Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2404973 C 2009/12/22

(11)(21) 2 404 973

(12) BREVET CANADIEN
CANADIAN PATENT

(13) C

(86) Date de dépôt PCT/PCT Filing Date: 2002/01/25
(87) Date publication PCT/PCT Publication Date: 2002/09/06
(45) Date de délivrance/Issue Date: 2009/12/22
(85) Entrée phase nationale/National Entry: 2002/10/01
(86) N° demande PCT/PCT Application No.: US 2002/003047
(87) N° publication PCT/PCT Publication No.: 2002/069139
(30) Priorités/Priorities: 2001/02/23 (US60/270,950); 2001/05/25 (US60/293,854); 2001/11/19 (US09/992,624)

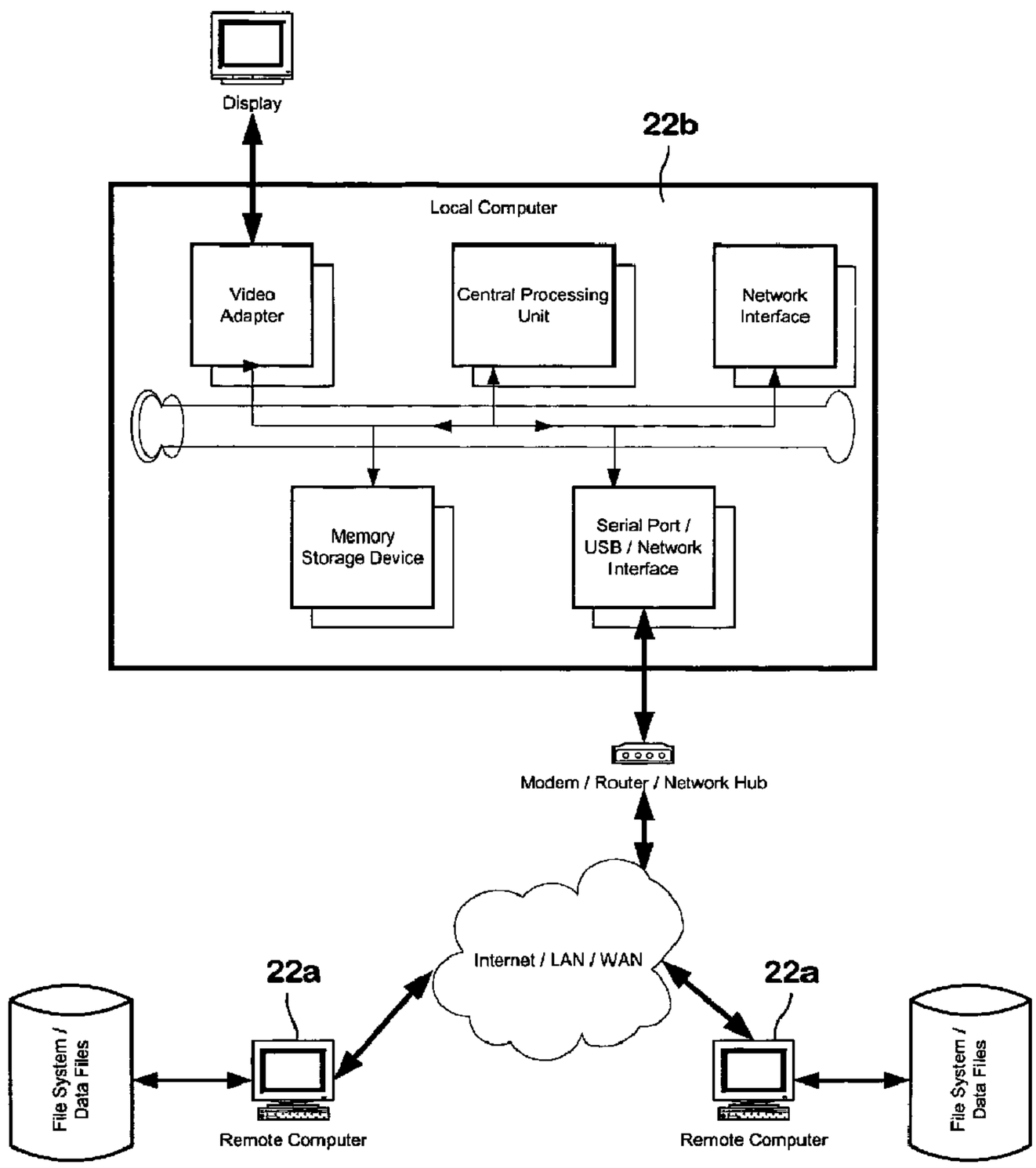
(51) Cl.Int./Int.Cl. *G06F 9/00* (2006.01), *G06F 9/44* (2006.01)

(72) Inventeurs/Inventors:
HIEW, FEN, US;
SCHROEDER, EDWIN M., US

(73) Propriétaire/Owner:
COMPLEMENTSOFT LLC, US

(74) Agent: OGILVY RENAULT LLP/S.E.N.C.R.L.,S.R.L.

(54) Titre : SYSTEME ET PROCEDE PERMETTANT DE CREER ET D'ENTRETENIR UN CODE LOGICIEL
(54) Title: SYSTEM AND METHOD FOR GENERATING AND MAINTAINING SOFTWARE CODE



(57) Abrégé/Abstract:
An Integrated Development Environment having numerous cooperating modules which together provide a system and method for generating and maintaining software, in particular, the software for data development and data manipulation languages. Among

CA 2404973 C 2009/12/22

(11)(21) 2 404 973

(13) C

(57) Abrégé(suite)/Abstract(continued):
other modules, the Integrated Development Environment includes a document manager for use in transparently retrieving related software files. Retrieved software files can be modified using an editor. The editing process may be enhanced through the use of a visualizer that allows the program flow and/or the data flow of the of the software files to be displayed.

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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
6 September 2002 (06.09.2002)

PCT

(10) International Publication Number
WO 02/069139 A2

(51) International Patent Classification[7]: G06F 9/00

(21) International Application Number: PCT/US02/03047

(22) International Filing Date: 25 January 2002 (25.01.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/270,950	23 February 2001 (23.02.2001)	US
60/293,854	25 May 2001 (25.05.2001)	US
09/992,624	19 November 2001 (19.11.2001)	US

(71) Applicant: **COMPLEMENTSOFT LLC** [US/US]; 3930 N. Southport, Unit 3S, Chicago, IL 60613 (US).

(72) Inventors: **HIEW, Fen**; 2525 Arbor Court, Mendota Heights, MN 55120 (US). **SCHROEDER, Edwin, M.**; 3930 N. Southport, Unit 3S, Chicago, IL 60613 (US).

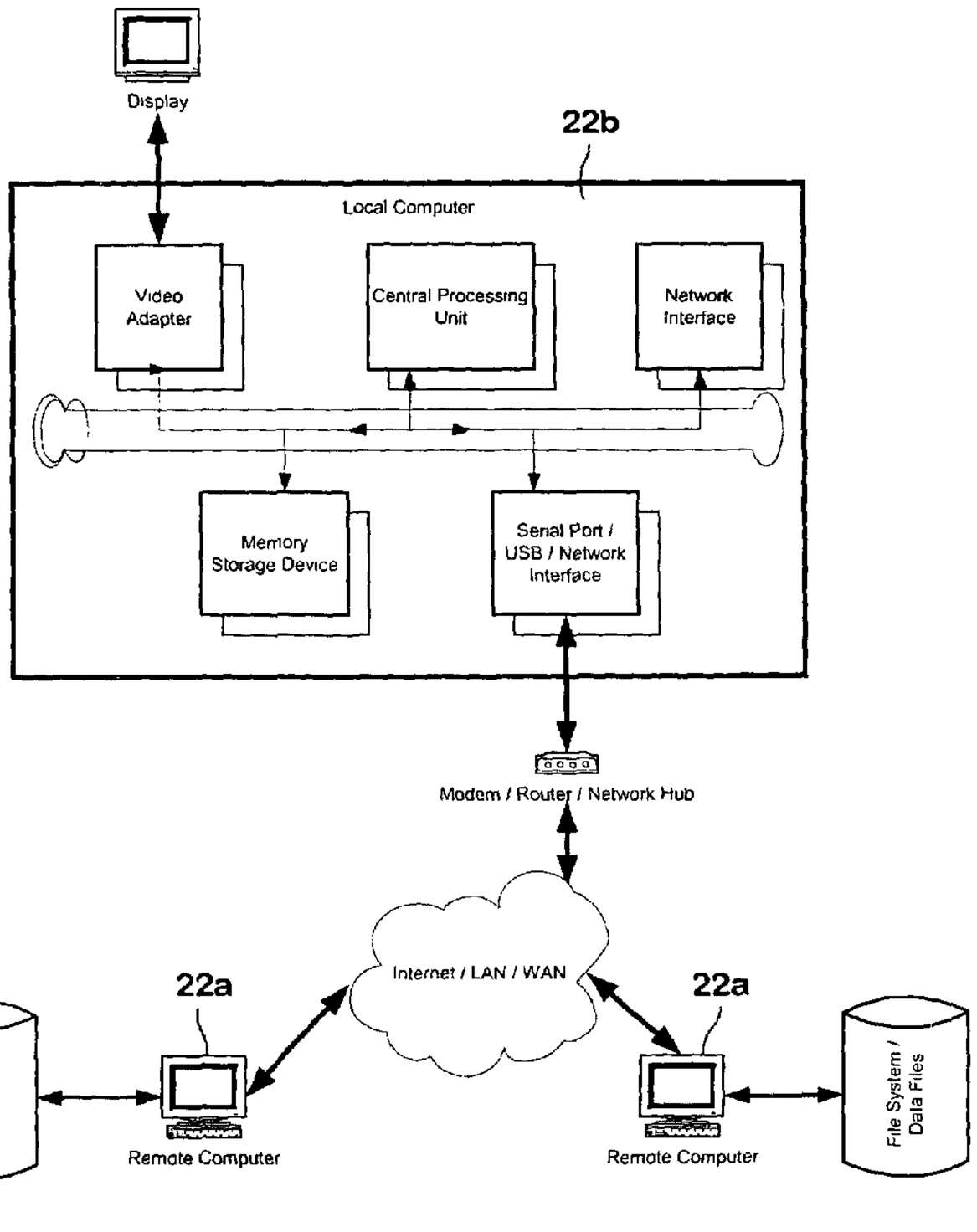
(74) Agents: **JAROSIK, Gary, R.** et al.; Altheimer & Gray, 10 S. Wacker Drive, Suite 4000, Chicago, IL 60606 (US).

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

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM),

[Continued on next page]

(54) Title: SYSTEM AND METHOD FOR GENERATING AND MAINTAINING SOFTWARE CODE



(57) Abstract: An Integrated Development Environment having numerous cooperating modules which together provide a system and method for generating and maintaining software, in particular, the software for data development and data manipulation languages. Among other modules, the Integrated Development Environment includes a document manager for use in transparently retrieving related software files. Retrieved software files can be modified using an editor. The editing process may be enhanced through the use of a visualizer that allows the program flow and/or the data flow of the of the software files to be displayed.

WO 02/069139 A2

WO 02/069139 A2

European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

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

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

SYSTEM AND METHOD FOR GENERATING AND MAINTAINING SOFTWARE CODE

BACKGROUND OF THE INVENTION

This invention relates generally to software maintenance and development tools and, more particularly, to an extensible, language independent software development tool having a graphical user interface, i.e., a GUI Integrated Development Environment.

The evolution of data manipulation and data management systems, such as SAS®, SPSS® and SQL®, and relational database management systems, such as IBM® DB2 UDB® and the Oracle® RDBMS, has resulted in several high-level software languages that are inconsistent and, in some cases, unstructured. Based on these inconsistencies and the unstructured nature of some of these languages, database management languages may be difficult to use, edit and debug. Moreover, because of the lack of a standardized syntax among these programming languages, it has been difficult for users of these languages to share code. This is particularly the case with SAS®.

Since the advent of personal computers and the GUI Interfaces such as Windows® Interfaces, it has become increasingly necessary for businesses to develop or purchase customized software in order to support specific business strategies or processes. This, in turn, has led to the implementation of a variety of software development tools and Integrated Development Environments ("IDEs"). Generally, these software development tools assist users and programmers in editing, debugging and developing software for specific programming languages. Software providers of data manipulation languages or systems have, however, failed to provide a single comprehensive software development tool capable of assisting users in the editing,

visualizing, debugging and development of software. Furthermore, the creation of graphical development tools for particular programming languages, such as, SAS®, has been inhibited by the intricacies of the programming language itself.

Over the years, point solutions have been introduced to address specific issues. For example, there are a few GUI based ftp software packages to make the job of transferring and managing code easier. There are also a few GUI based editor software packages to make editing code easier. Still further, there are software packages that one can use to manually diagram program flow and data flow. What is lacking, however, is a software package that integrates these best of breed point solutions and integrates them in such a way that they work seamlessly with each other, e.g., to have an editor that is integrated with a ftp package so that the editor can edit files that are located in a remote server or to have the editor integrated with the diagramming package so the user does not have to manually generate and update program flow or data flow diagrams.

Thus, a need exists for an Integrated Development Environment for generating and maintaining software code, in particular, for data manipulation centric languages. More specifically, a need exists for a system and method for exchanging, editing, debugging, visualizing and developing SAS®, SPSS®, SQL®, DB2 UDB®, Oracle® RDBMS and other relational database management system software.

SUMMARY OF THE INVENTION

In accordance with these needs, the present invention is embodied in an Integrated Development Environment for generating and maintaining source code (software programmed in a software language), in particular, programmed in data manipulation languages. Generally, the system in which the Integrated Development Environment resides includes a local computer capable of exchanging files with a remote computer via a network system, i.e., a Local Area Network, a Wide Area Network, or the Internet. The local computer preferably hosts the Integrated Developers Environment which is further comprised of a document manager for transferring files and otherwise providing enhanced file management functions, such as, version synchronization across multiple platforms. The document manager works in connection with a server module, a site manager and a connectivity layer which is part of the Integrated Development Environment to connect to remote computers, to

transparently exchange files with the remote computer and to manage server profiles and connection information that is related to remote computers and transferred files.

Once a file is transferred to the local computer, an editor, which is included as part of and integrated with the Integrated Development Environment, can modify the code associated with the file. In addition, the editor is also capable of creating new files and provides many advanced editing features such as visual execution break points, standardized formatting of files, and line numbering to name a few. A visualizer, i.e., a software tools that reads the code and generates diagrams and graphical representation of the program flow, data flow or the logic of the code, is also integrated and included as part of the Integrated Development Environment. Program flow diagrams are comprised of program block icons and arrows to depict the code's program flow. Data flow diagrams are comprised of icons depicting data processing steps and arrows to depict the flow of the data through the program. Preferably, the visualizer and editor are integrated so that changes made to the code in the editor are immediately reflected in the visualizer and vice-versa. The visualizer can also read information from execution logs and execution outputs to display the execution path for selected code and automatically display insightful debugging and optimization information for the selected code to the user.

To assist in developing new code or editing existing code, the Integrated Development Environment further includes a template manager that allows the user to browse through a repository of existing code or templates and to copy templates into the selected code for editing.

For allowing the editor to process code that is written in different Data Manipulation System programming languages and for creating the program flow icons, the Integrated Development Environment additionally includes a parser layer. The parser layer detects the type of code in the selected file and activates the rules and logic that apply to the corresponding Data Manipulation System programming language.

As will become apparent from the detailed description that follows, the subject Integrated Development Environment provides, among others, the following unique functions: seamlessly exchanging with and executing files on local and remote computers, where the site manager is capable of compiling connection information for remote computers necessary to achieve the seamless transfer and execution of files;

automatically generating program flow and data flow diagrams, where the program flow and data flow diagrams can be viewed at various levels of abstraction and where the user is capable of utilizing a step-wise function to collapse or expand the levels of abstraction to view; and automatically parsing the execution log to automatically match errors and warnings in the log file to the appropriate corresponding lines of code in the program file in order to ease the ability to correct the error, to visually highlight problematic areas, and to generate user customizable error messages and debugging advice for such problematic areas.

A better understanding of these and other objects, advantages, features, properties and relationships of the invention will be obtained from the following detailed description and accompanying drawings which set forth an illustrative embodiment and which are indicative of the various ways in which the principles of the invention may be employed.

BRIEF DESCRIPTION OF DRAWINGS

For a better understanding of the invention, reference may be had to a preferred embodiment shown in the following drawings in which:

Figure 1 is a diagram illustrating an exemplary computer network for generating, maintaining and executing computer code;

Figure 2 is a diagram illustrating exemplary components of the local computer;

Figure 3 is an exemplary screen shot depicting a graphical user interface displaying a Menu bar, a Tool bar, a Display area (in this case the Editor) and a Navigation bar;

Figure 4 a – f are exemplary screen shots and corresponding tables depicting icons that are representative of menu items included on the Menu bar, along with the name of each menu item and a functional description of pertinent menu items;

Figure 5 is a diagram illustrating the tool bar and a corresponding table describing the pertinent functions associated with the buttons included on the Tool bar;

Figure 6 is an exemplary screen shot depicting an Editor window in Full Screen mode, along with corresponding output and log file tabs;

Figure 7 is an exemplary screen shot depicting a tree view, along with the corresponding code segments;

Figure 8 is an exemplary screen shot depicting a template manager window with available templates;

Figure 9 is an exemplary screen shot depicting a program flow for a selected file, along with arrows that indicate the flow of data within the program flow;

Figure 10 is an exemplary screen shot depicting a server module window configured for automated login and including session tabs for Server A and Server B;

Figure 11 is an exemplary screen shot depicting a site manager window;

Figure 12 is an exemplary screen shot depicting a document manager window;

Figure 13 is an exemplary screen shot depicting a search panel for locating files;

Figure 14a is an exemplary screen shot depicting an Enhanced Editor Options window;

Figure 14b is an exemplary screen shot depicting a Color tab window for customizing the font colors associated with the code displayed by the Editor;

Figure 14c is an exemplary screen shot depicting a General tab window for configuring the usage of an external editor in-lieu of the built-in editor, enabling and disabling line numbering for the code displayed by the Editor, and selecting print options;

Figure 14d is an exemplary screen shot depicting an Execution Configuration tab for configuring the execution mode and execution location for selected files;

Figure 15 is an exemplary screen shot depicting the template window as it is displaying available web-based templates to the user;

Figure 16 is an exemplary screen shot of a web-based template that is already configured;

Figure 17 is an exemplary screen shot depicting a data flow for a selected file;

Figure 18a-c are a series of exemplary screen shots depicting operation of a step-wise function and various data flows as the data flows are being collapsed;

Figure 19 is an exemplary screen shot depicting the visualizer employing the Split Screen view;

Figure 20 is an exemplary screen shot depicting the visualizer with a problematic code section and a corresponding program flow icon being displayed in red;

Figure 20a is an exemplary screen shot depicting the visualizer with a problematic code section and a corresponding error log for that code section;

Figure 20b is an exemplary screen shot depicting a debugging hint associated with a problematic code section;

Figure 20c is an exemplary screen shot depicting an error message and a corresponding debugging hint being provided by the message manager;

Figure 21 depicts a predefined class structure for recognizing and displaying tokens, which is employed by a file parser to parse a selected file; and

Figure 22 depicts an exemplary file that has been parsed and the corresponding class structure of the parsed file.

DETAILED DESCRIPTION

Turning now to the Figures, wherein like reference numerals refer to like elements, there is illustrated an Integrated Development Environment having numerous cooperating modules which together provide a system and method for generating and maintaining software, in particular, the software for data development and data manipulation languages. Although not required, the system and method will be described in the general context of a computer network 20, illustrated in Fig. 1, and computer executable instructions being executed by general purpose computing devices within the computer network 20. In this regard, the general purpose computing devices may comprise one or more remote computers 22a, and one or more local computers 22b, hosting an integrated software application 30. The computer network 20 can also include one or more databases 24. It should be appreciated that the network components could be described as having client and server relationships, as generally known in the art.

To allow the local computers 22b to generate and maintain code written in various programming languages, the integrated software application 30 will reside on the local computer 22b. Further, as shown in Fig. 2, it is preferable that the integrated software application 30 execute on a Java Virtual Machine ("JVM") which acts as an interface between the integrated software application and the operating system for the

local computer 22b. Although the operating system for the local computer 22b is preferably Windows® based, it should be understood that the local computer 22b could employ any one of the currently existing operating systems, such as LINUX®, UNIX®, MAC OS®, etc.

For editing, generating and maintaining software (i.e., program code), the local computers 22b include a graphical user interface 40. As shown in Fig. 3, the graphical user interface 40 is further comprised of a menu bar 42, a tool bar 44, a display area 46 and a navigation bar 48. Fig. 4a-4e shows exemplary drop-down menu items included on the menu bar, along with a brief description of the functionality associated with those options. Figure 4f shows the "Right-Click" menu. The tool bar 44 is further comprised of and displays several buttons, including a template manager button 44a, which serves as a link to a template manager 100. In addition, Fig. 5 shows exemplary buttons included on the tool bar 44, along with a brief description of the functionality associated with those buttons. The navigation bar 48 is further comprised of and displays a document manager button 48a, a site manager button 48b, an editor button 48c, a visualizer button 48d, a database manager button 48f, and a server button 48e, which serve as links to modules corresponding with these respective buttons, i.e., a document manager 60, a site manager 70, an editor 80, a visualizer 120, a database manager 121, and a server module 160, each of which will be described in greater detail below.

As will be appreciated by those of skill in the art, the computers 22a, 22b need not be limited to personal computers, but may include hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, minicomputers, mainframe computers, personal digital assistants, cellular telephones or the like depending upon their intended end use within the system. For performing the procedures described hereinafter, the computer executable instructions may be written as routines, programs, objects, components, and/or data structures that perform particular tasks. Within the computer network 20, the computer executable instructions may reside on a single computer 22a, 22b or the tasks performed by the computer executable instructions may be distributed among a plurality of the computers 22a, 22b. Therefore, while described in the context of a computer network, it should also be understood that the present invention may be embodied in a

stand-alone, general purpose computing device that need not be connected to a network.

To perform the particular tasks in accordance with the computer executable instructions, the computers 22a, 22b may include, as needed, a video adapter, a processing unit, a system memory, and a system bus that couples the system memory to the processing unit. The video adapter allows the computers 22a, 22b to support a display, such as a cathode ray tube ("CRT"), a liquid crystal display ("LCD"), a flat screen monitor, a touch screen monitor or similar means for displaying textual and graphical data to a user. The display allows a user to view information, such as, code, file directories, error logs, execution logs and graphical user interface tools.

The computers 22a, 22b may further include read only memory (ROM), a hard disk drive for reading from and writing to a hard disk, a magnetic disk drive for reading from and writing to a magnetic disk, and/or an optical disk drive for reading from and writing to a removable optical disk. The hard disk drive, magnetic disk drive, and optical disk drive may be connected to the system bus by a hard disk drive interface, a magnetic disk drive interface, and an optical disk drive interface, respectively. The drives and their associated computer-readable media provide a means of non-volatile storage for the computer executable instructions and any other data structures, program modules, databases, etc. utilized during the operation of the computers 22a, 22b.

To connect the computers 22a, 22b within the computer network 20, the computers 22a, 22b may include a network interface or adapter. When used in a wide area network, such as the Internet, the computers 22a, 22b typically include a modem or similar device. The modem, which may be internal or external, is connected to the system bus via a serial port interface. It will be appreciated that the described network connections are exemplary and that other means of establishing a communications link between the computers 22a, 22b may be used. For example, the system may also include a wireless access interface that receives and transmits information via a wireless communications medium, such as a cellular communications network, a satellite communications network, or another similar type of wireless network. It should also be appreciated that the network interface will be capable of employing TCP/IP, FTP, SFTP, Telnet SSH, HTTP, SHTTP, RSH, REXEC, etc. and other network connectivity protocols.

For seamlessly transferring files, the document manager 60 is utilized. The document manager 60 is a file management program that performs many enhanced file management functions, such as recognizing related files, (e.g. , execution log files, output files, include files, etc.) and managing related files as a unit, regardless of the location of such files. For example, if the user causes the document manager 60 to transfer a file from the remote computer 22a to the local computer 22b, the document manager 60 may also determine whether any related files exist, i.e., files that are or will be used in conjunction with the file that the user is transferring, and transfer those files as well. As will be described in further detail below, these enhanced file management functions, including managing related files and storing information, such as origination and timestamp information about these files, allow the user to exchange files with and execute code on local computers 22b and/or remote computers 22a. To accomplish the enhanced file management functions, the document manager 60 may also include an intelligent module 62, a security layer 66 and a file transfer program 68.

The intelligent module 62 enables the document manager 60 to track pertinent file transfer information 64 that enable the performance of enhanced file management functions, including, uploading files from a source computer and returning/downloading an edited version of the same file to the same source computer for storage, execution, etc. To accomplish the functions of returning edited files to the source computer and executing edited files remotely, the file transfer information 64 may include a source computer identifier, a file directory identifier, a content identifier, a timestamp or other information capable of allowing the tracking of the type or path of files that are transferred between computers 22a, 22b or within a single computer 22b.

To ensure that the integrity and security of the network 20 is maintained, a security layer 66 is utilized. The security layer 66, which is part of the site manager 70, may save server login information supplied by the site manager 70 thereby allowing subsequent file exchanges to be handled transparently. The security layer 66 may also be capable of configuring the integrated software application 30 to work with firewalls. The site manager 70 gathers server login information and similar information for use by the document manager 60, the intelligent module 62 and the security layer 66, and interacts closely with each of these modules. Preferably,

remote file access and file execution are handled centrally through the site manager 70.

To maintain existing software and develop new software, the editor 80 allows the user to perform standard text editing functions, including, mouse placement of the cursor, click-and-drag text selection and standard Windows® key combinations for cutting, copying and pasting data. While editing code that has been previously executed, the associated log files and output files that are stored in the same directory 104 as the code will automatically be opened and each of these files will include a corresponding tab 84a, 84b, as shown in Fig. 6. The user can access each of these files by clicking on the corresponding tab 84a or 84b and switch between several related files by clicking on different tabs. The editor 80, the document manager 60 and the site manager 70 cooperate to track the association between edited code, the log files and output files that are generated by the edited code, and the remote system on which edited code may be executed. The editor 80 may also employ language-specific syntax checking and auto-correct functions to enhance the software development capability of the integrated software application 30.

To make the code more manageable and allow the user to see a more abstract version of the code, the editor 80 also provides a means to invoke a tree view 90, which may be displayed near the editor 80. The tree view 90 depicts individual procedures and data blocks as active elements 82 and shows the code at a high level. Since each active element 82 is representative of a larger code segment, as shown in Fig. 7, the user can navigate throughout the code by selecting active elements 82 and thereby displaying the larger code segment associated with the selected active element 82 (i.e., the editor will automatically scroll to the region where the code segment is visible). To make it easier to cut, copy or move code segment, the user can right-click on an active elements in the tree. This will cause the editor to automatically scroll to the region where the code segment is located and highlight the entire code segment. The user can then cut or copy the highlighted code segment to the clipboard.

For enabling users to add pre-existing lines of code or templates 106 to a program being edited or created, the editor 80 also includes the template manager 110, which may be accessible to the user by clicking on the Template button on the tool bar 44 menu. As shown in Figure 8, the template manager window 102 includes a directory 104 of templates 106, which are both pre-built and written by the user and

which may be examined by navigating the directory 104 in a manner that is well-known in the art. The template manager 110 allows the user to insert templates 106 directly into the code that is being created/edited. In addition, the template manager 110 also allows the user to create and organize their own templates 106. Still further, the template manager 110 could be used in connection with the Internet and a web browser to view and retrieve templates stored at remote locations, such as web sites, bulletin boards, etc. The Template Manager 100 can store the templates (both built-in and those developed by the user) locally and/or centrally on a remote server making it easy for the entire workgroup or enterprise to share these templates.

In addition, the editor 80 may also provide an auto-complete function capable of automatically generating code based on templates 106 managed by the template manager 100. For example, as soon as a user keys in a recognized key word, the editor 80 will automatically perform a look-up function for that keyword in the directory 104. If a template 106 includes a matching keyword, then the editor 80 may automatically paste the template into the display area. If more than one template 106 matches the keyword, then a pop-up window will be displayed presenting the user with the option of selecting one of the matching templates. It should also be appreciated that the user customize the integrated software application 30 by enabling or disabling the auto-complete feature.

For viewing the program flow and data flow of a selected program, as will be further described below, the visualizer 120, in connection with a parser layer 140, reads, parses and displays the code for the selected program, representing each program and data block with a program flow icon 126. As illustrated in Fig. 9, arrows connect these program flow icons 126 to generally illustrate the flow of data.

Assuming that the designated code segment has been executed previously and that the execution log is readily available, the user may also "mouse-over" the program flow icon to display the comments and execution statistics, i.e., CPU usage, number of row processed, etc., associated with the block. Right-clicking on the program flow icon will activate specific functions or options associated with the icon. For example, the dataset icon (i.e., an icon that represents a dataset or a table) will display a list of data inspection or data discovery functions, including ad hoc SQL® queries. Ad hoc SQL® queries that are selected may be executed by the respective engine. These engines (e.g., SAS, DB2 UDB, or other RDBMS servers) can resides

locally on the user's workstation or remotely on the LAN/WAN and the results, once retrieved will be displayed in a pop-up output window.

To access and interact with remote computers 22a via command lines, the server module 160 acts as a robust terminal emulator. The server module 160 allows users to open one or more sessions thereby simultaneously gaining access to one or more hosts/remote computers 22a. In addition, the user has the option of executing edited code on a remote computer 22a, by employing the server module 160, in connection with the site manager 70, to connect the local computer 22b to the remote computer 22a, as will be described in more detail below. Once a session is opened, a terminal tab for the respective session can be created and displayed to the user by the server module 160, as illustrated in Fig. 10. It should be understood by those with skill in the art that the server module 160, the site manager 70 and the document manager 60 all preferably interact with one another to effectuate the transfer of code between the remote server computers 22a and the local computer 22b. It should also be understood that each of these modules could be combined or further divided to form one single module or additional modules.

The site manager 70 assists the local computer 22b with access to the remote computers 22a. For example, the document manager 60 and the server module 160 use the site manager 70 to collect the connection information 78 necessary for the local computer 22b to make a LAN/WAN connection to a remote computer 22a using either FTP, SFTP, Telnet SSH, HTTP, SHTTP, RSH, REXEC and other TCP/IP connectivity protocols. Additionally, to avoid manually entering the connection information 78 each time a different remote computer 22a is accessed, the site manager 70 is also capable of saving connection information 78 for different servers. Thus, the site manager 70 creates a virtual computing environment by expanding the computing boundary of the local computer 22b to include remote computers 22a and making various computing resources across the LAN/WAN seamlessly available for use by the local computer 22b. As shown in Fig. 11, the site manager window 72 may be divided into a first panel 74 and a second panel 76. The first panel 74 displays a tree structure that depicts the system configuration 74a for each of the remote computers 22a that are available to the local computer 22b. The second panel 76 displays the connection information 78 for a selected remote computer 22a, including

but not limited to the following fields: Profile Name 78a, Host Address 78b, User 78c, Password 78d, FTP port 78e and Telnet Port 78f .

The site manager 70 also includes a PASV mode and Firewall option which can be selected by the user. PASV mode is supported as an option to allow file transfer programs ("FTPs") to work with a firewall. This feature reverses the connection between the remote computer 22a and the local computer 22b allowing many users whose sessions are hosted behind firewalls to use the document manager 60. Therefore, PASV mode is useful for certain types of firewalls that do not allow FTP servers to initiate data connections back to the connected client. If the Firewall option is selected, the firewall configuration parameters from the Firewall tab of the Option Dialog will be used to initiate a connection to this profile. Moreover, if the local computer 22b resides behind a firewall that limits or restricts FTP access and the Firewall mode is selected, the necessary connection information 78 and Firewall selections for the local computer 22b must be entered, i.e., a host name, port number, user ID and password. In accordance with procedures that are generally known in the art, the Firewall Type box permits the user to select the command required by the firewall to initiate an FTP session from the site manager 70. In addition to providing options that enhance the network connectivity of the local computer 22b, the Option button also provides tools and wizards for automatically searching the LAN/WAN for designated resources. These tools and wizards simplify the network/server connection setup between the local computer 22b and the remote computers 22a.

For parsing the code, the parser layer 140 is provided. The file parser 142 retrieves code utilizing the services provided by the document manager 60 and examines the words/tokens which comprise the code 144. The file parser 142 also identifies and tags the tokens 144 to indicate that they are one of a variety of token classes, such as, a keyword token, a newline token, a quotestring token, a macro token and a comment token. Since the integrated software application 30 is language aware, the parser layer 140 may accommodate varying language types. Thus, although the preferred embodiment of the present invention is specifically designed to include modules for parsing SAS® and SQL code, it should be appreciated that the modules can be included for parsing and interpreting other language types, such as, SPSS®, DB2 UDB® Store Procedures, ORACLE® PL/SQL, etc.

For managing and interacting with database files, a database manager 220 may be provided which automatically interacts with the local ODBC registry to display a list of ODBC resources. In addition, the database manager 220 may automatically search the LAN/WAN for RDBMS servers and display the list of RDBMS servers that are found locally or remotely. The database manager also allows the user to select and drill down to a desired database and automatically connect to the selected database, unless a login screen is required to connect to the database. Once connected to the selected database, the Database Manager will display a list of tables which are stored and available in the database. The user can then right-click on a particular table to gather statistics or to retrieve sample data from those tables. The user can also drill down on a particular table to display the columns of the tables. Similarly, the user can click on a particular table to gather statistics (e.g., frequency counts, min max, distinct value, average, etc.) or to retrieve sample data from those tables. The user can also drag and drop tables onto a work area to graphically generate SQL® statements to retrieve data from the database. Additional data manipulation functions, e.g., pivot, crosstab, and templates will be available to help users inspect, transform, import, export, map, format, transport and derive new data based on existing data. The database manager 220 may also include a meta data manager 222 to help the user document the business or logical definition of the data, and the technical or physical definition of the data. The meta data manager 222 may also assist the user in tracking the lineage of the data, i.e., the source of the data, the changes to the data that were made and the destination of the data.

To further exemplify the functionality of the integrated software application 30, the following section will discuss an exemplary software development session, along with the modules used to effectuate the respective functions.

This hypothetical session commences with activation of the integrate software application 30. Once the integrated software application 30 is launched, the graphical user interface 40 will be displayed and the user will have the opportunity to open modules in the navigation bar 48 or engage functions represented by buttons in the menu bar 42 or the tool bar 44. For determining to which computer the user desires access, the user would begin by activating the site manager 70, which causes the site manager window 72 to be displayed to the user. The site manager window 72 includes a first panel 74 and a second panel 76. As shown in Fig. 11, the first panel

74 of the site manager window 72 displays the defined/available remote computers 22a for the user's selection. As discussed above, the second panel 76 of the site manager window 72 displays a server profile for a selected remote computer 22a, comprising connection information 78, including, but not limited to, the following fields: Profile Name 78a, Host Address 78b, User, Password, FTP port 78e and Telnet Port 78f . As will be appreciated, the site manager can be configured to search the network for servers or certain types of servers and display the found servers for selection by the user.

The server profiles 170 may be populated by selecting a folder icon 172 to use as a parent directory 174. Once a parent directory 174 is opened, the user may click on the New button 176 to create a new server profile 170. Therefore, it should also be understood that the site manager 70 allows the user to create new server profiles 170 by entering the necessary connection information 78. To edit an existing server profile 170, the user must select a remote computer 22a and click the Edit button 178. After the Edit button 178 is selected, the cursor may be moved to the Profile Name 78a for editing or modification, or the user may click on the Option button for Advance Network/Server Connectivity Configuration Settings.

The text displayed in the Profile Name 78a field can be a profile name 78a that was selected in the first panel 74 of the site manager window 72 or text the user enters in creating a profile for a remote computer 22a or host. The Host Address 78b field may contain an IP address or a resolvable DNS host name of the FTP, SFTP, Telnet, SSH, REXEC, HTTP, or SHTTP server for the respective host definition. The User 78c field may contain the login name that the user will enter to access the remote server computer FTP account. If the selected remote computer 22a accepts anonymous FTP requests, "anonymous" may be entered in this field. The Password 78d field may contain a password 78d for the remote server computer FTP account, unless an anonymous login is used; then, the Password 78d field may remain blank. The FTP port 78e field is set by default to 21, since most server computers 22a accept FTP connection requests on port 21, however, the FTP Port 78e field may be changed as needed. The Telnet Port 78f field is set by default to 23, since most server computers 22a accept Telnet connection requests on port 23, however, the Telnet Port 78f field may also be changed as needed.

Once the user selects a remote computer 22a to access, the connection information 78 for the selected remote computer 22a is returned by the site manager 70 and displayed to the user. Then, the user is given the option of connecting to the selected remote computer 22a by selecting the Connect Telnet button. When the Connect Telnet button is selected, the server module 160 is activated. To connect to the remote computer 22a, the server module 160 utilizes the connectivity layer 180. More specifically, as is known in the art, the connectivity layer 180 utilizes the connection information 78 with TCP/IP and other similar networking protocols to interconnect with the remote computer 22a. Once the server module 160 has made an initial connection with the selected remote computer 22a, a server module window 162 is displayed to the user. The server module window 162 displays the name of the remote computer 22a to which the user is trying to connect.

Prior to opening a session on a particular host, the user may be required to login or the server module 160 may automatically log the user on (depending on the settings that the user has selected in the server profile 170, i.e., save login and password). Fig. 10 shows a server module window 162 that is configured for automated login, with a home path preset to the integrated software application's SAS® directory. The combination of the site manager 70, the server module 160 and the connectivity layer 170, enable the user to seamlessly access SAS® documents, log files and output files stored at the remote computer 22b.

Once the local computer 22b accesses a remote computer 22a using the combination of the site manager 70, the server module 160 and the connectivity layer 170, the document manager 60 will be activated automatically. Moreover, to view the available files for a selected computer, the document manager window 170 may be displayed to the user. As shown in Fig. 12, the document manager window 170 includes a local panel 172 and a remote panel 174. Each of these panels, 172, 174 displays the available files for the respective computers 22b, 22a as a file tree structure, as shown in Fig. 12. The user may then select one or more of the available files for editing or visualizing. In the preferred embodiment of the present invention, files that are located in the local panel 172 are resident on the local computer 22b and files that are located in the remote panel 174 are resident on the remote computer 22a, but it should be appreciated that such designations are for explanatory purposes only

and that other variations of network systems and computer designations could be employed.

The document manager 60 also allows the local computer 22b to perform various Windows® commands, such as, creating new files, renaming, deleting and opening existing files, printing files, copying, cutting and pasting information to the Windows® clipboard, and other standard commands that are generally known in the art. In addition, the document manager 60 provides a Search Option 69. As shown in Fig. 13, selecting the Search Option 69 opens the search option window 69a and allows users to search any file system that is accessible to the local workstation/local computer 22b, in accordance with generally known file searching techniques, including by exact or partial file name, modification date or date range, and content.

To open a file displayed in the local panel 172, the user may double-click on the selected file, or highlight the selected file and start the desired module (i.e., the editor 80 or the visualizer 120). If the user double-clicks on a file located in the local panel 172, the editor 80 will automatically open and display the file. To open and edit a file displayed in the remote panel 174, it is preferable that the user drag a selected file from the remote panel 174 of the document manager 60 and drop the selected file in the local panel 172 of the document manager 60. It should also be understood by those with skill in the art that other methods for opening and moving files may be employed.

To retrieve a file that is stored on a remote computer 22a, the document manager 60 may utilize a file transfer program 64, in connection with the site manager 70 and the connectivity layer 180. For example, as is known in the art, the document manager 60 may send a request to the file transfer program 64 to import the selected file from a particular remote computer 22a. In response to this request, the file transfer program 64 may communicate with the remote computer 22a via the connectivity layer 180 and instruct the remote computer 22a to send the selected file to the local computer 22b. When the selected file is being transferred from the remote computer 22a to the local computer 22b, the intelligent module 62 of the document manager 60 may compile file transfer information 64, as noted above. It should also be understood that, while it is preferred to have the intelligent module 62 compile file transfer information 64, other similar modules may be equally capable of accomplishing the task of compiling file transfer information 64 for the system 20.

Again, for exemplary purposes only, it is assumed that the user has chosen to edit the code before visualizing it. As mentioned above, to begin editing files, the user may either double-click on the selected file, which automatically opens the editor 80, or the user may highlight the selected file and then open the editor 80. To open the editor 80, the user need only double-click on the Editor button 48c located in the navigation bar 48 of the graphical user interface 40, as shown in Fig. 3.

As shown in Fig. 6, once the editor 80 begins the editing process (the default setting is Full Screen mode), the contents of a selected file are returned from the document manager 60 and displayed to the user beside a navigation bar 48 bar. For editing the code and implementing advanced code-handling capabilities, the editor 80 also includes features, such as, language-aware syntax highlighting, warning and error log file highlighting, automatic line numbering, automatic completion of code segment based on program templates, tree views of program blocks and code templates 106 . By highlighting key code statements, the language-aware syntax highlighting makes the code easier to read and edit. In addition, by highlighting the code, based on information contained within the warning and error log files, errors in the code are more apparent and the code is easier to trouble-shoot.

To allow the user to customize, edit and maintain error messages and the corresponding debugging hints, the message manager 88 is provided. More specifically, the message manager 88 allows the user to create, update and delete error messages. For example, the user can click on any error message in the log and in response to this selection, the message manager 88 may compare the selected error message to an existing repository of error messages The repository of error messages can be stored locally and/or centrally in a remote server accessible and sharable by the entire workgroup or the entire enterprise. If the selected error message matches one of the existing error messages, a debugging window will pop-up. The debugging window will display the original error message, along with any corresponding debugging hints that are associated with the original error message as provided by the message manager 88. If the selected error message does not match one of the existing error messages, the message manager 88 may automatically read the new message into the repository of error messages. Additionally, the user may have the option to associate debugging hints with the new error message.

To further add to the utility of the editor 80, the user can open the Enhanced Editor Options window 84 by clicking the Editor Configuration button 86 in the Options pull-down menu. The Enhanced Editor Options window 84 is further comprised of a General tab 84a, a Font tab 84b, a Color tab 84c and an Execution tab 84d. As is generally understood in the art and as shown in Fig. 14a, the font type, size and style of the code may also be customized by the user by selecting the Font tab 84b in the Enhanced Editor Option window 84. Additionally, as shown in Fig. 14b, the Color tab 84c of the Enhanced Editor Options window 84 may be selected to allow the user to select the colors that are used in connection with various syntactically significant language elements in the SAS® program file, thereby allowing the user to customize the language-aware syntax highlighting and warning and error log file highlighting functions. To further configure the user workspace, the editor 80 allows the user to choose whether or not to display line numbers in connection with editing the code. For example, as shown in Fig. 14c, the General tab 84a can be selected by the user and may provide options for selecting a line numbering button representative of disabling or enabling the line numbering, specifying printing options or indicating the use of alternate editor. For allowing the integrated software application 30 to configure the execution path for selected files, the Execution tab 84d can be selected by the user and the execution mode and execution location may be defined (i.e., Locally or Remotely), as illustrated in Fig. 14d. By configuring the execution path for selected files, the user can execute the code on local or remote computers 22b, 22a or engines.

Once the selected code is displayed on the editor window 82, the user can begin to edit the code. Although the default setting for the editor window 82 is the Full Screen mode, the user can also select a Split Screen mode. The Split Screen mode splits the editor window 82 into two panels, i.e., the tree view panel 82a and the code view panel 82b. To enhance the ability of the user to navigate within the body of the code, the tree view panel 82a displays the tree view 90, which is comprised of a group of active elements 92. The tree view panel 82a displays a high-level representation of the code and the code view panel 82b displays the corresponding code that is associated with a highlighted active element 92. To this end, the editor 80 works in connection with the parser layer 140 to parse the code in a manner that allows the editor 80 to display a tree view 90 comprised of active elements 92 that are

representative of code section that perform important functions, such as, the introduction of new variables or the execution of append, print or similar functions. Thus, the user can navigate through the code by clicking on the desired active element 92, which causes the code that corresponds with that active element to be displayed in the code view panel 82b.

The user can also add pre-existing code to the program being edited, by opening the template manager 100. The template manager 100 allows the user to select existing templates 106 and to integrate the code associated with those templates 106 into the code by simply clicking on the paste button or by utilizing standard cut and paste tools, or similar functions, as will be explained in greater detail below.

Once the template button 44a is selected, the template manager 100 is activated and the template manager window 101 is opened. It should be appreciated that the template manager 100 may be a separate browser; therefore, the template manager window 101 may remain open throughout the users session. The template manager window 101 may be further comprised of a template directory panel 101a and a template code panel 101b. The template directory panel 101a may display a directory 104 of template folders 105 or available templates 106. To display the code associated with an available template 106, the user may select a file folder containing available templates 106 from the directory 104 and select an available template 106, which is located within the selected file folder. Once a template 106 is selected, the template code 106a associated with the template 106 is returned to the template manager 100 and displayed in the template code panel 101b.

To insert the template code 106a associated with a selected template 106 directly into the code that is being edited, the user may click the Copy button 102 on the template manager window 101; this places the template code 106a associated with the selected template 106 into the Windows® clipboard. Then, the user should place the cursor in the desired location within the code and select the Paste button 103. Or the user may simply click the paste button and the template will be copied directly into the current location of the cursor in the document.

The template manager 100 may also allow the user to create new template folders 105 for storing templates 106, to create new templates 106 and to edit existing templates 106. These templates can reside on either the local drive or the network drive. In addition, for allowing the template manager 100 to import additional

templates 106 from remote computers 22a, the template manager 100 also enables the user to browse web-based templates 108 . As shown in Fig. 15, once the web-based templates 108 are accessed and displayed on the template manager window 101, the web-based templates 108 may be utilized in a manner similar to the templates 106. As shown in Fig. 16, additional web-based templates 108 may be added to the template manager 100 by the user. It should also be understood that the templates 106 are web enabled, i.e., the user can associate links or URLs with the template during creation. Thus, users may be capable of clicking on a link and causing the template manager 100 to automatically activate a web browser to display any content associated with the link.

Once the user has completed the editing or development of the code, the user can execute and debug the code. In addition, the code can be executed locally or remotely. As is known in the art, the user may also use the editor 80 to set break points 112 in the code. As shown in Fig. 23, break points 112 can be set by selecting the line numbers that correspond to the intended location of the break points 112. It should also be appreciated that by setting break points 112 in the code, the code can be executed in its entirety or as a block of code with the additional option of skipping specified code segments within the selected blocks.

The site manager 70, the document manager 60 and the editor 80 work in conjunction with one another to execute the code. For example, the site manager 70 manages the connections between the local computer 22b and the remote host/remote computer 22a; the document manager 60 tracks the associations between the code, the log files and the output files that are generated and the remote system on which the code is to be executed; and the editor 80 allows the user to modify the code. Moreover, for assisting users in developing and maintaining software, the editor 80 and the visualizer 120 have an integrated relationship. For example, each of the active elements 92 displayed in the tree view 90 may also be represented in the visualizer 120 and any changes made in the editor 80 can be simultaneously reflected in the visualizer 120.

To execute code remotely when the code resides on a remote computer 22a, the code is first copied to a directory on the local computer 22b. Next, the user should open the local copy of the code. After the local copy is opened and returned to the editor 80, the code view panel 82b will display the code. The user can also use the

document property wizard to specify the server and the location in which they intend to execute the code. The server or location may be selected by browsing the server list in the site manager 70.

To execute code remotely when the code resides on a local workstation/local computer 22b, the user first instructs the site manager 70 to make a connection to the remote computer 22a. Then, the user may cause the document manager 60 to copy the code from the local computer 22b to a directory 104 on the remote computer 22a. To select code for execution, the user can drag an icon representing the selected code from the local panel 172 of the document manager 60 window and drop the icon in the remote panel 174 of the document manager 60 window. It should also be appreciated that the selected code could be copied by other means that are generally known in the art. Next, the user should open the local copy of the code. After the local copy is opened, the editor window 82 will display the code.

To execute the code or code segments, the user selects the Execute Program button 44a on the tool bar 44 or in the Right-Click menu. Once the Execute Program button 44a is selected, the server module 160 connects to the proper computer 22a, 22b using the connectivity layer 180 and causes the execution script, which may be defined in the site manager 70, to operate on the selected code. After execution of the code is complete, a system prompt will be displayed. The user may then type "exit" to return to the editor 80 window; the document manager 60 automatically transfers the session's .log an .lst files to the same directory 104 from which the code was copied and displays them as separate tabs on the Editor 80 window. The user can also configure the remote execution function to that it will exit automatically.

As shown in Figs. 9 and 17, the visualizer 120 may be used to show the program flow 122 or the data flow 124 of the selected code on the visualizer window 121. Additionally, the user will be able to toggle between the program flow display 122a and the data flow display 124a by selecting the View Program Flow button 122b or View Data Flow button 124b, respectively. As shown in Fig. 9, the program flow 122 is displayed as a default and displays program flow icons 126, which are graphical representations of code sections, in the order that they occur in the code.

For generating the program flow icons 126, a document view engine 200 is provided. The document view engine 200 operates in conjunction with the parser layer 140 to parse the code. Using information provided by the parser layer 140, the

document view engine can intelligently recognize and arrange individual procedures and data blocks on the visualizer window 121 and represent the procedures and data blocks as program flow icons 126. The document view engine 200 may also allow users to assign meaning and attributes to tokens 144, which are identified by the parser layer 140. By assigning meanings and attributes to tokens 144, the document view engine 200 allows the visualizer to create program flows 122 and data flows 124.

As shown in Fig. 17, the visualizer 120 may also show the data flow 124 of the subject code. The data flow 124 is generated by parsing the code, tracing the flow of the data through the code and displaying individual processes and data blocks in separate columns with arrows that connect the program flow icons 126 and indicate the direction of the data flow. In addition, as shown in Fig. 18, the data flow visualizer also allows the user to combine flows in a step-wise manner, thereby following the flow of the data from start to finish. As the user clicks the Step-wise button 121a located on the visualizer window 121, shown in Fig. 17, the visualizer 120 and the document view engine 200 may re-generate the data flow 124a on the visualizer window 121 and collapse the number of program flow icons 126 that comprise the data flow 124, ultimately showing the entire lineage of the data (i.e., where the data came from, how the data has been processed and where the data was stored). Thus, the visualizer 120 enables the user to view a representation of the flow of data during execution of selected code one step or program statement at a time. It should also be appreciated that the user may take a single data flow 124 and reverse the step-wise function, thereby expanding the number of program flow icons 126 that comprise the data flow 124 and showing the data flow 124 for individual sections or blocks of code.

It should also be appreciated that the integrated software application 30 allows changes to the code to be made textually or visually, i.e., by using the editor 80 or the visualizer 120, respectively. Editing, creating or developing new code visually using the visualizer is achieved by reverse engineering the exiting code or code templates, i.e., SAS®, SQL®, SPSS®, DB2 UDB®, Oracle® RDBMS and UNIX® Scripts, and displaying the code visually using icons. As the user manipulate these icons visually, code will be generated. Any changes to the code via the visual interface can be forward engineered to assume a textual format capable of being executed on the

respective Data Development and Data Management System. Therefore, the integrated software application 30 is capable of producing a textual file that is derived from a visual model and executing the derived textual file.

As shown in Fig. 19, the visualizer window 121 may also be configured to display a split view. When the visualizer window 121 is in split view mode, the visualizer window 121 will be comprised of a flow panel 121b and an visualizer code panel 121c. The flow panel 121b will display either the program flow 122 or the data flow 124, and the visualizer code panel 121c will display the code for program flow icons 126, which are shown in the flow panel 121b. Moreover, by clicking on an icon in the program flow window or data flow window, the user can cause the editor 80 window 100 to display the portion of the code that corresponds to the program flow icon 126 that was selected. This is especially helpful in debugging the code.

For example, as shown in Fig. 20, when elements appear in red, this signifies that the highlighted element contains an error in the underlying code. If the user selects the highlighted element, a pop-up window may be displayed to the user depicting the code associated with the highlighted element, along with the corresponding Error log, i.e., the editor 80 will scroll to the corresponding section of code and display the code to the user, while also displaying the Error log in close proximity to the code. It should be appreciated by those with skill in the art that during the user session, the user may freely navigate between the document manager 60, the editor 80 and the visualizer 120, as needed, by selecting the corresponding icon in the navigation bar 48.

It should also be understood that the user can visualize the execution log. Visualizing the execution log will show the exact path of the program flow 122 and data flow 124. The program flow 122 and the data flow 124 will be exact because they are based on the actual execution of the code. This in turns provides additional debugging and optimization information, such as, the code section that will get executed, how much data is being processed, the execution time for the code, the external files, library or macros that are referenced by the code, the format for the fully instantiated macro, etc.

For parsing the code, the document manager 60 first determines the file type for a selected file, i.e. the SAS®, SPSS®, SQL®, DB2 UDB®, Oracle® RDBMS etc. After the file type for a selected portion of code is determined, the parser layer 140

deploys the corresponding file parser 142, e.g., a file parser 142 that corresponds to, in this case, one of a variety of data manipulation and/or data management programming languages. By deploying the appropriate file parser 142, the parser layer 140 also activates the respective rules and logic that correspond to the detected programming language. Therefore, users are capable of developing, editing and maintaining code that can be executed by more than one data manipulation and/or data management program.

As mentioned earlier, the parser layer 140 is capable of processing varying file types, as the integrated software application 30 has been designed to be language aware. For example, as mentioned above, the document manager 60 recognizes file types and the integrated software application 30 includes enhanced, standalone productivity tools, such as, generic text editors, the Windows File Manager and File Transfer programs 68. In addition, these productivity tools are also designed to work seamlessly with the integrated software application 30 by implementing XML protocols, as is generally known in the art. Thus, although the preferred embodiment of the present invention is designed to interact with SAS® code, it should also be appreciated that the system will also be capable of parsing and interpreting other file types, such as, SPSS®, SQL®, DB2 UDB®, Oracle® RDBMS, etc.

As is generally known in the art, after the file parser 142 retrieves code from the document manager 60, the file parser 142 breaks the code document 146 into individual words/tokens 144. Based on the class of the individual tokens 144, the file parser 142 identifies and tags the tokens 144. Tokens 144 can be tagged to indicate that they are one of a variety of classes, such as, a keyword token 144a, a newline token 144b, a quotestring token 144c, a macro token 144d and a comment token 144e. By tagging the tokens 144, the parser layer 140 enables the document view engine 200 to recognize and display the program flow 122. As shown in Fig. 21, the document view engine 200 employs a predefined class structure for recognizing and displaying the tokens 144 provided by the file parser 142. An example of the class structure as it might be implemented in a parsed document/file is shown in Fig. 22.

While specific embodiments of the present invention have been described in detail, it will be appreciated by those skilled in the art that various modifications and alternatives to those details could be developed in light of the overall teachings of the disclosure. For example, the processes described with respect to computer executable

instructions can be performed in hardware or software without departing from the spirit of the invention. Furthermore, the order of all steps disclosed in the figures and discussed above has been provided for exemplary purposes only. Therefore, it should be understood by those skilled in the art that these steps may be rearranged and altered without departing from the spirit of the present invention. In addition, it is to be understood that all patents discussed in this document are to be incorporated herein by reference in their entirety. Moreover, while the present invention may be described in terms of a particular programming language, it should also be understood that the present invention may be programmed in various other software languages. Accordingly, the particular arrangement disclosed is meant to be illustrative only and not limiting as to the scope of the invention which is to be given the full breadth of the appended claims and any equivalents thereof.

CLAIMS

What is claimed is:

1. A computer system, comprising:

a document manager for retrieving source code programmed using one of a plurality of types of data manipulation languages;

an editor for displaying the retrieved source code and providing a means for a user to edit the retrieved source code;

a parser layer which detects the one of the plurality of types of data manipulation languages in which the retrieved source code is programmed and which activates rules and logic applicable to the detected one of the plurality of types of data manipulation languages; and

a visualizer dynamically linked to the editor for displaying graphical representations of flows within the retrieved source code using the rules and logic applicable to the detected one of the plurality of types of data manipulation languages and activated by the parser, wherein the editor, parser layer and visualizer cooperate such that edits made to the source code using the editor are automatically reflected in the graphical representations of flows displayed by the visualizer and edits made to the graphical representations of flows in the visualizer are automatically reflected in the source code displayed by the editor.

2. A computer system as recited in claim 1, wherein the graphical representations of flows depict data flows.

3. A computer system as recited in claim 1, wherein the graphical representations of flows depict program flows.

4. A computer system as recited in claim 1, wherein the graphical representations of data flows are expandable and collapsible.

5. A computer system as recited in claim 1, wherein the document manager retrieves all files related to the source code to be edited.

6. A computer system as recited in claim 1, wherein the document manager comprises a site manager and a connectivity layer for retrieving source code from one or more remote computers.

7. A computer system as recited in claim 6, wherein the document manager comprises a security layer for managing secure connections with the one or more remote computers.

8. A computer system as recited in claim 1, wherein the editor comprises a template manager for allowing preprogrammed segment of source code to be placed within the source code being edited.

9. A computer system as recited in claim 8, wherein the template manager is adapted to automatically correct segments of the source code.

10. A computer system as recited in claim 8, wherein the template manager is adapted to automatically generate segments of the source code.

11. A computer system as recited in claim 1, further comprising a means for allowing the source code to be executed both locally and remotely.

12. A computer system as recited in claim 11, wherein the parser layer further examines error log files generated by the means for allowing the source code to be executed to determine segments of the source code determined to include errors.

13. A computer system as recited in claim 12, wherein the visualizer cooperates with the parser layer to change the appearance of displayed flows as a function of the source code segments determined to have errors.

14. A computer system as recited in claim 12, wherein the editor cooperates with the parser layer to change the appearance of portions of the displayed source code as a function of the software segments determined to have errors.

15. A computer system as recited in claim 12, further comprising a message manager cooperating with the parser layer for displaying debugging hints as a function of the source code segments determined to have errors.

16. A computer system as recited in claim 15, wherein the message manager allows a user to edit and maintain debugging hints for a variety of different errors.

17. A computer readable media having computer executable instructions for examining and editing software, the instructions performing steps comprising:

retrieving software comprised of source code that is programmed in one of a plurality of types of data manipulation programming languages;

displaying the retrieved source code and providing a means for a user to edit the displayed, retrieved source code;

detecting the one of the plurality of types of data manipulation languages in which the retrieved source code is programmed;

activating rules and logic applicable to the detected one of the plurality of types of data manipulation languages to parse the source code and to create a flow representation of the source code;

displaying the flow representation of the source code and providing a means for the user to edit the displayed flow representation of the source code; and

causing edits made to the displayed source code to be automatically reflected in the displayed flow representation of the source code and causing edits made to the displayed flow representation of the source code to be automatically reflected in the displayed source code.

18. The computer readable media as recited in claim 17, wherein the flow representation depicts data flow.

19. The computer readable media as recited in claim 17, wherein the flow representation depicts program flow.

20. The computer readable media as recited in claim 17, wherein the instructions allow the user to expand and collapse the flow representation.

21. The computer readable media as recited in claim 17, wherein the instructions allow the user to search a network for the software to be retrieved.

22. The computer readable media as recited in claim 17, wherein the instructions allow for retrieving of an error log file generated as a result of an execution of the software and further function to parse the error log file to determine segments of the software having an error and update the flow representations to indicate which segments are determined to have an error.

23. The computer readable media as recited in claim 22, wherein the instructions display debugging hints corresponding to the errors determined to be within the software.

24. The computer readable media as recited in claim 23, wherein the instructions allow for retrieving of the debugging hints by browsing a network.

25. The computer readable media as recited in claim 24, wherein the instructions allow for the customization of the debugging hints.

26. The computer readable media as recited in claim 24, wherein the instructions display a plurality of code templates to allow the user to select and insert one or more of the code templates into the software.

27. The computer readable media as recited in claim 26, wherein the instructions allow for browsing a network to retrieve the code templates for selection by the user.

28. The computer readable media as recited in claim 27, wherein the instructions display code within a code template in response to a request by the user.

29. The computer readable media as recited in claim 27, wherein the instructions automatically correct segments of the source code.

30. The computer readable media as recited in claim 27, wherein the instructions automatically generate segments of the source code.

31. The computer readable media as recited in claim 17, wherein the flow representation comprises individual flow elements representative of designated segments of the source code.

32. The computer readable media as recited in claim 31, wherein the instructions display comments and execution statistics associated with each flow element in response to a mouse-over selection of the respective flow element.

33. The computer readable media as recited in claim 17, wherein the instructions display a list of data inspection functions in response to selection of a dataset icon.

34. The computer readable media as recited in claim 17, wherein the instructions display a list of data discovery functions in response to selection of a dataset icon.

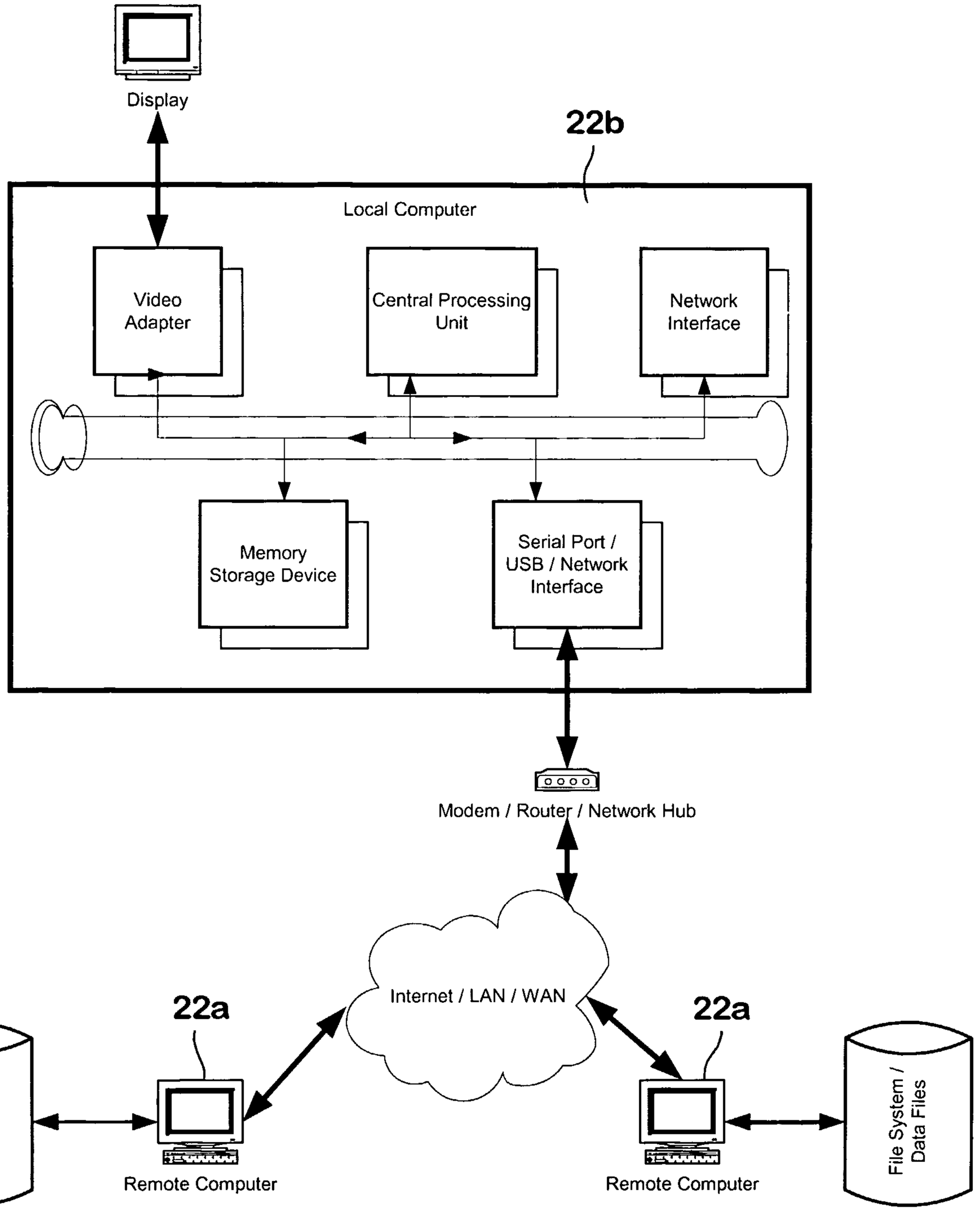

Display
22b
Local Computer
Video Adapter
Central Processing Unit
Network Interface
Memory Storage Device
Serial Port / USB / Network Interface
Modem / Router / Network Hub
Internet / LAN / WAN
22a
22a
File System / Data Files
Remote Computer
Remote Computer
File System / Data Files

FIG. 1

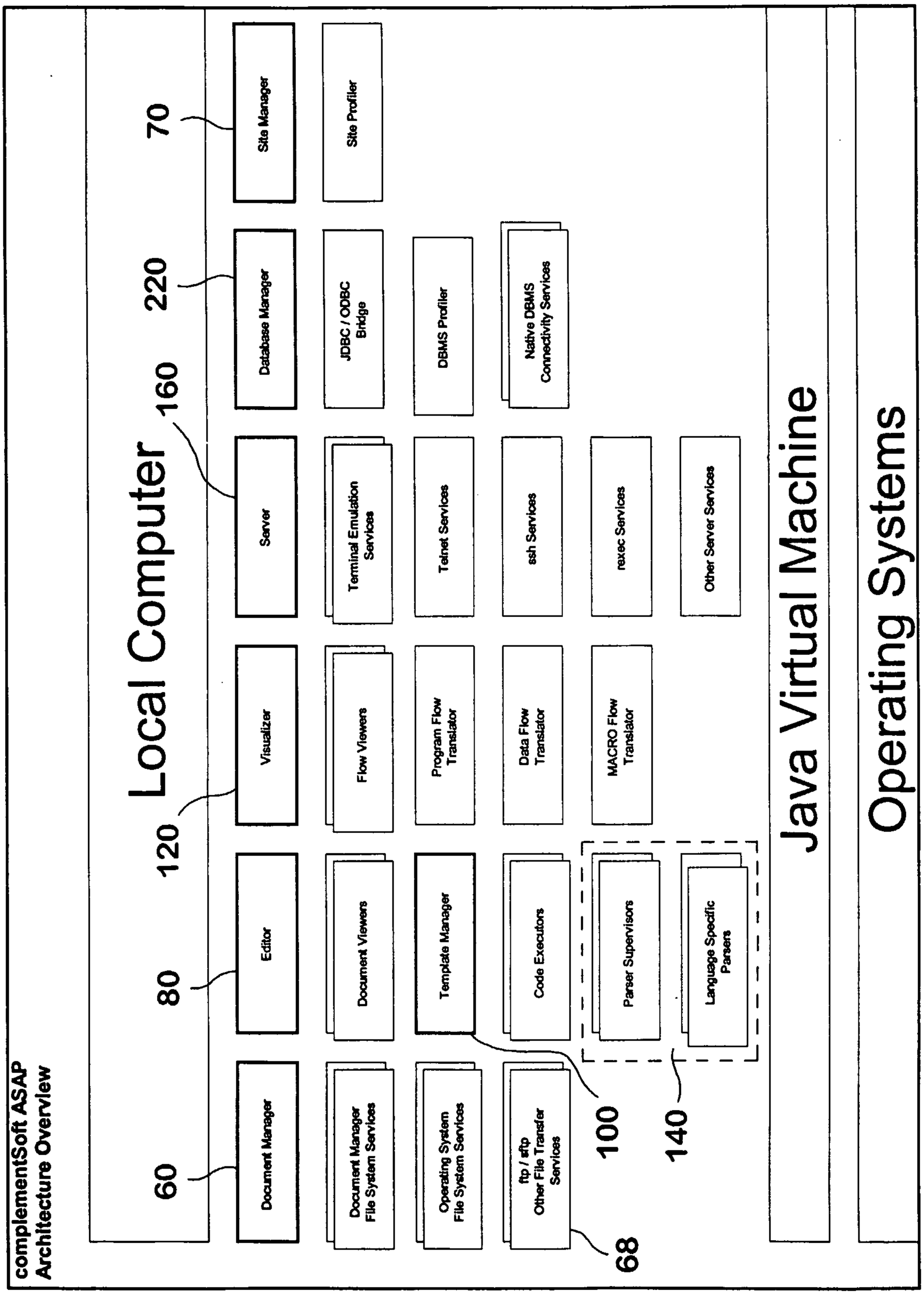

complementSoft ASAP
Architecture Overview
Local Computer
Java Virtual Machine
Operating Systems
Server
Terminal Emulation Services
Telnet Services
ssh Services
rexec Services
Other Server Services
Database Manager
JDBC / ODBC Bridge
DBMS Profiler
Native DBMS Connectivity Services
Site Manager
Site Profiler
Visualizer
Flow Viewers
Program Flow Translator
Data Flow Translator
MACRO Flow Translator
Editor
Document Viewers
Template Manager
Code Executors
Parser Supervisors
Language Specific Parsers
Document Manager
Document Manager File System Services
Operating System File System Services
ftp / sftp Other File Transfer Services
60
80
120
160
220
70
100
140
68

FIG. 2

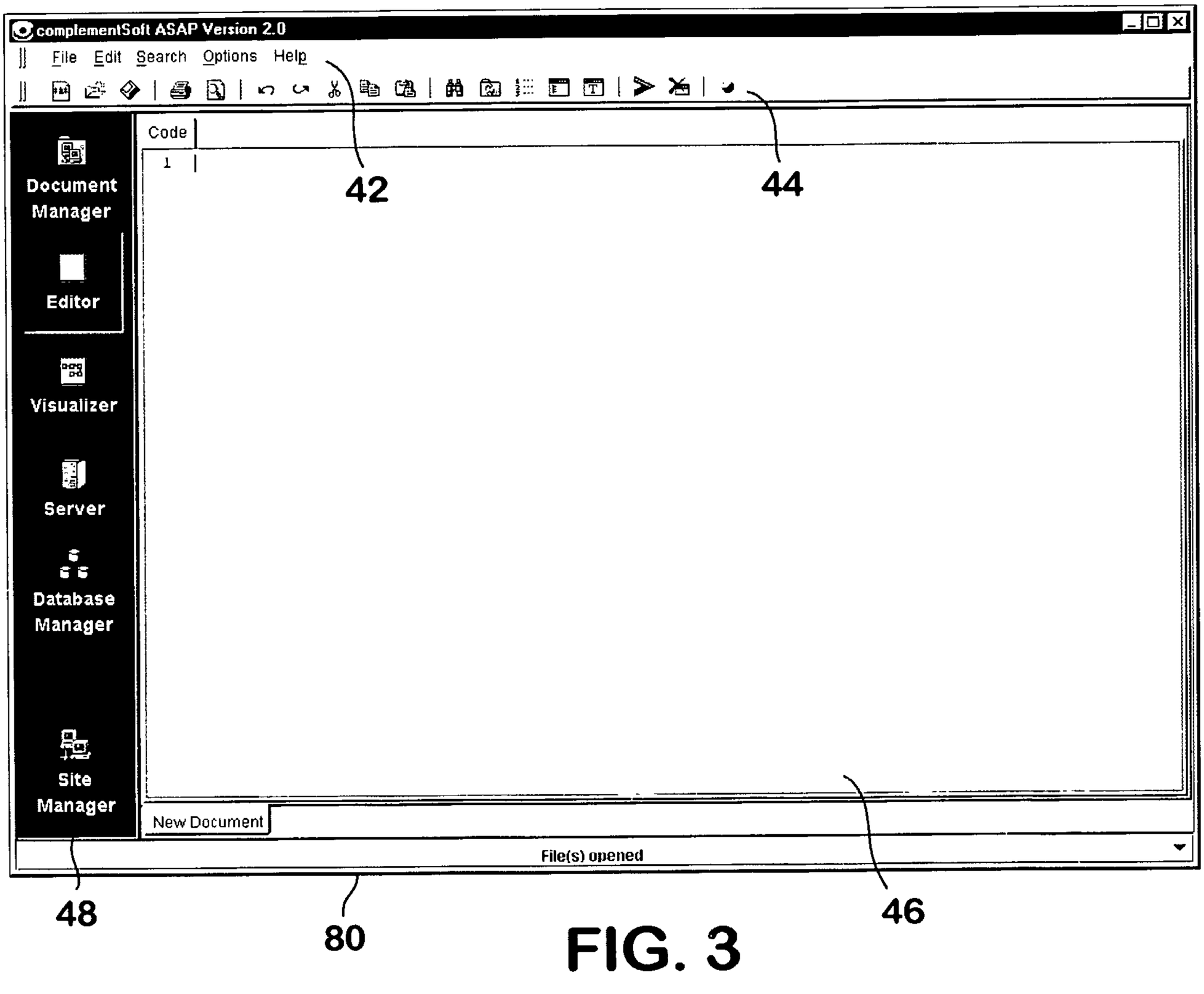
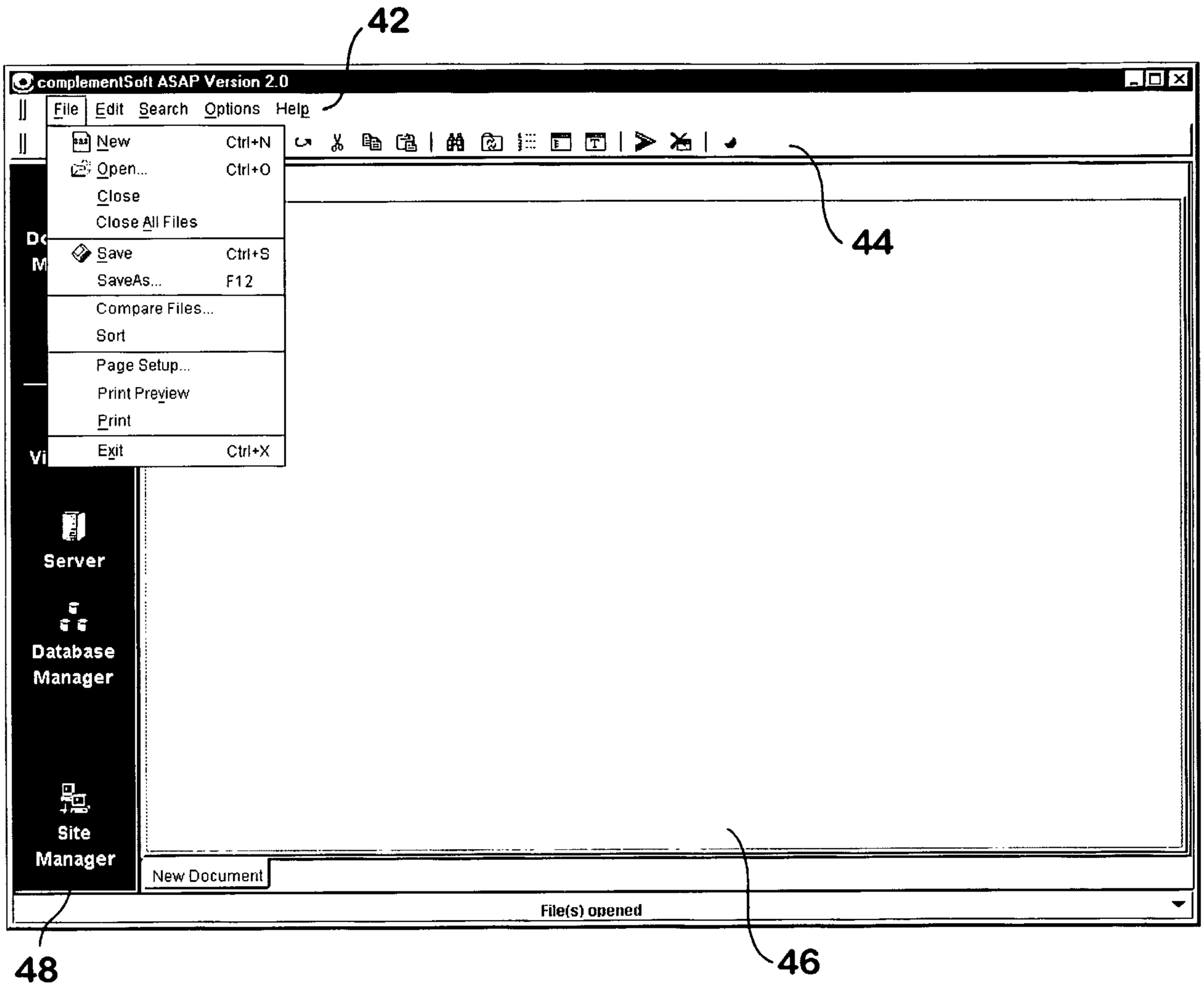
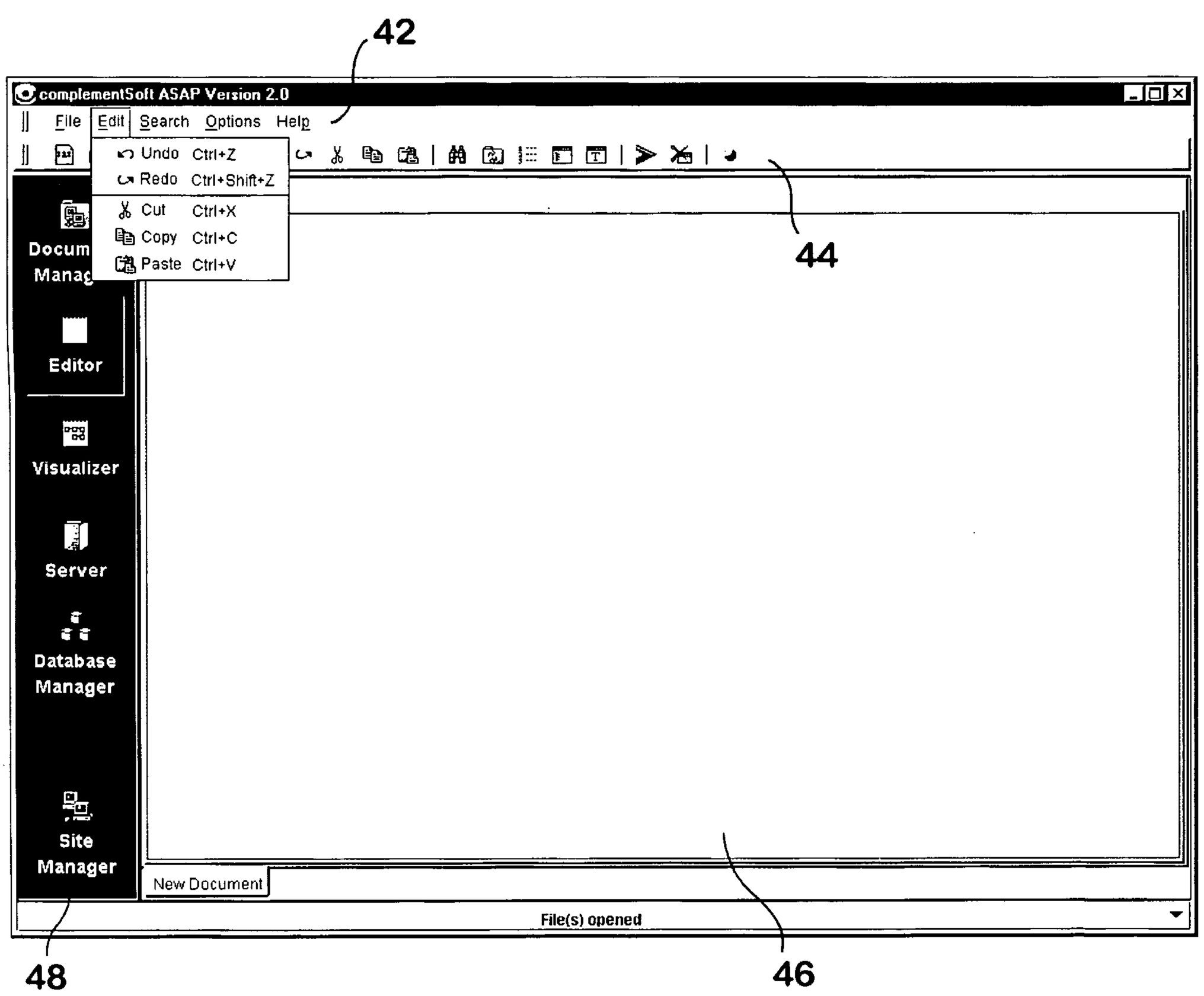

complementSoft ASAP Version 2.0
File Edit Search Options Help
Code
1
42
44
Document Manager
Editor
Visualizer
Server
Database Manager
Site Manager
New Document
File(s) opened
48
80
46

FIG. 3



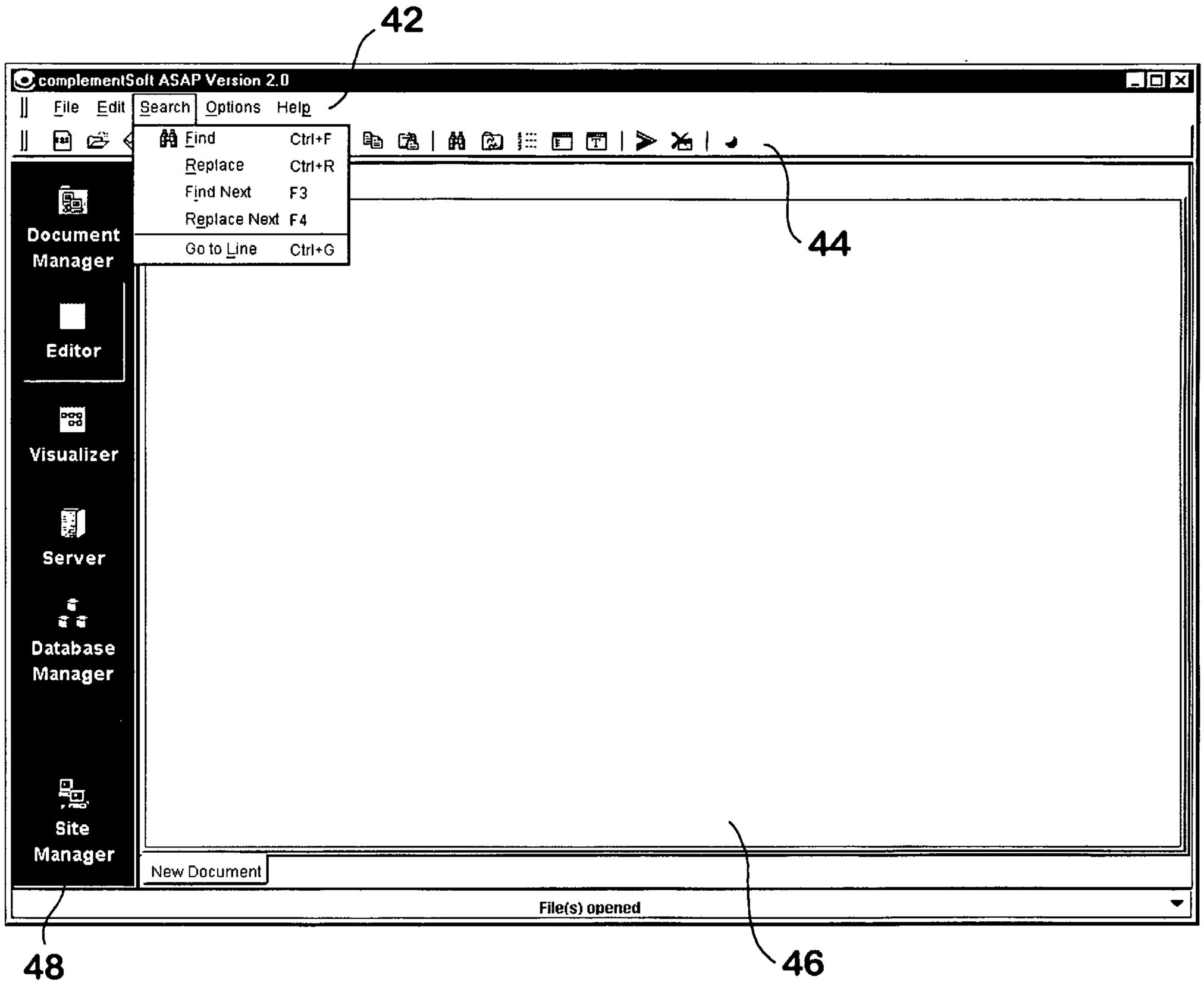
Menu Item	Functionality
New	Create a new document (a program)
Open	Open an existing document
Close	Close the currently opened document
Save	Save the currently open document
Save As	Save the currently open document using a different name or in a different location
Print Preview	Print Preview of the currently open (active) document
Print	Print the currently open (active) document
Exit	Exit the application

FIG. 4a



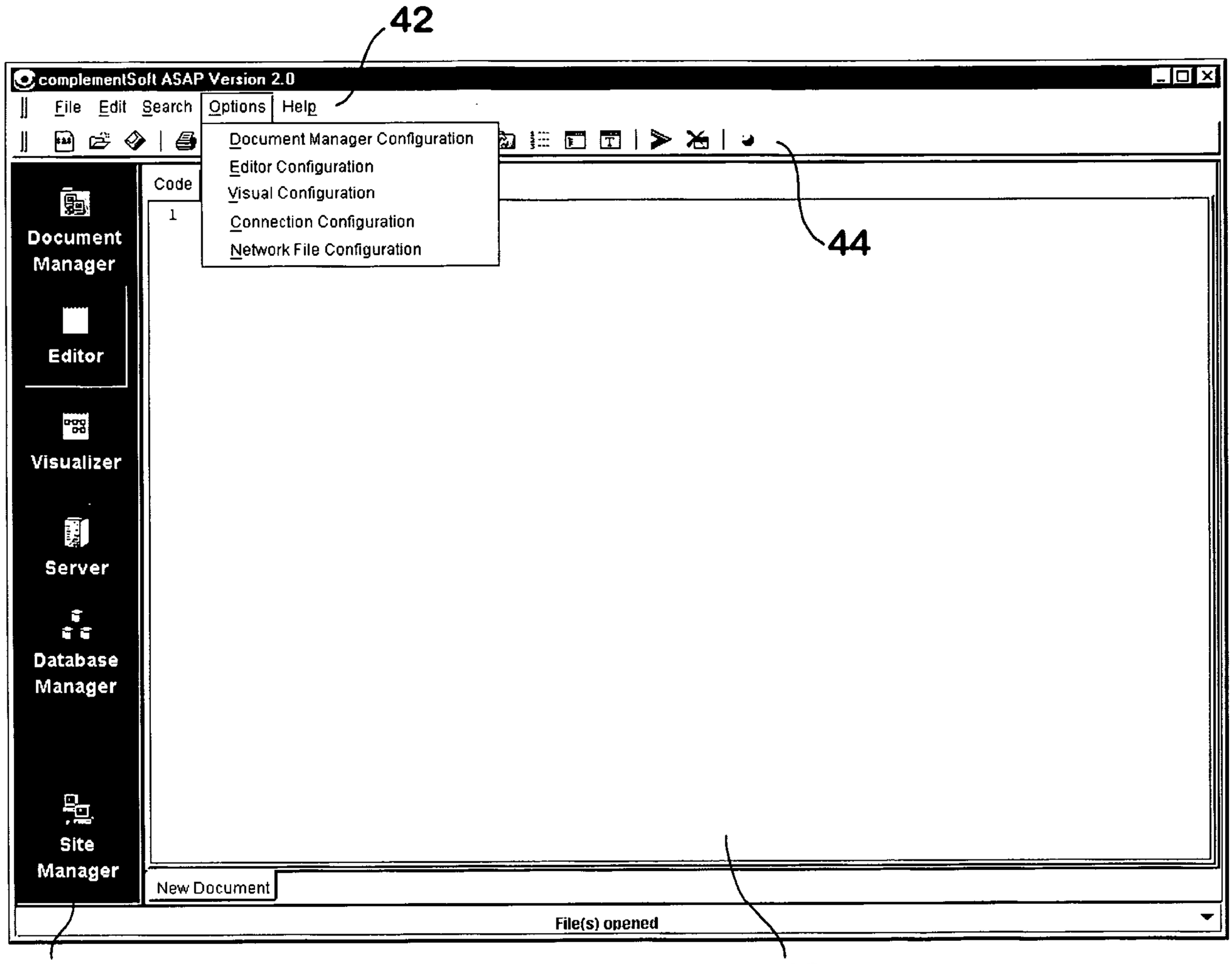
Menu Item	Functionality
Undo	
Redo	
Cut	
Copy	
Paste	

FIG. 4b



Menu Item	Functionality
Find	
Replace	
Find Next	
Replace Next	
Goto Line	Go to the n line in the currently active document

FIG. 4c



Menu Item	Functionality
Document Manager Configuration	Activate the Document Manager Configurator
Editor Configuration	Activate the Editor Configurator
Visual Configuration	Activate the Visual Configurator
Connection Configuration	Activate the Connection Configurator
Network File Configuration	Activate the Network File Configurator

FIG. 4d

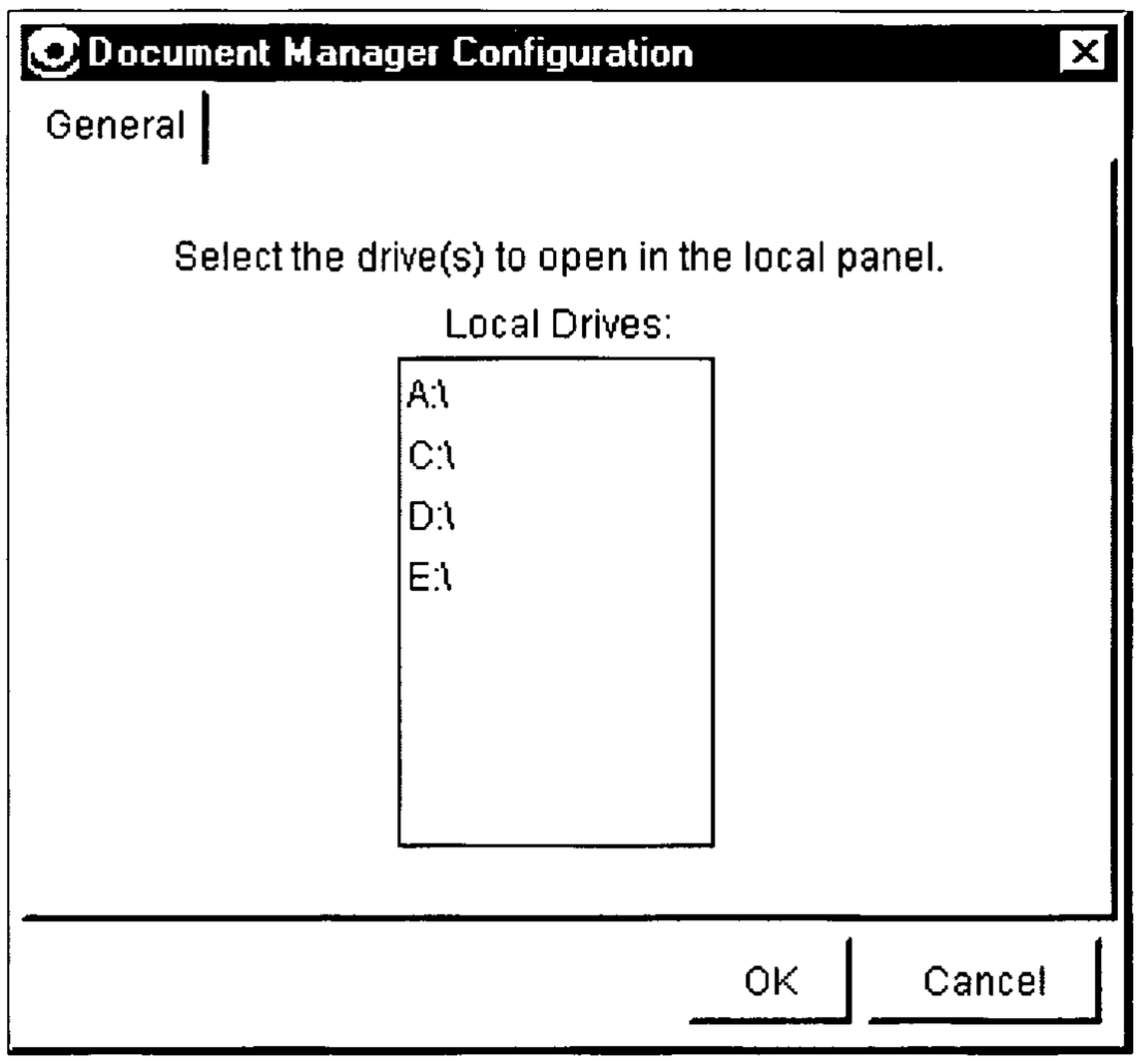

Document Manager Configuration
General
Select the drive(s) to open in the local panel.
Local Drives:
A:\
C:\
D:\
E:\
OK
Cancel

FIG. 4da

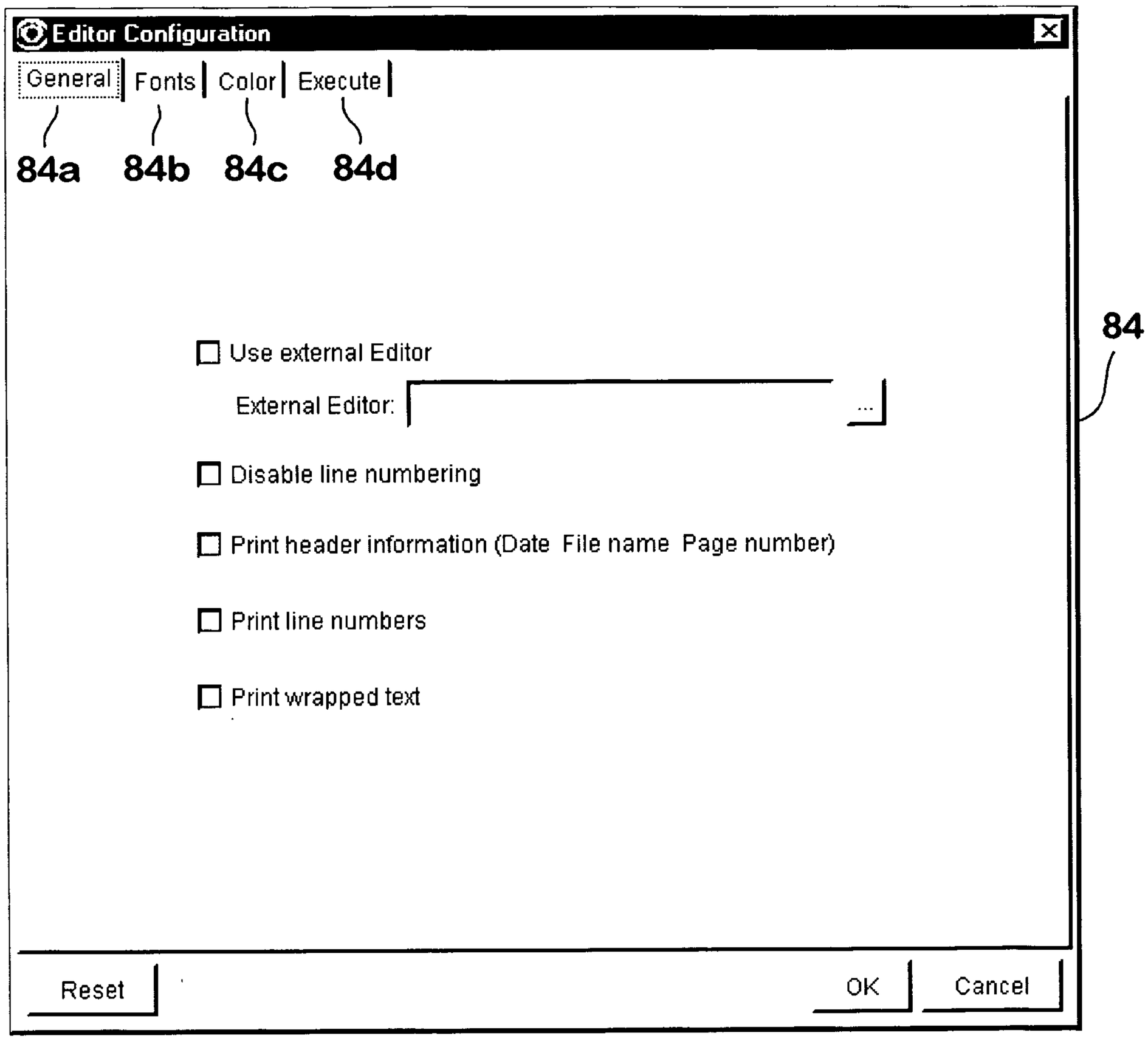

Editor Configuration
General
Fonts
Color
Execute
84a
84b
84c
84d
84
Use external Editor
External Editor:
...
Disable line numbering
Print header information (Date File name Page number)
Print line numbers
Print wrapped text
Reset
OK
Cancel

FIG. 4db

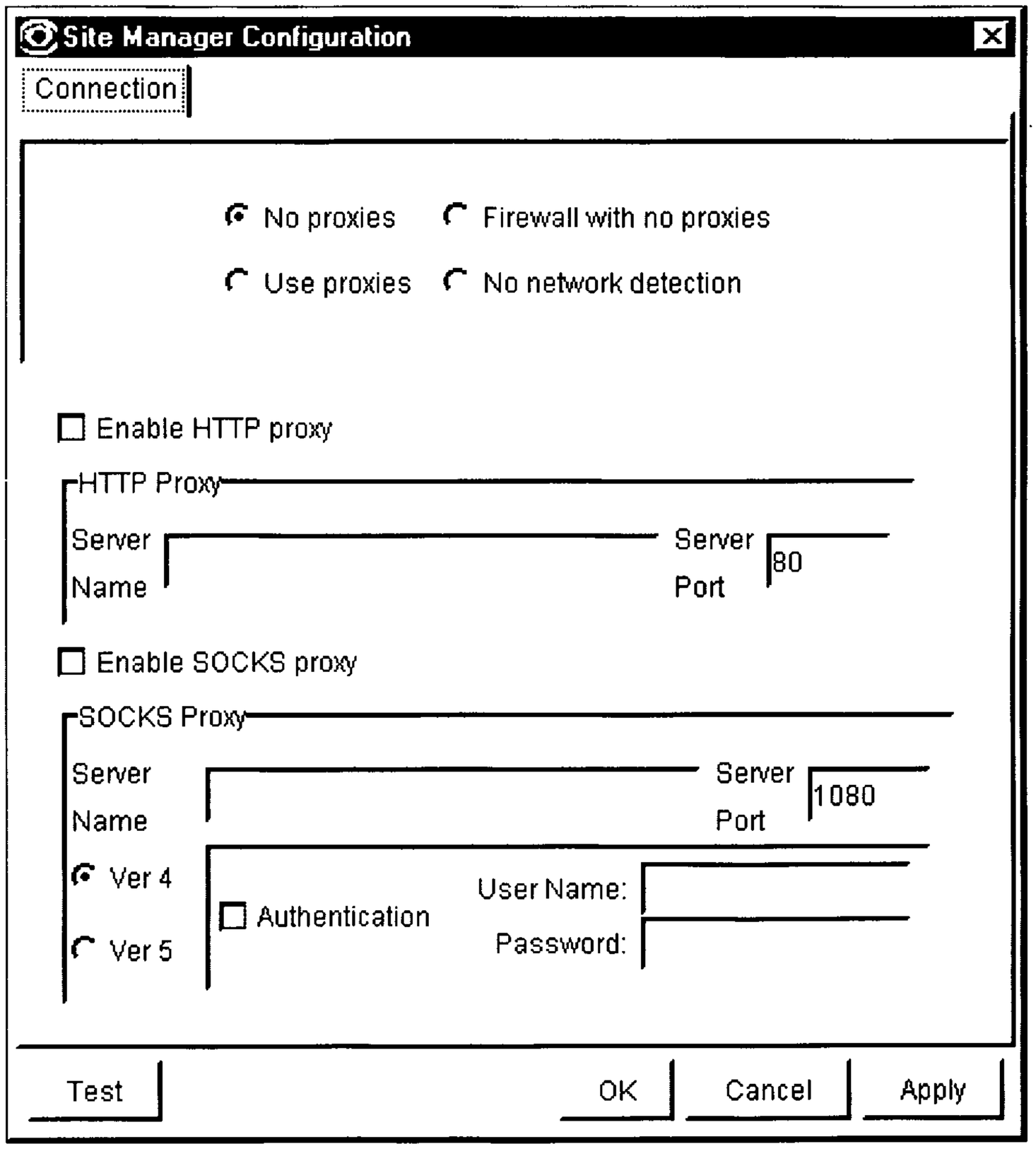

Site Manager Configuration
Connection
No proxies
Firewall with no proxies
Use proxies
No network detection
Enable HTTP proxy
HTTP Proxy
Server Name
Server Port
80
Enable SOCKS proxy
SOCKS Proxy
Server Name
Server Port
1080
Ver 4
Ver 5
Authentication
User Name:
Password:
Test
OK
Cancel
Apply

FIG. 4dc

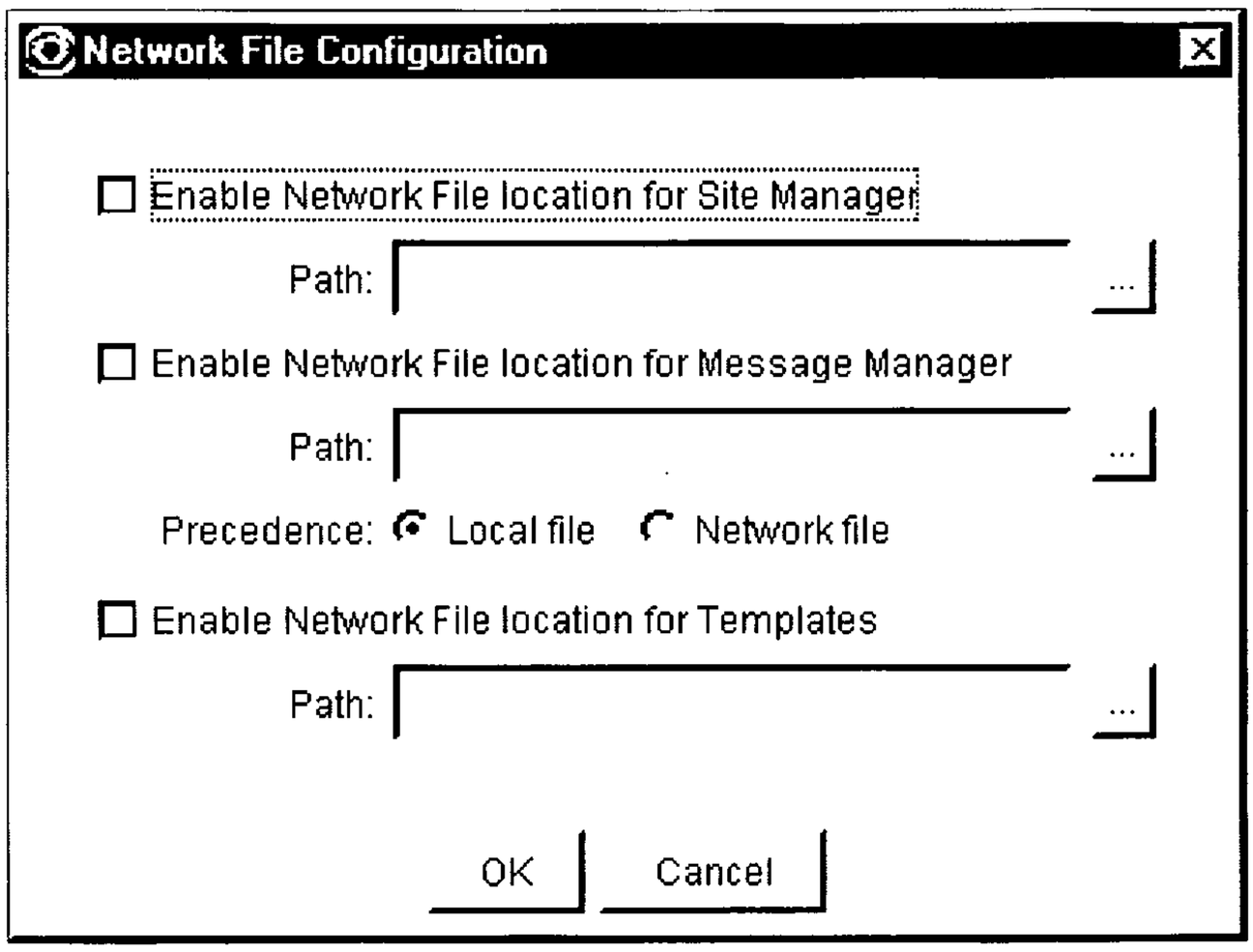
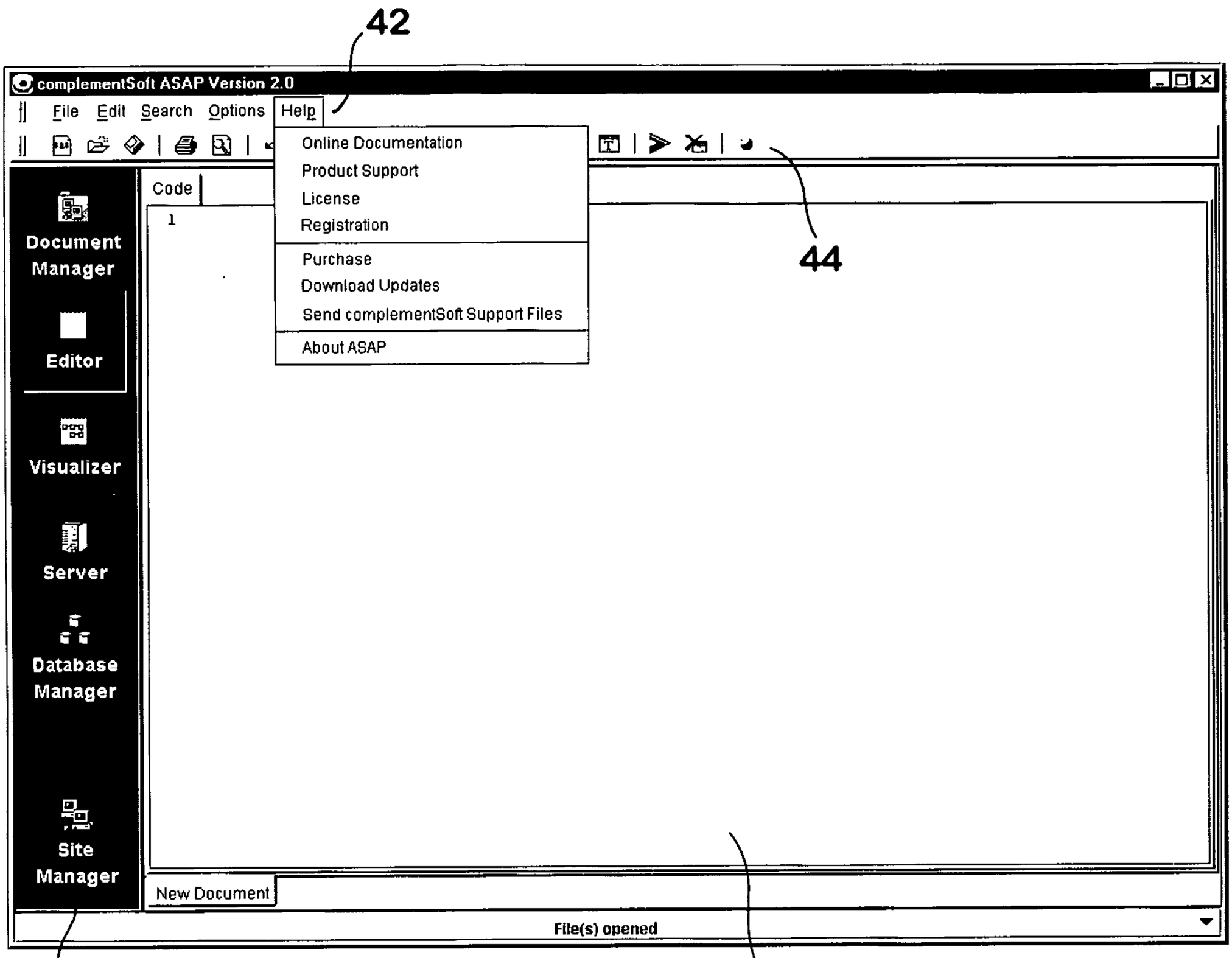

Network File Configuration
Enable Network File location for Site Manager
Path:
...
Enable Network File location for Message Manager
Path:
...
Precedence: Local file Network file
Enable Network File location for Templates
Path:
...
OK
Cancel

FIG. 4dd



Menu Item	Functionality
On-line Documentation	Show On-line Documentation
Product Support	Displays the Product Support Page with e-mail address, How to . . . and Frequently Asked Questions
License	Detailed license information with the ability to renew and/or upgrade the license
Registration	Perform on-line product registration
Purchase	Displays the complementSoft e-store
Download Updates	Retrieve Product Updates
Send complementsoft Support files	E-mail support files to complementSoft
About ASAP	Version and copyright information

FIG. 4e

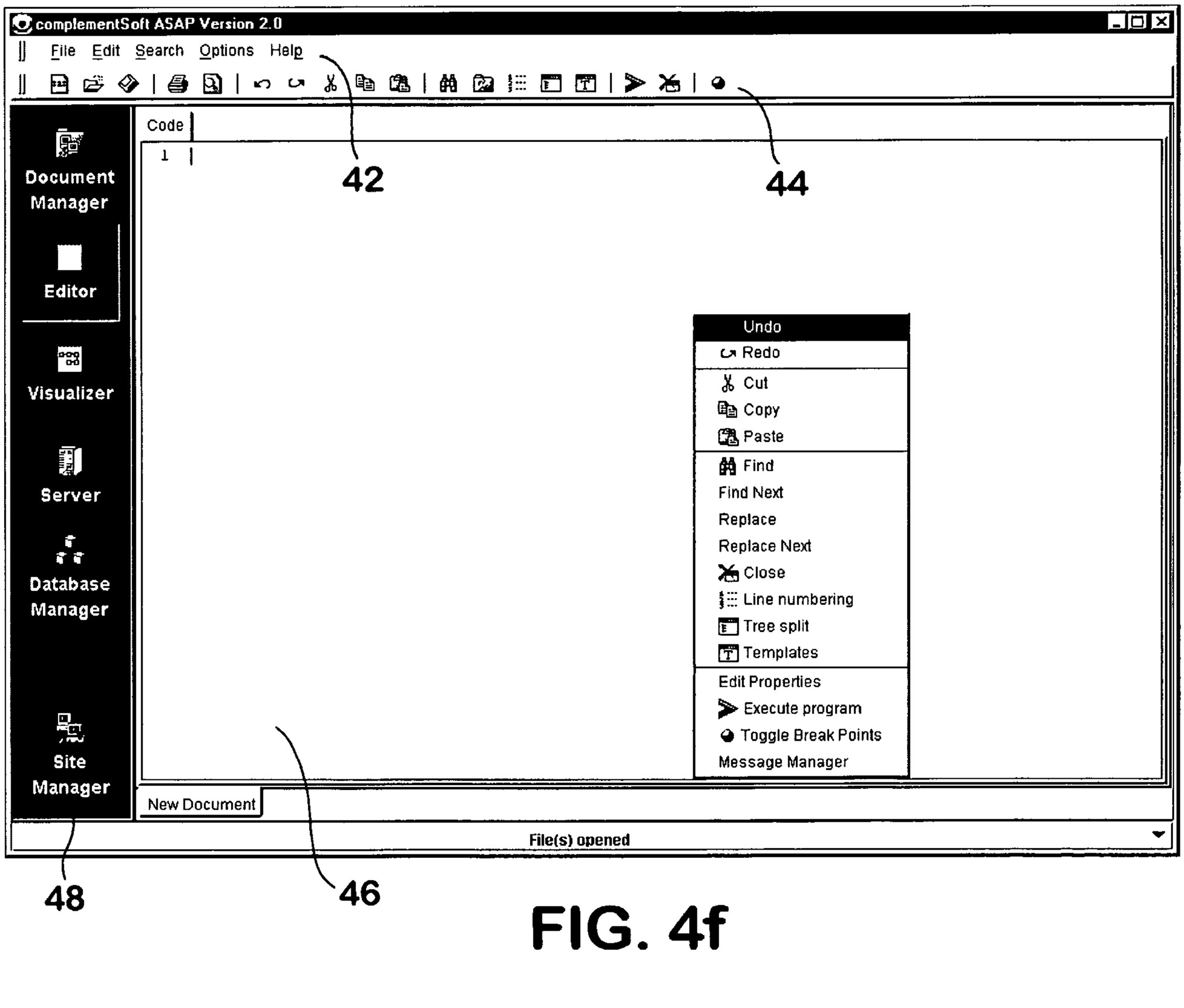
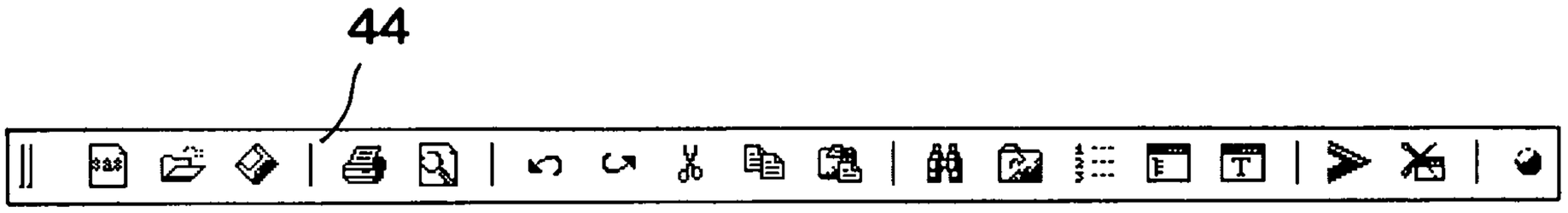

complementSoft ASAP Version 2.0
File Edit Search Options Help
Code
1
42
44
Document Manager
Editor
Visualizer
Server
Database Manager
Site Manager
Undo
Redo
Cut
Copy
Paste
Find
Find Next
Replace
Replace Next
Close
Line numbering
Tree split
Templates
Edit Properties
Execute program
Toggle Break Points
Message Manager
New Document
File(s) opened
48
46

FIG. 4f

Menu Item	Functionality
Undo	
Redo	
Cut	
Copy	
Paste	
Find	
Find Next	
Replace	
Replace Next	
Close	Close the currently active document
Line numbering	Turn line numbering on of off
Tree Split	Activate the Tree View split screen mode
Template Manager	Activate the Template Manager
Edit Properties	Edit the properties (timestamp, originating location, etc.) of the document
Execute Program	Execute the currently active program (i.e., the one that the user is currently editing) The user can further specify the execution mode (i.e., how to execute, with which option turn on/off, etc) and execution location (i.e., where to execute, which server, which execution engine, which location) If the user has previously activated Break-Point execution (see below), then the execution will be carried out based out the break-points, i.e., start at where the user indicate that it should start, end at where the user indicate that it should end and skipping statements as indicated by the user.
Toggle Break Points	Activate / deactivate Break-Point execution. Setup Break-Points if they have not been previously setup. Reset/Remove Break-Points if they have been previously setup
Message Manager	Activate the Message Manager to add, remove, or update the messages in the message repository

FIG. 4f (continued)



Menu Item	Functionality
New	
Open	
Save	
Print	
Print Preview	
Undo	
Redo	
Cut	
Copy	
Paste	
Find	Find a word
Refresh	Refresh the Editor display
Toggle Line Numbers	Show or hide line numbers
Toggle Tree View	Enable or disable Tree View
Template Manager	Activate the Template manager
Execute Program	Execute the currently active program
Close	Close the currently active program
Toggle Break Point Execution	Enable of disable Break Point Execution

FIG. 5

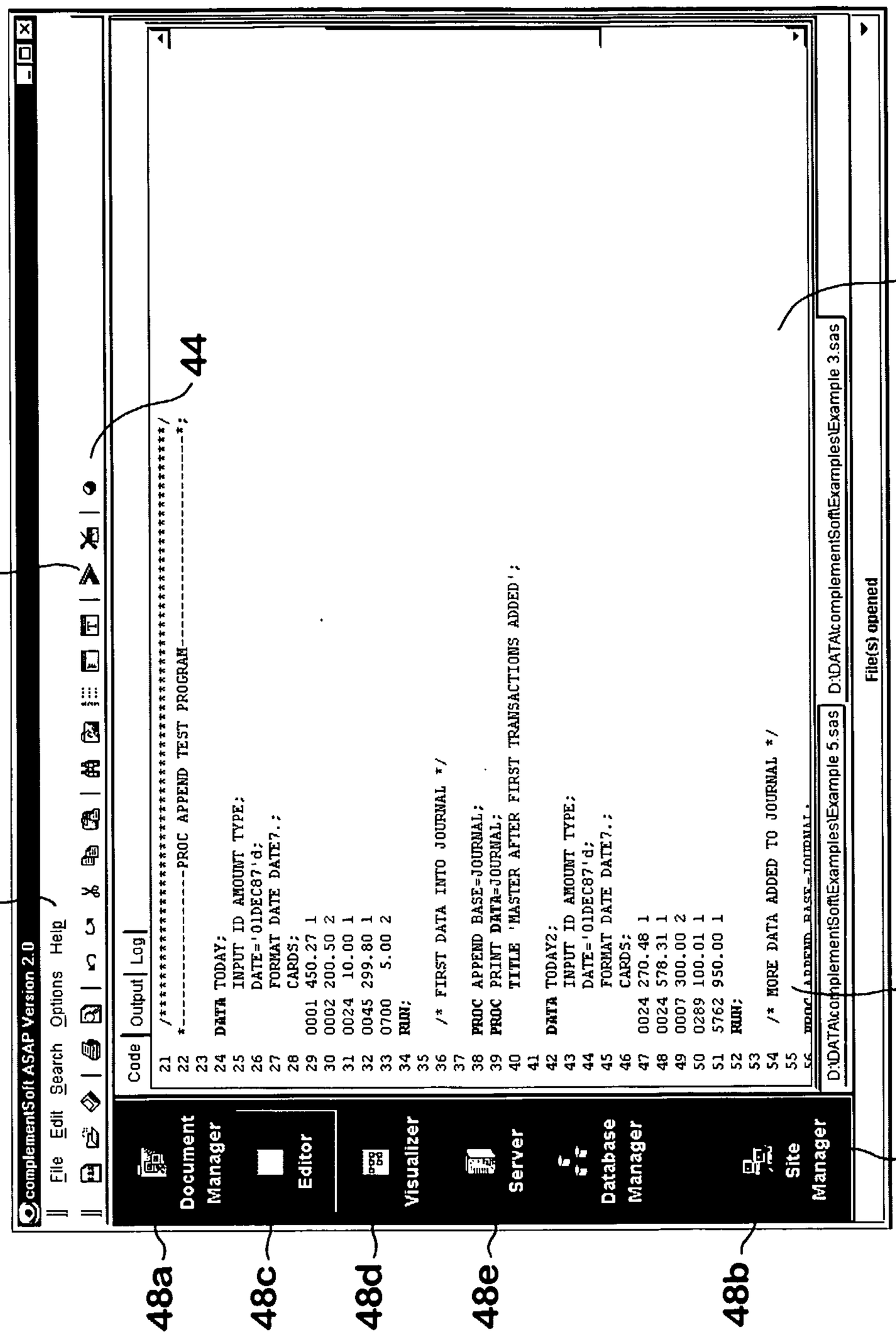

42
44a
44
80
82
48
48a
48c
48d
48e
48b
complementSoft ASAP Version 2.0
File Edit Search Options Help
Code Output Log
Document Manager
Editor
Visualizer
Server
Database Manager
Site Manager
D:\DATA\complementSoft\Examples\Example 5.sas
D:\DATA\complementSoft\Examples\Example 3.sas
File(s) opened

FIG. 6

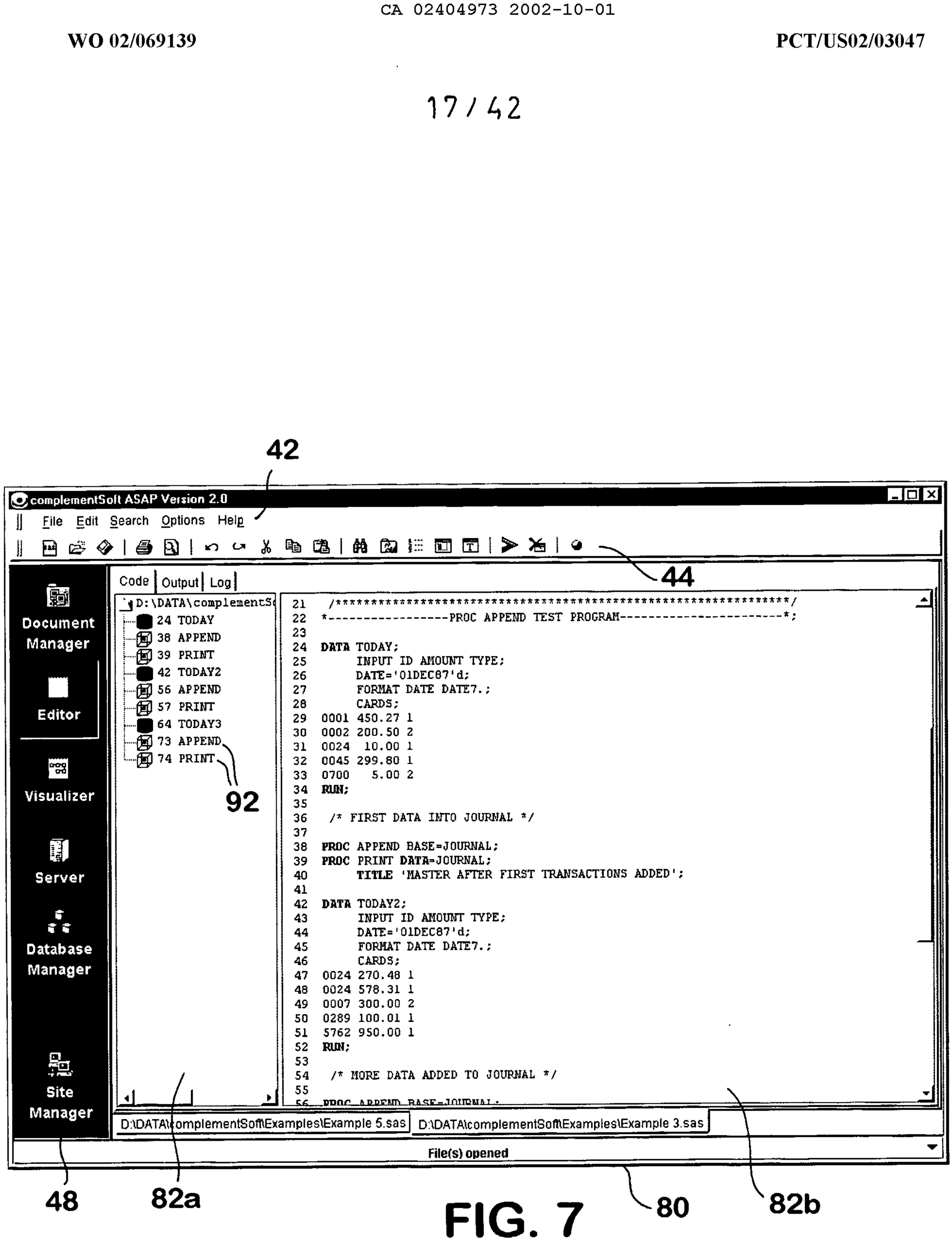

42
complementSoft ASAP Version 2.0
File Edit Search Options Help
44
Code Output Log
D:\DATA\complementS
24 TODAY
38 APPEND
39 PRINT
42 TODAY2
56 APPEND
57 PRINT
64 TODAY3
73 APPEND
74 PRINT
92
Document Manager
Editor
Visualizer
Server
Database Manager
Site Manager
21 /*--*/
22 *-----------------PROC APPEND TEST PROGRAM----------------------*;
23
24 DATA TODAY;
25 INPUT ID AMOUNT TYPE;
26 DATE='01DEC87'd;
27 FORMAT DATE DATE7.;
28 CARDS;
29 0001 450.27 1
30 0002 200.50 2
31 0024 10.00 1
32 0045 299.80 1
33 0700 5.00 2
34 RUN;
35
36 /* FIRST DATA INTO JOURNAL */
37
38 PROC APPEND BASE=JOURNAL;
39 PROC PRINT DATA=JOURNAL;
40 TITLE 'MASTER AFTER FIRST TRANSACTIONS ADDED';
41
42 DATA TODAY2;
43 INPUT ID AMOUNT TYPE;
44 DATE='01DEC87'd;
45 FORMAT DATE DATE7.;
46 CARDS;
47 0024 270.48 1
48 0024 578.31 1
49 0007 300.00 2
50 0289 100.01 1
51 5762 950.00 1
52 RUN;
53
54 /* MORE DATA ADDED TO JOURNAL */
55
D:\DATA\complementSoft\Examples\Example 5.sas
D:\DATA\complementSoft\Examples\Example 3.sas
File(s) opened
48
82a
80
82b

FIG. 7

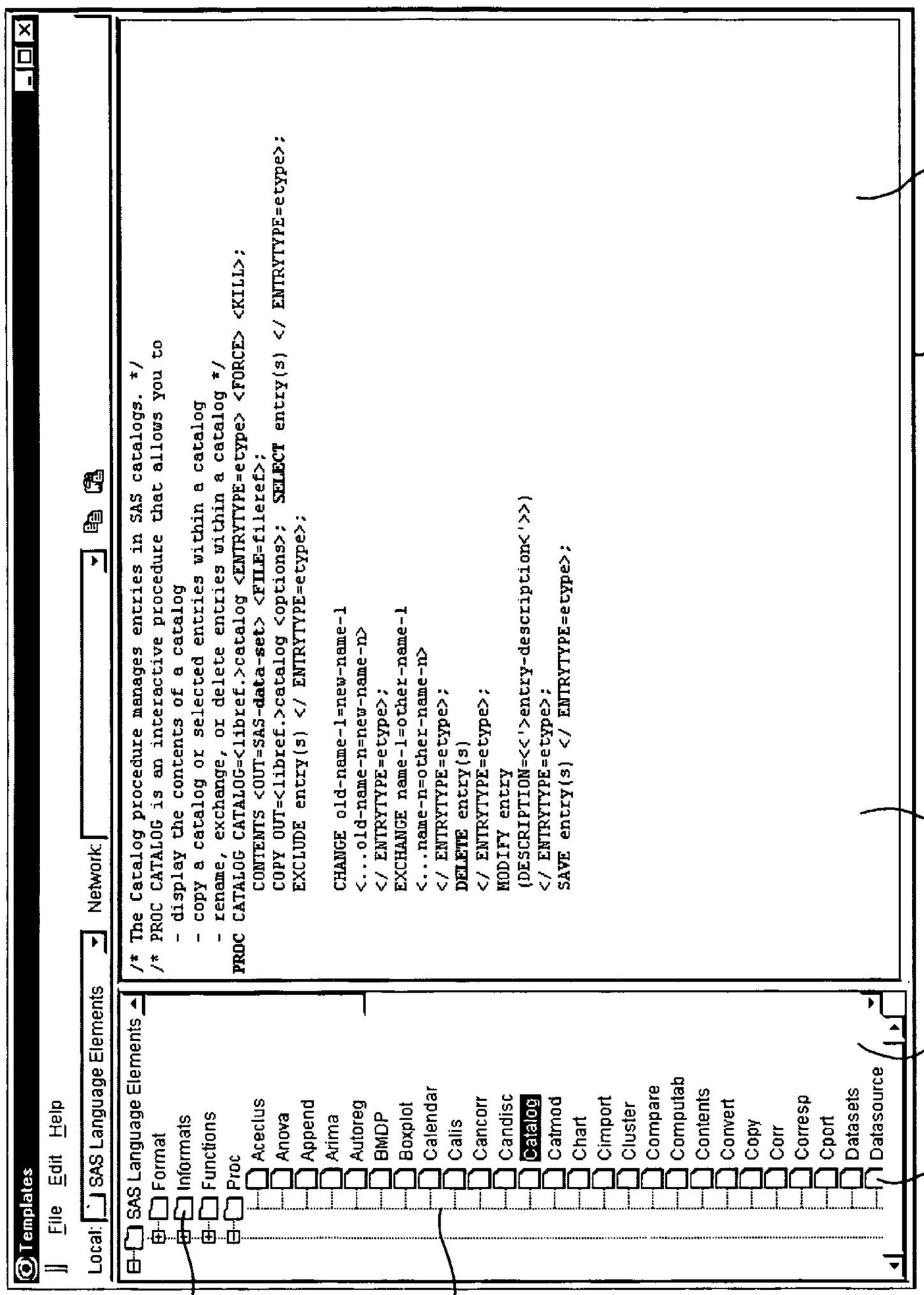

Templates
File Edit Help
Local: SAS Language Elements
Network:
SAS Language Elements
Format
Informats
Functions
Proc
Aceclus
Anova
Append
Arima
Autoreg
BMDP
Boxplot
Calendar
Calis
Cancorr
Candisc
Catalog
Catmod
Chart
Cimport
Cluster
Compare
Computab
Contents
Convert
Copy
Corr
Corresp
Cport
Datasets
Datasource
/* The Catalog procedure manages entries in SAS catalogs. */
/* PROC CATALOG is an interactive procedure that allows you to
- display the contents of a catalog
- copy a catalog or selected entries within a catalog
- rename, exchange, or delete entries within a catalog */
PROC CATALOG CATALOG=<libref.>catalog <ENTRYTYPE=etype> <FORCE> <KILL>;
CONTENTS <OUT=SAS-data-set> <FILE=fileref>;
COPY OUT=<libref.>catalog <options>; SELECT entry(s) </ ENTRYTYPE=etype>;
EXCLUDE entry(s) </ ENTRYTYPE=etype>;
CHANGE old-name-1=new-name-1
<...old-name-n=new-name-n>
</ ENTRYTYPE=etype>;
EXCHANGE name-1=other-name-1
<...name-n=other-name-n>
</ ENTRYTYPE=etype>;
DELETE entry(s)
</ ENTRYTYPE=etype>;
MODIFY entry
(DESCRIPTION=<<'>entry-description<'>>)
</ ENTRYTYPE=etype>;
SAVE entry(s) </ ENTRYTYPE=etype>;
100
101
101a
101b
104
105
106

FIG. 8

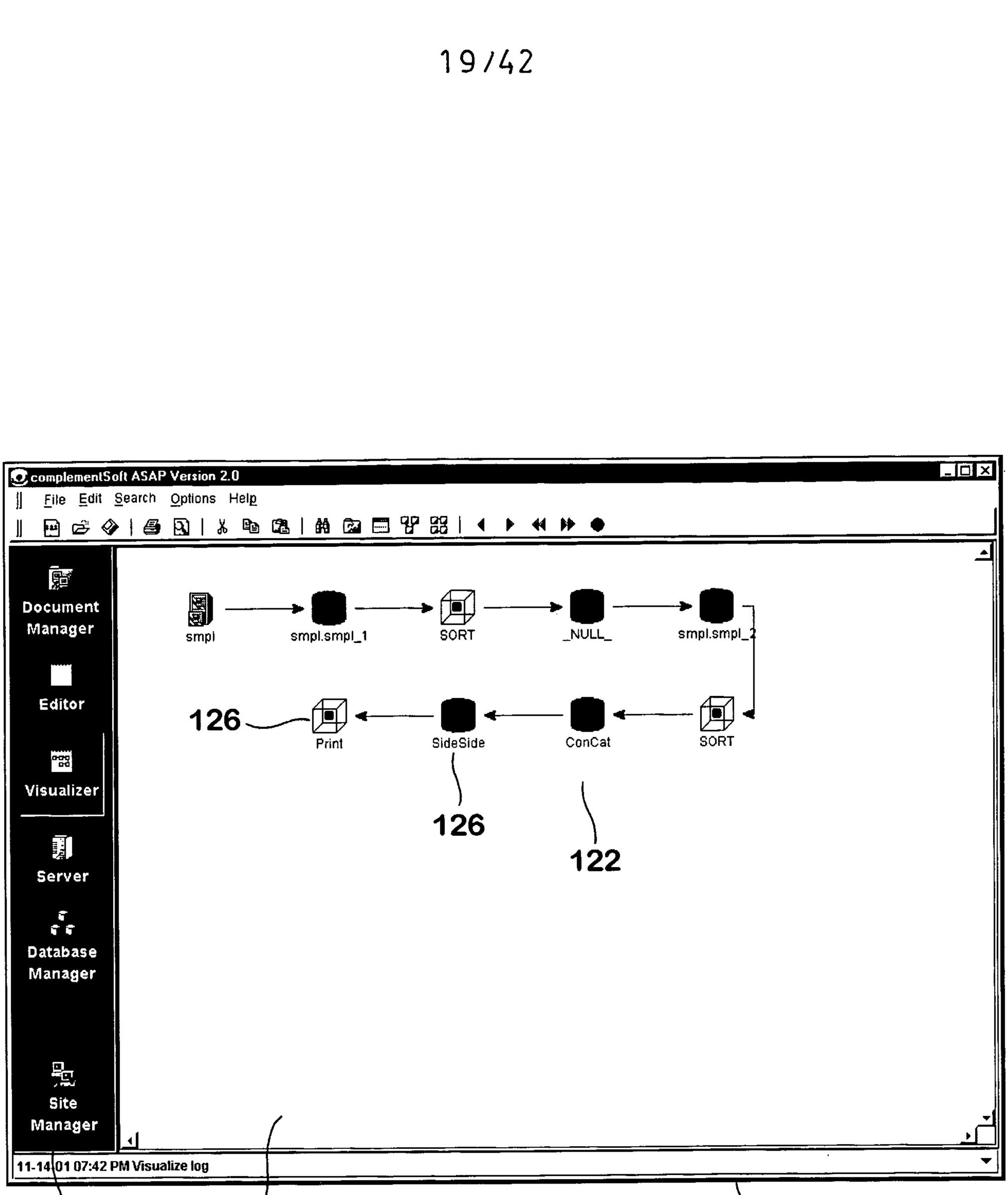

complementSoft ASAP Version 2.0
File Edit Search Options Help
Document Manager
Editor
Visualizer
Server
Database Manager
Site Manager
smpl
smpl.smpl_1
SORT
NULL
smpl.smpl_2
126
Print
SideSide
ConCat
SORT
126
122
11-14-01 07:42 PM Visualize log
48
121
120

FIG. 9

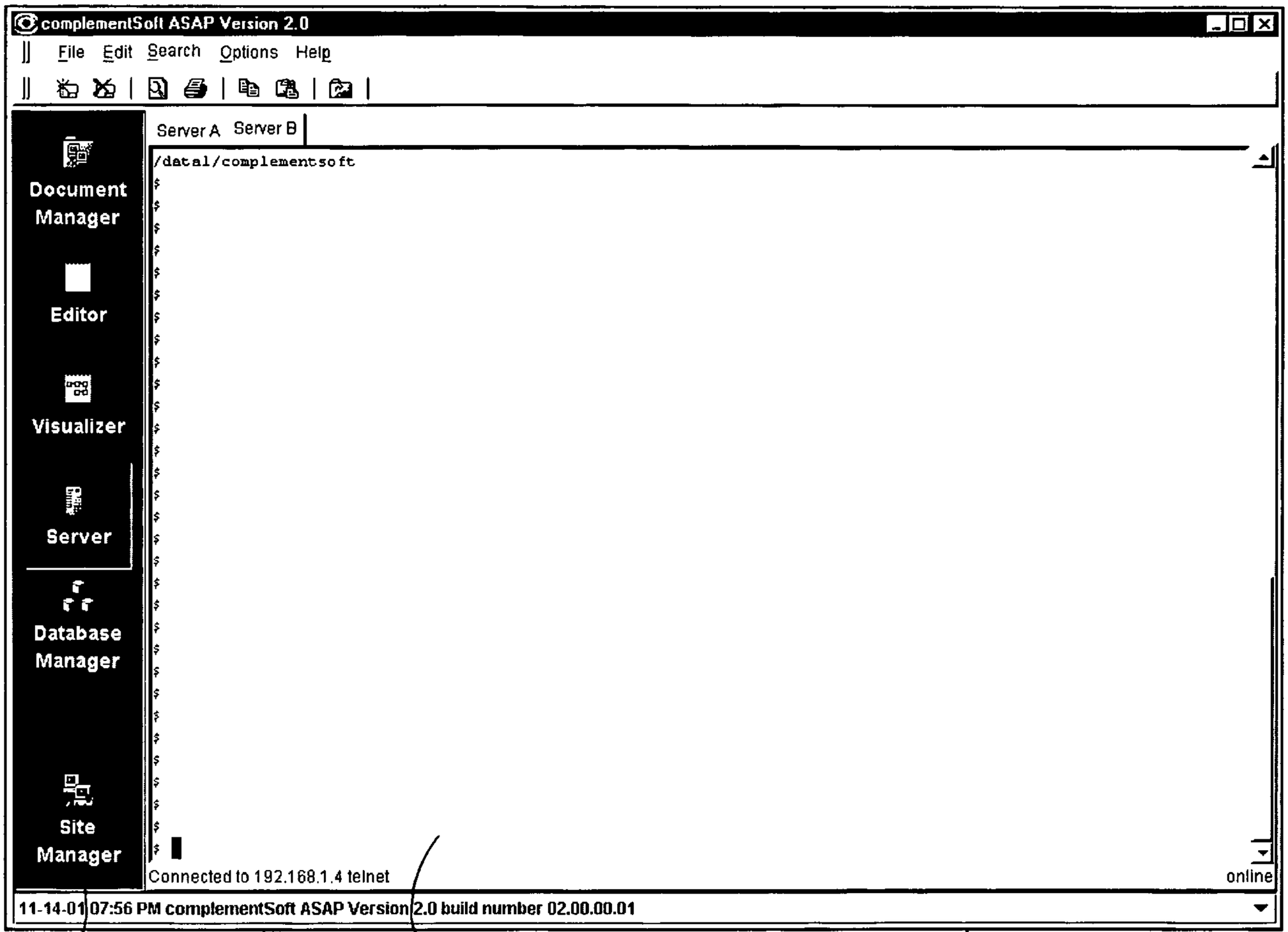

complementSoft ASAP Version 2.0
File Edit Search Options Help
Server A Server B
/data1/complementsoft
Document Manager
Editor
Visualizer
Server
Database Manager
Site Manager
Connected to 192.168.1.4 telnet
online
11-14-01 07:56 PM complementSoft ASAP Version 2.0 build number 02.00.00.01
48
162
160

FIG. 10

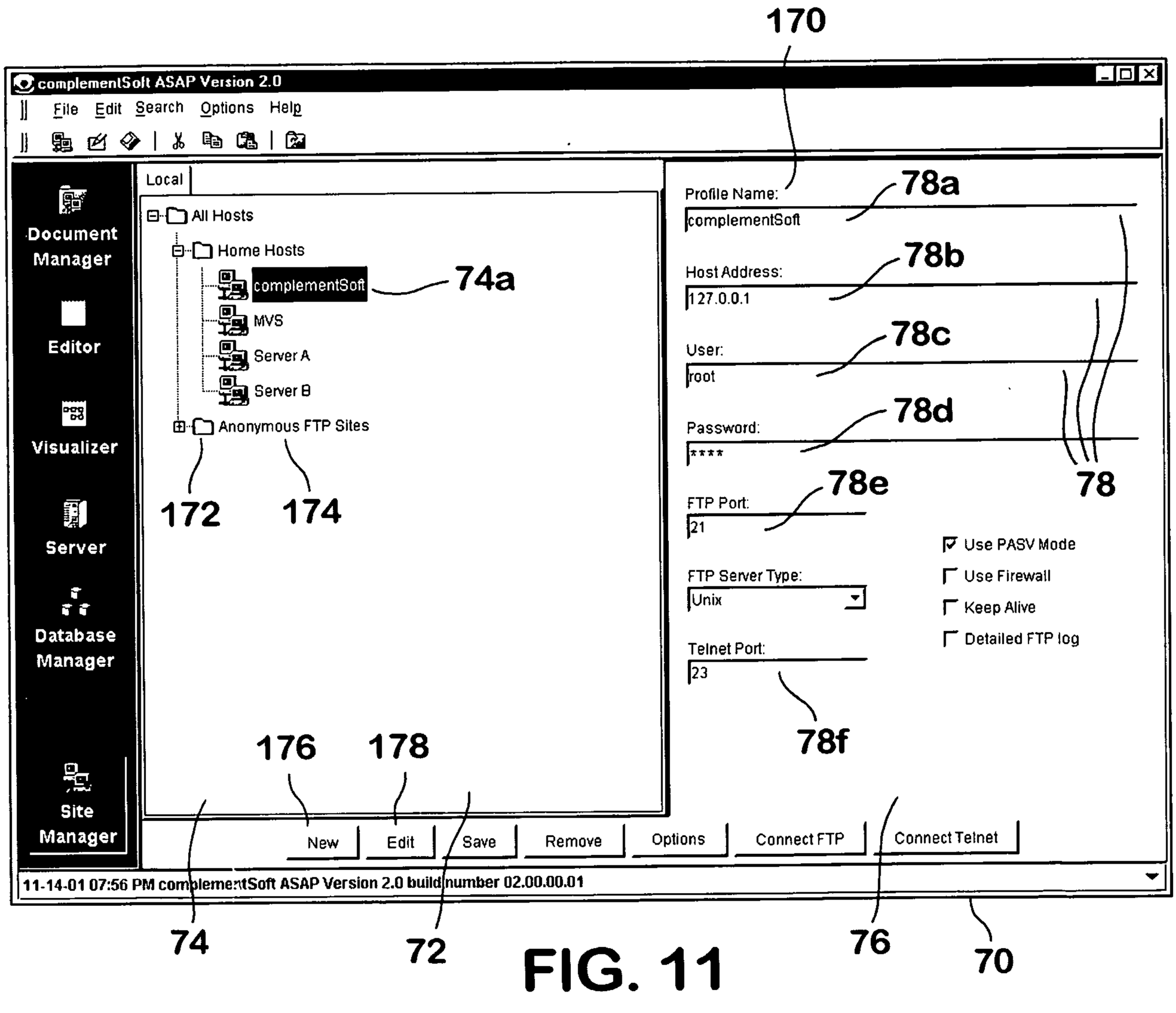

170
complementSoft ASAP Version 2.0
File Edit Search Options Help
Document Manager
Editor
Visualizer
Server
Database Manager
Site Manager
Local
All Hosts
Home Hosts
complementSoft
MVS
Server A
Server B
Anonymous FTP Sites
74a
172
174
Profile Name:
78a
complementSoft
Host Address:
78b
127.0.0.1
User:
78c
root
Password:
78d
****
78
FTP Port:
78e
21
Use PASV Mode
Use Firewall
Keep Alive
Detailed FTP log
FTP Server Type:
Unix
Telnet Port:
23
78f
176
178
New
Edit
Save
Remove
Options
Connect FTP
Connect Telnet
11-14-01 07:56 PM complementSoft ASAP Version 2.0 build number 02.00.00.01
74
72
76
70

FIG. 11

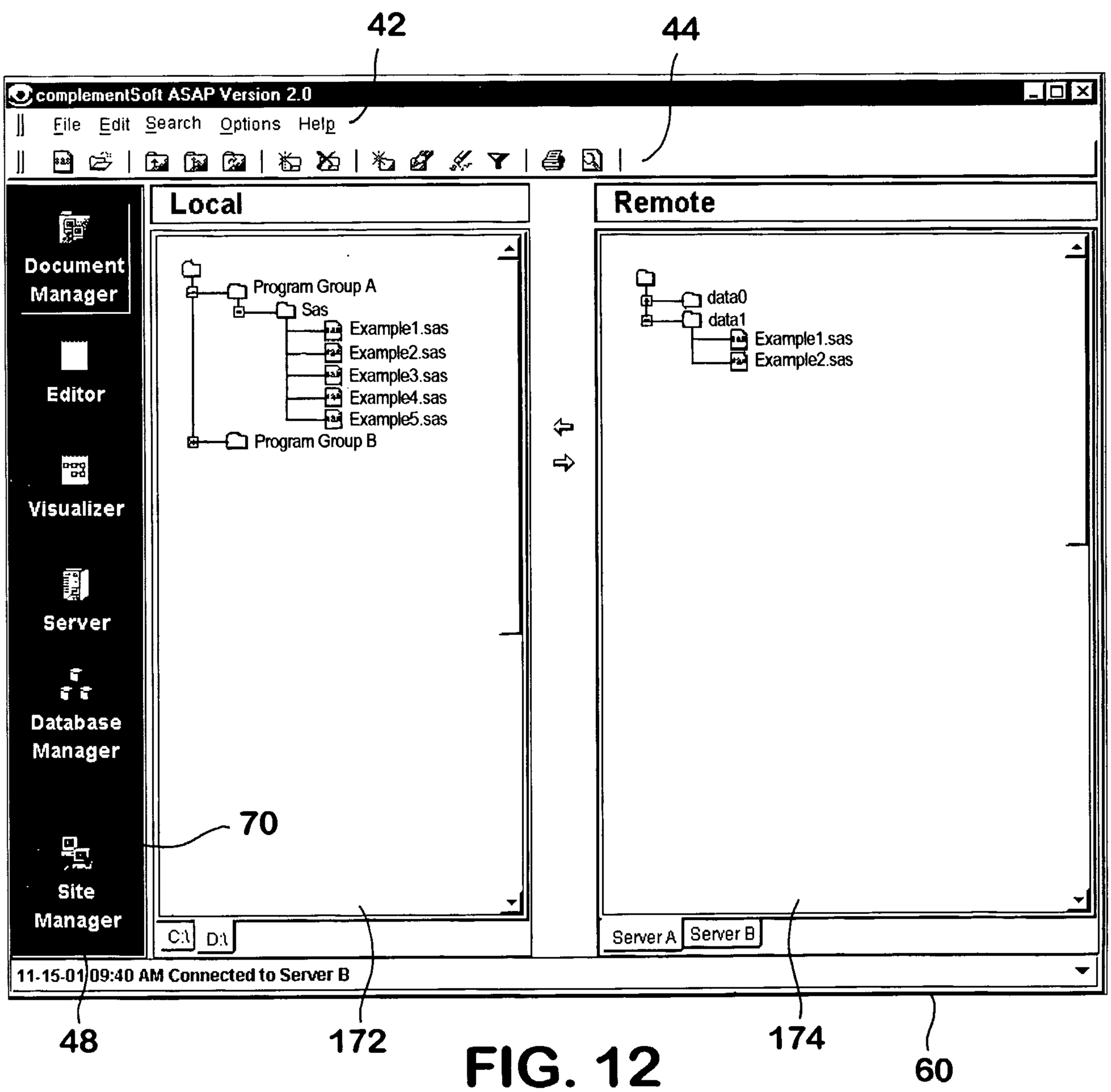

42
44
complementSoft ASAP Version 2.0
File Edit Search Options Help
Document Manager
Editor
Visualizer
Server
Database Manager
Site Manager
70
Local
Program Group A
Sas
Example1.sas
Example2.sas
Example3.sas
Example4.sas
Example5.sas
Program Group B
C:\ D:\
Remote
data0
data1
Example1.sas
Example2.sas
Server A Server B
11-15-01 09:40 AM Connected to Server B
48
172
174
60

FIG. 12

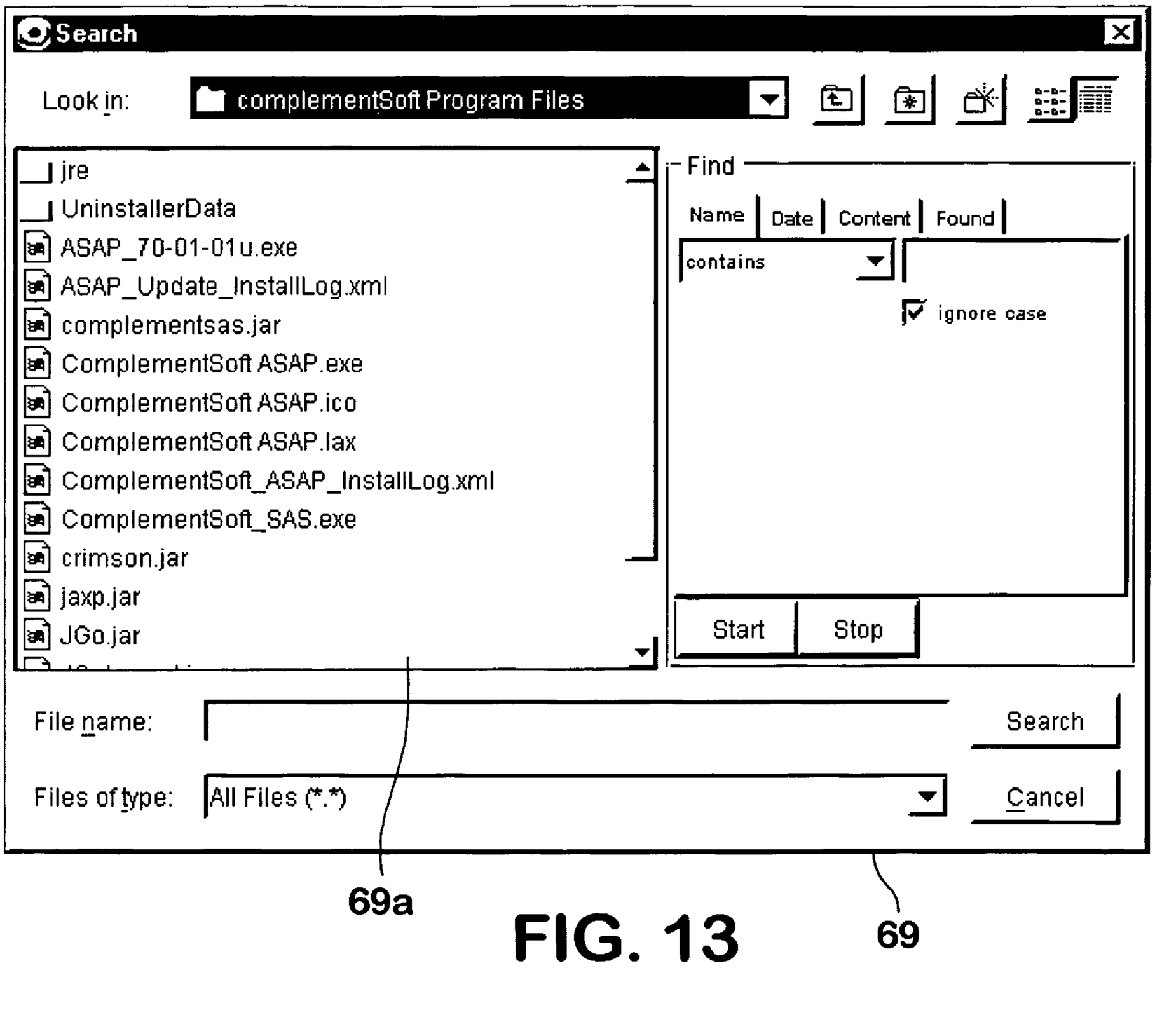

Search
Look in:
complementSoft Program Files
jre
UninstallerData
ASAP_70-01-01u.exe
ASAP_Update_InstallLog.xml
complementsas.jar
ComplementSoft ASAP.exe
ComplementSoft ASAP.ico
ComplementSoft ASAP.lax
ComplementSoft_ASAP_InstallLog.xml
ComplementSoft_SAS.exe
crimson.jar
jaxp.jar
JGo.jar
Find
Name
Date
Content
Found
contains
ignore case
Start
Stop
File name:
Search
Files of type:
All Files (*.*)
Cancel
69a
69

FIG. 13

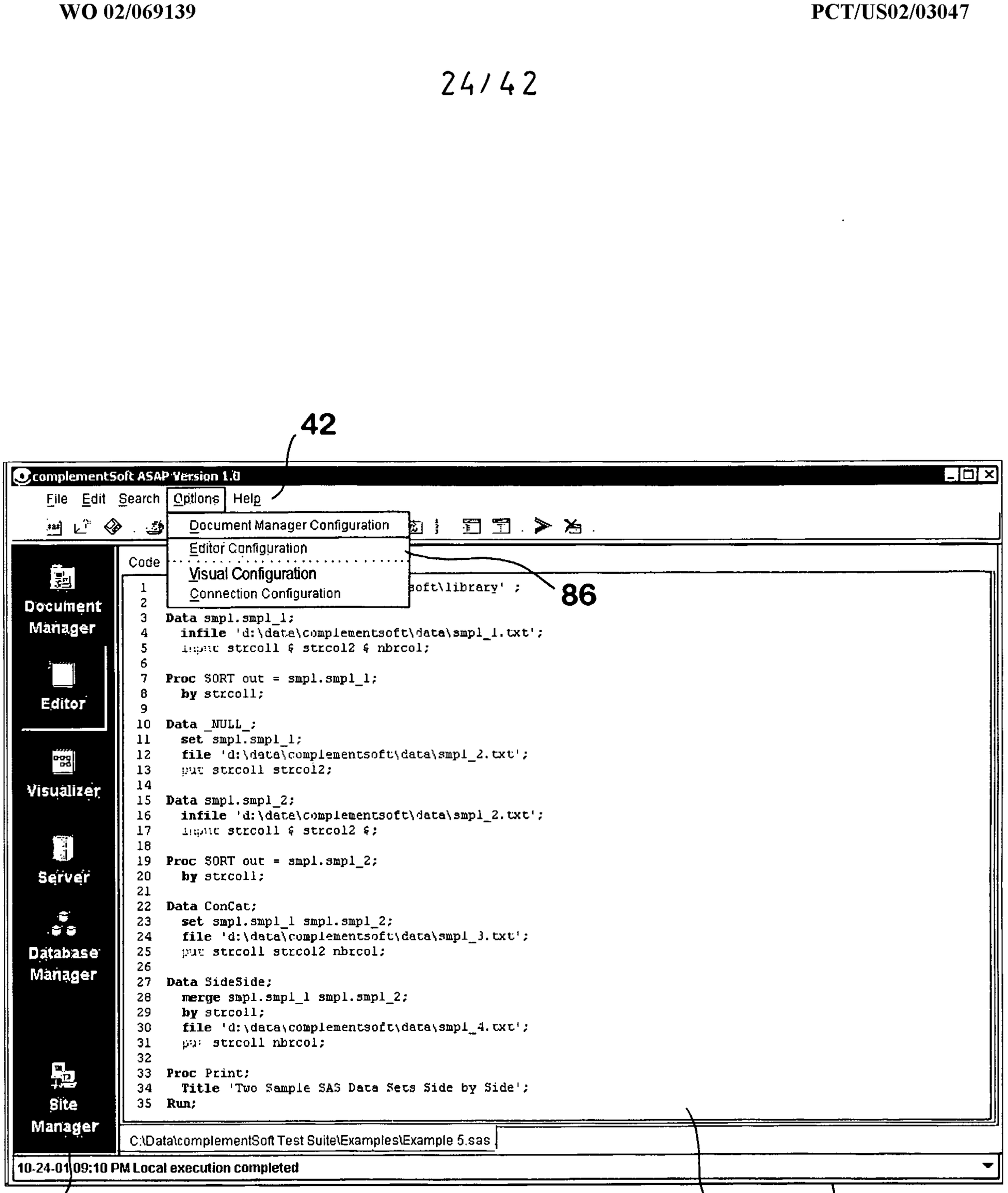

42
complementSoft ASAP Version 1.0
File Edit Search Options Help
Document Manager Configuration
Editor Configuration
Visual Configuration
Connection Configuration
86
Code
Document Manager
Editor
Visualizer
Server
Database Manager
Site Manager
1 soft\library' ;
2
3 Data smpl.smpl_1;
4 infile 'd:\data\complementsoft\data\smpl_1.txt';
5 input strcol1 $ strcol2 $ nbrcol;
6
7 Proc SORT out = smpl.smpl_1;
8 by strcol1;
9
10 Data _NULL_;
11 set smpl.smpl_1;
12 file 'd:\data\complementsoft\data\smpl_2.txt';
13 put strcol1 strcol2;
14
15 Data smpl.smpl_2;
16 infile 'd:\data\complementsoft\data\smpl_2.txt';
17 input strcol1 $ strcol2 $;
18
19 Proc SORT out = smpl.smpl_2;
20 by strcol1;
21
22 Data ConCat;
23 set smpl.smpl_1 smpl.smpl_2;
24 file 'd:\data\complementsoft\data\smpl_3.txt';
25 put strcol1 strcol2 nbrcol;
26
27 Data SideSide;
28 merge smpl.smpl_1 smpl.smpl_2;
29 by strcol1;
30 file 'd:\data\complementsoft\data\smpl_4.txt';
31 put strcol1 nbrcol;
32
33 Proc Print;
34 Title 'Two Sample SAS Data Sets Side by Side';
35 Run;
C:\Data\complementSoft Test Suite\Examples\Example 5.sas
10-24-01 09:10 PM Local execution completed
48
82
80

FIG. 14a

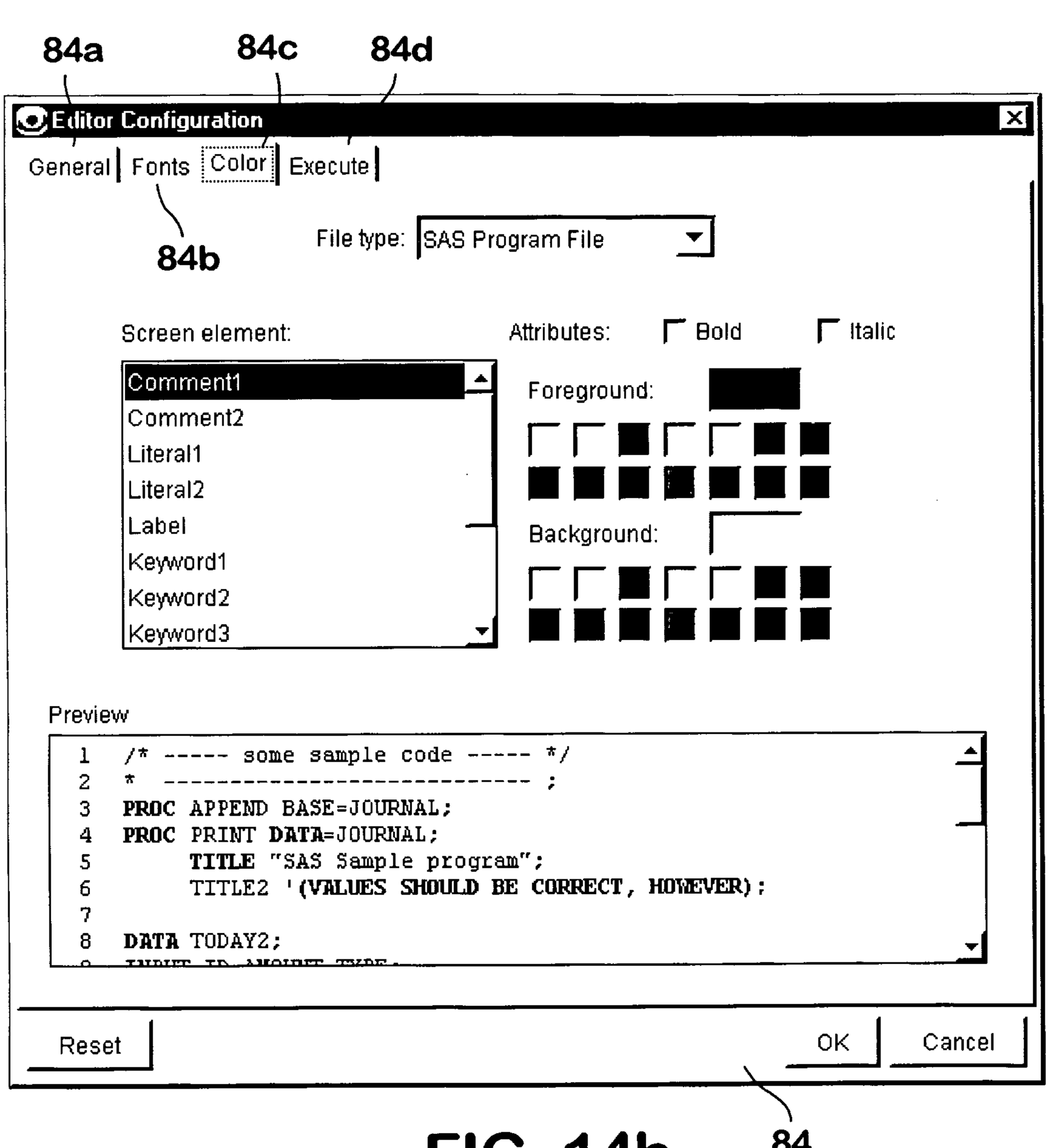

84a
84c
84d
Editor Configuration
General
Fonts
Color
Execute
84b
File type:
SAS Program File
Screen element:
Comment1
Comment2
Literal1
Literal2
Label
Keyword1
Keyword2
Keyword3
Attributes:
Bold
Italic
Foreground:
Background:
Preview
1 /* ----- some sample code ----- */
2 * ---------- ;
3 PROC APPEND BASE=JOURNAL;
4 PROC PRINT DATA=JOURNAL;
5 TITLE "SAS Sample program";
6 TITLE2 '(VALUES SHOULD BE CORRECT, HOWEVER);
7
8 DATA TODAY2;
Reset
OK
Cancel
84

FIG. 14b

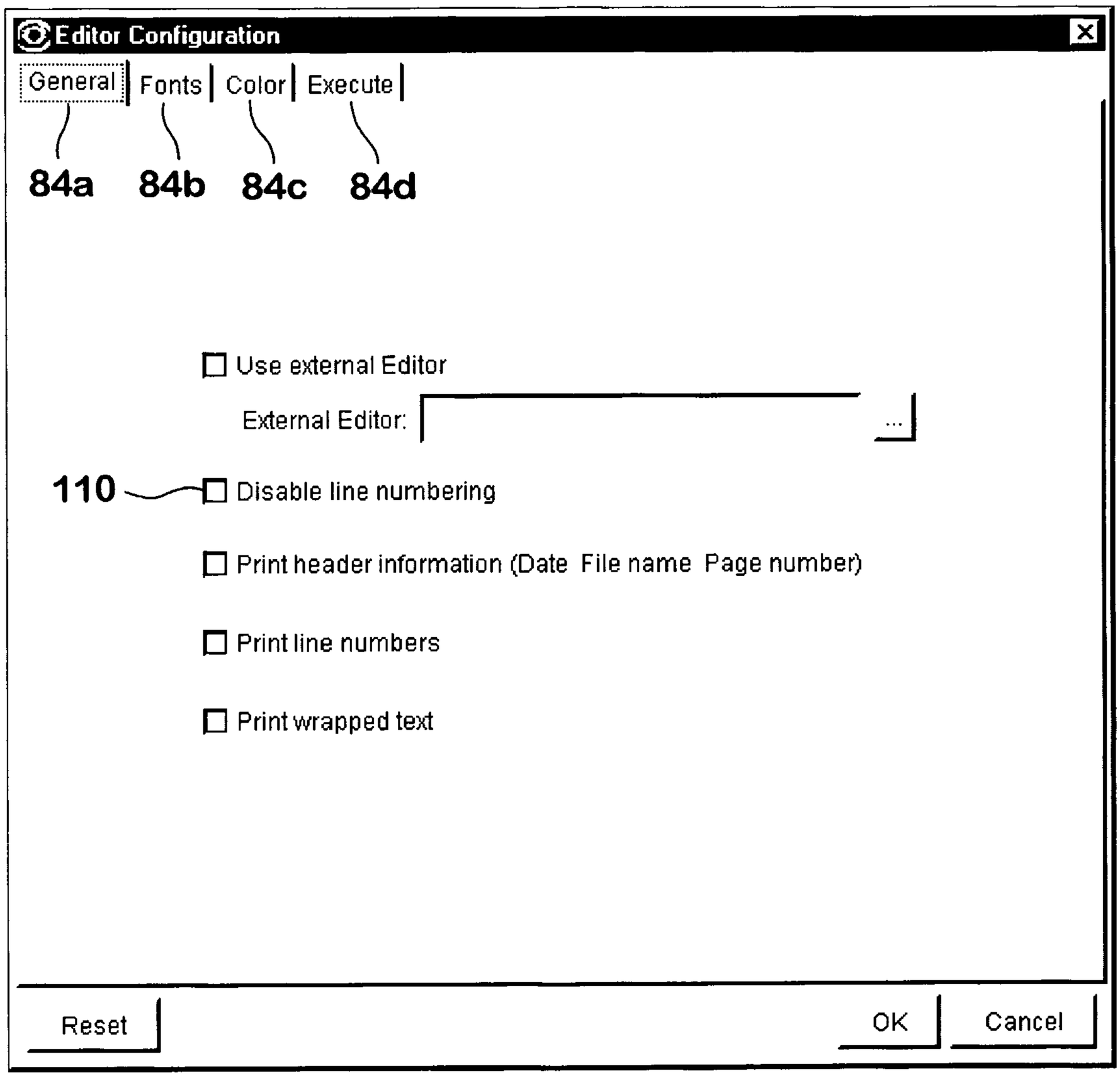

Editor Configuration
General
Fonts
Color
Execute
84a
84b
84c
84d
Use external Editor
External Editor:
...
110
Disable line numbering
Print header information (Date File name Page number)
Print line numbers
Print wrapped text
Reset
OK
Cancel

FIG. 14c

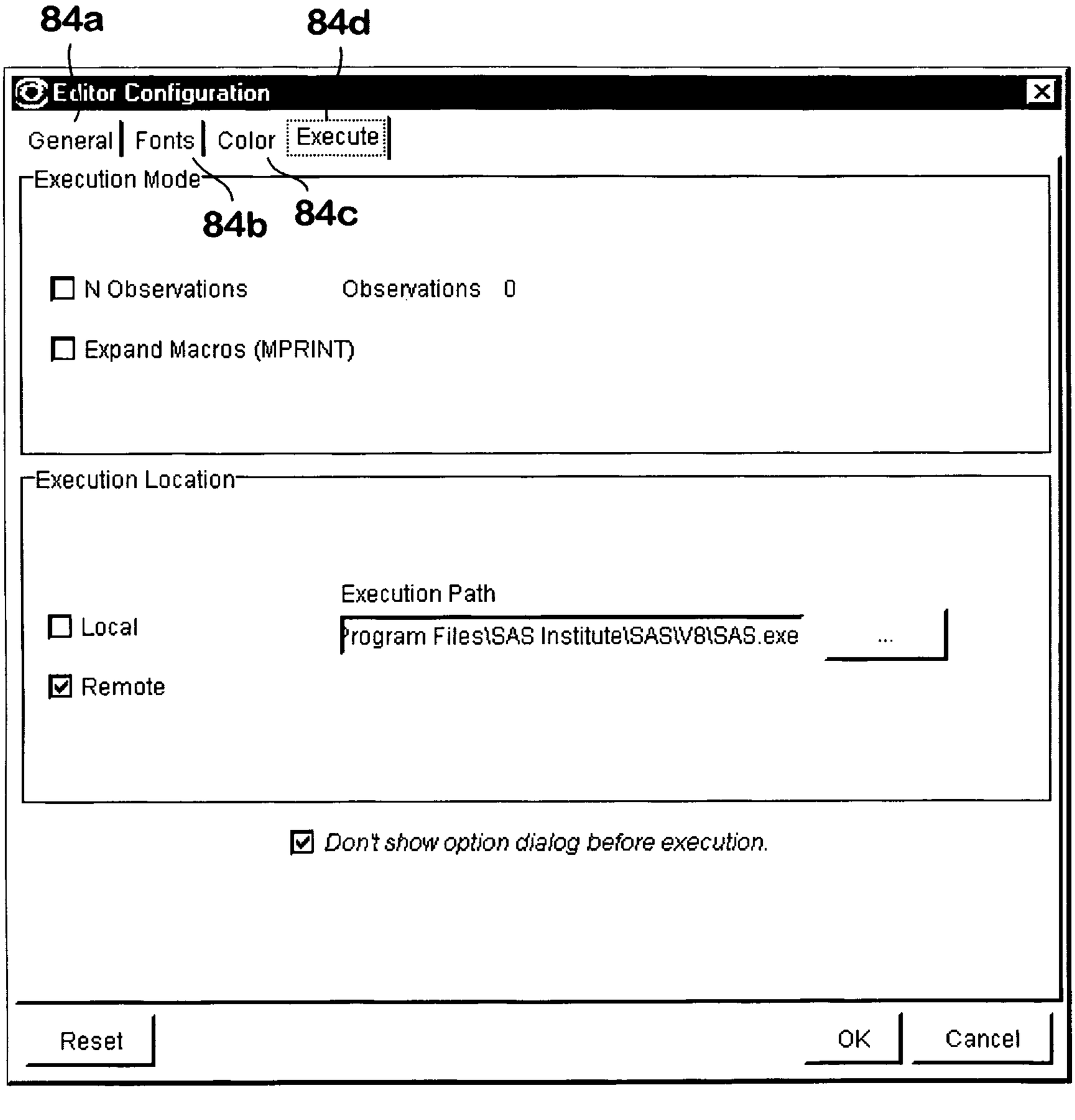

84a
84d
Editor Configuration
General
Fonts
Color
Execute
Execution Mode
84b
84c
N Observations
Observations 0
Expand Macros (MPRINT)
Execution Location
Execution Path
Local
Program Files\SAS Institute\SAS\V8\SAS.exe
...
Remote
Don't show option dialog before execution.
Reset
OK
Cancel

FIG. 14d

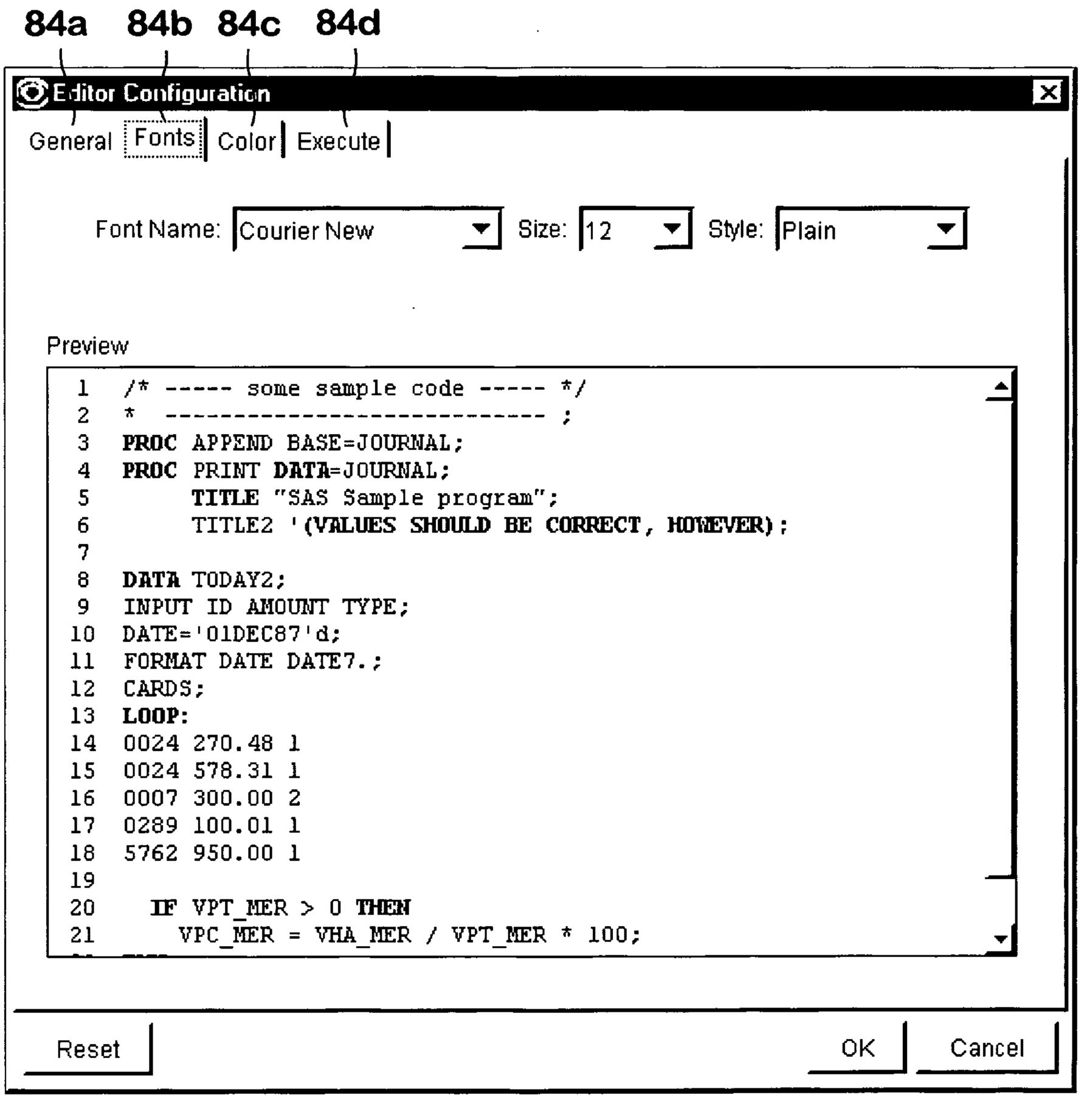

84a
84b
84c
84d
Editor Configuration
General
Fonts
Color
Execute
Font Name: Courier New
Size: 12
Style: Plain
Preview
1 /* ----- some sample code ----- */
2 * ---------- ;
3 PROC APPEND BASE=JOURNAL;
4 PROC PRINT DATA=JOURNAL;
5 TITLE "SAS Sample program";
6 TITLE2 '(VALUES SHOULD BE CORRECT, HOWEVER);
7
8 DATA TODAY2;
9 INPUT ID AMOUNT TYPE;
10 DATE='01DEC87'd;
11 FORMAT DATE DATE7.;
12 CARDS;
13 LOOP:
14 0024 270.48 1
15 0024 578.31 1
16 0007 300.00 2
17 0289 100.01 1
18 5762 950.00 1
19
20 IF VPT_MER > 0 THEN
21 VPC_MER = VHA_MER / VPT_MER * 100;
Reset
OK
Cancel

FIG. 14e

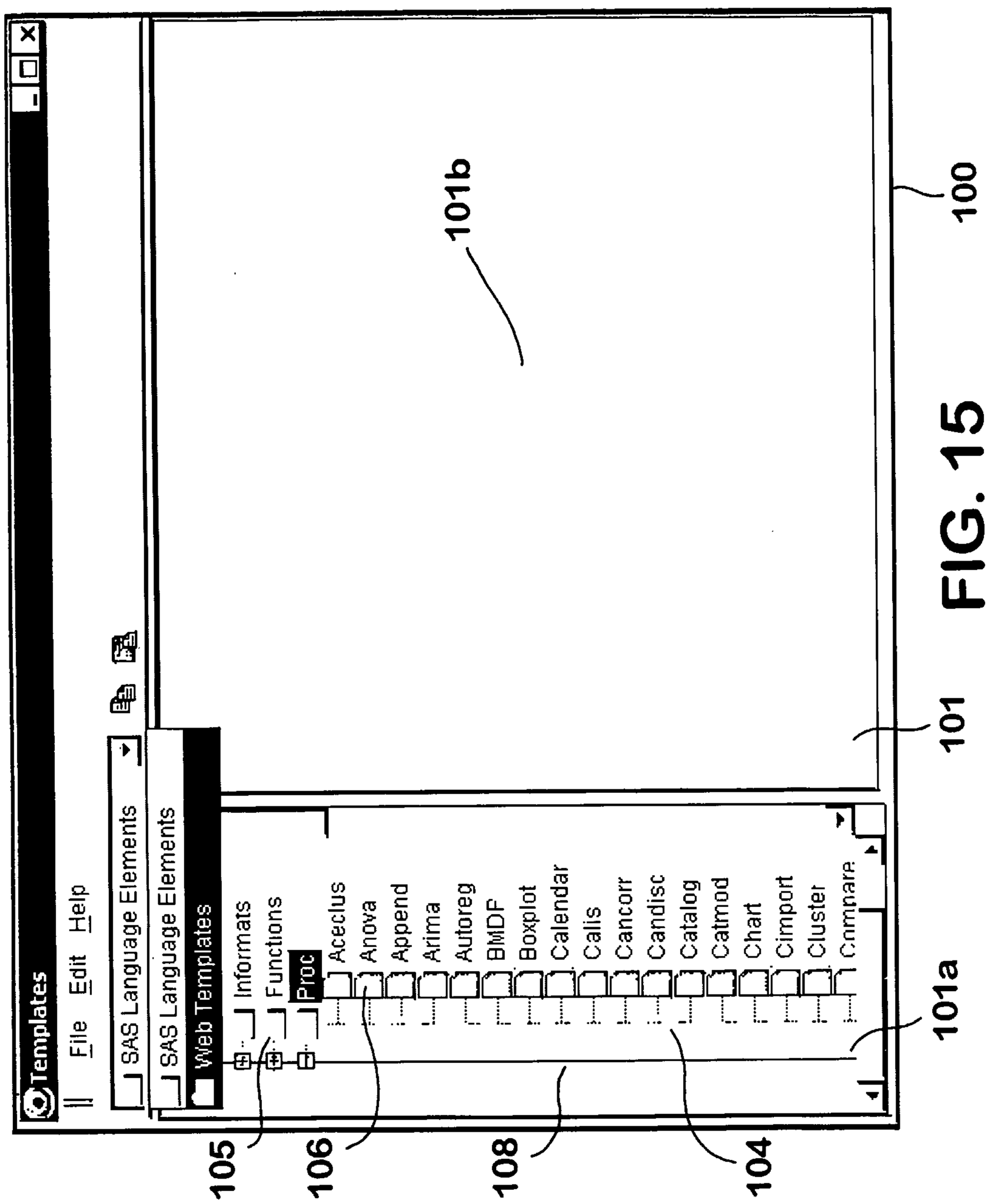

Templates
File Edit Help
SAS Language Elements
SAS Language Elements
Web Templates
Informats
Functions
Proc
Aceclus
Anova
Append
Arima
Autoreg
BMDF
Boxplot
Calendar
Calis
Cancorr
Candisc
Catalog
Catmod
Chart
Cimport
Cluster
Cnmpare
101b
100
101
101a
105
106
108
104

FIG. 15

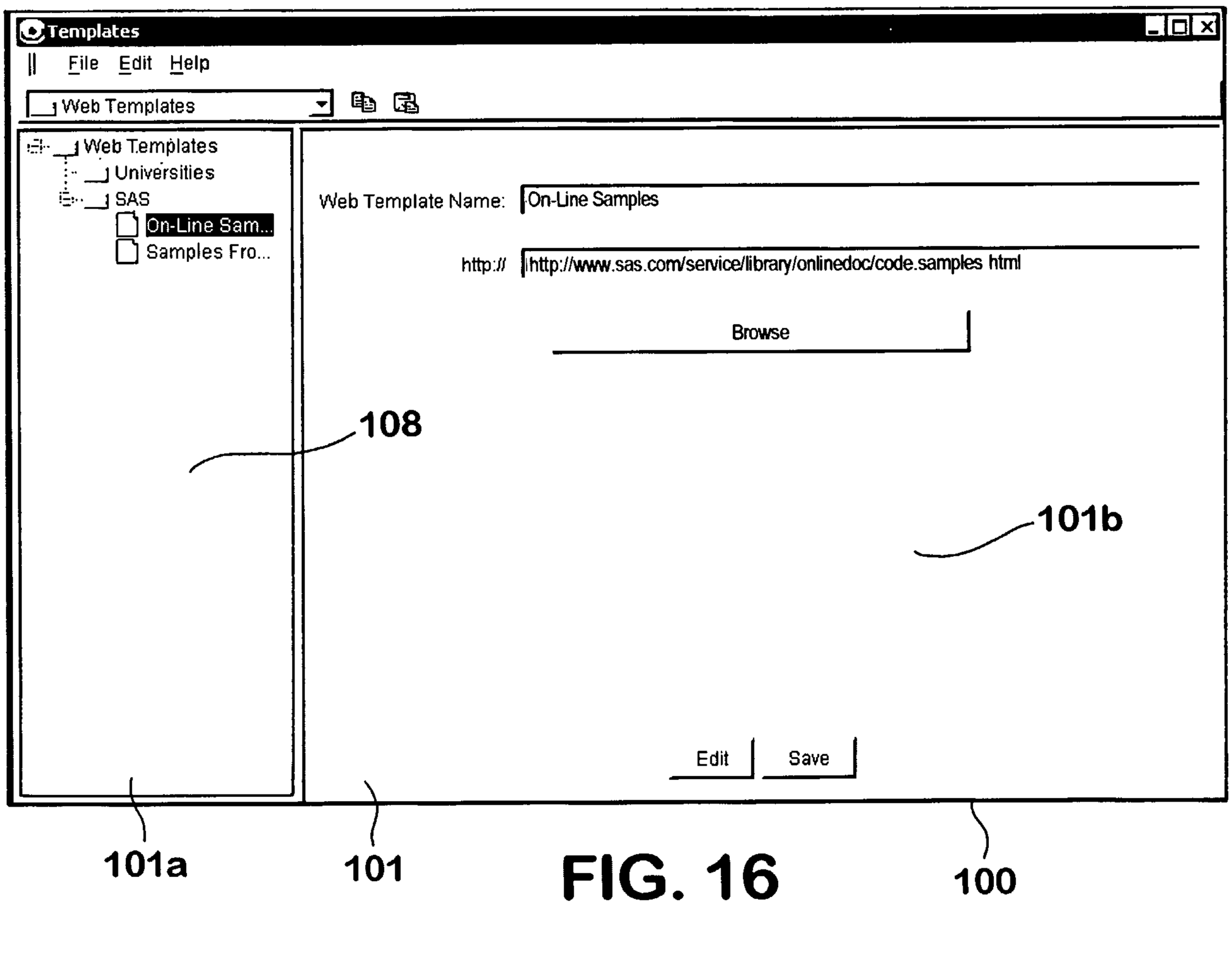

Templates
File Edit Help
Web Templates
Web Templates
Universities
SAS
On-Line Sam...
Samples Fro...
Web Template Name:
On-Line Samples
http://
http://www.sas.com/service/library/onlinedoc/code.samples html
Browse
108
101b
Edit
Save
101a
101
100

FIG. 16

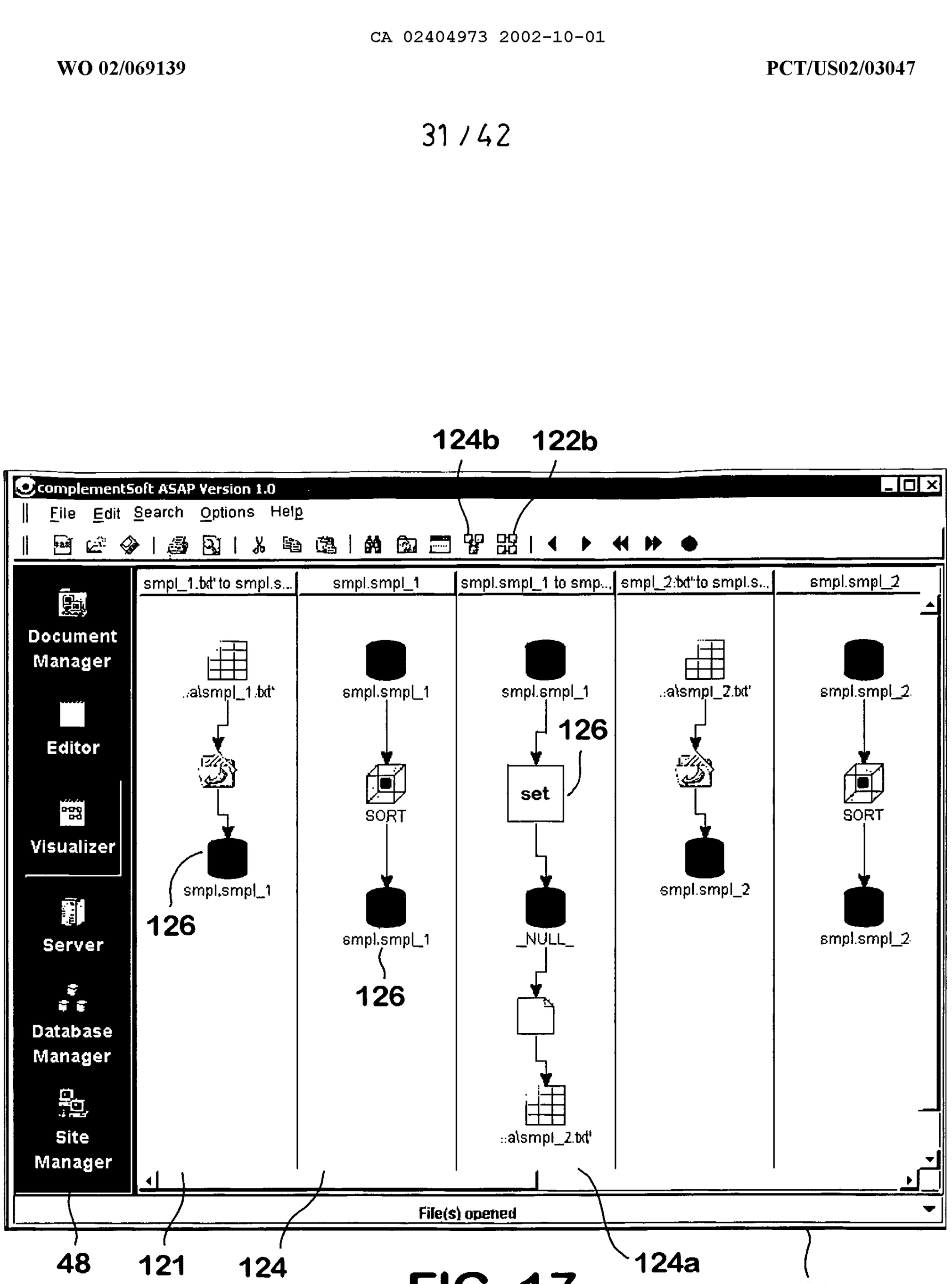

124b
122b
complementSoft ASAP Version 1.0
File Edit Search Options Help
Document Manager
Editor
Visualizer
Server
Database Manager
Site Manager
smpl_1.txt' to smpl.s...
smpl.smpl_1
smpl.smpl_1 to smp...
smpl_2.txt' to smpl.s...
smpl.smpl_2
..a\smpl_1.txt'
smpl.smpl_1
smpl.smpl_1
..a\smpl_2.txt'
smpl.smpl_2
126
set
SORT
SORT
smpl.smpl_1
smpl.smpl_1
smpl.smpl_2
smpl.smpl_2
126
126
NULL
..a\smpl_2.txt'
File(s) opened
48
121
124
124a
120

FIG. 17

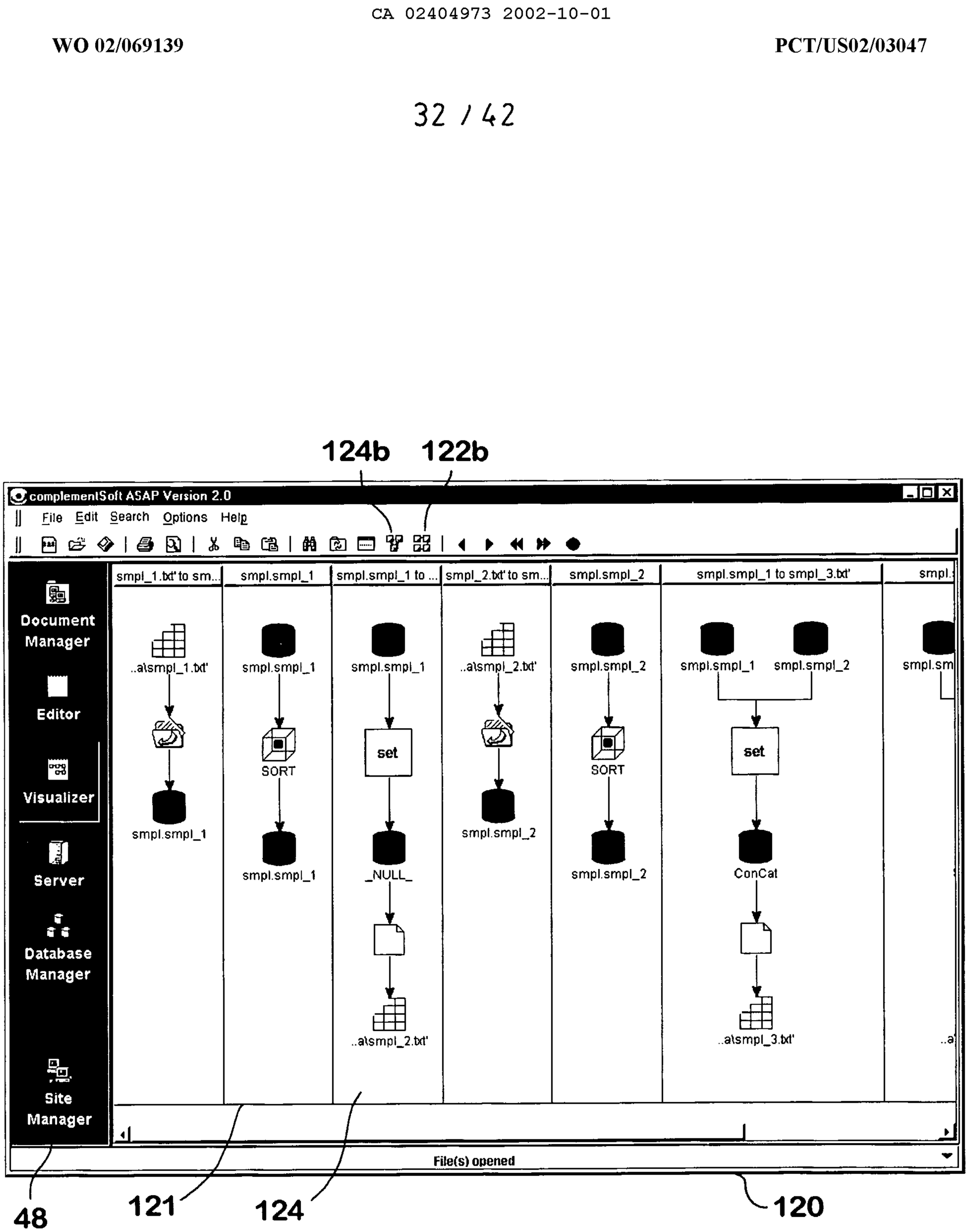

124b
122b
complementSoft ASAP Version 2.0
File Edit Search Options Help
Document Manager
Editor
Visualizer
Server
Database Manager
Site Manager
smpl_1.txt' to sm...
smpl.smpl_1
smpl.smpl_1 to ...
smpl_2.txt' to sm...
smpl.smpl_2
smpl.smpl_1 to smpl_3.txt'
..a\smpl_1.txt'
smpl.smpl_1
SORT
set
NULL
..a\smpl_2.txt'
ConCat
..a\smpl_3.txt'
File(s) opened
48
121
124
120

FIG. 18a

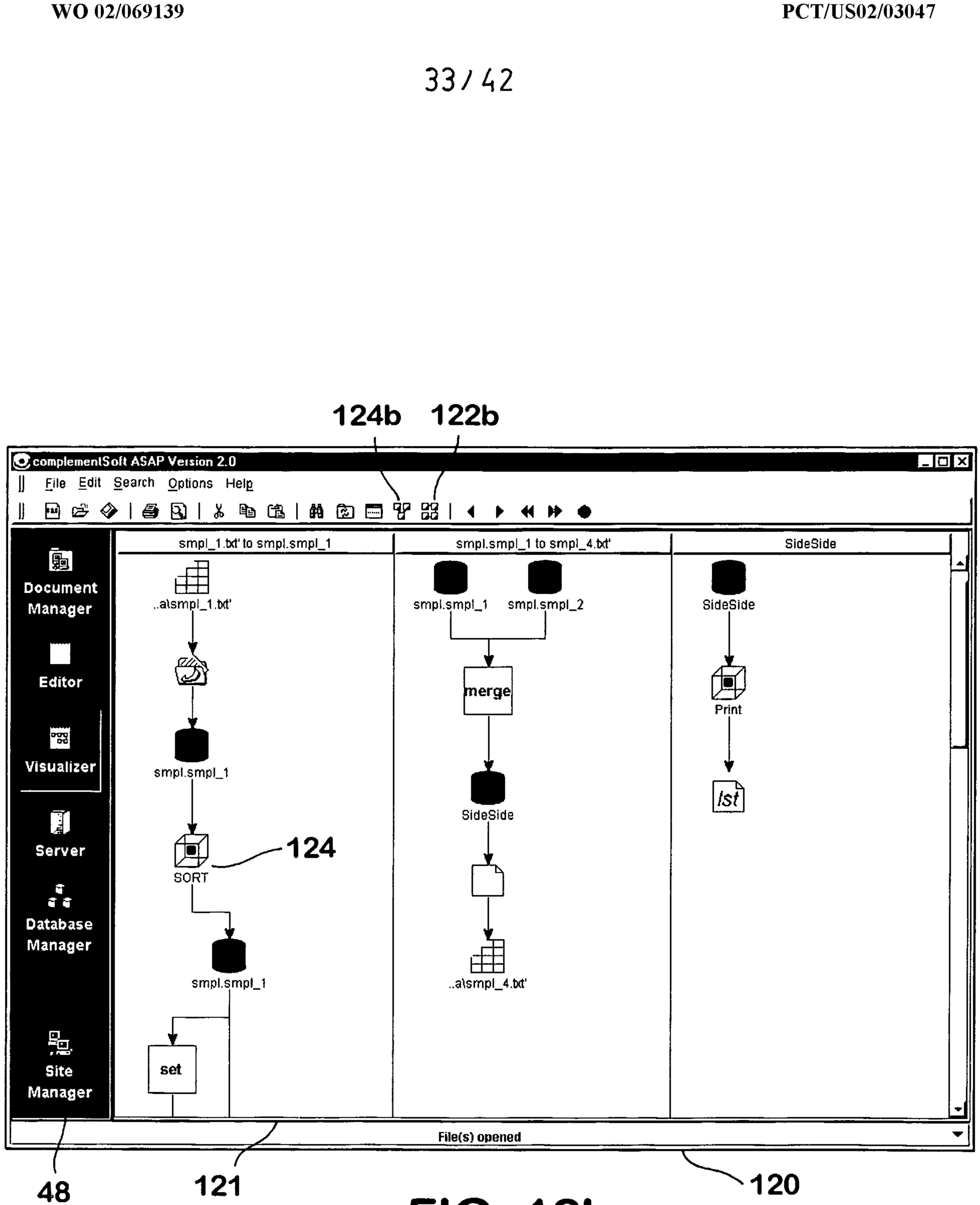

124b
122b
complementSoft ASAP Version 2.0
File Edit Search Options Help
Document Manager
Editor
Visualizer
Server
Database Manager
Site Manager
smpl_1.txt' to smpl.smpl_1
smpl.smpl_1 to smpl_4.txt'
SideSide
..a\smpl_1.txt'
smpl.smpl_1
SORT
124
smpl.smpl_1
set
smpl.smpl_1
smpl.smpl_2
merge
SideSide
..a\smpl_4.txt'
SideSide
Print
lst
File(s) opened
48
121
120

FIG. 18b

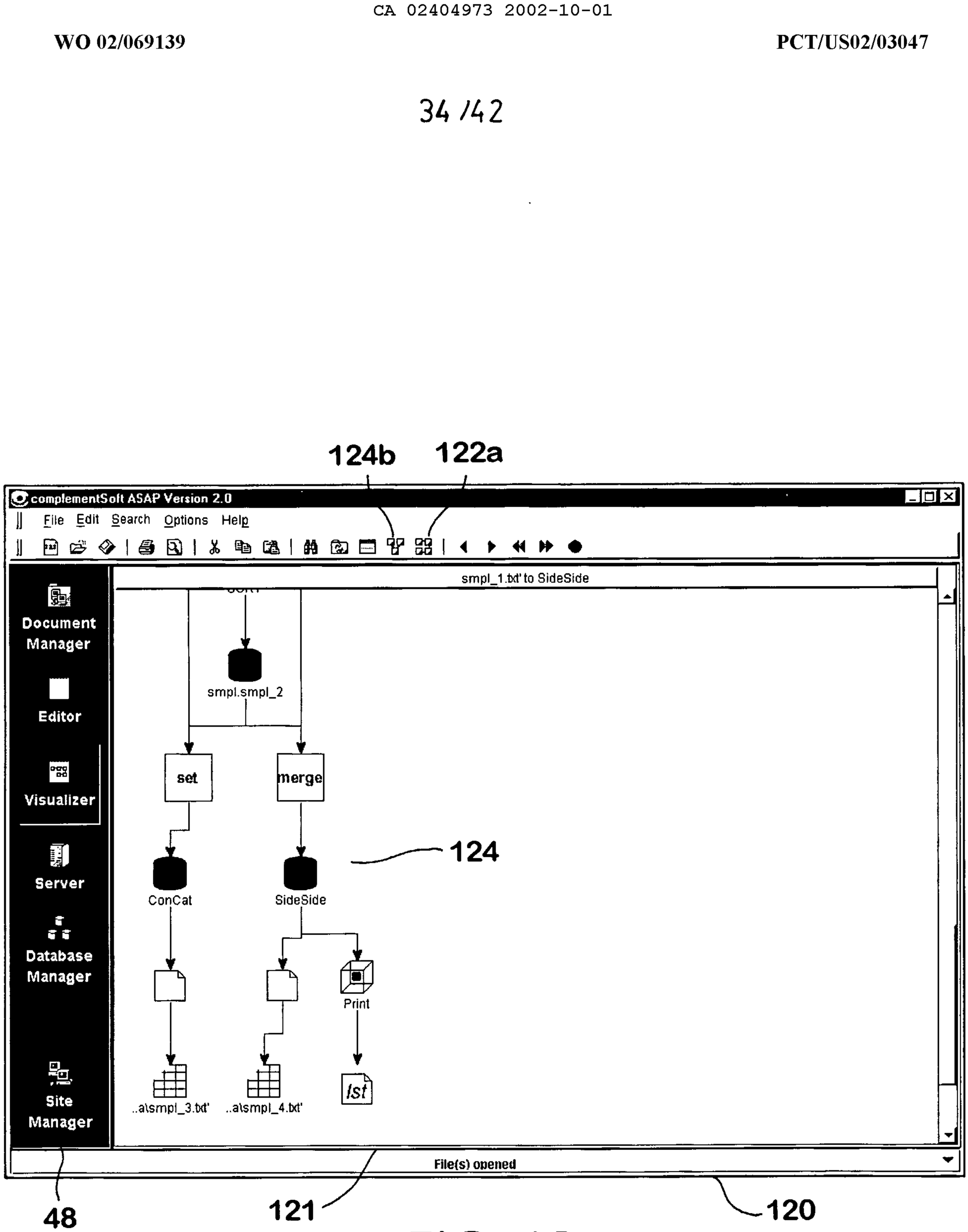

124b
122a
complementSoft ASAP Version 2.0
File Edit Search Options Help
Document Manager
Editor
Visualizer
Server
Database Manager
Site Manager
smpl_1.txt' to SideSide
smpl.smpl_2
set
merge
ConCat
SideSide
Print
..a\smpl_3.txt'
..a\smpl_4.txt'
lst
124
File(s) opened
48
121
120

FIG. 18c

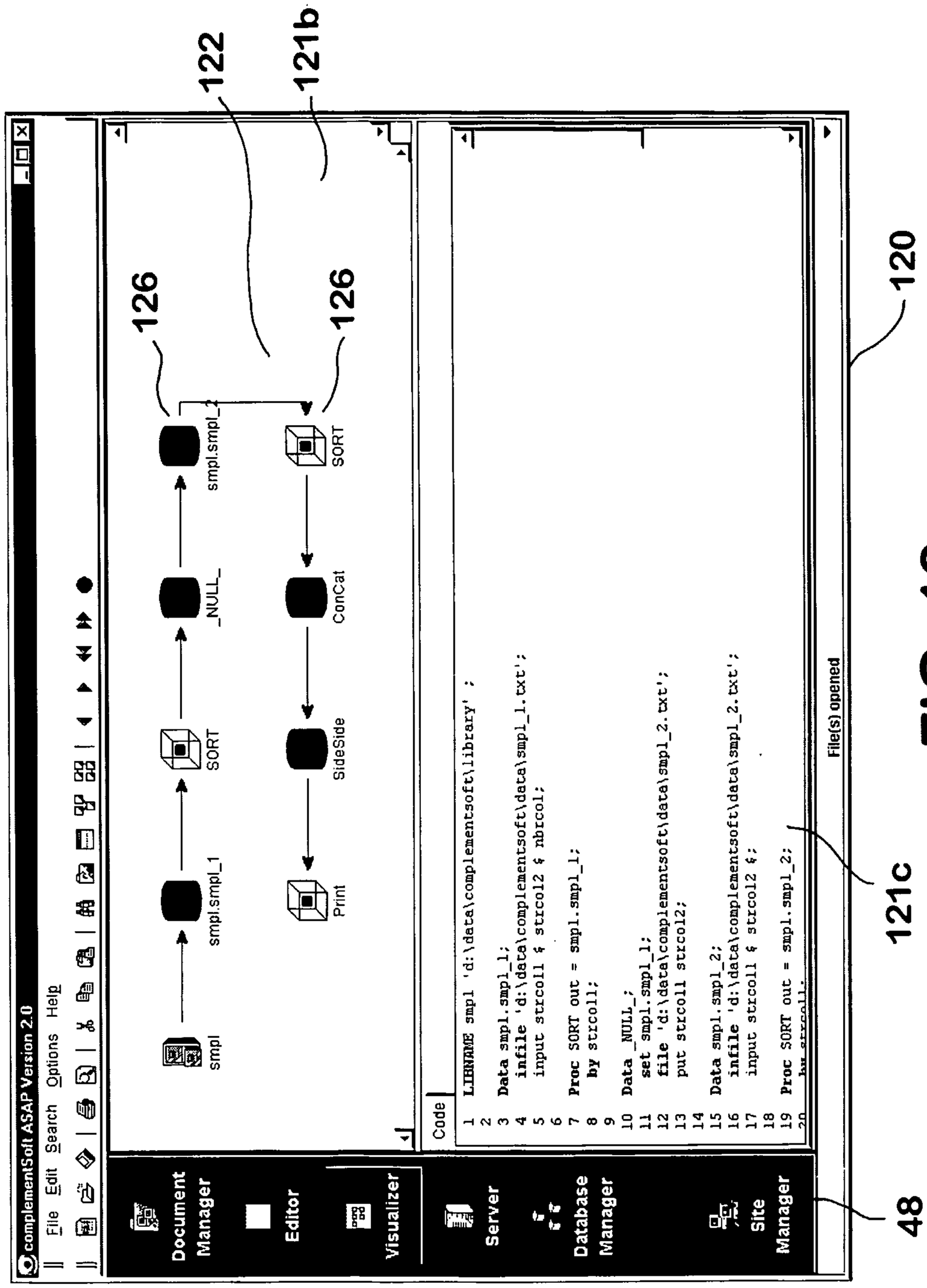

122
121b
126
126
120
smpl.smpl_2
SORT
NULL
ConCat
smpl
smpl.smpl_1
SORT
Print
SideSide
complementSoft ASAP Version 2.0
File Edit Search Options Help
Code
1 LIBNAME smpl 'd:\data\complementsoft\library' ;
2
3 Data smpl.smpl_1;
4 infile 'd:\data\complementsoft\data\smpl_1.txt';
5 input strcol1 $ strcol2 $ nbrcol;
6
7 Proc SORT out = smpl.smpl_1;
8 by strcol1;
9
10 Data _NULL_;
11 set smpl.smpl_1;
12 file 'd:\data\complementsoft\data\smpl_2.txt';
13 put strcol1 strcol2;
14
15 Data smpl.smpl_2;
16 infile 'd:\data\complementsoft\data\smpl_2.txt';
17 input strcol1 $ strcol2 $;
18
19 Proc SORT out = smpl.smpl_2;
20 by strcol1;
File(s) opened
Document Manager
Editor
Visualizer
Server
Database Manager
Site Manager
121c
48

FIG. 19

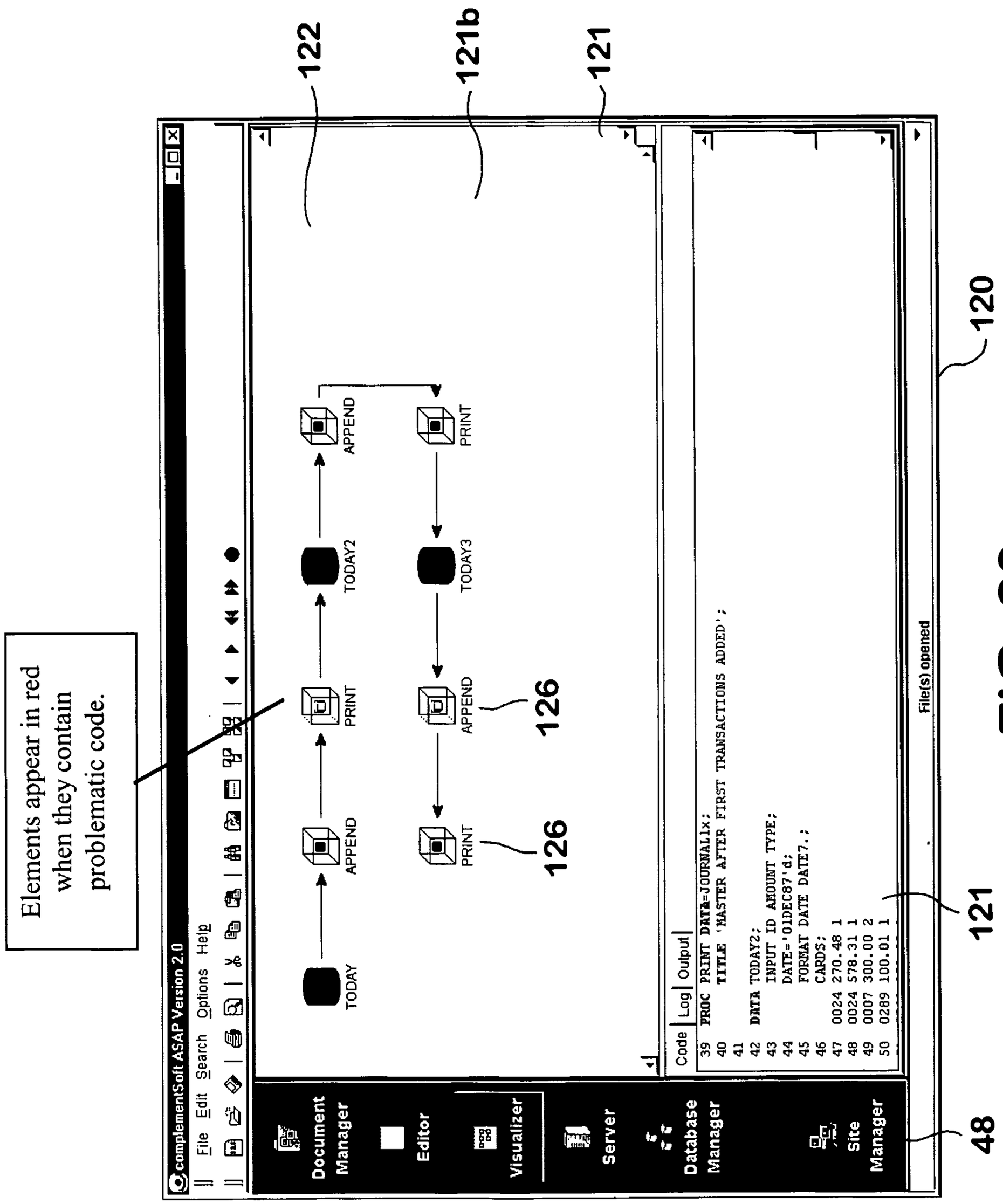

Elements appear in red when they contain problematic code.
22
21b
21
120
complementSoft ASAP Version 2.0
File Edit Search Options Help
Document Manager
Editor
Visualizer
Server
Database Manager
Site Manager
TODAY
APPEND
PRINT
TODAY2
PRINT
APPEND
TODAY3
126
126
Code Log Output
39 PROC PRINT DATA=JOURNAL1x;
40 TITLE 'MASTER AFTER FIRST TRANSACTIONS ADDED';
41
42 DATA TODAY2;
43 INPUT ID AMOUNT TYPE;
44 DATE='01DEC87'd;
45 FORMAT DATE DATE7.;
46 CARDS;
47 0024 270.48 1
48 0024 578.31 1
49 0007 300.00 2
50 0289 100.01 1
File(s) opened
121
48

FIG. 20

```
Code & Log
 39  PROC PRINT DATA=JOURNAL1x;
 40       TITLE 'MASTER AFTER FIRST TRANSACTIONS ADDED';
 41
 42  DATA TODAY2;
 43       INPUT ID AMOUNT TYPE;
 44       DATE='01DEC87'd;
 45       FORMAT DATE DATE7.;
 46       CARDS;
 47  0024 270.48 1
 48  0024 578.31 1
 49  0027 288.99 2

38          PROC APPEND BASE=JOURNAL;

NOTE: Appending WORK.TODAY to WORK.JOURNAL.
NOTE: BASE data set does not exist. DATA file is being copied to BASE file.
□2                                        The SAS System          08:22 Friday,

NOTE: There were 5 observations read from the dataset WORK.TODAY.
NOTE: The data set WORK.JOURNAL has 5 observations and 4 variables.
NOTE: PROCEDURE APPEND used:
      real time           2.14 seconds


39          PROC PRINT DATA=JOURNAL1x;
ERROR: File WORK.JOURNAL1X.DATA does not exist.
```

FIG. 20a

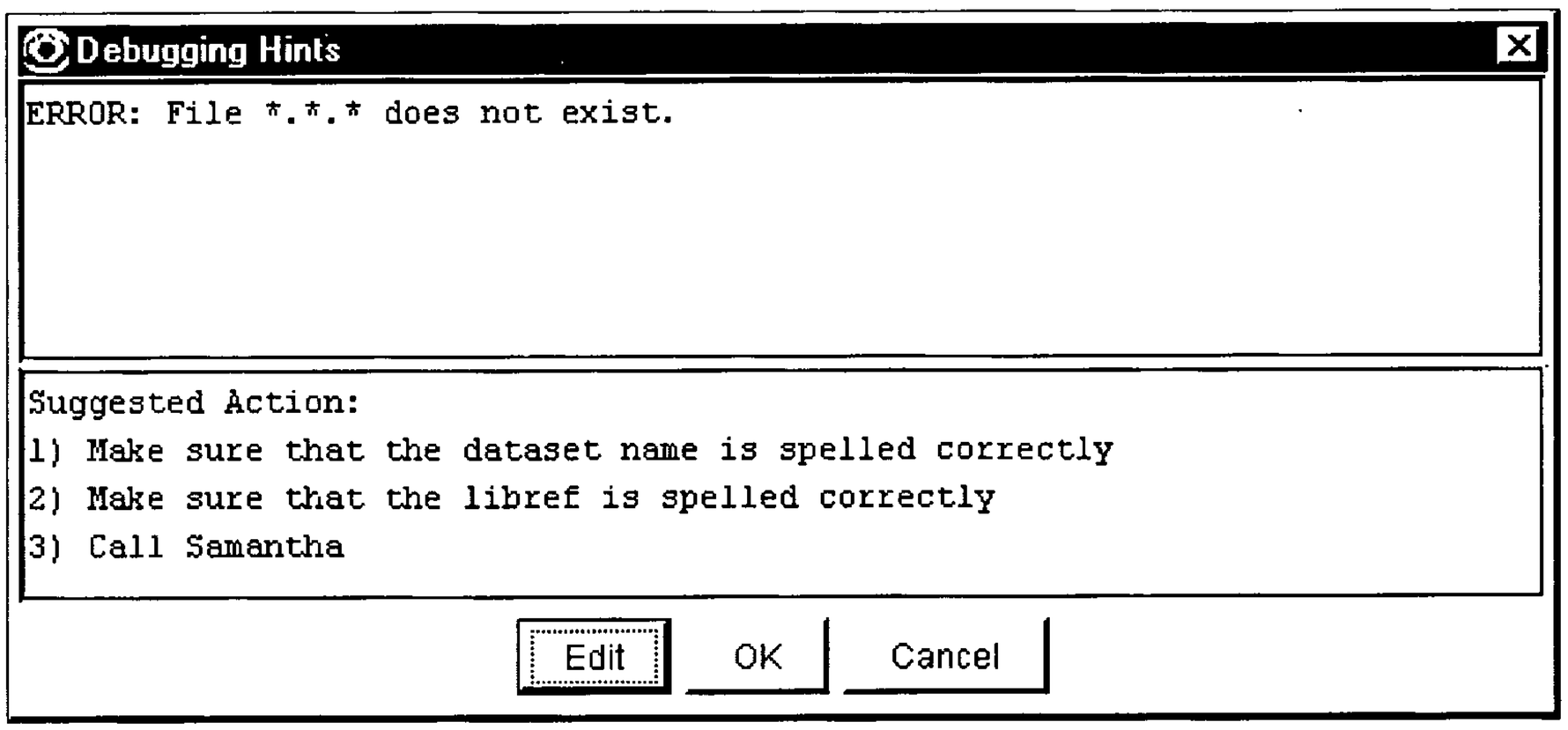

Debugging Hints
ERROR: File *.*.* does not exist.
Suggested Action:
1) Make sure that the dataset name is spelled correctly
2) Make sure that the libref is spelled correctly
3) Call Samantha
Edit
OK
Cancel

FIG. 20b

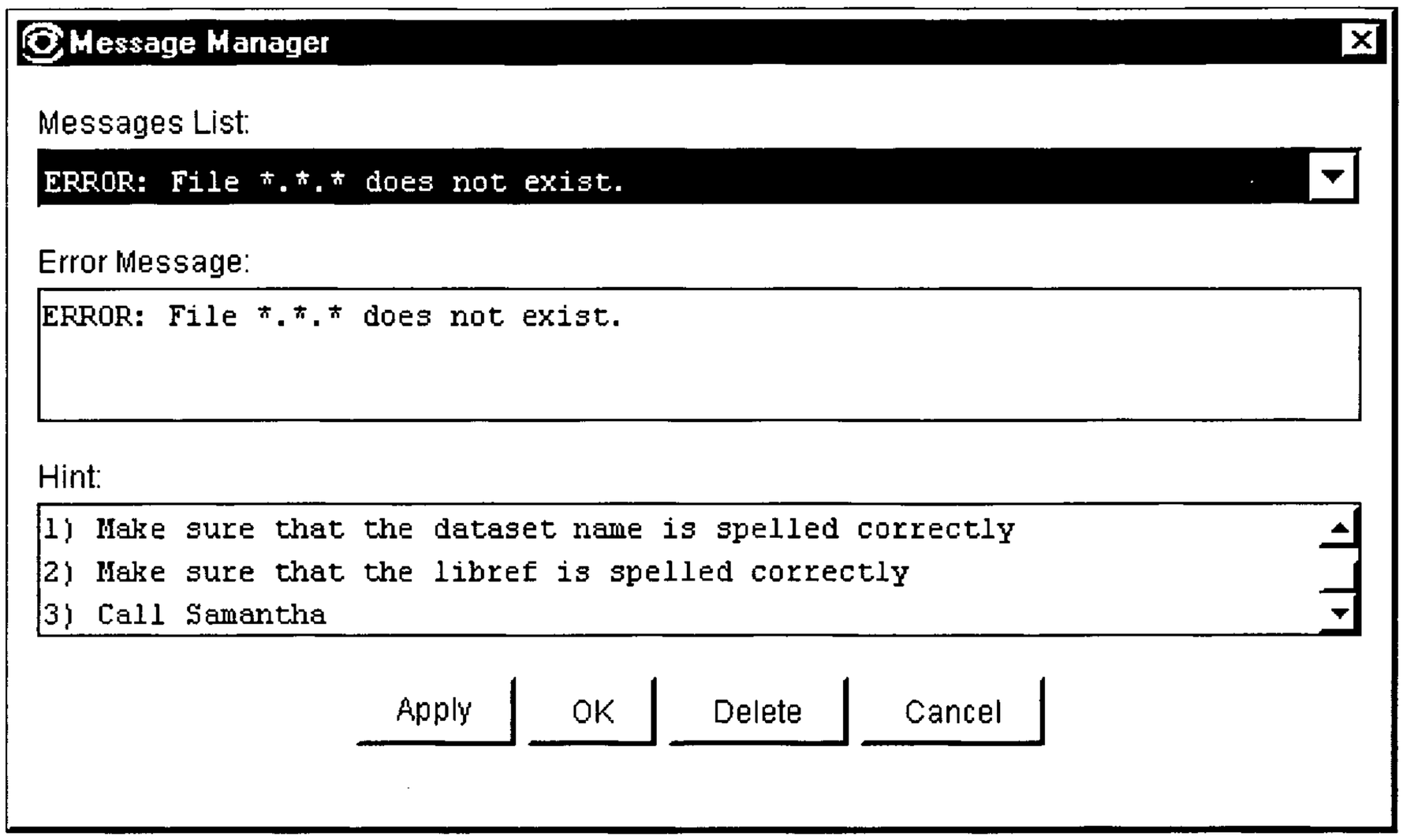

Message Manager
Messages List:
ERROR: File *.*.* does not exist.
Error Message:
ERROR: File *.*.* does not exist.
Hint:
1) Make sure that the dataset name is spelled correctly
2) Make sure that the libref is spelled correctly
3) Call Samantha
Apply
OK
Delete
Cancel

FIG. 20c

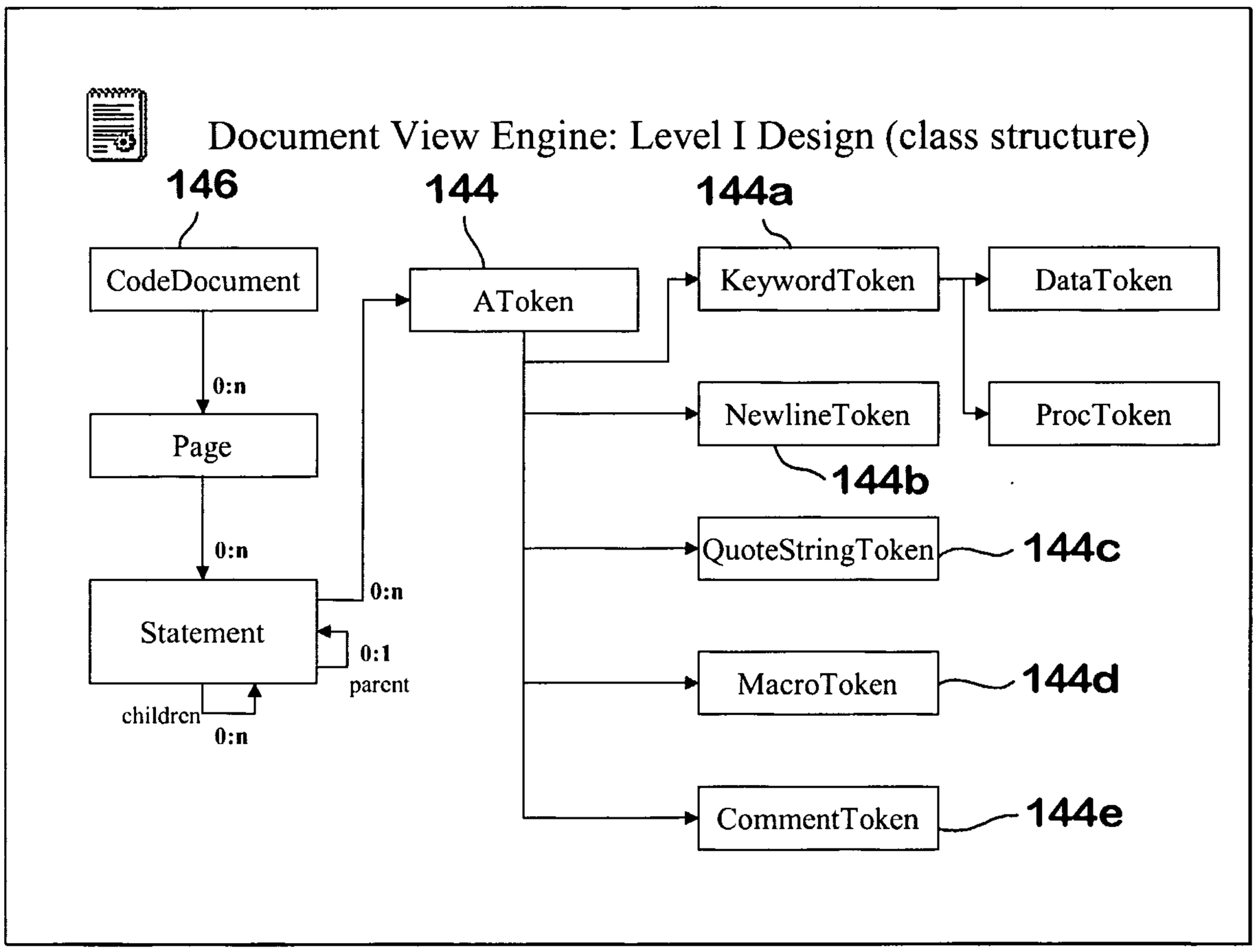

Document View Engine: Level I Design (class structure)
146
144
144a
CodeDocument
AToken
KeywordToken
DataToken
0:n
Page
NewlineToken
ProcToken
144b
0:n
QuoteStringToken
144c
0:n
Statement
0:1
parent
MacroToken
144d
children
0:n
CommentToken
144e

FIG. 21

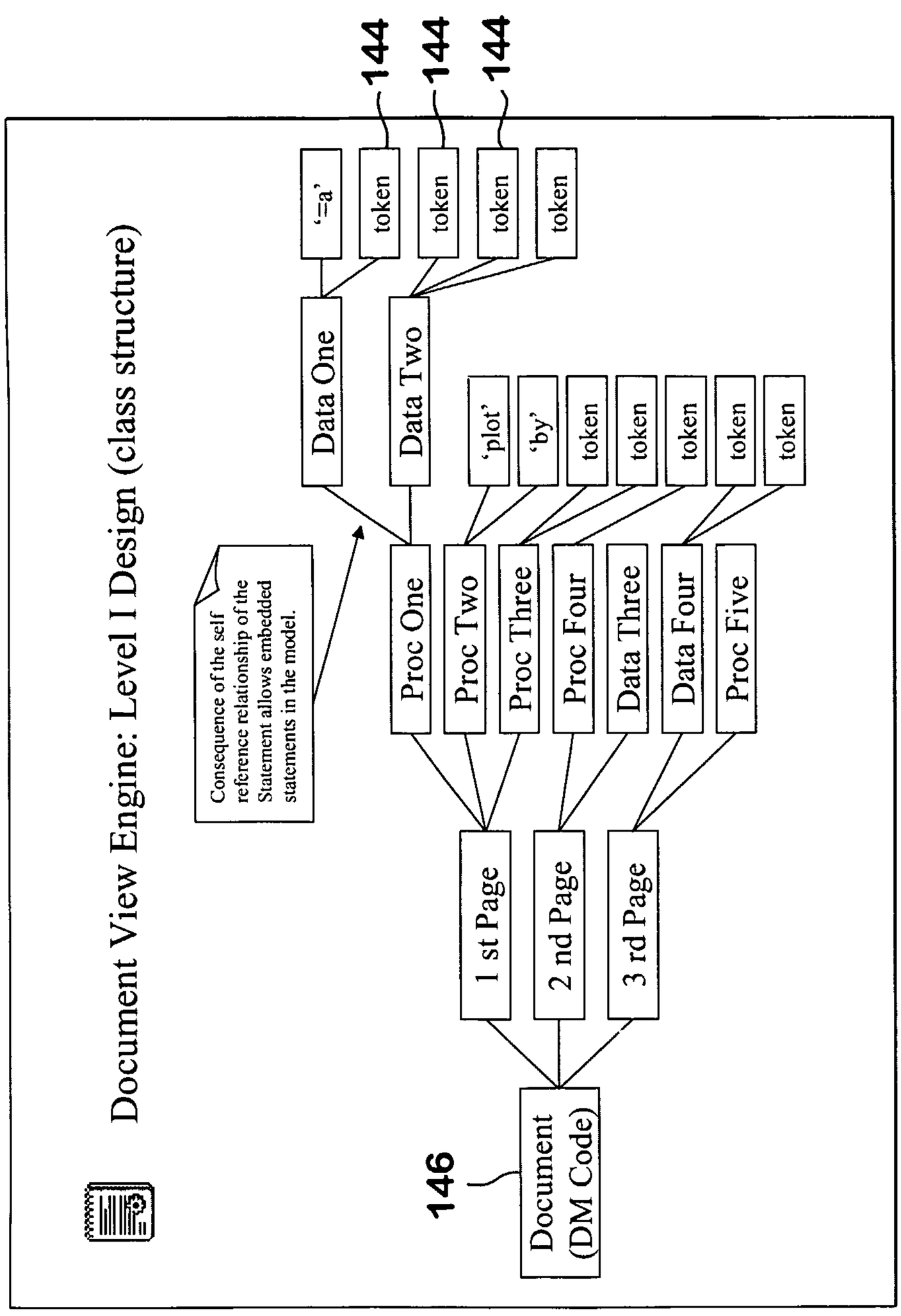

Document View Engine: Level I Design (class structure)
144
144
144
'=a'
token
token
token
token
Data One
Data Two
'plot'
'by'
token
token
token
token
token
Consequence of the self reference relationship of the Statement allows embedded statements in the model.
Proc One
Proc Two
Proc Three
Proc Four
Data Three
Data Four
Proc Five
1 st Page
2 nd Page
3 rd Page
146
Document (DM Code)

FIG. 22

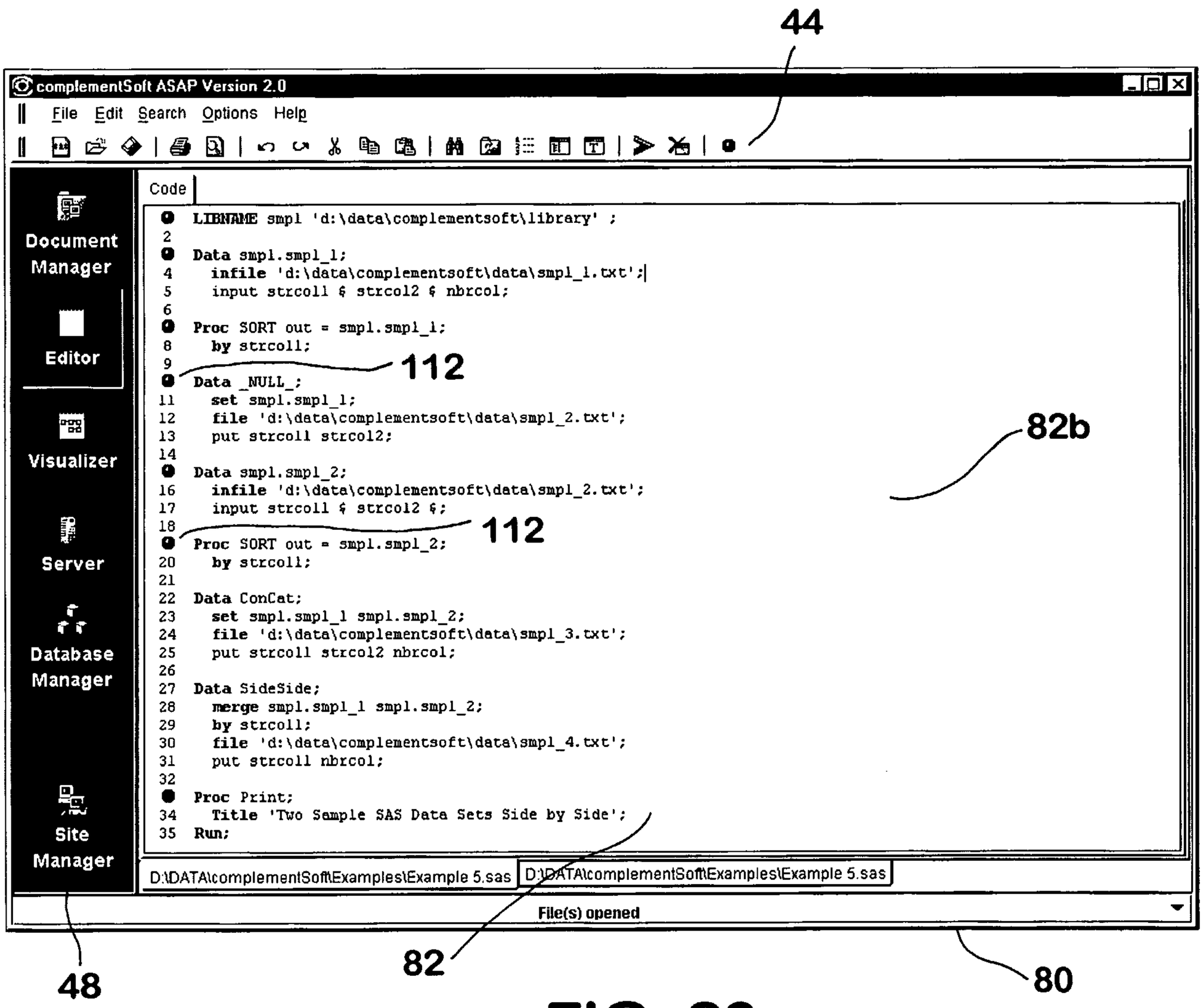
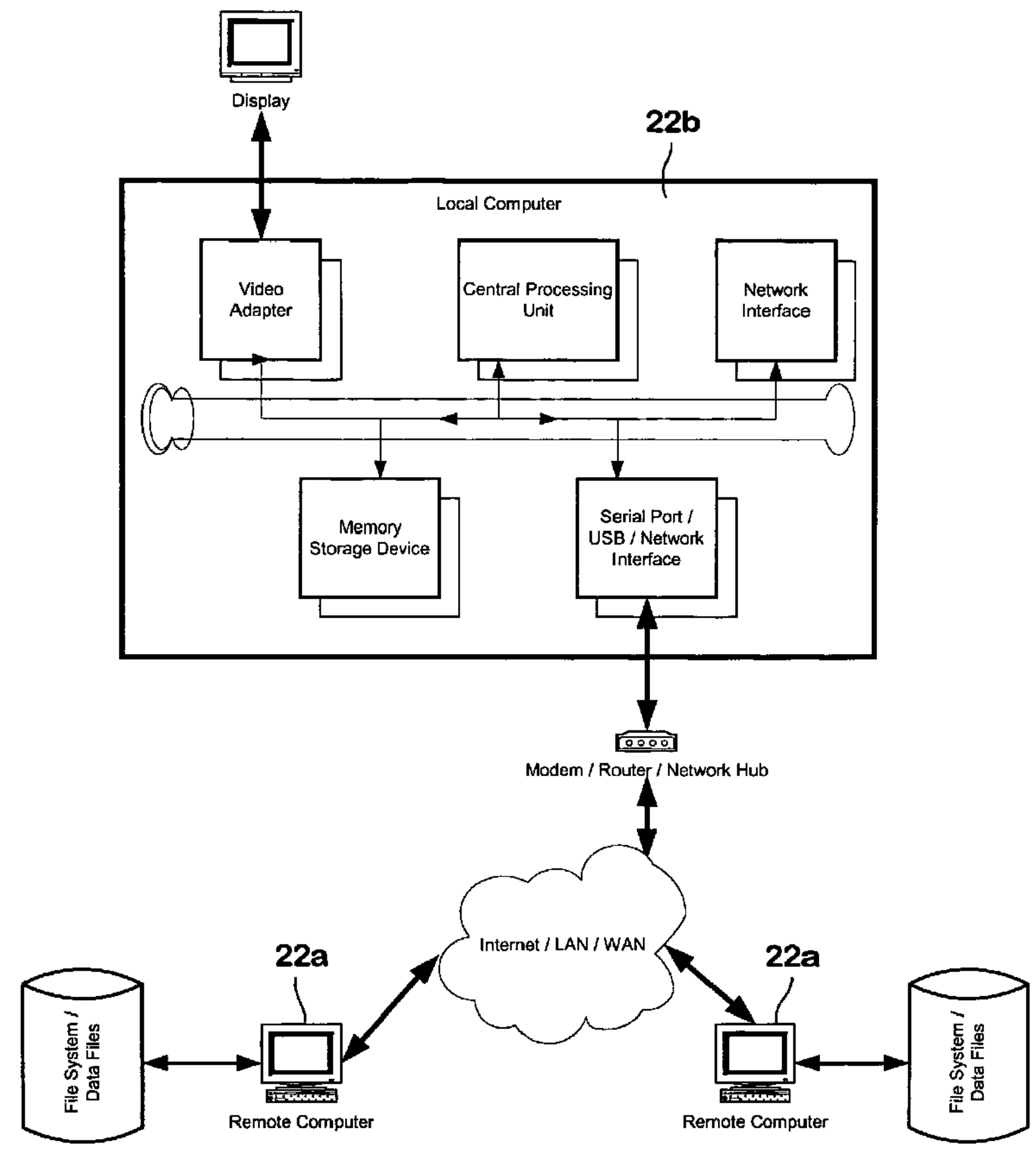

44
complementSoft ASAP Version 2.0
File Edit Search Options Help
Code
LIBNAME smpl 'd:\data\complementsoft\library' ;
Data smpl.smpl_1;
infile 'd:\data\complementsoft\data\smpl_1.txt';
input strcol1 $ strcol2 $ nbrcol;
Proc SORT out = smpl.smpl_1;
by strcol1;
112
Data _NULL_;
set smpl.smpl_1;
file 'd:\data\complementsoft\data\smpl_2.txt';
put strcol1 strcol2;
82b
Data smpl.smpl_2;
infile 'd:\data\complementsoft\data\smpl_2.txt';
input strcol1 $ strcol2 $;
112
Proc SORT out = smpl.smpl_2;
by strcol1;
Data ConCat;
set smpl.smpl_1 smpl.smpl_2;
file 'd:\data\complementsoft\data\smpl_3.txt';
put strcol1 strcol2 nbrcol;
Data SideSide;
merge smpl.smpl_1 smpl.smpl_2;
by strcol1;
file 'd:\data\complementsoft\data\smpl_4.txt';
put strcol1 nbrcol;
Proc Print;
Title 'Two Sample SAS Data Sets Side by Side';
Run;
Document Manager
Editor
Visualizer
Server
Database Manager
Site Manager
D:\DATA\complementSoft\Examples\Example 5.sas
D:\DATA\complementSoft\Examples\Example 5.sas
File(s) opened
48
82
80

FIG. 23


Display
22b
Local Computer
Video Adapter
Central Processing Unit
Network Interface
Memory Storage Device
Serial Port / USB / Network Interface
Modem / Router / Network Hub
Internet / LAN / WAN
22a
22a
File System / Data Files
Remote Computer
Remote Computer
File System / Data Files