DISPLAYED FILE:
DISPLAYED OBJECT:
X,Y POSITION:
Z POSITION:
CURSOR X,Y POSITION:
FILE NAME:
PATH:
POSITION:
CREATION DATE/TIME:
LAST UPDATE DATE/TIME:
TYPE:
FILE NAME:
PATH:
POSITION:
CREATION DATE/TIME:
LAST UPDATE DATE/TIME:
TYPE
808

Figure 8

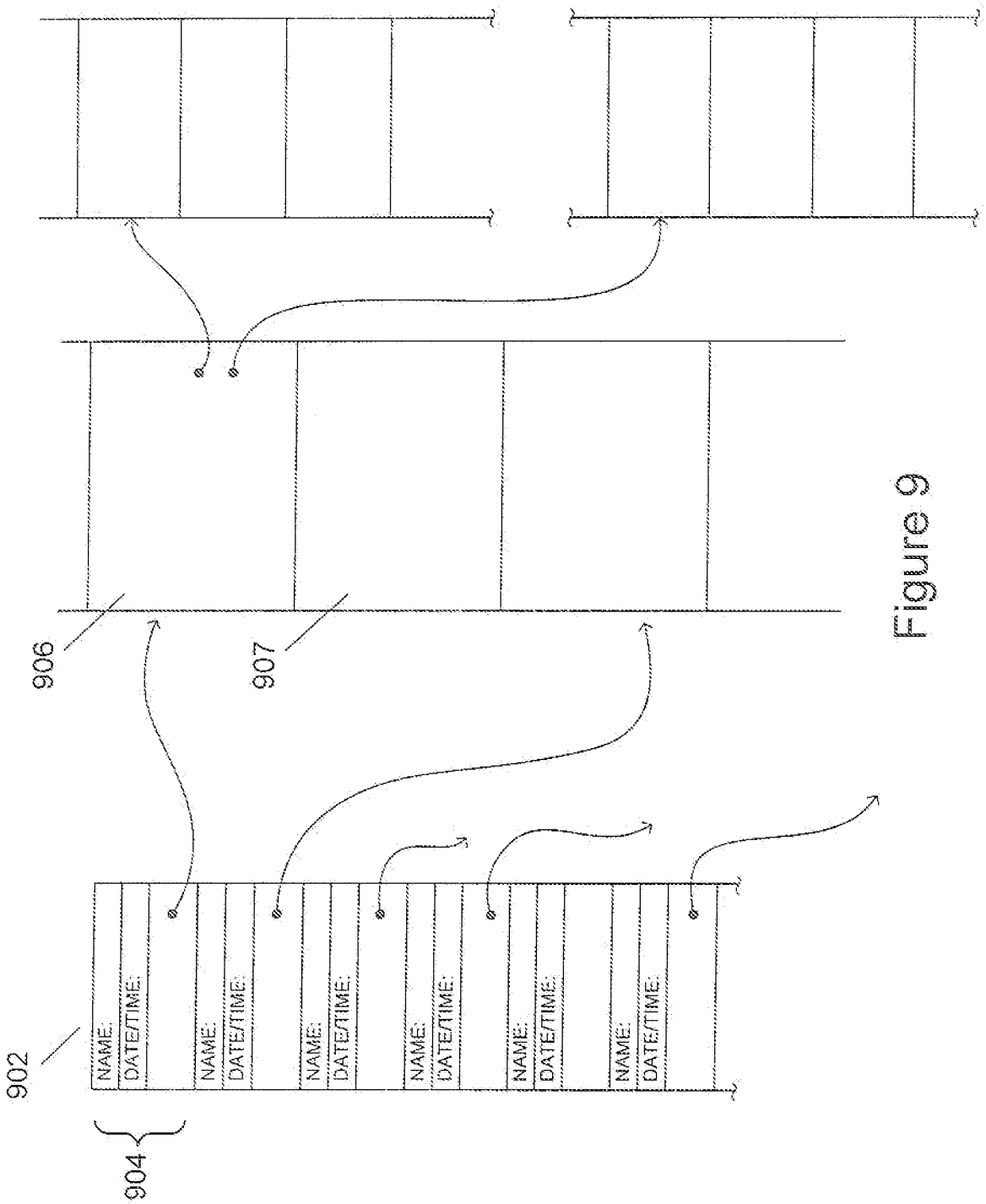


Figure 9

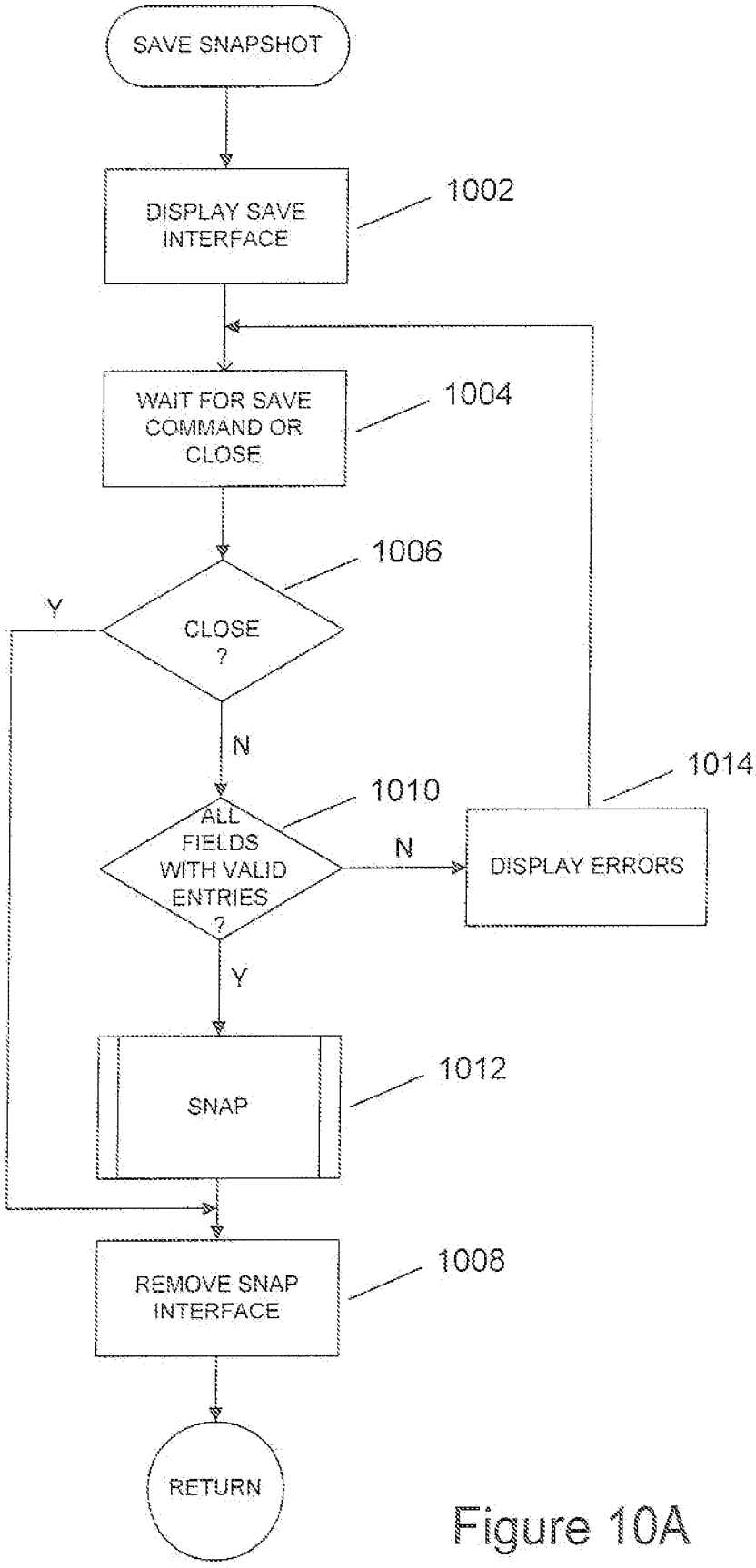

SAVE SNAPSHOT
DISPLAY SAVE INTERFACE
1002
WAIT FOR SAVE COMMAND OR CLOSE
1004
1006
CLOSE ?
Y
N
1010
ALL FIELDS WITH VALID ENTRIES ?
N
DISPLAY ERRORS
1014
Y
SNAP
1012
REMOVE SNAP INTERFACE
1008
RETURN

Figure 10A

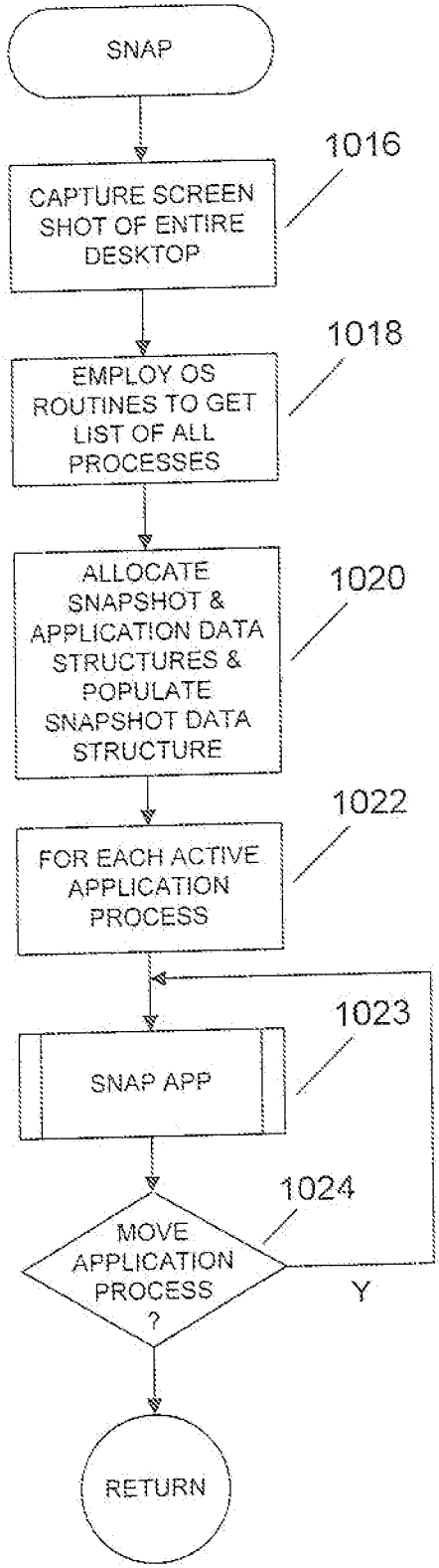

SNAP
1016
CAPTURE SCREEN SHOT OF ENTIRE DESKTOP
1018
EMPLOY OS ROUTINES TO GET LIST OF ALL PROCESSES
1020
ALLOCATE SNAPSHOT & APPLICATION DATA STRUCTURES & POPULATE SNAPSHOT DATA STRUCTURE
1022
FOR EACH ACTIVE APPLICATION PROCESS
1023
SNAP APP
1024
MOVE APPLICATION PROCESS ?
Y
RETURN

Figure 10B

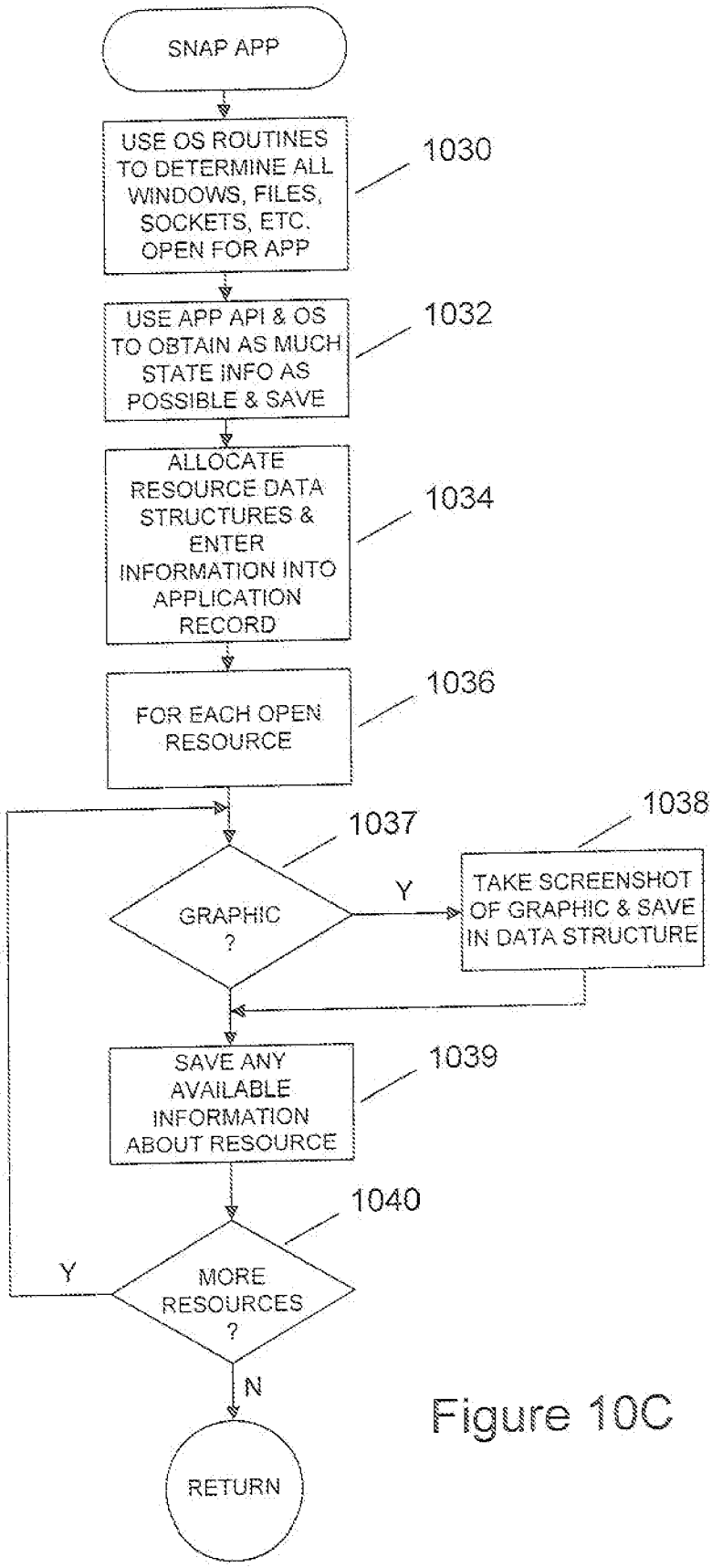

SNAP APP
USE OS ROUTINES TO DETERMINE ALL WINDOWS, FILES, SOCKETS, ETC. OPEN FOR APP
1030
USE APP API & OS TO OBTAIN AS MUCH STATE INFO AS POSSIBLE & SAVE
1032
ALLOCATE RESOURCE DATA STRUCTURES & ENTER INFORMATION INTO APPLICATION RECORD
1034
FOR EACH OPEN RESOURCE
1036
1037
GRAPHIC ?
Y
1038
TAKE SCREENSHOT OF GRAPHIC & SAVE IN DATA STRUCTURE
SAVE ANY AVAILABLE INFORMATION ABOUT RESOURCE
1039
1040
MORE RESOURCES ?
Y
N
RETURN

Figure 10C

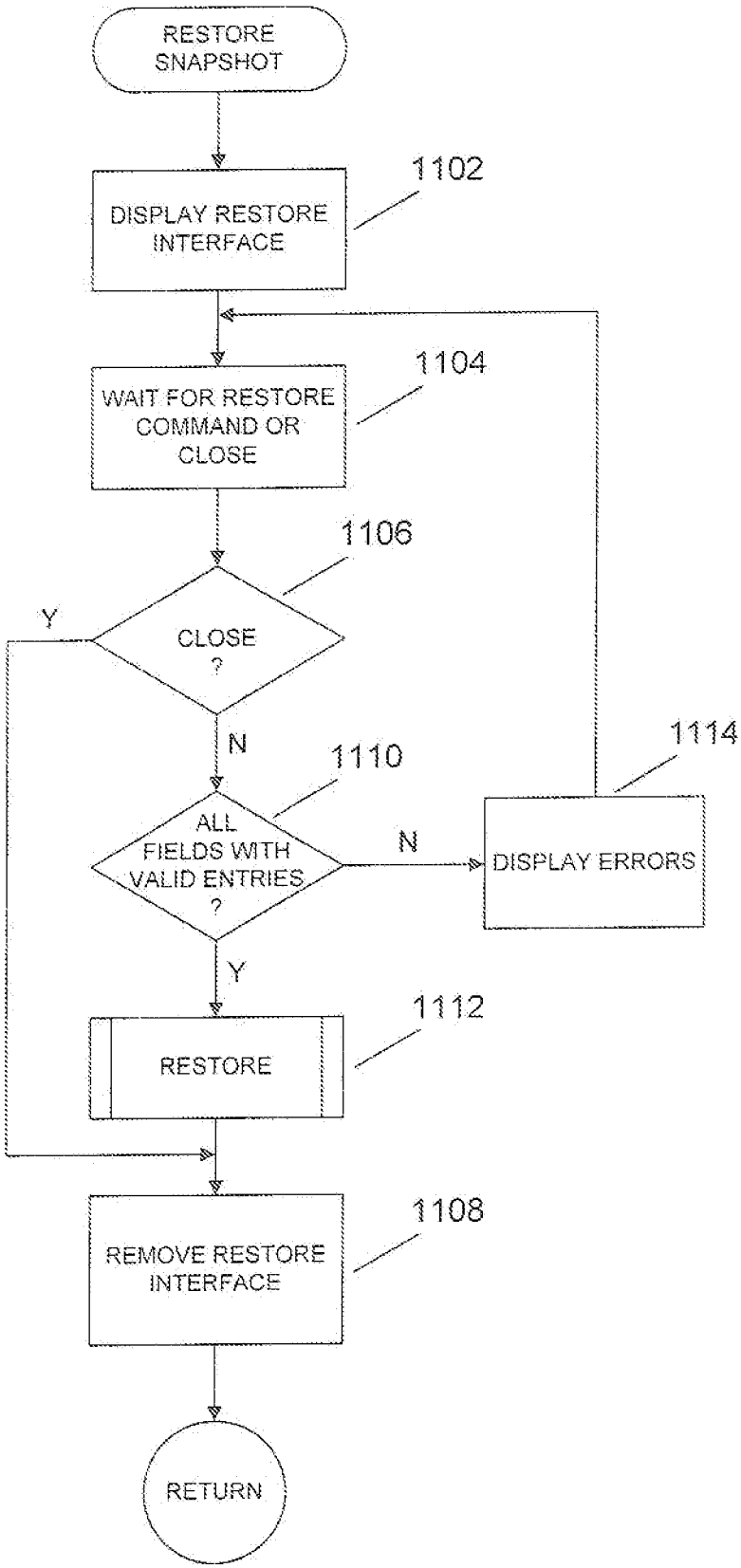

RESTORE SNAPSHOT
DISPLAY RESTORE INTERFACE
1102
WAIT FOR RESTORE COMMAND OR CLOSE
1104
CLOSE ?
1106
Y
N
ALL FIELDS WITH VALID ENTRIES ?
1110
N
DISPLAY ERRORS
1114
Y
RESTORE
1112
REMOVE RESTORE INTERFACE
1108
RETURN

Figure 11A

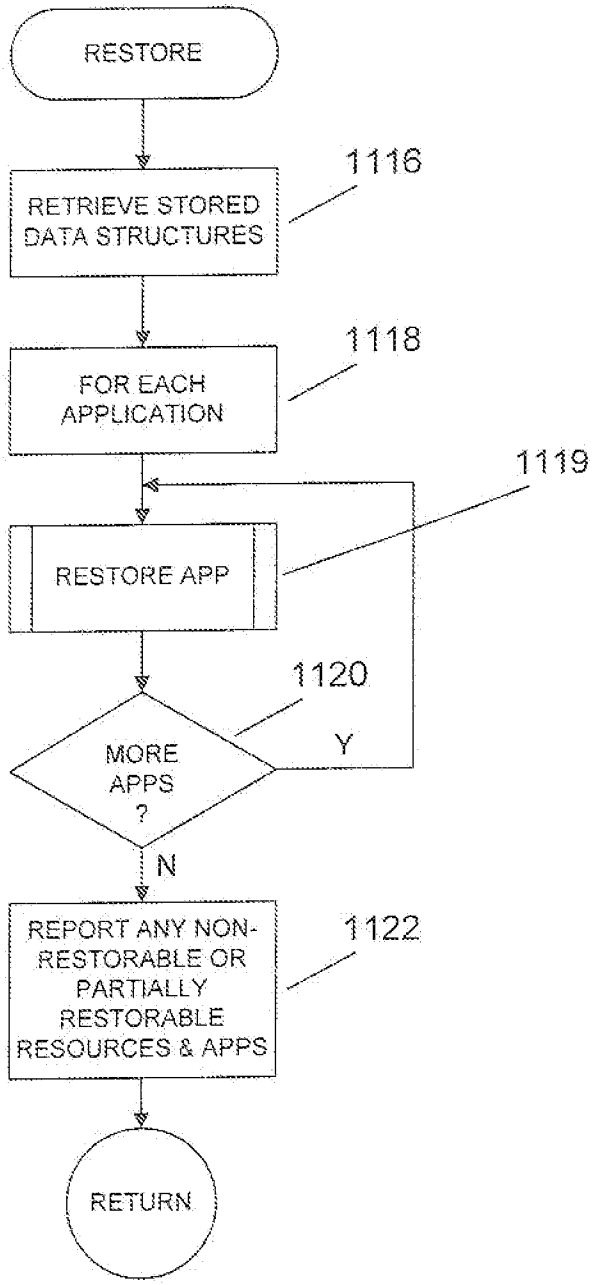

RESTORE
RETRIEVE STORED DATA STRUCTURES
1116
FOR EACH APPLICATION
1118
RESTORE APP
1119
MORE APPS ?
1120
Y
N
REPORT ANY NON-RESTORABLE OR PARTIALLY RESTORABLE RESOURCES & APPS
1122
RETURN

Figure 11B

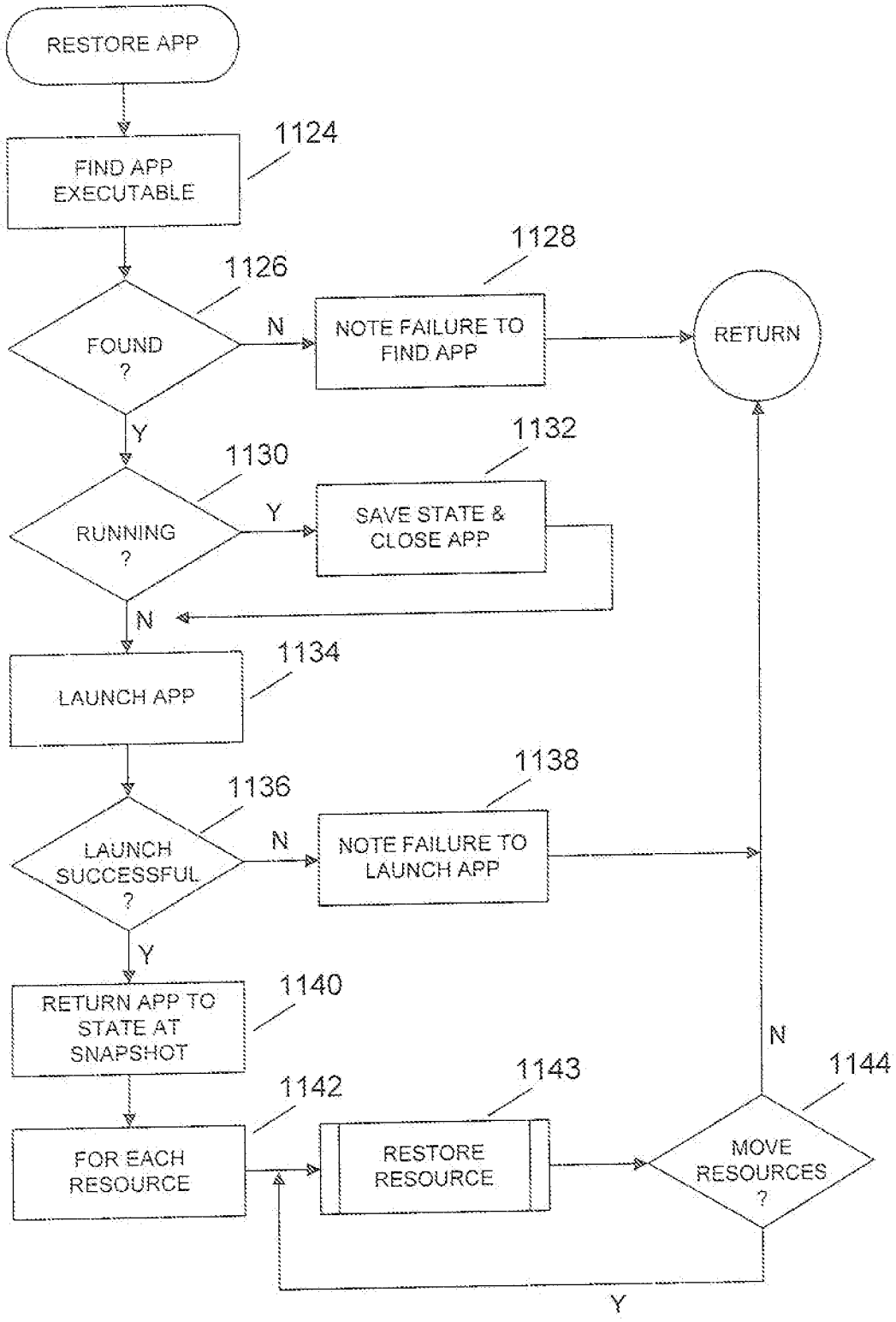

RESTORE APP
FIND APP EXECUTABLE
1124
FOUND ?
1126
N
NOTE FAILURE TO FIND APP
1128
RETURN
Y
RUNNING ?
1130
Y
SAVE STATE & CLOSE APP
1132
N
LAUNCH APP
1134
LAUNCH SUCCESSFUL ?
1136
N
NOTE FAILURE TO LAUNCH APP
1138
Y
RETURN APP TO STATE AT SNAPSHOT
1140
FOR EACH RESOURCE
1142
RESTORE RESOURCE
1143
MOVE RESOURCES ?
1144
N
Y

Figure 11C

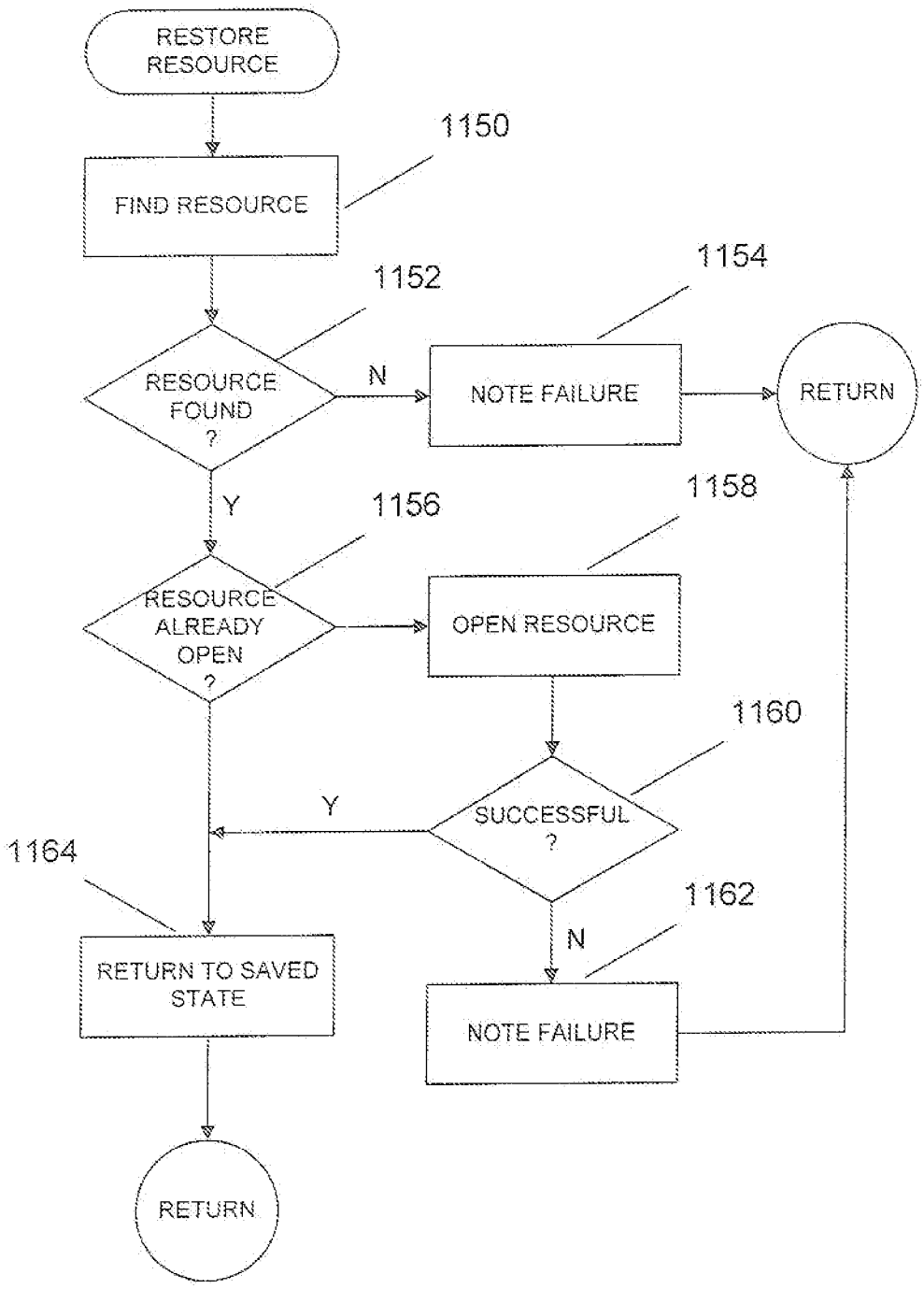

RESTORE RESOURCE
1150
FIND RESOURCE
1152
RESOURCE FOUND ?
N
1154
NOTE FAILURE
RETURN
Y
1156
RESOURCE ALREADY OPEN ?
1158
OPEN RESOURCE
1160
SUCCESSFUL ?
Y
1164
RETURN TO SAVED STATE
N
1162
NOTE FAILURE
RETURN

Figure 11D

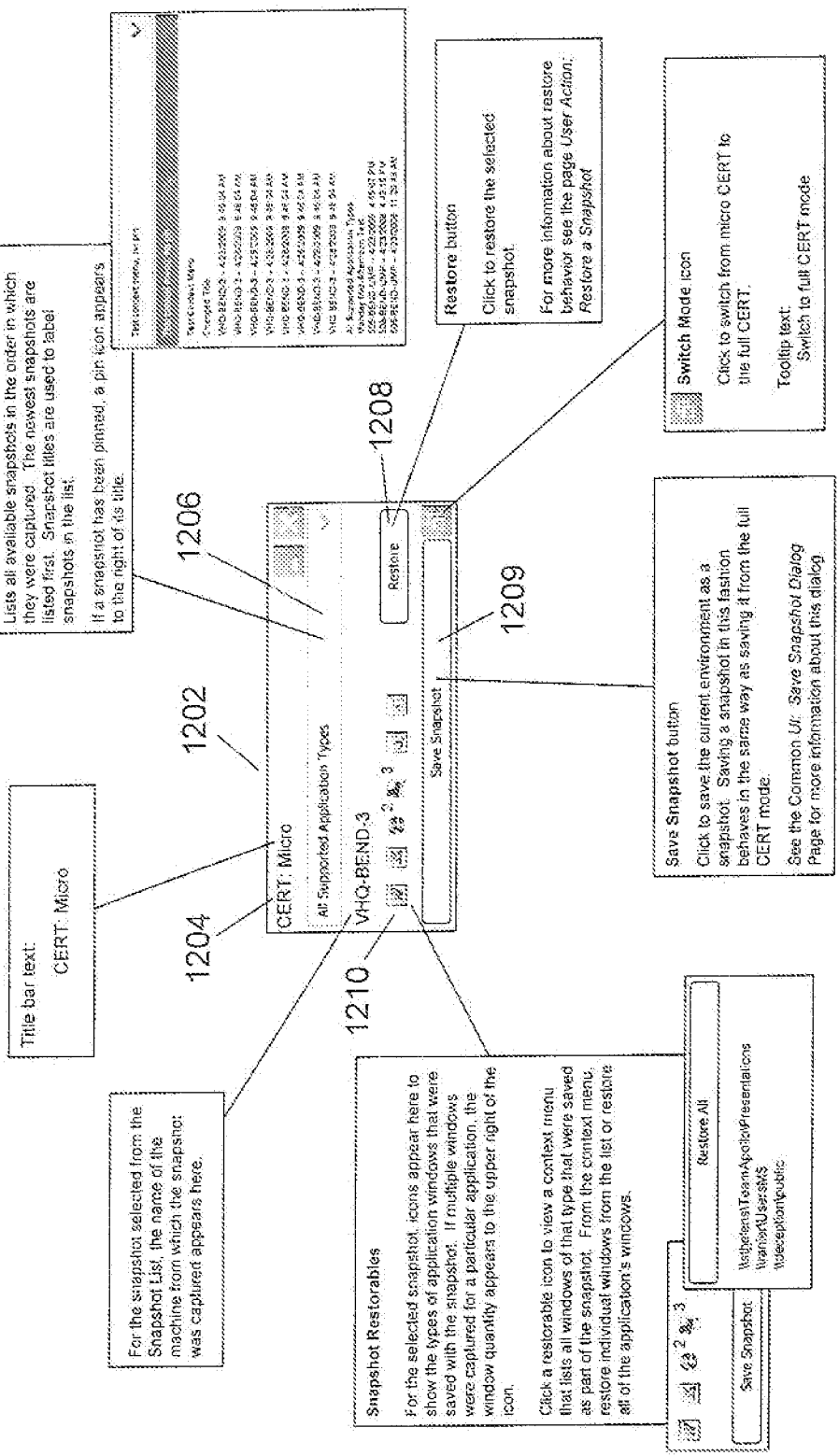

Snapshot List
Lists all available snapshots in the order in which they were captured. The newest snapshots are listed first. Snapshot titles are used to label snapshots in the list.
If a snapshot has been pinned, a pin icon appears to the right of its title.
Title bar text:
CERT: Micro
For the snapshot selected from the Snapshot List, the name of the machine from which the snapshot was captured appears here.
CERT: Micro
All Supported Application Types
VHQ-BEND-3
Restore
Save Snapshot
Restore button
Click to restore the selected snapshot.
For more information about restore behavior see the page User Action: Restore a Snapshot
Switch Mode icon
Click to switch from micro CERT to the full CERT.
Tooltip text:
Switch to full CERT mode
Save Snapshot button
Click to save the current environment as a snapshot. Saving a snapshot in this fashion behaves in the same way as saving it from the full CERT mode.
See the Common UI: Save Snapshot Dialog Page for more information about this dialog
Snapshot Restorables
For the selected snapshot, icons appear here to show the types of application windows that were saved with the snapshot. If multiple windows were captured for a particular application, the window quantity appears to the upper right of the icon.
Click a restorable icon to view a context menu that lists all windows of that type that were saved as part of the snapshot. From the context menu, restore individual windows from the list or restore all of the application's windows.
Restore All
Save Snapshot
1202
1204
1206
1208
1209
1210

Figure 12A

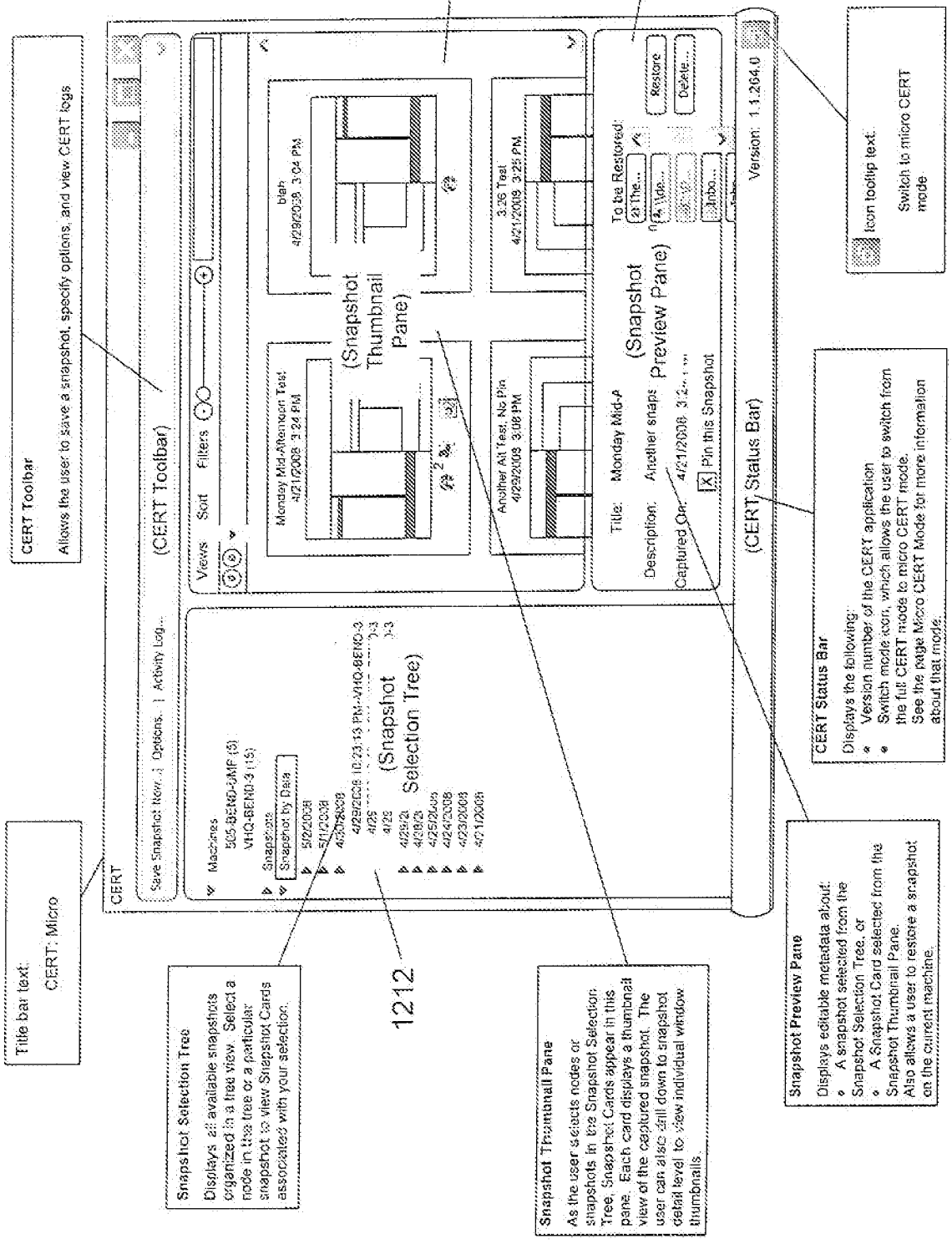

Title bar text:
CERT: Micro
CERT Toolbar
Allows the user to save a snapshot, specify options, and view CERT logs
(CERT Toolbar)
(Snapshot Selection Tree)
(Snapshot Thumbnail Pane)
(Snapshot Preview Pane)
(CERT Status Bar)
1212
1213
1214
Snapshot Selection Tree
Displays all available snapshots organized in a tree view. Select a node in the tree or a particular snapshot to view Snapshot Cards associated with your selection.
Snapshot Thumbnail Pane
As the user selects nodes or snapshots in the Snapshot Selection Tree, Snapshot Cards appear in this pane. Each card displays a thumbnail view of the captured snapshot. The user can also drill down to snapshot detail level to view individual window thumbnails.
Snapshot Preview Pane
Displays editable metadata about:
A snapshot selected from the Snapshot Selection Tree, or
A Snapshot Card selected from the Snapshot Thumbnail Pane.
Also allows a user to restore a snapshot on the current machine.
CERT Status Bar
Displays the following:
Version number of the CERT application
Switch mode icon, which allows the user to switch from the full CERT mode to micro CERT mode.
See the page Micro CERT Mode for more information about that mode.
Icon tooltip text:
Switch to micro CERT mode

Figure 12B

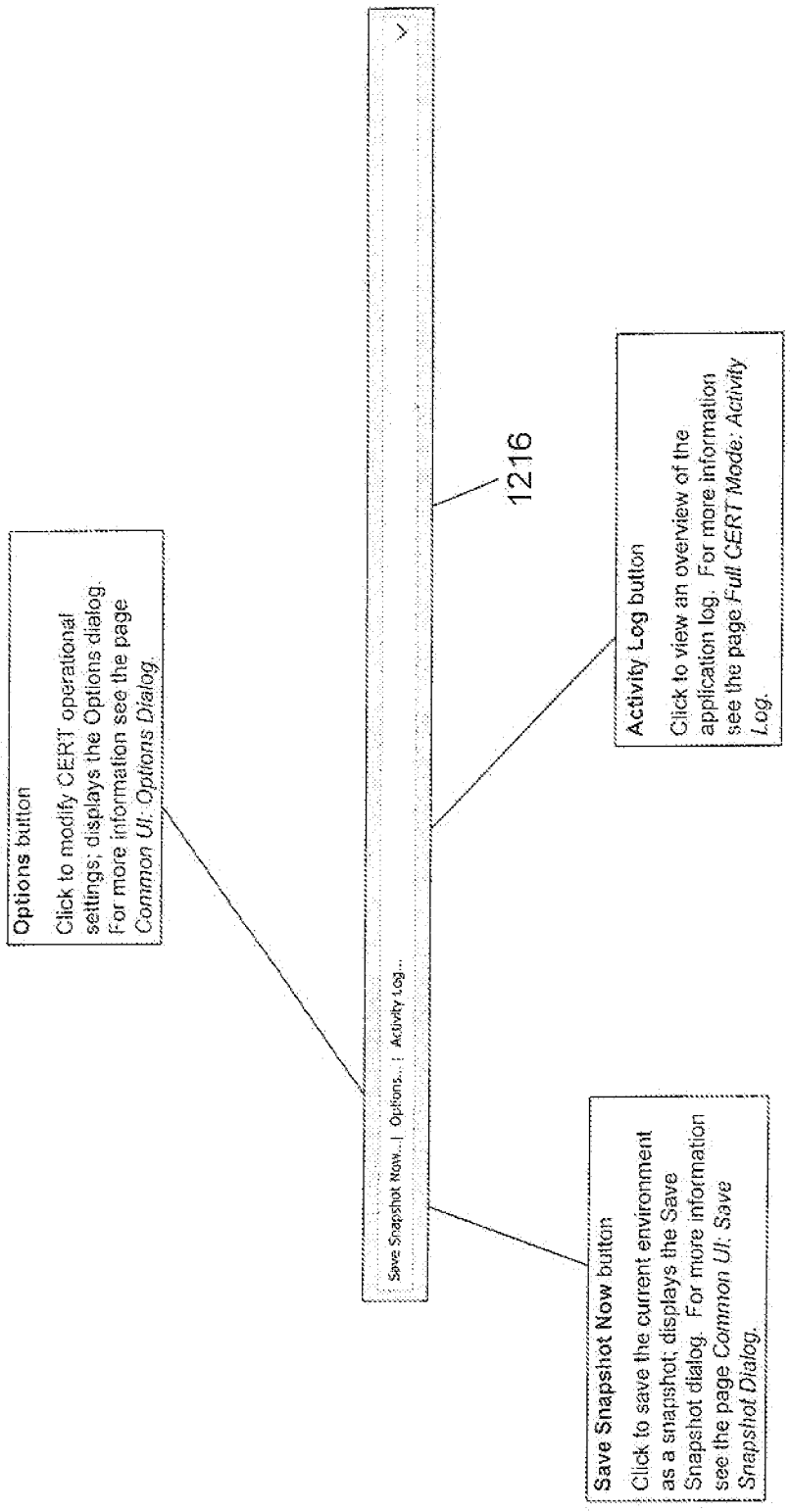

Options button
Click to modify CERT operational settings; displays the Options dialog. For more information see the page Common UI: Options Dialog.
Save Snapshot Now... | Options... | Activity Log...
1216
Activity Log button
Click to view an overview of the application log. For more information see the page Full CERT Mode: Activity Log.
Save Snapshot Now button
Click to save the current environment as a snapshot; displays the Save Snapshot dialog. For more information see the page Common UI: Save Snapshot Dialog.

Figure 12C

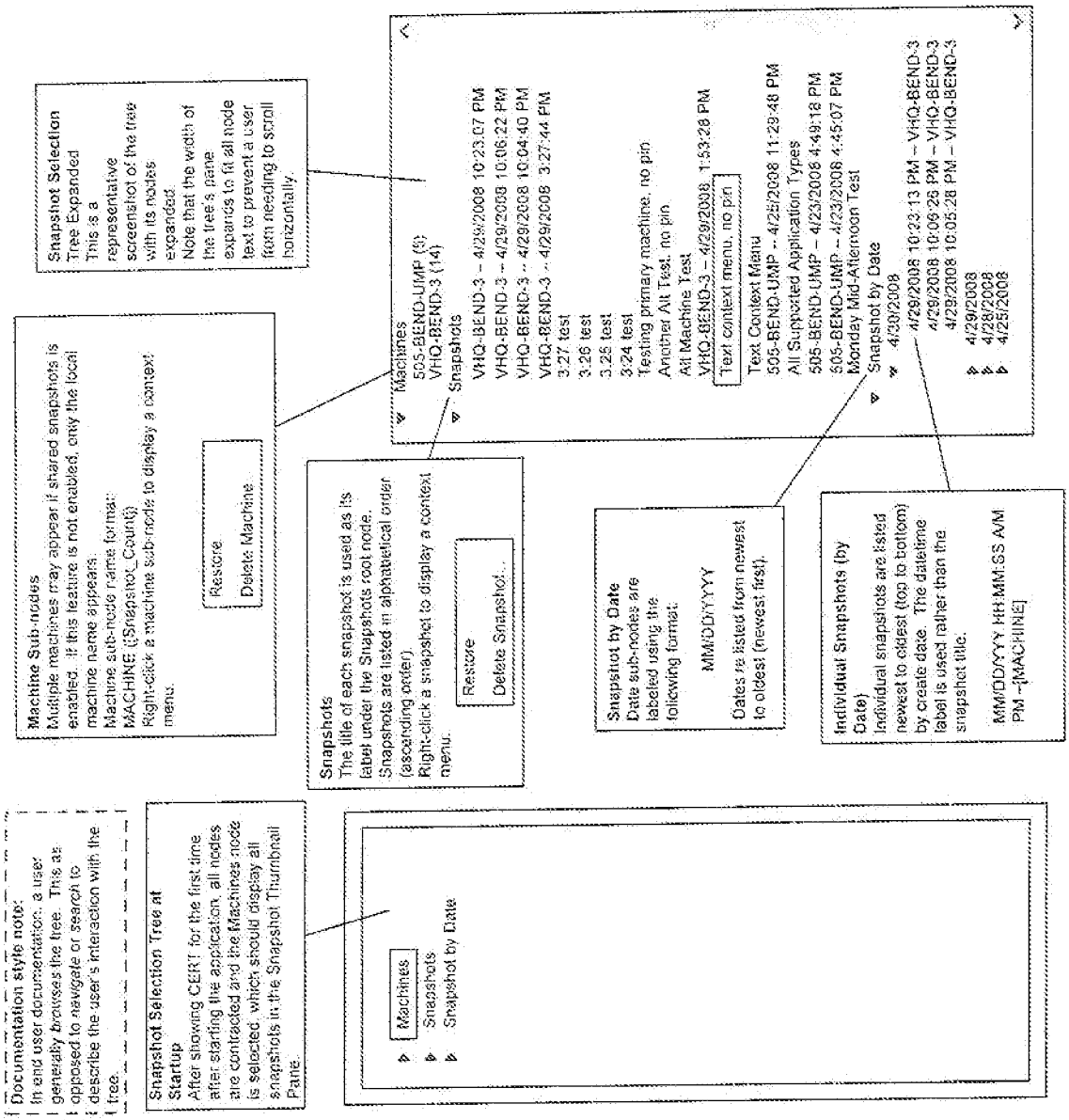

Selection Tree Terminology
The user browses a tree structure to find snapshots. The tree consists of root nodes and each one can contain either snapshots or additional sub-nodes.
> Root node contracted
v Root node expanded
> Sub-node contracted
v Sub-node expanded
Snapshot
Snapshot
Documentation style note:
In end user documentation, a user generally browses the tree. This as opposed to navigate or search to describe the user's interaction with the tree.
Selection Tree Structure
Three root nodes allow the user to browse available snapshots.
The Machines root node presents all snapshots organized by the machine from which the snapshots were saved.
The Snapshots root node lists all available snapshots.
The Snapshot by Date root node groups snapshots by the date on which these snapshots were created (as opposed to the date last modified).
Selection Behavior
A selected node or snapshot appears shaded in purple.
When a user selects a node in the tree, all snapshots contained within that node and its sub-nodes should be displayed in the Snapshot Thumbnail Pane, the first snapshot listed is selected by default, and the Snapshot Thumbnail Pane is scrolled to its top-most position.
When a user selects a snapshot in the tree, its snapshot card appears in the Snapshot Thumbnail Pane.
Context Menus
The user can right-click a snapshot in the tree as well as any sub-nodes of the root nodes.
Snapshot Selection Tree at Startup
After showing CERT for the first time after starting the application, all nodes are contracted and the Machines node is selected, which should display all snapshots in the Snapshot Thumbnail Pane.
Machines
Snapshots
Snapshot by Date
Machine Sub-nodes
Multiple machines may appear if shared snapshots is enabled. If this feature is not enabled, only the local machine name appears.
Machine sub-node name format:
MACHINE ((Snapshot_Count))
Right-click a machine sub-node to display a context menu.
Restore
Delete Machine
Snapshots
The title of each snapshot is used as its label under the Snapshots root node.
Snapshots are listed in alphabetical order (ascending order).
Right-click a snapshot to display a context menu.
Restore
Delete Snapshot...
Snapshot by Date
Date sub-nodes are labeled using the following format:
MM/DD/YYYY
Dates are listed from newest to oldest (newest first).
Individual Snapshots (by Date)
Individual snapshots are listed newest to oldest (top to bottom) by create date. The datetime label is used rather than the snapshot title.
MM/DD/YYY HH:MM:SS A/M PM -[MACHINE]
Machines
505-BEND-UMP (5)
VHQ-BEND-3 (14)
Snapshots
VHQ-BEND-3 – 4/29/2008 10:23:07 PM
VHQ-BEND-3 – 4/29/2008 10:06:22 PM
VHQ-BEND-3 – 4/29/2008 10:04:40 PM
VHQ-BEND-3 – 4/29/2008 3:27:44 PM
3:27 test
3:26 test
3:25 test
3:24 test
Testing primary machine, no pin
Another Alt Test, no pin
Alt Machine Test
VHQ-BEND-3 – 4/29/2008 1:53:28 PM
Text context menu, no pin
Text Context Menu
505-BEND-UMP – 4/25/2008 11:29:48 PM
All Supported Application Types
505-BEND-UMP – 4/23/2008 4:49:18 PM
505-BEND-UMP – 4/23/2008 4:45:07 PM
Monday Mid-Afternoon Test
Snapshot by Date
4/30/2008
4/29/2008 10:23:13 PM – VHQ-BEND-3
4/29/2008 10:06:26 PM – VHQ-BEND-3
4/29/2008 10:05:28 PM – VHQ-BEND-3
4/29/2008
4/28/2008
4/25/2008
Snapshot Selection
Tree Expanded
This is a representative screenshot of the tree with its nodes expanded.
Note that the width of the tree's pane expands to fit all node text to prevent a user from needing to scroll horizontally.

Figure 12D

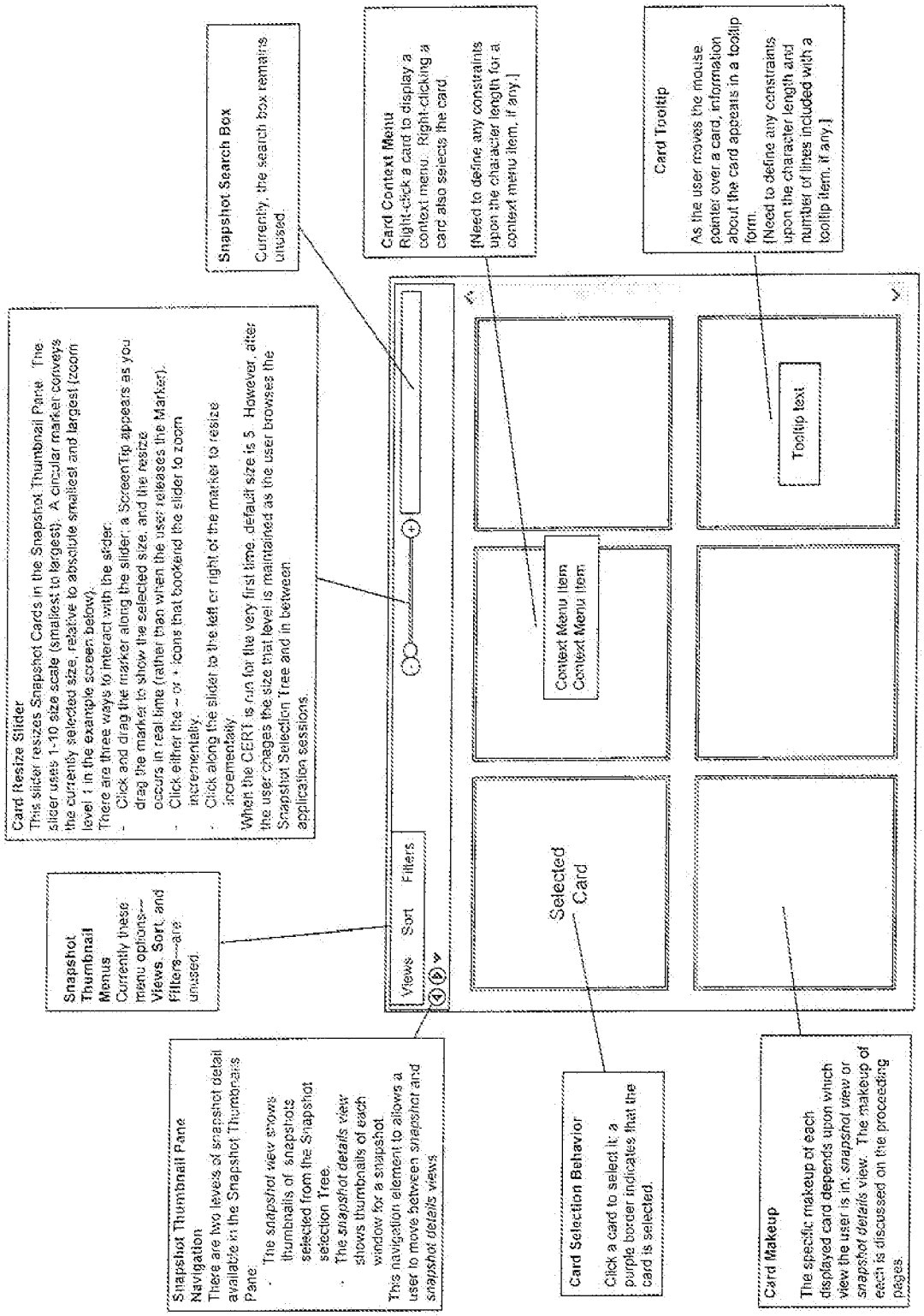

Card Resize Slider
This slider resizes Snapshot Cards in the Snapshot Thumbnail Pane. The slider uses 1-10 size scale (smallest to largest). A circular marker conveys the currently selected size, relative to absolute smallest and largest (zoom level 1 in the example screen below).
There are three ways to interact with the slider:
- Click and drag the marker along the slider; a ScreenTip appears as you drag the marker to show the selected size, and the resize occurs in real time (rather than when the user releases the Marker).
- Click either the – or + icons that bookend the slider to zoom incrementally.
- Click along the slider to the left or right of the marker to resize incrementally.
When the CERT is run for the very first time, default size is 5. However, after the user changes the size that level is maintained as the user browses the Snapshot Selection Tree and in between application sessions.
Snapshot Search Box
Currently, the search box remains unused.
Card Context Menu
Right-click a card to display a context menu. Right-clicking a card also selects the card.
[Need to define any constraints upon the character length for a context menu item, if any.]
Card Tooltip
As the user moves the mouse pointer over a card, information about the card appears in a tooltip form.
[Need to define any constraints upon the character length and number of lines included with a tooltip item, if any.]
Snapshot Thumbnail Menus
Currently these menu options—Views, Sort, and Filters—are unused.
Views
Sort
Filters
Context Menu Item
Context Menu Item
Tooltip text
Selected Card
Snapshot Thumbnail Pane Navigation
There are two levels of snapshot detail available in the Snapshot Thumbnails Pane:
- The *snapshot view* shows thumbnails of snapshots selected from the Snapshot selection Tree.
- The *snapshot details view* shows thumbnails of each window for a snapshot.
This navigation element to allows a user to move between *snapshot* and *snapshot details* views.
Card Selection Behavior
Click a card to select it; a purple border indicates that the card is selected.
Card Makeup
The specific makeup of each displayed card depends upon which view the user is in: *snapshot view* or *snapshot details* view. The makeup of each is discussed on the proceeding pages.

Figure 12E

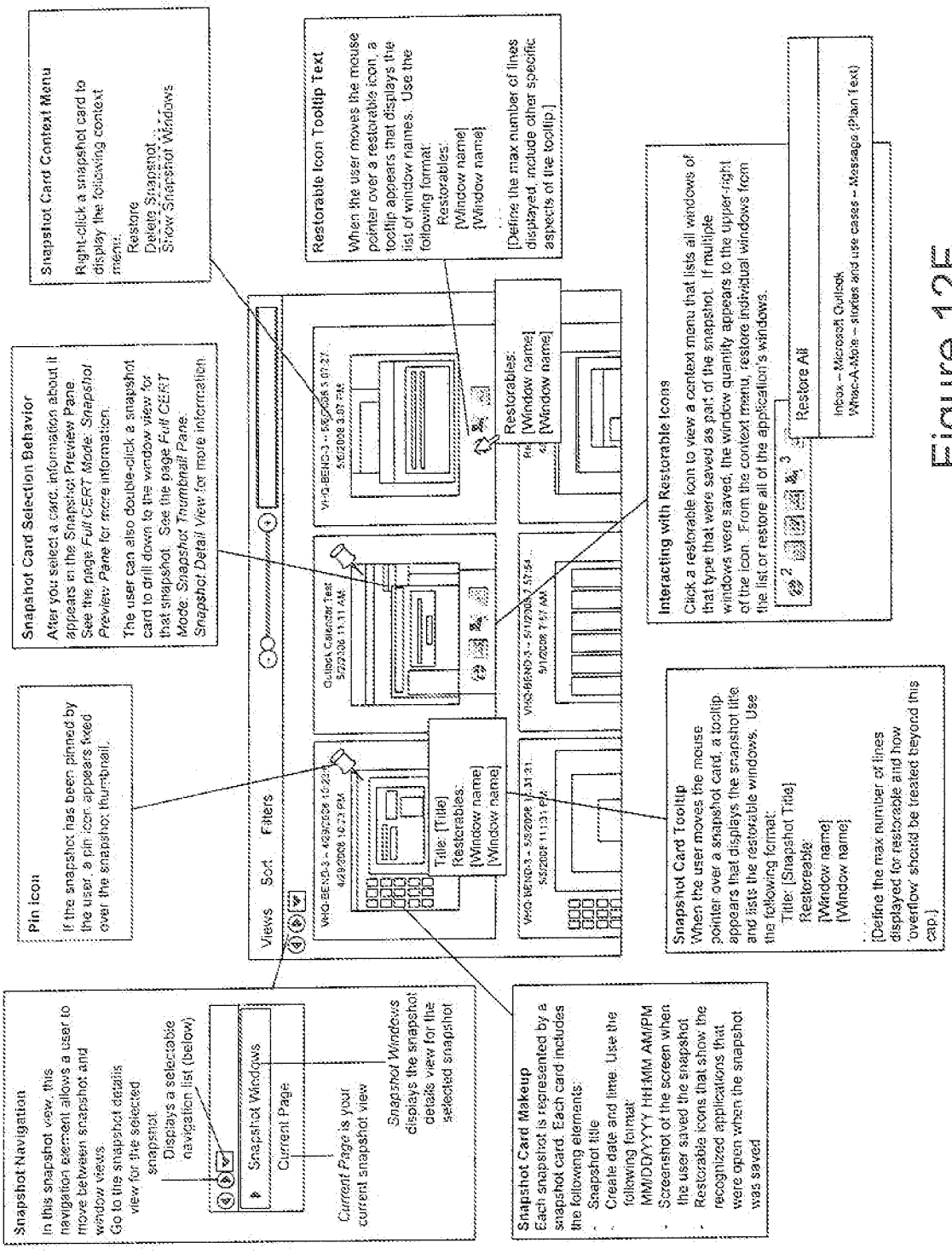

Snapshot Navigation
In this snapshot view, this navigation element allows a user to move between snapshot and window views.
Go to the snapshot details view for the selected snapshot.
Displays a selectable navigation list (below)
Snapshot Windows
Current Page
Current Page is your current snapshot view
Snapshot Windows displays the snapshot details view for the selected snapshot
Pin icon
If the snapshot has been pinned by the user, a pin icon appears fixed over the snapshot thumbnail.
Snapshot Card Selection Behavior
After you select a card, information about it appears in the Snapshot Preview Pane
See the page Full CERT Mode: Snapshot Preview Pane for more information.
The user can also double-click a snapshot card to drill down to the window view for that snapshot. See the page Full CERT Mode: Snapshot Thumbnail Pane: Snapshot Detail View for more information.
Snapshot Card Context Menu
Right-click a snapshot card to display the following context menu:
Restore
Delete Snapshot
Show Snapshot Windows
Restorable Icon Tooltip Text
When the user moves the mouse pointer over a restorable icon, a tooltip appears that displays the list of window names. Use the following format:
Restorables:
[Window name]
[Window name]
...
[Define the max number of lines displayed, include other specific aspects of the tooltip.]
Views
Sort
Filters
Title: [Title]
Restorables:
[Window name]
[Window name]
Restorables:
[Window name]
[Window name]
Interacting with Restorable Icons
Click a restorable icon to view a context menu that lists all windows of that type that were saved as part of the snapshot. If multiple windows were saved, the window quantity appears to the upper-right of the icon. From the context menu, restore individual windows from the list or restore all of the application's windows.
Restore All
Inbox – Microsoft Outlook
What-A-Mole – stories and use cases – Message (Plain Text)
Snapshot Card Tooltip
When the user moves the mouse pointer over a snapshot card, a tooltip appears that displays the snapshot title and lists the restorable windows. Use the following format:
Title: [Snapshot Title]
Restorable:
[Window name]
[Window name]
...
[Define the max number of lines displayed for restorable and how 'overflow' should be treated beyond this cap.]
Snapshot Card Makeup
Each snapshot is represented by a snapshot card. Each card includes the following elements:
Snapshot title
Create date and time. Use the following format:
MM/DD/YYYY HH:MM AM/PM
Screenshot of the screen when the user saved the snapshot
Restorable icons that show the recognized applications that were open when the snapshot was saved

Figure 12F

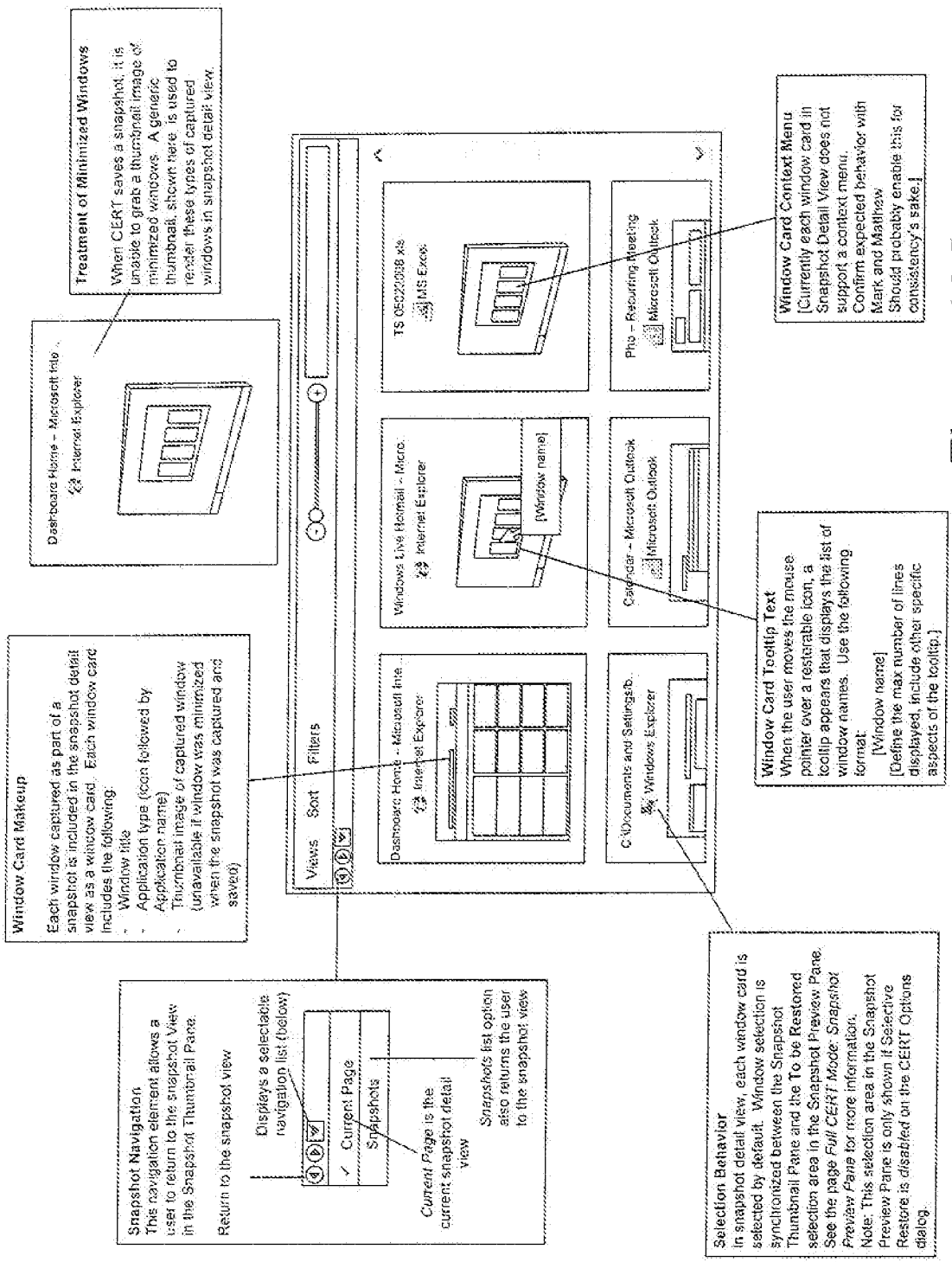

Window Card Makeup
Each window captured as part of a snapshot is included in the snapshot detail view as a window card. Each window card includes the following:
- Window title
- Application type (Icon followed by Application name)
- Thumbnail image of captured window (unavailable if window was minimized when the snapshot was captured and saved)
Treatment of Minimized Windows
When CERT saves a snapshot, it is unable to grab a thumbnail image of minimized windows. A generic thumbnail, shown here, is used to render these types of captured windows in snapshot detail view.
Dashboard Home - Microsoft Inte...
Internet Explorer
Views
Sort
Filters
Windows Live Hotmail - Micro...
Internet Explorer
[Window name]
Calendar - Microsoft Outlook
Microsoft Outlook
C:\Documents and Settings\b...
Windows Explorer
TS 080220938.xls
MS Excel
Php - Recurring Meeting
Microsoft Outlook
Window Card Context Menu
[Currently each window card in Snapshot Detail View does not support a context menu. Confirm expected behavior with Mark and Matthew
Should probably enable this for consistency's sake.]
Window Card Tooltip Text
When the user moves the mouse pointer over a restorable icon, a tooltip appears that displays the list of window names. Use the following format:
[Window name]
[Define the max number of lines displayed, include other specific aspects of the tooltip.]
Snapshot Navigation
The navigation element allows a user to return to the snapshot View in the Snapshot Thumbnail Pane.
Return to the snapshot view
Displays a selectable navigation list (below)
Current Page
Snapshots
Current Page is the current snapshot detail view
Snapshots list option also returns the user to the snapshot view
Selection Behavior
In snapshot detail view, each window card is selected by default. Window selection is synchronized between the Snapshot Thumbnail Pane and the To be Restored selection area in the Snapshot Preview Pane. See the page Full CERT Mode: Snapshot Preview Pane for more information.
Note: This selection area in the Snapshot Preview Pane is only shown if Selective Restore is disabled on the CERT Options dialog.

Figure 12G

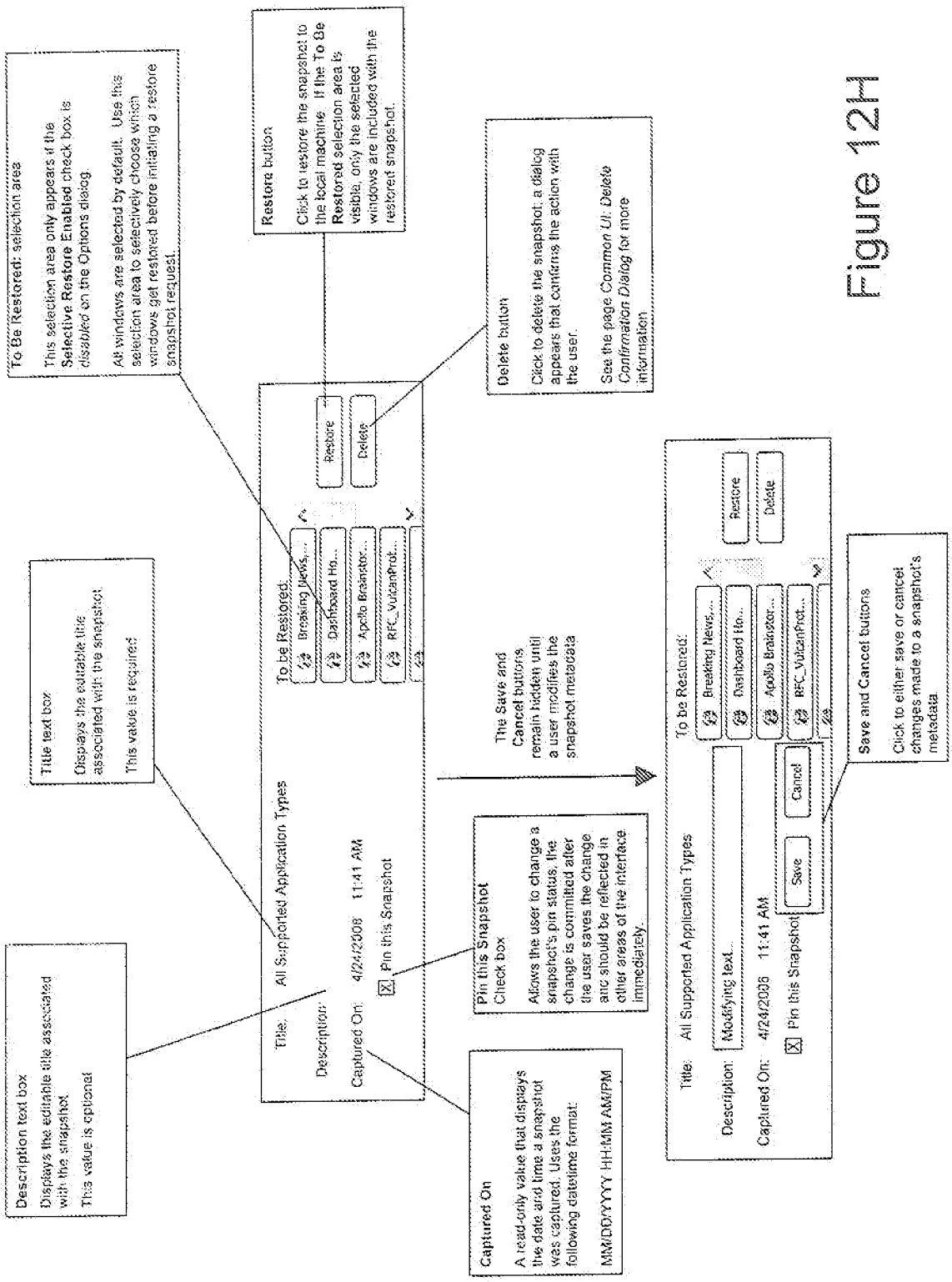

To Be Restored: selection area
This selection area only appears if the Selective Restore Enabled check box is disabled on the Options dialog.
All windows are selected by default. Use this selection area to selectively choose which windows get restored before initiating a restore snapshot request.
Restore button
Click to restore the snapshot to the local machine. If the To Be Restored selection area is visible, only the selected windows are included with the restored snapshot.
Delete button
Click to delete the snapshot; a dialog appears that confirms the action with the user.
See the page Common UI: Delete Confirmation Dialog for more information
Title text box
Displays the editable title associated with the snapshot
This value is required
Description text box
Displays the editable title associated with the snapshot
This value is optional
Title: All Supported Application Types
Description:
Captured On: 4/24/2008 11:41 AM
Pin this Snapshot
To be Restored:
Breaking News...
Dashboard Ho...
Apollo Brainstor...
RFC_VulcanProt...
Restore
Delete
The Save and Cancel buttons remain hidden until a user modifies the snapshot metadata
Pin this Snapshot
Check box
Allows the user to change a snapshot's pin status. The change is committed after the user saves the change and should be reflected in other areas of the interface immediately.
Captured On
A read-only value that displays the date and time a snapshot was captured. Uses the following datetime format:
MM/DD/YYYY HH:MM AM/PM
Title: All Supported Application Types
Description: Modifying text...
Captured On: 4/24/2008 11:41 AM
Pin this Snapshot
Save
Cancel
Save and Cancel buttons
Click to either save or cancel changes made to a snapshot's metadata

Figure 12H

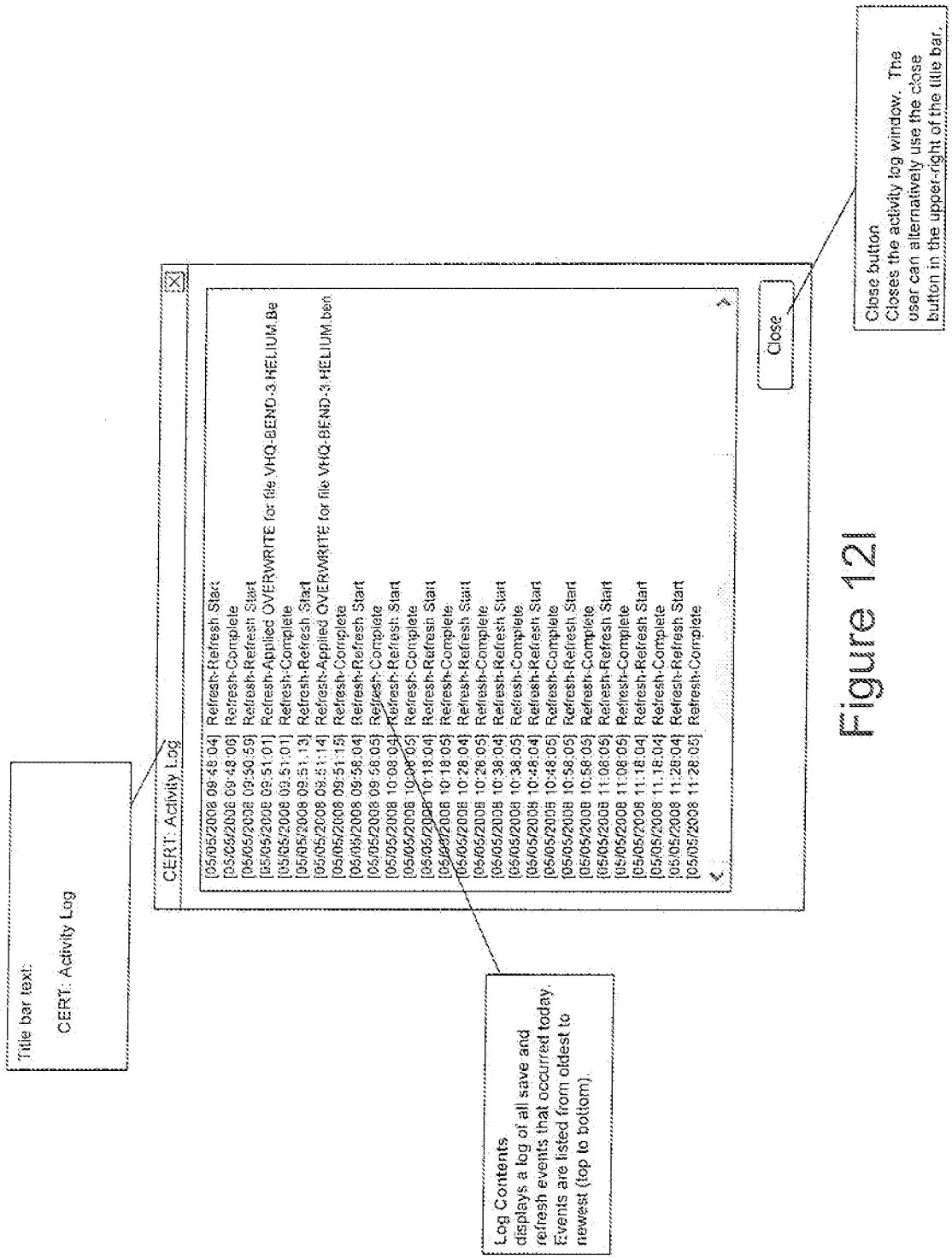

Title bar text:
CERT: Activity Log
CERT: Activity Log
[05/05/2008 09:48:04] Refresh-Refresh Start
[05/05/2008 09:48:06] Refresh-Complete
[05/05/2008 09:50:59] Refresh-Refresh Start
[05/05/2008 09:51:01] Refresh-Applied OVERWRITE for file VHQ-BEND-3.HELIUM.Be
[05/05/2008 09:51:01] Refresh-Complete
[05/05/2008 09:51:13] Refresh-Refresh Start
[05/05/2008 09:51:14] Refresh-Applied OVERWRITE for file VHQ-BEND-3.HELIUM.ben
[05/05/2008 09:51:15] Refresh-Complete
[05/05/2008 09:58:04] Refresh-Refresh Start
[05/05/2008 09:58:05] Refresh-Complete
[05/05/2008 10:08:04] Refresh-Refresh Start
[05/05/2008 10:08:05] Refresh-Complete
[05/05/2008 10:18:04] Refresh-Refresh Start
[05/05/2008 10:18:05] Refresh-Complete
[05/05/2008 10:28:04] Refresh-Refresh Start
[05/05/2008 10:28:05] Refresh-Complete
[05/05/2008 10:38:04] Refresh-Refresh Start
[05/05/2008 10:38:05] Refresh-Complete
[05/05/2008 10:48:04] Refresh-Refresh Start
[05/05/2008 10:48:05] Refresh-Complete
[05/05/2008 10:58:04] Refresh-Refresh Start
[05/05/2008 10:58:05] Refresh-Complete
[05/05/2008 11:08:04] Refresh-Refresh Start
[05/05/2008 11:08:05] Refresh-Complete
[05/05/2008 11:18:04] Refresh-Refresh Start
[05/05/2008 11:18:04] Refresh-Complete
[05/05/2008 11:28:04] Refresh-Refresh Start
[05/05/2008 11:28:05] Refresh-Complete
Close
Close button
Closes the activity log window. The user can alternatively use the close button in the upper-right of the title bar.
Log Contents
displays a log of all save and refresh events that occurred today. Events are listed from oldest to newest (top to bottom).

Figure 12I

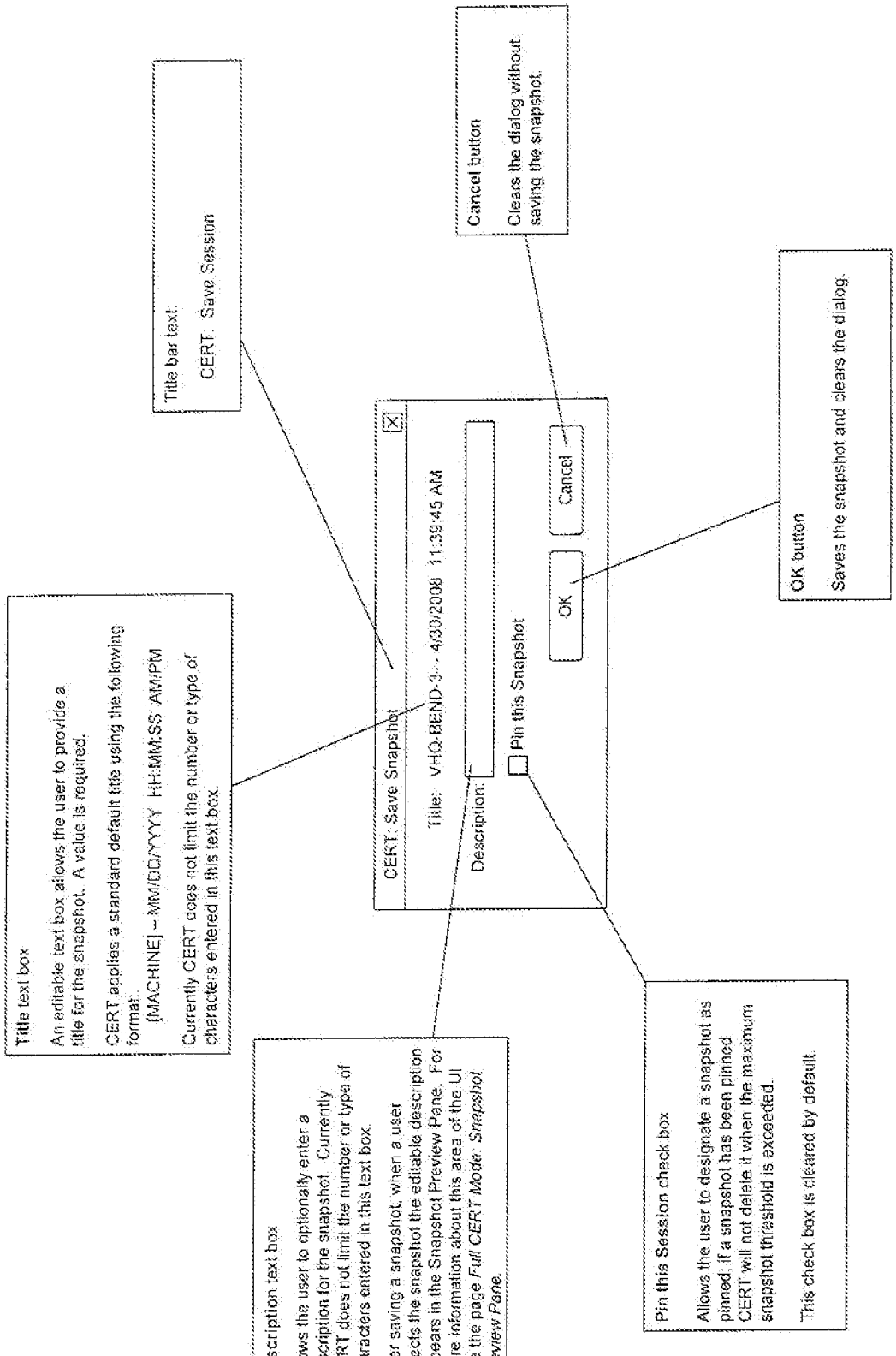

Title text box
An editable text box allows the user to provide a title for the snapshot. A value is required.
CERT applies a standard default title using the following format:
[MACHINE] – MM/DD/YYYY HH:MM:SS AM/PM
Currently CERT does not limit the number or type of characters entered in this text box.
Title bar text
CERT: Save Session
Cancel button
Clears the dialog without saving the snapshot.
CERT: Save Snapshot
Title: VHQ-BEND-3... 4/30/2008 11:39:45 AM
Description:
Pin this Snapshot
OK
Cancel
OK button
Saves the snapshot and clears the dialog.
Description text box
Allows the user to optionally enter a description for the snapshot. Currently CERT does not limit the number or type of characters entered in this text box.
After saving a snapshot, when a user selects the snapshot the editable description appears in the Snapshot Preview Pane. For more information about this area of the UI see the page Full CERT Mode: Snapshot Preview Pane.
Pin this Session check box
Allows the user to designate a snapshot as pinned; if a snapshot has been pinned CERT will not delete it when the maximum snapshot threshold is exceeded.
This check box is cleared by default.

Figure 12J

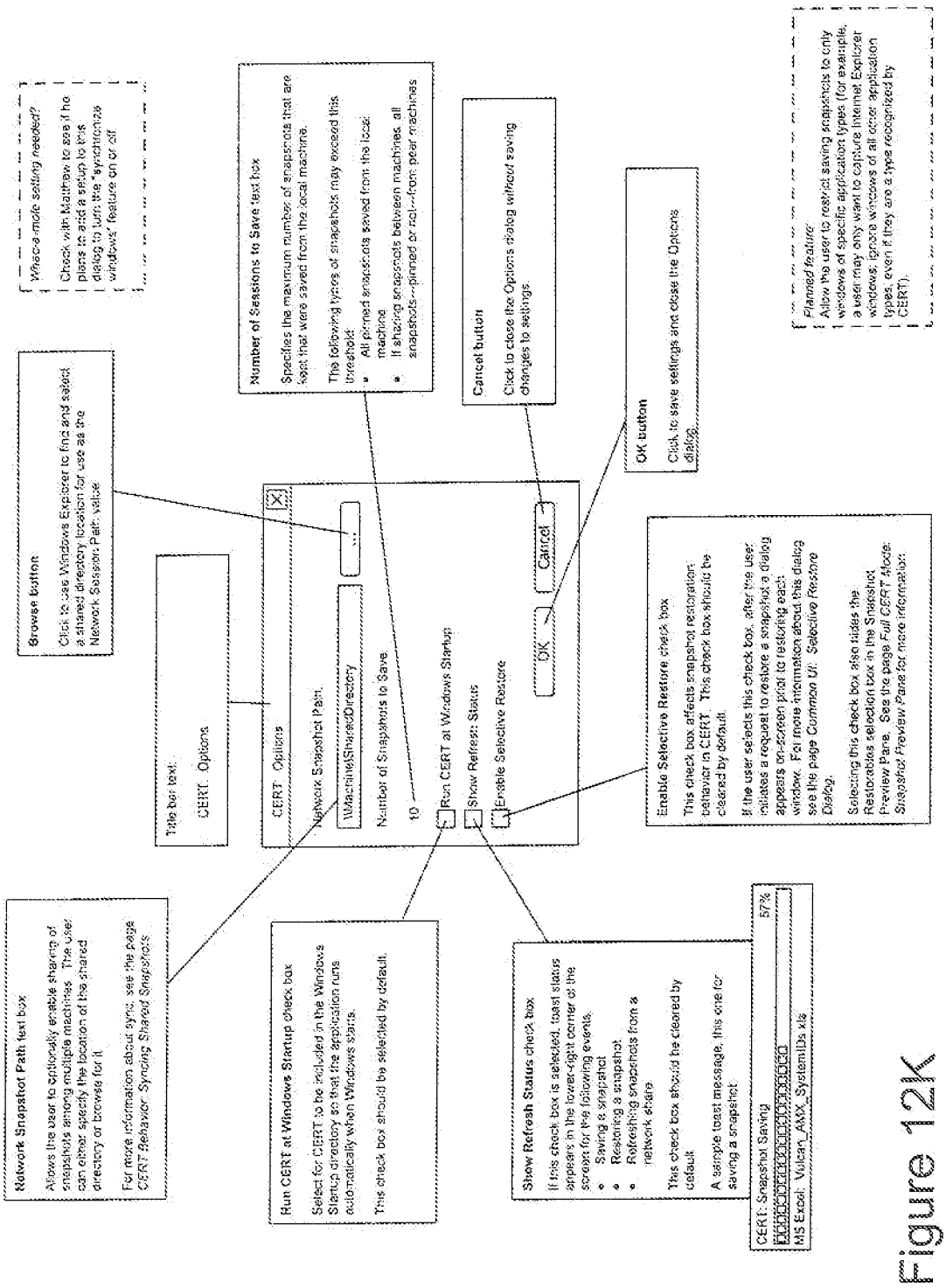

Network Snapshot Path text box
Allows the user to optionally enable sharing of snapshots among multiple machines. The user can either specify the location of the shared directory or browse for it.
For more information about sync, see the page CERT Behavior: Syncing Shared Snapshots.
Run CERT at Windows Startup check box
Select for CERT to be included in the Windows Startup directory so that the application runs automatically when Windows starts.
This check box should be selected by default.
Show Refresh Status check box
If this check box is selected, toast status appears in the lower-right corner of the screen for the following events.
Saving a snapshot
Restoring a snapshot
Refreshing snapshots from a network share
This check box should be cleared by default
A sample toast message, this one for saving a snapshot
CERT: Snapshot Saving
57%
MS Excel: Vulcan_AMX_SystemIds.xls
Title bar text:
CERT: Options
Browse button
Click to use Windows Explorer to find and select a shared directory location for use as the Network Session Path value.
CERT: Options
Network Snapshot Path:
\\Machine\Shared\Directory
Number of Snapshots to Save
10
Run CERT at Windows Startup
Show Refresh Status
Enable Selective Restore
OK
Cancel
Number of Sessions to Save text box
Specifies the maximum number of snapshots that are kept that were saved from the local machine.
The following types of snapshots may exceed this threshold:
All pinned snapshots saved from the local machine
If sharing snapshots between machines, all snapshots—pinned or not—from peer machines
Cancel button
Click to close the Options dialog without saving changes to settings
OK button
Click to save settings and close the Options dialog
Enable Selective Restore check box
This check box affects snapshot restoration behavior in CERT. This check box should be cleared by default.
If the user selects this check box, after the user initiates a request to restore a snapshot a dialog appears on-screen prior to restoring each window. For more information about this dialog see the page Common UI: Selective Restore Dialog.
Selecting this check box also hides the Restorables selection box in the Snapshot Preview Pane. See the page Full CERT Mode: Snapshot Preview Pane for more information.
Where-is-mole setting needed?
Check with Matthew to see if he plans to add a setup to this dialog to turn the "synchronize windows" feature on or off
Planned feature:
Allow the user to restrict saving snapshots to only windows of specific application types (for example, a user may only want to capture Internet Explorer windows; ignore windows of all other application types, even if they are a type recognized by CERT).

Figure 12K

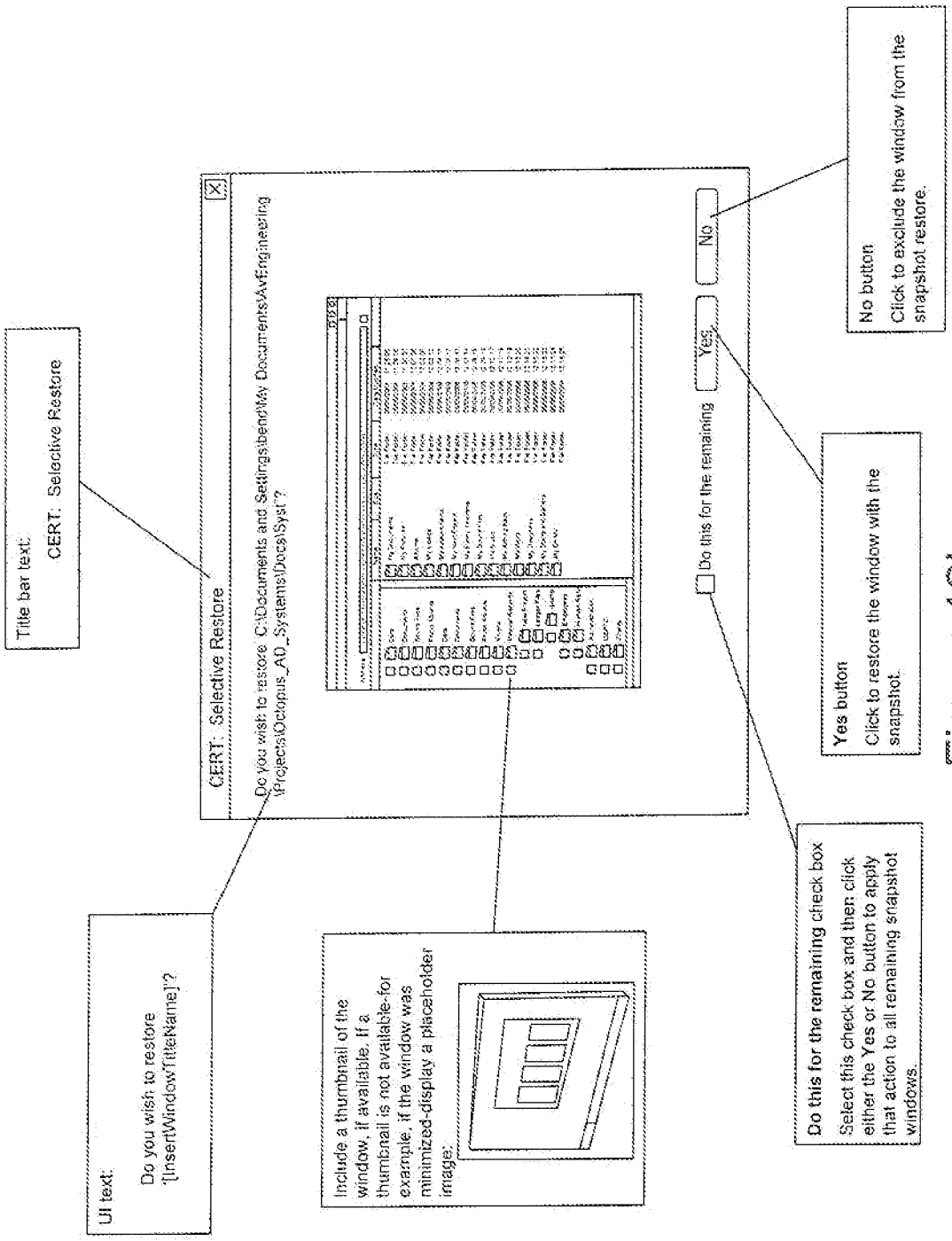

Title bar text:
CERT: Selective Restore
CERT: Selective Restore
Do you wish to restore C:\Documents and Settings\bendWay Documents\Awf\engineering \Projects\Octopus_AD_Systems\Docs\Syst'?
Yes
No
Do this for the remaining
UI text:
Do you wish to restore '[InsertWindowTitleName]'?
Include a thumbnail of the window, if available. If a thumbnail is not available-for example, if the window was minimized-display a placeholder image:
Yes button
Click to restore the window with the snapshot.
No button
Click to exclude the window from the snapshot restore.
Do this for the remaining check box
Select this check box and then click either the Yes or No button to apply that action to all remaining snapshot windows.

Figure 12L

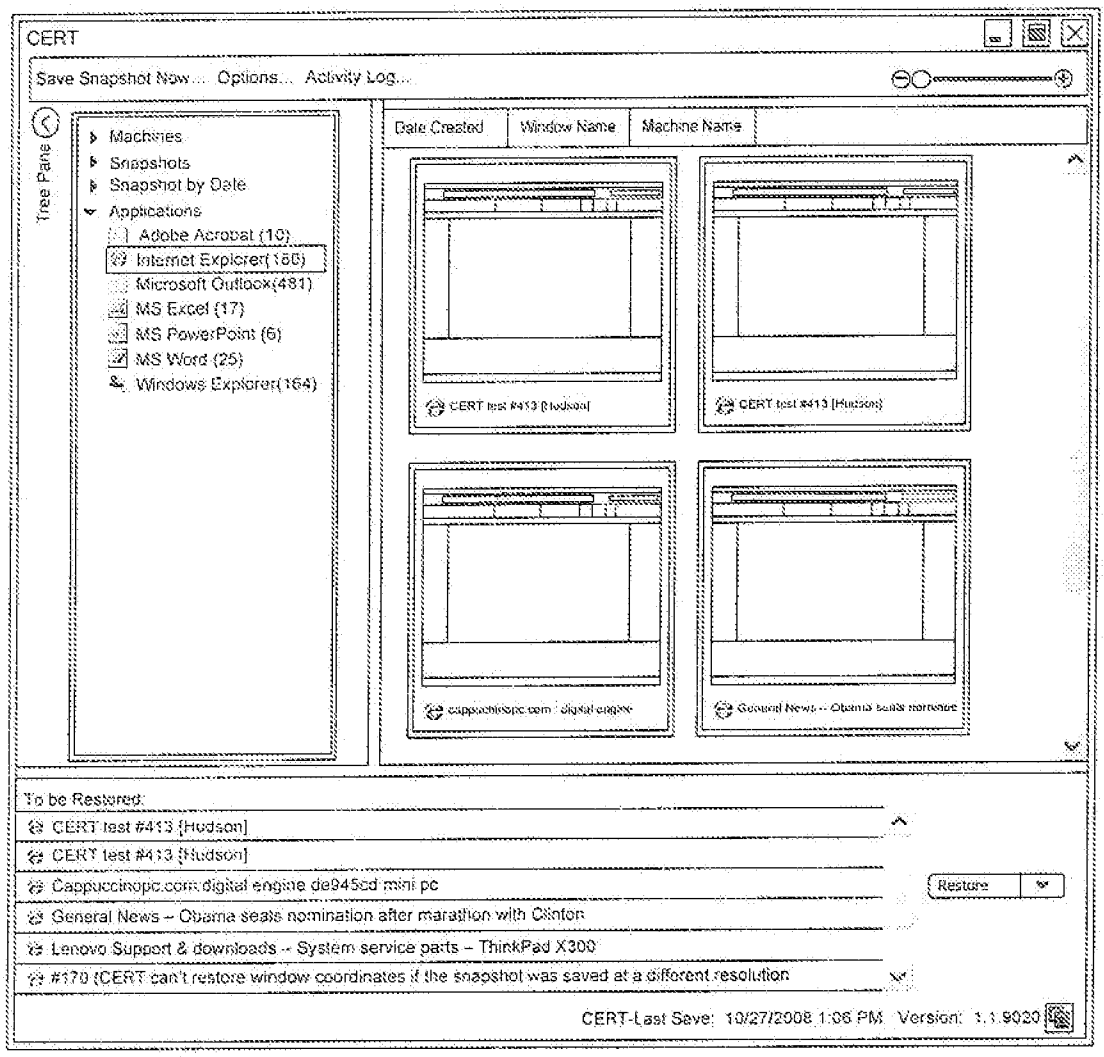

CERT
Save Snapshot Now... Options... Activity Log...
Tree Pane
Machines
Snapshots
Snapshot by Date
Applications
Adobe Acrobat (10)
Internet Explorer(185)
Microsoft Outlook(481)
MS Excel (17)
MS PowerPoint (6)
MS Word (25)
Windows Explorer(164)
Date Created
Window Name
Machine Name
To be Restored:
CERT test #413 [Hudson]
CERT test #413 [Hudson]
Cappuccinopc.com: digital engine de945cd mini pc
General News – Obama seals nomination after marathon with Clinton
Lenovo Support & downloads -- System service parts – ThinkPad X300
#170 (CERT can't restore window coordinates if the snapshot was saved at a different resolution
Restore
CERT-Last Save: 10/27/2008 1:06 PM Version: 1.1.9020

Figure 12M