



Frederick P. Fish  
1855-1930

W.K. Richardson  
1859-1951

# FISH & RICHARDSON P.C.

12390 El Camino Real  
San Diego, California  
92130

Telephone  
858 678-5070

Facsimile  
858 678-5099

Web Site  
www.fr.com

November 14, 2005

Attorney Docket No.: 17776-002US4/

Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

Presented for filing is a new continuation patent application of:

Applicant: MICHAEL DE ANGELO

Title: SYSTEM AND METHOD FOR CREATING AND MANIPULATING  
INFORMATION CONTAINERS WITH DYNAMIC REGISTERS

The prior application is assigned of record to Pattern Intelligence, Inc.,  
a Delaware corporation, by virtue of an assignment submitted to the Patent and  
Trademark Office and recorded on June 9, 2005 at 016675/0763.

Enclosed are the following papers, including those required to receive a filing date  
under 37 CFR §1.53(b):

	<u>Pages</u>
Specification	59
Claims	8
Abstract	1
Drawing(s)	30

Enclosures:

- Small entity statement. This application is entitled to small entity status.
- Application Cover Page.
- Postcard.


This application is a continuation (and claims the benefit of priority under 35 USC 120) of U.S. application serial no. 09/284,113, filed April 7, 1999 and claims the benefit of PCT/US99/01988 filed January 28, 1999 and of U.S. Patent Application No. 60/073,209, filed January 30, 1998. The disclosure of the prior application is considered part of (and is incorporated by reference in) the disclosure of this application.

CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. EV399296562US

November 14, 2005  
Date of Deposit





AUSTIN  
BOSTON  
DALLAS  
DELAWARE  
NEW YORK  
SAN DIEGO  
SILICON VALLEY  
TWIN CITIES  
WASHINGTON, DC

Commissioner for Patents  
November 14, 2005  
Page 2

			<u>Small Entity</u>	<u>Large Entity</u>	
Basic Filing Fee			150	300	\$150
Search Fee			250	500	\$250
Examination Fee			100	200	\$100
Total Claims 30	over 20	<b>10 x \$25</b>	25	50	\$250
Independent Claims 12	over 3	<b>9 x \$100</b>	100	200	\$900
Fee for Multiple Dependent claims			180	360	\$0
Fee for each additional 50 pages of Specification and Drawings over 100			125	250	\$0
		$(58+30-100)/50 = 0 \times$			
Total Filing fee					\$1650

A check for the filing fee is enclosed. Please apply any other required fees or any credits to deposit account 06-1050, referencing the attorney docket number shown above.

If this application is found to be incomplete, or if a telephone conference would otherwise be helpful, please call the undersigned at (858) 678-5070.


Kindly acknowledge receipt of this application by returning the enclosed postcard.

Please direct all correspondence to the following:

**20985**

PTO Customer Number

Respectfully submitted,



Carl A. Kukkonen, III  
Reg. No. 42,773

Enclosures

CAK/vzw  
10570293.doc

APPLICATION  
FOR  
UNITED STATES LETTERS PATENT

TITLE: SYSTEM AND METHOD FOR CREATING AND  
MANIPULATING INFORMATION CONTAINERS WITH  
DYNAMIC REGISTERS

APPLICANT: MICHAEL DE ANGELO

CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. EV399296562US

November 14, 2005  
Date of Deposit

**SYSTEM AND METHOD FOR CREATING AND MANIPULATING  
INFORMATION CONTAINERS WITH DYNAMIC REGISTERS**

**CROSS-REFERENCE TO RELATED APPLICATIONS**

[0001] The present application is a continuation of U.S. Patent Application No. 09/284,113, entitled System And Method For Creating And Manipulating Information Containers With Dynamic Registers, filed on April 7, 1999, which is incorporated herein in its entirety, and claims the benefit of PCT/US99/01988 filed January 28, 1999 and of U.S. Patent Application No. 60/073,209, filed January 30, 1998.

**BACKGROUND OF THE INVENTION**

[0002] 1. Field of the Invention

The present invention relates generally to computer systems in a multi-user mainframe or mini computer system, a client server network, or in local, wide area or public networks, and in particular, to computer networks for creating and manipulating information containers with dynamic interactive registers in a computer, media or publishing network, in order to manufacture information on, upgrade the utility of, and develop intelligence in, a computer network by offering the means to create and manipulate information containers with dynamic registers.

[0003] 2. Description of the Related Art

In the present day, querying and usage of information resources on a computer network is accomplished by individuals directing a search effort by submitting key words or phrases to be compared to those key words or phrases contained in the content or description of that information resource, with indices and contents residing in a fixed location unchanging except by human input. Similarly, the class of storage medium upon which information resides, its class and subclass organizational structures, and its routes of access all remain fundamentally unaltered by ongoing user queries and usage. Only the direct and intended

intervention of the owner of the information content or computer hosting site changes these parameters, normally accomplished manually by programmers or systems operators at their own discretion or the discretion of the site owner.

[0004] There exists currently in the art a limited means of interfacing a computer user with the information available on computer networks such as the world wide web. Primarily, these means are search engines. Search engines query thousands or tens of thousands of index pages per second to suggest the location of information while the user waits. While factual information can be accessed, the more complex, particular or subtle the inquiry, the more branches and sub-branches need to be explored in a time consuming fashion in order to have any chance of success. Further, there are no such automatic devices that reconstruct the information into more useful groupings or makes it more accessible according to factors attached to the content by the content creator such as the space or time relevancy of its content, or factors attached to the content by the system's compilation and analysis of the accumulated biography of that specific content's readership.

[0005] The utility of wide area and public computer networks is thus greatly limited by the static information model and infrastructure upon which those networks operate.

[0006] One problem is that on a wide area or public network, specific content such as a document remains inert, except by the direct intervention of users, and is modified neither by patterns or history of usage on the network, or the existence of other content on the network.

[0007] Another problem is that content does not reside in an information infrastructure conducive to reconstruction by expert rule-based, fuzzy logic, or artificial

intelligence based systems. Neither the intelligence of other information users nor the expert intelligence of an observant network computer system can be utilized in constructing, or re-constructing information resources. Where content resides in a fixed location and structure, “information” becomes something defined by the mind of the information provider rather than the mind of the information user, where the actual construction and utility of information exists. Information remains, like raw ore, in an unrefined state.

[0008] Another problem is that the class of storage medium upon which data resides cannot be system or user managed and altered according to the actual recorded and analyzed hierarchically graded usage of any given information resource residing on that storage medium except by statistical analysis of universal, undefined “hits” or visits to that page or site.

[0009] Another problem is that information resource groupings remain fixed on the given storage medium location according to the original installation by the resource author, not altered according to the actual recorded and analyzed hierarchically graded usage of that given information resource. Content itself remains inert, with no possibility of evolution.

[0010] A further problem with the prior art is that neither the search templates generated by those more knowledgeable in a given field of inquiry, nor the search strategies historically determined to be successful, or system-constructed according to analyses of search strategies historically determined to be successful, are available to inquiring users. A search template is here defined as one or more text phrases, graphics, video or audio bits, alone or in any defined outline or relational format designed to accomplish an inquiry. Internet or wide area network search may return dozens of briefs to a keyword or key phrase

inquiry sometimes requiring the time-consuming examination of multiple information resources or locations, with no historical relation to the success of any given search strategy.

[0011] A further problem is that there is limited means to add to, subtract from, or alter the information content of documents, databases, or sites without communicating with the owners or operators of those information resources, e.g., contacting, obtaining permission, negotiating and manually altering, adding or subtracting content. Additionally, once so altered, there is not a means to derive a proportionate value, and thereby a proportionate royalty as the information is used.

[0012] A final problem is that the physical residence of a body of data or its cyberspace location may not serve its largest body of users in the most expedient manner of access. Neither the expert intelligence of other information users nor the expert intelligence of an observant computer system is presently utilized by inherent network intelligence to analyze, re-design and construct access routes to information medium except by statistical analysis of universal, undefined "hits" or visits to that page or site.

[0013] Therefore, there is a need for a system and methods for creating and manipulating information containers with dynamic interactive registers defining more comprehensive information about contained content in a computer, media or publishing network, in order to manufacture information on, upgrade the utility of, and develop intelligence in, a computer network by providing a searching user the means to utilize the searches of other users or the historically determined and compiled searches of the system, a means to containerize information with multiple registers governing the interaction of that container, a means to re-classify the storage medium and location of information resources

resident on the network, a means to allow the reconstruction of content into more useful formations, and a means to reconstruct the access routes to that information.

### **SUMMARY OF THE INVENTION**

**[0014]** The present invention is a system and methods for manufacturing information on, upgrading the utility of, and developing intelligence in, a computer or digital network, local, wide area, public, corporate, or digital-based, supported, or enhanced physical media form or public or published media, or other by offering the means to create and manipulate information containers with dynamic registers.

**[0015]** The system of the present invention comprises an input device, an output device, a processor, a memory unit, a data storage device, and a means of communicating with other computers, network of computers, or digital-based, supported or enhanced physical media forms or public or published media. These components are preferably coupled by a bus and configured for multi-media presentation, but may also be distributed throughout a network according to the requirements of highest and best use.

**[0016]** The memory unit advantageously includes an information container made interactive with dynamic registers, a container editor, a search interface, a search engine, a search engine editor, system-wide hierarchical container gateways interacting with dynamic container registers, a gateway editor, a register editor, a data collection means with editor, a data reporting means with editor, an analysis engine with editor, an executing engine with editor, databases, and a means of communicating with other computers as above. These



components may reside in a distributed fashion in any configuration on multiple computer systems or networks.

[0017] The present invention advantageously provides a container editor for creating containers, containerizing storing information in containers and defining and altering container registers. A container is an interactive nestable logical domain configurable as both subset and superset, including a minimum set of attributes coded into dynamic interactive evolving registers, containing any information component, digital code, file, search string, set, database, network, event or process, and maintaining a unique network-wide lifelong identity.

[0018] The container editor allows the authoring user to create containers and encapsulate any information component in a container with registers, establishing a unique network lifelong identity, characteristics, and parameters and rules of interaction. The authoring user defines and sets the register with a starting counter and/or mathematical description by utilizing menus and simple graphing tools or other tools appropriate to that particular register. The registers determine the interaction of that container with other containers, system components, system gateways, events and processes on the computer network.

[0019] Containers and registers, upon creation, may be universal or class-specific. The editor provides the means to create system-defined registers as well as the means to create other registers. The editor enables the register values to be set by the user or by the system, in which case the register value may be fixed or alterable by the user upon creation. Register values are evolving or non-evolving for the duration of the life of the container on

the system. Evolving registers may change through time, space, interaction, system history and other means.

**[0020]** System-defined registers comprise: (1) an historical container register, logging the history of the interaction of that container with other containers, events and processes on the network, (2) an historical system register, logging the history of pertinent critical and processes on the network, (3) a point register accumulating points based upon a hierarchically rated history of usage, (4) an identity register maintaining a unique network wide identification and access location for a given container, (5) a brokerage register maintaining a record of ownership percentage and economic values, and others.

**[0021]** The present invention also includes user-defined registers. User defined registers may be created wholly by the user and assigned a starting value, or simply assigned value by the user when that register is pre-existent in the system or acquired from another user, and then appended to any information container, or detached from any container.

**[0022]** Exemplary user-defined registers comprise (1) a report register, setting trigger levels for report sequences, content determination and delivery target, (2) a triple time register, consisting of a range, map, graph, list, curve or other representation designating time relevance, actively, assigning the time characteristics by which that container will act upon another container or process, passively, assigning the time characteristics by which that container be acted upon by another container or process, and neutrally, assigning the time characteristics by which that container will interact with another container or process, (3) a triple space register, consisting of a range, map, graph, list, curve or other representation designating the domain and determinants of space relevance, actively, assigning the space characteristics by which that content will act upon another container or process, passively,

assigning the space, characteristics by which that content will be acted upon by another container or process, and neutrally, assigning the space characteristics by which that container will interact with another container or process, (4) a domain of influence register, determining the set, class and range of containers upon which that container will act, (5) a domain of receptivity register, determining the set, class and range of containers allowed to act upon that container, (6) a domain of neutrality register, determining the set, class and range of containers with which that container will interact, (7) a domain of containment register, determining the set, class and range of containers which that container may logically encompass, (8) a domain of inclusion register, determining the set, class and range of containers by which that container might be encompassed, (9) an ownership register, recording the original ownership of that containers, (10) a proportionate ownership register, determining the proportionate ownership of that containers, (11) a creator profile register, describing the creator or creators of that container, (12) an ownership address register, maintaining the address of the creator or creators of that container, (13) a value register, assigning a monetary or credit value to that container, and (14) other registers created by users or the system.

**[0023]** Containers are nestable and configurable as both subset and superset and may be designated hierarchically according to inclusive range, such as image component, image, image file, image collection, image database, or if text, text fragment, sentence, paragraph, page, document, document collection, document, database, document library, or any arrangement wherein containers are defined as increasingly inclusive sets of sets of digital components.

[0024] The present invention also includes, structurally integrated into each container, or strategically placed within a network at container transit points, unique gateways, nestable in a hierarchical or set and class network scheme. Gateways gather and store container register information according to system-defined, system-generated, or user determined rules as containers exit and enter one another, governing how containers system processes or system components interact within the domain of that container, or after exiting and entering that container, and governing how containers, system components and system processes interact with that unique gateway, including how data collection and reporting is managed at that gateway. The gateways record the register information of internally nested sub and superset containers, transient containers and search templates, including the grade of access requested, and, acting as an agent of an analysis engine and execution engine, govern the traffic and interaction of those containers and searches with the information resource of which they are the gateway and other gateways. The gateways' record of internally nested and transient container registers, and its own interaction with those containers, is made available, according to a rules-based determination, to the process of the analysis engine by the data collection and/or data reporting means.

[0025] The present invention also includes a means of data storage at any given gateway.

[0026] The present invention also includes a data collection means, residing anywhere on the network, or located at one or more hierarchical levels of nestable container gateways for gathering information from other gateways and analysis engines according to system, system-generated or user determined rules. The data collection means manages the gathering of data regarding network-wide user choices, usage and information about

information, by collecting it from container and gateway registers as those containers and gateways pass through one another. Such statistics as frequency, pattern, and range of time, space and logical class is collected as directed by the analysis engine, and made that data available to the analysis engine by advancing it directly to the analysis engine, or incrementally, to the next greater hierarchically inclusive collection level. The rules of data collection may be manually set or altered by the system manager, or set by the system and altered by the system in its evolutionary capacity.

[0027] The present invention also includes a data reporting means, located at one or more hierarchical levels of nestable container gateways for submitting information to other gateways and analysis engines according to system, system-generated or user determined rules. The data reporting means manages the sending of data from the registers, gateways and search templates in a frequency, pattern, and range of time, space and logical class as directed by the analysis engine, and makes that data available to the analysis engine by advancing it directly to the analysis engine, or incrementally to the next greater hierarchically inclusive reporting level. The rules of data collection may be manually set or altered by the system manager, or set by the system and altered by the system in its evolutionary capacity. The data reporting means may be established to work in concert, in redundancy, or in contiguous or interwoven threads of hierarchically nested containers.

[0028] The present invention also includes an analysis engine that receives, reports and collects information regarding the interaction of user searches with gateways and container registers, as well as container registers with other container registers, and container registers with gateways. The analysis engine analyzes the information submitted by the gateways and instructs the execution engine to create new information containers, content

assemblages, storage schemes, access routes, search templates, and gateway instructions. The analysis engine includes an editor that provides a system manager with a means of editing the operating principles of that engine, governing data reporting, data collection, search template loading, gateway instructions, and other.

[0029] The present invention also includes an execution engine, fulfilling the instructions of the analysis engine, to create new information containers, content sun and superset assemblages, storage schemes, access routes, search templates, and gateway instructions. The execution engine includes an editor that provides a system manager with a means of editing the operating principles of that engine, governing data reporting, data collection, search template loading, gateway instructions, and other.

[0030] The present invention also includes a search interface or browser. The search interface provides a means for a searching user to submit, record and access search streams or phrases generated historically by himself, other users, or the system. Search streams or phrases of other users are those that have been historically determined by the system to have the highest probability of utility to the searching user. Search streams or phrases generated by the system are those that have been constructed by the system through the analysis engine based upon the same criteria.

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

[0031] FIG. 1 is a block diagram of a first and preferred embodiment of a system constructed according to the present invention.

[0032] FIG. 2 A is block diagram of a preferred embodiment of the memory unit.

[0033] FIG. 2 B is an exemplary embodiment of a computer network showing computer servers, personal computers, workstations, Internet, Wide Area Networks, Intranets in relationship with containers and gateways.

[0034] FIG. 2B1 is an exemplary embodiment of a computer network showing computer servers, personal computers, workstations, Internet, Wide Area Networks, Intranets in relationship with containers and gateways and exemplary locations of gateway storage in proximity to one or more of the various sites.

[0035] FIGS. 2C through 2H are exemplary embodiments in block diagram form of computer network components showing a possible placement of nested containers, computer servers, gateways, and the software components named in Fig. 2 A on a network.

[0036] FIG. 3A is a graphical representation for one embodiment of a container having a plurality of containers nested within that container.

[0037] FIG. 3C is a drawing showing elements that might be logically encapsulated by a container. FIG. 4 is a drawing of an information container showing a gateway and registers logically encapsulating containerized elements.

[0038] FIG. 5 is a flowchart showing a preferred method for the containerization process and container editor operating on the communication device.

[0039] FIG. 6 is a flowchart showing a preferred method for searching for containers within a node.

[0040] FIG. 7 is a flowchart further showing a preferred method for searching for containers over one or more gateways.

[0041] FIG. 8 is a flowchart showing a method for performing the data collection and reporting on containers.

[0042] FIG. 9 is a flowchart showing the operation of the analysis engine.

[0043] FIG. 10 is a flowchart showing the operation of the execution engine.

[0044] FIG. 11 is a flowchart showing the operation of the gateway editor.

[0045] FIG. 12 is a flowchart showing the operation of the gateway process.

[0046] FIG. 13A is a drawing showing an example of nested containers,

gateways, registers, analysis engines and an execution engine prior to container reconstruction as depicted in 13 B, 13 C and 13 D.

[0047] FIG. 13B is a drawing showing the reconstructed nested containers of Figure 13A.

[0048] FIG. 13C is a drawing showing further reconstruction of nested containers, with a container relocated to reside within another container.

[0049] FIG. 13D is a drawing showing a flowchart of the reconstruction process

[0050] FIG. 14 is a drawing showing the screen interface of the container editor.

[0051] FIG. 15 is a drawing showing the screen interface of the gateway editor.

[0052] FIG. 16 is a drawing showing the screen interface of the search interface.

[0053] FIG. 17 is a drawing of a generic application program showing a drop-down menu link, and a button link to the containerization process or container editor.

## **DESCRIPTION OF THE PREFERRED EMBODIMENT**

[0054] THE SYSTEM

[0055] Referring now to FIG. 1, a preferred embodiment of a system 10 for creating and manipulating information containers with dynamic interactive registers in a computer, media, or publishing network 201 in order to manufacture information on, upgrade



the utility of, and develop intelligence in that network 201, is shown. The system 10 preferably comprises an input device 24, an output device 16, a processor 18, a memory unit 22, a data storage device 20, and a communication device 26 operating on a network 201. The input device 24, an output device 16, a processor 18, a memory unit 22, a data storage device 20, are preferably coupled together by a bus 12 in a von Neumann architecture. Those skilled in the art will realize that these components 24, 16, 18, 22, 20, and 26 may be coupled together according to various other computer architectures including any physical distribution of components linked together by the communication device 26 without departing from the spirit or scope of the present invention, and may be infinitely nested or chained, both as computer systems within a network 202, and as networks within networks 201.

**[0056]** The output device 16 preferably comprises a computer monitor for displaying high-resolution graphics and speakers for outputting high fidelity audio signals. The output device 16 is used to display various user interfaces 110, 125, 210, 300, 510, 610, 710, as will be described below, for searching for and containerizing information, and editing the container gateways, containers, container registers, the data reporting means and the data collection means, and the search, analysis and execution engines. The author uses the input device 24 to manipulate icons, text, charts or graphs, or to select objects or text, in the process of packaging, searching or editing in a conventional manner such as in the Macintosh or Windows operating systems.

**[0057]** The processor 18 preferably executes programmed instruction steps, generates commands, stores data and analyzes data configurations according to programmed instruction steps that are stored in the memory unit 22 and in the data storage device 20. The processor 22 is preferably a microprocessor such as the Motorola 680(x)0, the Intel 80(x)86

or Pentium, Pentium II, and successors, or processors made by AMD, or Cyrix CPU of the any class.

**[0058]** The memory unit 22 is preferably a predetermined amount of dynamic random access memory, a read-only memory, or both. The memory unit 22 stores data, operating systems, and programmed instructions steps, and manages the operations of all hardware and software components in the system 10 and on the network 201, utilizing the communication device 26 whenever necessary or expeditious to link multiple computer systems 202 within the network 201.

**[0059]** The data storage device 20 is preferably a disk storage device for storing data and programmed instruction steps. In the exemplary embodiment, the data storage device 20 is a hard disk drive. Historical recordings of network usage are stored on distributed and centralized data storage devices 20.

**[0060]** The preferred embodiment of the input device 24 comprises a keyboard, microphone, and mouse type controller. Data and commands to the system 10 are input through the input device 24.

**[0061]** The present invention also includes a communication device 26. The communication device 26 underlies and sustains the operations of, referring now also to Fig 2 the analysis 400 and execution 500 engines, the data reporting 600 and collection 700 means, the container editor 110, the search interface 300, and the search engine 320, providing the means to search, access, move, copy, utilize or otherwise perform operations with and on data. The communication device 26 utilizes one or more of the following technologies: modem, infrared, microwave, laser, photons, electrons, wave phenomena, cellular carrier, satellite, laser, router hub, direct cabling, physical transport, radio, broadcast or cable TV or

other to communicate with other computers, digital-supported television, computer networks, or digital-based or supported public or published media, or physical media forms, on any a local, wide area, public, or any computer-based computer supported, or computer interfaced network, including but not limited to the Internet. It also allows for the functioning and distribution of any container 100 or container component herein described to reside anywhere on any computer system in any configuration on that local, wide area, public, or corporate computer-based or computer related network, or digital-based or supported media form.

[0062] Referring now to Figure 2 A, a preferred embodiment of the memory unit 22 is shown. The memory unit includes: an interactive information container 100, a container editor 110, container registers 120, a container register editor 125, system-wide hierarchical container gateways 200, gateway storage 205, gateway editors 210, engine editors 510, a search interface 300, search engine 320, analysis engine 400, execution engine 500, a data reporting module, 600, a data reporting editor 610, a data collection module 700, a data collection editor 710, screen interfaces (GUI's) 936, menu or access buttons from generic computer programs 937, and databases 900, all residing in memory optimized between a data storage means 20 such as magnetic, optical, laser, or other fixed storage, and a memory means 22 such as RAM. The memory unit 22 functions by operating on communications network 12 with a communication device 26 on multiple computer systems 202 within the network 201. These components will be described first briefly in the following paragraphs, then in more detail with reference to Figures 3 A through 17.

[0063] Those skilled in the art will realize that these components might also be stored in contiguous blocks of memory, and that software components or portions thereof may reside in the memory unit 22 or the data storage means 20.

[0064] The present invention includes information containers 100 as noted above. The information container 100 is a logically defined data enclosure which encapsulates any element or digital segment (text, graphic, photograph, audio, video, or other), or set of digital segments, or referring now to FIG. 3 C, any system component or process, or other containers or sets of containers. A container 100 at minimum includes in its construction a logically encapsulated portion of cyberspace, a register and a gateway. A container 100 at minimum encapsulates a single digital bit, a single natural number or the logical description of another container, and at maximum all defined cyberspace, existing, growing and to be discovered, including but not limited to all containers, defined and to be defined in cyberspace. A container 100 contains the code to enable it to interact with the components enumerated in 2 A, and to reconstruct itself internally and manage itself on the network 201.

[0065] The container 100 also includes container registers 120. Container registers 120 are interactive dynamic values appended to the logical enclosure of an information container 100, and serve to govern the interaction of that container 100 with other containers 100, container gateways 200 and the system 10, and to record the historical interaction of that container 100 on the system 10. Container registers 120 may be values alone or contain code to establish certain parameters in interaction with other containers 100 or gateways 200.

[0066] The present invention also includes container gateways 200. Container gateways 200 are logically defined gateways residing both on containers 100 and independently in the system 10. Gateways 200 govern the interactions of containers 100 within their domain, and alter the registers 120 of transiting containers 100 upon ingress and egress.

[0067] The present invention also includes container gateway storage 205 to hold the data collected from registers 120 of transient containers 100 in order to make it available to the data collection means 700 and the data reporting means 600, and to store the rules governing the operations of its particular gateway 200, governing transiting containers upon ingress and egress, and governing the interactive behavior of containers 100 within the container 100 to which that gateway 200 is attached. Gateway storage 205 may be located on gateways 200 themselves, containers 100 or anywhere on the network 202, 201, including but not limited to Internet, Intranet, LAN, WAN, according to best analysis and use.

[0068] The memory unit 22 also includes an execution engine 500 to perform the functions on the system 10 as directed by the analysis engine after its analysis of data from the data reporting means 600, the data collection means 700, and the search interface 300.

[0069] The memory unit 22 also includes a search interface 300, by which the user enters, selects or edits search phrases or digital strings to be used by the search engine 320 to locate containers 100.

[0070] The memory unit 22 also includes an analysis engine 400 which performs rules based or other analysis upon the data collected from the search interface 300 and the data collection 700 and data reporting 600 means.

[0071] The memory unit 22 also includes a data reporting means 600, by which means the information collected by gateways 200 from transient containers 100 is sent to the analysis engine 400.

[0072] The memory unit 22 also includes a data collection means 700, by which means the analysis engine 400 gathers the information collected by gateways 200 from transient containers 100.

[0073] The memory unit 22 also includes a container editor 110 for creating, selecting, acquiring, modifying and appending registers 120 and gateways 200 to containers 100, for creating, selecting, acquiring, and modifying containers, and for selecting content 01 to encapsulate.

[0074] The memory unit 22 also includes a register editor 125, for creating, selecting, acquiring and modifying container registers 120 and establishing and adjusting the values therein.

[0075] The memory unit 22 also includes a gateway editor 210, by which means the user determines the rules governing the interaction of a given gateway 210 with the registers 120 of transient containers 100, governing transiting containers upon ingress and egress, and governing the interactive behavior of containers within the container to which that gateway is attached.

[0076] The memory unit 22 also includes databases 900, by which means the analysis engine 400, the execution engine 500, the gateways 100, the editors 110, 125, 210, 510, 610, 710, and the search interface 300, store information for later use.

[0077] The memory unit 22 present invention also includes a search engine 320 by which means the user is able to locate containers 100 and, referring now to Fig. 4, containerized elements 01.

[0078] The memory unit 22 present invention also includes an engine editor 510, by which means the user establishes the rules and operating procedures for the analysis engine 400 and the execution engine 500.

[0079] The memory unit 22 present invention also includes a reporting means editor 610, by which means the user establishes the rules and schedule under which the

information collected by gateways 200 from transient containers 100 will be sent to the analysis engine 400.

[0080] The memory unit 22 present invention also includes a collection means editor 710, by which means the user establishes the rules and schedule under which the analysis engine 400 will gather the information collected by gateways 200 from transient containers 100.

[0081] The memory unit 22 present invention also includes screen interfaces (GUI's) 936, specifically designed to simplify and enhance the operations of the container editor 110, the gateway editor 210, and the search interface 300.

[0082] The present invention also includes a menu or button access 937, by which a user utilizing any generic computer program may access the system 10 or the container editor 110 from a menu selection(s) or button(s) within that program.

[0083] The present invention also includes a computer, media or publishing network 201, comprising computers, digital devices and digital media 202 and a communication device 26, within which the components enumerated in Fig. 2 A interact, compiling, analyzing, and altering containers 100 and the network 201 according to information gathered from container registers 120.

[0084] The memory unit 22 also includes one or more computers 202, by which means the components of Fig 1 sustain the operations described in Fig. 2 A.

[0085] The memory unit 22 also includes flat or relational databases 900, used where, and as required. Databases are used to store search phrases, search templates, system history for the analysis engine and execution engine, container levels and container, sites and digital elements, or any and all storage required to operate the system.

[0086] Referring now to FIG. 2 B, a drawing of a computer network 201 as a system 10, showing a possible placement of nested containers 100, computer servers, gateways 200, on the sites described below. (Note: Fig. 2 B utilizes in parts the same numbering scheme as Fig. 13 A, 13 B, 13 C, 13 D and as Fig. 2 A.) In FIG. 2 B various exemplary sites are shown, any or all of which might interact dynamically within the system. Site 1 shows a single workstation with a container and gateway connected to an Intranet. (Individual containers may be a floppy or CD-Rom to be downloaded or inserted.) Site 2 shows a server with a gateway in relationship to various containers.. Site 3 shows an Internet web page with a container residing on it. Site 4 shows a personal computer with containers and a gateway connected to the Internet. Site 5 shows a configuration of multiple servers and containers on a Wide Area Network.. Site 6 shows a workstations with a gateway and containers within a container connected to a Wide Area Network. Site 7 shows an independent gateway, capable of acting as a data collection and data reporting site as it gathers data from the registers of transiting containers, and as an agent of the execution engine as it alters the registers of transient containers. A container 100 contains the code to enable it to interact with the components enumerated in 2A, and to reconstruct itself internally and manage itself on the network 201. The code resides in and with the container in its registers and gateway definitions and controls. Additional system code resides in all sites to manage the individual and collective operation and oversight of the components enumerated in 2A, with the specific components distributed amongst the sites according to the requirements of optimization.



[0087] Referring now to Fig. 2 B 1 various exemplary sites are shown as described above in Fig. 2 B, with the addition of possible location of one or more gateway storage 205 locations.

[0088] Referring now to Figures 2 C through 2 H, various exemplary sites with one or more of the logical components of the system 10 in relationship are shown. Site 1 comprises an interactive information container 100, a container editor 110, container registers 120, a container register editor 125, system-wide hierarchical container gateways 200, gateway storage 205, gateway editors 210, engine editors 510, a search interface 300, search engine 320, analysis engine 400, execution engine 500, a data reporting means 600, a data reporting means editor 610, a data collection means 700, a data collection means editor 710, and databases 900, all residing on data storage means 20, utilizing the memory unit to function 22, operating on communications network 12 with a communication device 26..

[0089] Site 2 comprises an interactive information container 100, a container editor 110, container registers 120, a container register editor 125, system-wide hierarchical container gateways 200, gateway storage 205, gateway editors 210, engine editors 510, search engine 320, analysis engine 400, execution engine 500, a data reporting means 600, a data reporting means editor 610, a data collection means 700, a data collection means editor 710, and databases 900, all residing on data storage means 20, utilizing the memory unit to function 22, operating on communications network 12 with a communication device 26.

[0090] Site 3 comprises an interactive information container 100, a container editor 110, container registers 120, a container register editor 125, hierarchical container

gateways 200, gateway storage 205, gateway editors 210, and databases 900, all residing on data storage means 20, utilizing the memory unit to function 22, operating on communications network 12 with a communication device 26.

**[0091]** Site 4 comprises an interactive information container 100, a container editor 110, container registers 120, a container register editor 125, hierarchical container gateways 200, gateway storage 205, gateway editors 210, a search interface 300, and databases 900, all residing on data storage means 20, utilizing the memory unit to function 22, operating on communications network 12 with a communication device 26.

**[0092]** Site 5 comprises an interactive information container 100, container registers 120, a container register editor 125, hierarchical container gateways 200, gateway storage 205, and databases 900, all residing on data storage means 20, accessed and utilized by non-resident memory unit 22, operating on communications network 12 with a communication device 26.

**[0093]** Site 6 includes an independent analysis engine 400, execution engine 500, data collection means 700, and data reporting means 600 gateway editors 210, engine editors 510, a data reporting means editor 610, a data collection means 700, a data collection means editor 710, and databases 900, all residing on data storage means 20, utilizing the memory unit to function 22, operating on communications network 12 with a communication device 26.

**[0094]** Referring now to FIG. 3 A and FIG. 3 B, a block diagram of several nested information containers is shown, including examples of elements, e.g., code 1100, text 1200, audio 1300, video 1400, photograph 1500, graphic images 1600, and examples of possible container level classifications in increasing size, e.g., element 10900000, document

10800000, database 10700000, warehouse 10600000, domain 10500000, and continuing increasingly larger on Fig 3 (B), subject 10400000, field 10300000, master field 10200000, species 10100000. Containers may be infinitely nested and assigned any class, super class or sub class scheme and description by the creator of the container to govern nesting within that container. In addition to digital elements, containers may also include system process and components, including containerization itself.

[0095] Referring now to FIG. 3 C, a block diagram of an information container system is shown, listing, without any relationship indicated, some of the possible system components and processes, or sets thereof, that may be encapsulated as elements 01 in an information container 100. An information container 100 may include one or more of the following: any unique, container 100, gateway 200, output device 16, input device 24, output device process 160, input device process 240, data storage device 20, data storage device process 2000, processor 18, bus 12, content 01, search process 02, interface 04, memory unit 22, communication device 26, search interface 300, search process 98, network 201, class of device, process or content 999, class of process at any unique class of device 990, process at any unique device 99, editor 110, 125, 210, 510, 610, 710, engine 320, 400, 500, containerization process 1098, or process 08.

[0096] Any container may include (n) other containers, to infinity. The use of value evolving container registers 120 in conjunction with gateways 200, data reporting modules 600, data collection modules 700, the analysis engine 400, and the execution engine 500 provides the information container 100 with extensive knowledge of the use, operation of its internal contents, prior to, during and after those contents' residence within that container 100, and extensive knowledge of the use, operation and contents of the system 10 external to

itself, and allows the container 100 to establish and evolve its own identity and course of interaction on the system 10. Further, containers 100, as logical enclosures, can exist and operate independent of their digital contents, whether encapsulating audio, video, text, graphic, or other.

[0097] Referring now to FIG. 4, a block diagram of an information container 100 is shown. The information container 100 is a logically defined data enclosure which encapsulates any element, digital segment (text, graphic, photograph, audio, video, or other), set of digital segments as described above with reference to FIG. 3 (C), any system component or process, or other containers or sets of containers. The container 100 comprises the containerized elements 01, registers 120 and a gateway 200.

[0098] Registers 120 appended to an information container 110 are unique in that they operate independently of the encapsulated contents, providing rules of interaction, history of interaction, identity and interactive life to that container 100 through the duration of its existence on a network 201, without requiring reference to, or interaction with, its specific contents. They enable a container 100 to establish an identity independent of its contents. Additionally, registers 120 are unique in that their internal values evolve through interaction with other containers 100, gateways 200, the analysis engine 400, the execution engine 500, and the choices made by the users in the search interface 300, the container editor 110, the register editor 125, the gateway editor 210, the engine editor 510. Registers 120 are also unique in that they can interact with any register of a similar definition on any container 100 residing on the network 201, independent of that container's contents. Registers 120, once constructed, may be copied and appended to other containers 100 with their internal values reset, to form new containers. Register values, when collected at gateways 200 and

made available to the analysis engine 400 through the data collection means 700 and the data reporting means 600, provide an entirely new layer of network observation and analysis and operational control through the execution engine 500. Registers 120 accomplish not only a real time information about information system, but also a real time information about information usage on a network. Further, because the user base of a network determines usage, the system 10, in gathering information about information usage, is observing the choices of the human mind. When these choices are submitted to the analysis of a rules-based or other analysis engine 400, the system 10 becomes capable of becoming progressively more responsive to the need of the user base, in effect, learning to become more useful by utilizing the execution engine 500 to create system-wide changes by altering the rules of gateway 200 interaction and thereby altering the registers 120 of transient containers 100 and establishing a complete evolutionary cycle of enhanced utility.

[0099] Further, in establishing the pre-defined registers as described in the following four paragraphs, the following unique aspects of information about information are utilized for the first time: 1) the dynamic governance of information according to its utility through time, in active, passive and neutral aspects, as explained below; 2) the dynamic governance of information according to its utility through space in active, passive and neutral aspects, as explained below; 3) the dynamic governance of information according to its ownership, as explained below; 4) the dynamic governance of information according to its unique history of interaction as an identity on a network, as explained below; 5) the dynamic governance of information according to the history of the system on which it exists, as explained below; 6) the dynamic governance of information according to established rules of interaction, in active, passive and neutral aspects, as explained below; 7) the dynamic

governance of information according to the profile of its creator, as explained below; 8) the dynamic governance of information according to the value established by its ongoing usage, as explained below; 9) the dynamic governance of information according to its distributed ownership, as explained below; 10) the dynamic governance of information according to what class of information it might be incorporated into, and according to what class of information container it might incorporate, as explained below; 11) the dynamic governance of information according to self-reporting, as explained below.

[00100] Referring now to Fig 4, registers 120 may be (1) pre-defined, (2) created by the user or acquired by the user, or (3) system-defined or system-created. Pre-defined registers 120 are those immediately available for selection by the user within a given container editor as part of that container editor, in order that the user may append any of those registers 120 to a container 100 and define values for those registers 120 where required. Registers 120 created by the user are those conceived and created by a specific user or user group and made immediately available for selection by the user or user group in conjunction with any of a wide number of container editors, in order that the user may append any of those registers 120 to a container 100 and define values for those registers 120 where required. Registers 120 acquired by the user are those registers existing network-wide 201, created by the user base, that might be located and acquired by the user in order that the user may append any of those registers 120 to a container 100 and define values for those registers 120 where required. System-defined registers are those registers whose values are set and/or controlled by the system 10. System-created registers are those registers created by the system 10.

[00101] Registers 120 are user or user-base created or system-created values or ranges made available by the system 10 to attach to a unique container, and hold system-set, user-set, or system-evolved values. Values may be numeric, may describe domains of time or space, or may provide information about the container 100, the user, or the system 10. Registers 120 may be active, passive or interactive and may evolve with system use. Pre-defined registers include, but are not limited to, system history 110000, container history 101000, active time 102000, passive time 103000, neutral time 104000, active space 111000, passive space 112000, neutral space 113000, containment 105000, inclusion 106000, identity 114000, value 115000, ownership 107000, ownership addresses 116000, proportionate ownership 117000, creator profile 108000, receptivity 118000, influence 119000, points 109000, others 120000, reporting 121000, neutrality 122000, acquire 123000, create 124000, content title 125000, content key phrase(s) 126000, and content description 127000, security 12800, and parent rules 129000.

[00102] Pre-defined registers comprise an historical container register 101000, logging the history of the interaction of that container 100 with other containers, events and processes on the network 201, an historical system register 110000, logging the history of pertinent critical and processes on the network, a point register 109000 accumulating points based upon a hierarchically rated history of usage, an identity register 114000 maintaining a unique network wide identification and access location for a given container specifying a unique time and place of origin and original residence, a proportionate ownership register 117000 maintaining a record of ownership percentage and economic values, and others 120000.

[00103] User-defined registers include a report register 121000 setting trigger levels for report sequences, content determination and delivery target, three time registers, consisting of a range, map, graph, list, curve or other designating time relevance, 102000 assigning the time characteristics by which that container will act upon another container or process, 103000 assigning the time characteristics by which that container be acted upon by another container or process, and 104000 assigning the time characteristics by which that container will interact with another container or process, three space registers, consisting of a range, map, graph, list, curve or other designating the domain and determinants of space relevance, 111000 assigning the space characteristics by which that content will act upon another container or process, 112000 assigning the space, characteristics by which that content will be acted upon by another container or process, and 113000 assigning the space characteristics by which that container will interact with another container or process, a domain of influence register 119000, determining the set, class and range of containers upon which that container will act, a domain of receptivity register 118000, determining the set, class and range of containers allowed to act upon that container, a domain of neutrality register 122000, determining the set, class and range of containers with which that container will interact, a domain of containment register 105000, determining the set, class and range of containers which that container may logically encompass, a domain of inclusion 106000 register, determining the set, class and range of containers by which that container might be encapsulated, an ownership register 107000, recording the original ownership of that containers, a creator profile register 108000, describing the creator or creators of that container, an ownership address register 116000, maintaining the address of the creator or creators of that container, a value register 115000, assigning a monetary or credit value to that



container, other registers 120000 created by users or the system, a reporting register 121000, determining the content, scheduling and recipients of information about that container, a neutrality register 122000, an acquire register 123000, enabling the user to search and utilize other registers residing on the network, a create register 124000, enabling the user to construct a new register, a content title register 125000, naming the contents of the container, a content key register, 126000, identifying the container contents with a key phrase generated by the user and/or the system based upon successful usage of that phrase in conjunction with the utilization of the information within that container 100, a content description register 127000, identifying the container contents with additional description, a security register 128000, controlling container security, and a parent container register 129000, storing the rules governing container interaction as dictated by the parent (encapsulating) container.

**[00104]** The container also includes a gateway 200 and gateway storage 205.

**[00105]** Gateways 200 are logically defined passageways residing both on containers 100 and independently in the system 10. Gateways 200 govern the interactions of containers 100 encapsulated within their domain by reading and storing register 120 information of containers entering and exiting that container 100.

**[00106]** The present invention also includes container gateway storage 205. Gateway storage 205 stores information regarding the residence, absence, transience, and alteration of encapsulated and encapsulating containers 100, and their attached registers 120, holding the data collected from registers 120 of transient containers 100 in order to make it available to the data collection means 700 and the data reporting means 600, and storing the rules governing the operations of its particular gateway 200.

[00107] Referring now to FIG. 5, a flow chart of the preferred method for creating a container 100 is shown.

[00108] Input is received from the user selecting a container level through use of a drop-down menu 10100. A menu of all possible container classes within the subset and superset scheme of multiple hierarchically nested containers, i.e.; element, document, file, database, warehouse, domain, and more, is displayed on the output device 10200. Input is received from the user selecting a class 10300.

[00109] A graphic representation of a container in that class, with registers common to all containers as well as registers unique to its class is displayed 10301.

[00110] Input is received from the user choosing to “create” 10400, “edit” 10500, or “locate” 10600.

[00111] When the input of “create” 10400 is received from the user, a container template in that class appears 10410. Input from the user is then received adding or selecting a register 10540 to append to that container template. When input is received from the user adding a register, a list of registers that might be added to that class of container is made available to select 10550. Input is received from the user selecting a register 10560 and editing it 10570. The menu returns to “add or select” 10540.

[00112] If the input of “locate” 10600 is received from the user, the system prompts the user to enter the identity of the container or class of containers 10605. The system locates the container(s) 10610. Input is received from the user selecting a container 10620. The system prompts the user for a security code for permission to access the container for template use, or to alter its registers, or to alter its content 10630. . Input is received from the user entering a name and password providing access to one of the security levels 10640.

Input is received from the user editing the container accordingly by transition to step 10500 and performing the steps for editing.

[00113] If the input of “edit” 10500 is received, a list of containers available to edit at that level is shown 10510. Input is received from the user selecting a container 10520. That container appears, available to edit 10530. Input is received from the user selecting “add” or “select” registers 10540 by the user clicking on the graphically depicted register, or from a drop down menu. Input is received from the user selecting the register to edit 10560. Input is received from the user selecting “modify” or “delete” for that register 10565. If input is received from the user to “delete,” that register is severed from the container. If input is received from the user to “modify”, the register editor 10570 screen appropriate to that register appears, i.e., an x-y type graph to define a curve of relevant active time, in which the user manipulates the x-y termini, scale and curve, or a global map in which Input is received from the user selecting the locale of active space, whether zip code, city, county, state, country, continent, plant or other. When input is received from the user saving the definition, the screen returns to the main container screen to make another selection available. . Input is received from the user defining as many registers as he chooses. One of the registers may be named "new register." Input is received from the user selecting the new register, and if chosen by the user, defining a wholly unique and new kind of register by the user entering input into the register editor 125.

[00114] When the input is received from the user choosing to add a register, a list of registers that might be added to that class of container are made available to select 10550. Input is received from the user selecting a register 10560 and editing it 10570. The menu returns to “add or select” 10540, and in turn to Input – Select Container.

[00115] Input may then be received from the user choosing to add, modify, or delete the container contents 10700. Once the registers are defined, input is received from the user indicating completion and the interface reverts to the container editor. When input is received from the user choosing "select component" (to select the component to containerize) from the main menu bar 10700, a window appears allowing the user to select any file, component, or other container. If for example, the user were creating a warehouse container, and wishes to incorporate several databases into that container, input would then be received from the user selecting "database." The program would prompt the user for the location (directory) of that database or container. If the requested selection is not containerized, input may then be received from the user choosing to containerize the element at that time, after which the program returns to "select component." Once input is received from the user defining the database location, the program logically encases the directory or directories in the defined container. The above procedure may be repeated as many times as desired to include multiple databases within a single container. While logical simplicity would dictate that all containers within a container be of the same subset, it would be possible for input to be received from the user choosing containers of any subset to include in the container. When input is received from the user choosing "finished," the container is created with a unique network identity, preferably through some combination of exact time and digital device serial number, or centralized numbering system, or other means. The container 100 contains all digital code, including data and program software from the selected items or containers.

[00116] Input may then be received from the user to publish the container 11100 at a user-identified or system suggested location 11200 to be selected 11400.

[00117] Input is received from the user to "publish", from the main menu bar 11100. Input is received from the user choosing to leave the container where it was created, move or copy it to another drive, directory, computer, or network the user designates, or select the location from location options offered by the system 11200, or submit, or duplicate and submit, the container to the analysis engine 400 for intelligent inclusion in other containers, thus allowing the system to publish the container as instructed or choose the residence of the container 11400.

[00118] If input is received from the user to choosing to "move," or "copy" a browse function allows the user to name the new location or browse a list of possible locations. If input is received from the user choosing to "submit," a browser function allows the user to name the analysis search engine 310 or browse a list of possible analyses engines. When input is received from the user choosing the residence of the container 11300, the program restores the search interface screen.

[00119] Referring now to FIG. 6, a flow chart of the method for searching for containers 100.

[00120] When input is received from the user selecting "search interface" from the main title bar, the search interface screen appears. The user is given the choice of containerizing selected content or requesting that container levels be displayed 30100. From a drop down menu another menu appears allowing input to be received from the user selecting the container level 30200. Input is received from the user selecting the container level (from the smallest component to the whole system) 30300.

[00121] Input is received 30310 from the user selecting the phrases, containers or components, which then are re-submitted to the same process, until the input is received from the user selecting a specific site or container.

[00122] The search phrase, whether containerized or not, is submitted simultaneously to the search engine 30400 and the analysis engine 30500.

[00123] The screen then reports in a selection menu, the number of applicable sites found by the search engine 30410, the number of historically proven applicable sites found by the analysis engine 30410, the number of historically proven applicable containers at the selected container level or any container level found by the analysis engine 30410, and the number of historically proven new search phrases or digital segments found by the analysis engine 30320. Input is received from the user selecting one of the named sets above 30330. If input is received from the user choosing the search engine, the search interface lists the applicable site titles with a brief description 30410. If input is received from the user choosing the site list of the analysis, the search interface lists the applicable site titles with a brief description 30410. If input is received from the user choosing the container list of the analysis engine, the search interface lists the applicable container titles with a brief description 30410. If input is received from the user selecting a container 30420, the system offers the means to view titles and descriptions of sub-containers at any chosen class level. . If input is received from the user choosing the phrase list of the analysis engine, the search interface lists the applicable phrases or digital segments with a brief description 30320. The search and search result cycle repeats until input is received from the user choosing to go to an individual container or site.

[00124] Input is received from the user entering text or any digital string describing his search objectives into a text or search box. When input is received from the user submitting the search string, the system provides the option of containerizing the search through the container editor 110. Once the search container 101 is created, the system restores the search interface 300 screen the user.

[00125] Input is received from the user selecting “search”, “supported search” or “both” from another drop-down menu and from submitting the search. When input is received from the user selecting “search” 30310, the search phrase is submitted to the search engine 30400, which searches both content and the appropriate container registers, as pre-indexed in the search engine, and returns a list of appropriate locations, components or containers. When input is received from the selecting “supported search”, the search phrase is submitted to the analysis engine search support, which returns a list, in a drop-down menu, of search phrases or individual containers, for any and all container levels, used by other users or created by the system and known to be historically successful for the described effort and the described searching user, as per the results of the analysis search engine. Input is received from the user selecting a new search phrase or specific container from the drop down menu 30330. When input is received from the user choosing a new search phrase, that phrase is also submitted to the analysis engine 30500 which returns a list of pre-compiled historically proven sites, components or containers associated with that search phrase 30320. Input is received from the user choosing a selection 30420 and the system calls up that specific site, container or component. If input is received from the user selecting a specific site, container or component at any time during the search process, that element is called up by the system 30440.

[00126] Input is received from the user choosing to containerize a search or select a container level in which to search 30100. When input is received from the user choosing to containerize the search, the software moves to the container editor as described in Fig. 5, and then returns the user to the search interface screen. Input is received from the user selecting to search a specific container level or the whole network. The system shows the available levels 30200. Input is received from the user selecting a container level 30300, and entering the text or digital component comprising the search string 30310. The system searches the containers 30400 while simultaneously submitting the search string to the analysis engine 30500. While the system is accessing containers, sites or templates 30700, the analysis engine 30500 inquires of the appropriate database 30600 to access historically successful containers, sites or search templates corresponding to the search request 30700, which is then shown on another portion or option of the search interface, either as available containers or sites 30410 or as search template options 30320. On one portion or option of the search interface screen the corresponding containers or sites are listed and/or previewed for selection 30410. Input is received from the user selecting the container to access 30420. The system accesses that container 30430 and shows it on the screen 30440 for user review. Input is received from the user selecting an operation, i.e., preview, read, purchase, move, copy, lease, in any composed schedule with operations assigned specific values 30460, and the system obtains the specified result 30470. The selection of the operation including any interaction with any uniquely defined container 100 is recorded 30800 by the container gateway (Fig. 2 A, 200), stored in the gateway storage 205 and made available to the analysis engine (Fig. 9) by the data collection and reporting means (Fig. 8). Reporting and collection occurs on a regular basis according to user determined times or rules. The analysis engine compiles and analyzes



selections according to various rules-based systems applicable to the particular container area of residence in cyberspace.

[00127] Input is received from the user selecting the container or site 30410, proceeding as described above, or selecting a search template 30330, and editing it to re-enter the search 30310. All operations on Fig. 6 utilize the communication device 26 whenever necessary or expeditious.

[00128] Referring now to FIG. 7, a flow chart of the search process is shown. Steps in FIG. 7 repeated from FIG. 6 are given the same reference number as in FIG. 6 for convenience and ease of understanding. Fig. 7 commences with "SEARCH TRANSITS GATEWAY 32100", continuing from Fig. 6, "SYSTEM SEARCHES CONTAINERS 30400". The submitted search 32100 transits the gateway 200. The gateway 200 interacts with the container registers 32200. The gateways 200 store the information downloaded from the registers 32300, and the container registers are altered 32500. The container registers 120 then interact with the registers 120 of the encapsulated search, which registers, and the values set within, have been constructed and appended to the search through the search interface 32600. Values are exchanged and compared and operations performed under the rules governing both interacting containers 100, and the rules governing the search container 100 and any gateway 200. The search engine 320, operating under the principles and means of search engines presently existing as described elsewhere, then provides to the search interface 32600 a list of containers 100 meeting the requirements of the search and its appended registers, as well as additional search options 32900. The gateway 200 reports and makes available for collection to the analysis engine 400 the information obtained from the interaction 32400. On a periodic basis defined by the user or a rules-based system, the

analysis engine 400 (Fig. 9) stores in databases 900, analyzes and instructs the execution engine 500, and the execution engine 500 executes changes in the system components as defined below (Fig. 10). All operations on Fig. 7 utilize the communication device 26 whenever necessary or expeditious.

**[00129]** On the remaining figures, shapes referring to other figures, to operations external to the scope of the present figures, or to the subject of the present drawing, are indicated with dashed lines, and are shown only to place the described operations in the context of continuous and continual operations external to the drawing.

**[00130]** Referring now to FIG. 8, a flow chart of the preferred process for collecting and reporting information on containers is shown. The data reporting 600 and data collection 700 means utilizes subroutines within the analysis engines 400 and gateways 200 to submit and collect register information and sub level analysis to other analysis engines 400 or other gateways 200 of a higher (larger) logical set in a set pattern and frequency defined by the administrator.

**[00131]** Input is received from the user selecting "data reporting" 70100 from the "edit gateway" drop-down menu. Container levels are displayed 70200. Input is received from the user selecting container level 70300. A menu of all possible gateways 70320 and analysis engines 70330 residing on gateways on the above defined container class appears, depicted graphically as a tree of analysis engines and gateways at that container level. Input is received from the user selecting "source" from "source or destination." Input is received from the user 70400 selecting a container, containers, or class of container by clicking on the graphically depicted container(s) or container level on a display device. Input is received from the user 70410 selecting "destination" from "source or destination" Input is received

from the user 70500 selecting an analysis engine, analysis engines, or class of analysis engine by clicking on the graphically depicted analysis engine(s) or analysis engine level on a display device. A time scheduler is displayed. Input is received from the user 70510 selecting the reporting frequency for the selected gateways to report data to the selected engines. The data from the gateways is thenceforth continuously moved or copied to the analysis engines by the system 10 utilizing the execution engine 500 according to the defined schedule, rules and pattern 70420, 70520.

[00132] Input is received from the user selecting "choose container level" 70300 from the gateway editor drop-down menu. A menu 70320 appears listing the classes of containers on the system within the defined subset and superset scheme of multiple hierarchically nested containers, i.e.; element, document, file, database, warehouse, domain, appears. Input is received from the user selecting the class of containers. A graphic representation of that container level throughout the system appears. Input 70300 is received from the user selecting individual containers or all the containers in that class.

[00133] From the gateway editor drop-down menu input 70100 is received from the user selecting "data collecting" A menu of all possible gateways and analysis engines residing on gateways on the above defined container class appears, depicted graphically as a tree of analysis engines, and gateways at that container level. Input 70510 is received from the user selecting "source" from "source or destination." Input is received from the user selecting a container, containers, or class of container by clicking on the graphically depicted container(s) or container level. Input 70510 is received from the user selecting "destination" from "source or destination." Input 70510 is received from the user selecting an analysis engine, analysis engines, or class of analysis engine by clicking on the graphically depicted

analysis engine(s) or analysis engine level. A time scheduler appears. Input 70510 is received from the user selecting the collecting frequency for the selected engines to collect data from the selected gateways. The data from the gateways is thenceforth continuously moved or copied to the analysis engines by the system 10 utilizing the execution engine 500 according to the defined schedule, rules and pattern.

**[00134]** The data collection 700 means, utilizing the communication device 26 and an execution engine 500, comprises one or more subroutines or agents programmed to travel through the network collecting the accumulated data and analyses from selected analysis engines, gateways or selected subset level of analysis engines or gateways (as above) in a pattern and frequency defined by the gateway administrator at a given container level. Input 70510 is received from the user or administrator, defining the collection and reporting of data, thus controlling permission within his gateway, and being subject to permission levels defined by others beyond his gateway.

**[00135]** Input is received from the user or gateway administrator selecting collection or reporting 70100 and the system shows the container levels available 70200. Input is received from the user selecting a container level 70300. Input is received from the user selecting "gateway" 70400 or "engine" 70500. The system shows gateways 70320 or engines 70330 associated with that level. Input is received from the user editing the reporting parameters associated with a gateway or a class of gateways 70410 or an engine or class of engines 70510. Input is received from the user selecting the collecting frequency for the chosen engines. When input is received from the user choosing to user save the definition, the screen returns to the main container screen, step 70100 to make another selection available. Input is received from the user choosing to repeat the cycle, choosing

“destination” to describe the destination analysis engines and the data collecting frequency from those destination analysis engines. The data collection means 700 collects the accumulated gateway information in a pattern and frequency defined by the gateway administrator or user at a given container level.

[00136] The system utilizing the execution engine (see Fig. 10) distributes the new parameters to the gateways 70420 or engines 70520 by the communication device 26. Using the new parameters the gateways report to the analysis engines 70430 after, in some cases, conducting sub-analysis 70440, or using sub-analysis 70440 to submit directly to specified gateways under certain conditions and parameters, and the analysis engines collect from the gateways 70530. The analysis engine uploads, downloads and utilizes information to databases 900 to conducts its analysis.

[00137] The invention includes an analysis engine 400. Through the data reporting 600 means and data collection 700 the analysis engine 400 receives data and sub-analysis from the search interface and the gateways. Data includes, for each gateway 200, the frequency and grade of access, the description of the user accessing, the identity of the container 100 accessing, the register parameters, and the historically accumulated register data.

[00138] Referring now to FIG. 9, a flow chart of the operation of the analysis engine 400 is shown. Analysis engines 400 may reside at any gateway or anywhere in the system 10. The analysis engine 400, operating under its own programmed sequence, utilizing the communication device 26, works, by means of programmed rules of logical, mathematical, statistical or other analysis upon gateway and register information, in continuous interaction with the search process 410 and the data collection and reporting

process 420 to analyze, determine and compile instructions 40100 on container construction 40110 to containerize in an automated process 40115, on container contents 40120 to move, copy or delete containers 40125, on storage schemes 40130 to move or copy containers to new storage 40135, on access routes 40140 to alter gateway pointers to sought information 40145, on search templates 40150 to add, delete or change search phrases and the referenced objects indicated by those search phrases 40155 and on gateway instructions 40160 to alter gateway registers and pointers 40165.

[00139] Thus, analyses might include, but are not limited to, the physical locus of the users accessing, the demographic classification of the users accessing, the access frequency for a given container, the range or curve of time relevance affecting a container, the range or region of space relevance affecting a container 100, the number or number of a specific type of container 100 transiting a gateway 200, the hierarchically graded usage of containers 100 or container contents 01 compared with the demographic of those users accessing the container, the hierarchically graded usage of containers 100 or container contents 01 compared with search phrases entered into the search interface 300, the hierarchically graded usage of containers 100 or container contents 01 compared with search phrases entered into the search interface 300 compared with the demographic of the users accessing, the number of pertinent containers nested within a given container 100. Once an analysis is accomplished, the result is compared to pre-programmed rules triggering instruction sets (such as moving a container to nest within another container).

[00140] Instructions are then sent to the execution engine 40200, which utilizes the communication device 26 to execute the instructions derived from the analyses. These

containerized instructions transit the gateways 40300 and are utilized in the gateway process (Fig. 12)

[00141] Referring now to FIG. 10, a flow chart of the operation of the execution engine is shown. The execution engine 400, operating under its own programmed sequence in response to the instructions from the analysis engine 50100, utilizing the communication device 26, works in continuous process as its containerized execution instructions transit the gateways 50200 to create containers 50210 in an automated containerization process 50215, alter container contents 50230 by moving or copying containers to new containers 50235, to alter storage 50240 by moving or copying containers to new storage 50245, to alter access routes 50250 by altering gateway pointers 50255, to alter search templates 50260 by adding, changing and deleting search phrases and the referenced objects indicated by those search phrases 50265, to alter gateway instructions 50270 by altering gateway registers and pointers 50275. The execution works in a continuous loop with the gateway process 50300, the data collection and reporting process 50400 and the analysis engine process 50300.

[00142] The invention includes gateways 200. Gateways may be placed and reside anywhere on the network where containers transit. Gateways also reside on any or all containers. The gateway reads and stores the chosen register information from transient containers entering or exiting its logical boundaries. The resident analysis search engine, if any, performs the specified level of analysis. Data and analysis is both held for the collection means according to the pattern and timing specified in the data reporting 600 editor and submitted according to the pattern and timing specified in the data collection means editor 700.

[00143] The gateways are network-wide, hierarchical, and nestable, and reside with a container encompassing any component, digital code, file, search string, set, database, network, event or process and maintaining a unique lifelong network wide identity and unique in all the universe historical identity, or may be strategically placed at such container transit points to gather and store register information attached to any such container, according to system-defined, system-generated, or user determined rules residing in its registers defining the behavior of those containers and components as they exit and enter one another, or interact with one another or any system process or system component within the logical domain of that container, or after exiting and entering that container, or defining how they interact with that unique gateway.

[00144] Gateway's registers comprise both system-defined and user-defined registers, alterable by author, duration, location, network-wide history, individual container history and/or interaction with other containers, gateways, networks or media, and evolve according to that gateway's history on a computer network, or according to the network history of events and processes, or according to that information component's interaction with other information containers, components, system components, network events or processes.

[00145] Referring now to FIG. 11, a flow chart of the gateway editor is shown. From the main title bar input is received from the user selecting "containerize" or "gateway level" 20100. When input is received from the user selecting "containerize" the system enters the container editor process 110. When input is received from the user selecting "gateway," the system shows the gateway levels available 20200. A menu of all possible gateways within the subset and superset scheme of defined multiple hierarchically nested



gateways appears. Input is received from the user selecting the gateway level 20300. The system searches the gateways 20500 to locate the available gateway templates 20700 and the available gateways 20600. Input is received from the user selecting the gateway 20610 or gateway level template 20720. The system goes to the gateway 20620 or to the template 20720. A graphic representation of the chosen gateway 20630 or template 20730 appears. Input is received from the user to edit 20640 or create a gateway 20740. Once completed, input may be received from the user selecting "analysis level" from the gateway 200 drop-down menu, to select the level of analysis in a multi-level analysis sequence to be accomplished at the local level by a gateway-resident analysis engine. The user accesses the container editor to containerize (Fig. 5). Input is received from the user selecting the registers by clicking on the graphically depicted register, or from a drop down menu. Input is received from the user setting the registers as described elsewhere in ("container registers"). Input is received from the user selecting or defining the rules governing the interaction of that gateway with transient containers. Input is received from the user selecting or defining the rules governing the interaction of containers existing within the logical domain of the container 100 to which that gateway is attached. The user publishes the gateway (Fig. 5). Input is received from the user selecting "residence" from the main menu bar. ). Input is received from the user choosing to leave the gateway where it was created, move it to container on another drive, directory, computer, or network. If the user chooses "move," a browse function allows the user to name the new location or browse a list of possible locations. Once input is received from the user choosing the residence of the gateway, the program restores the search interface screen.

[00146] The invention includes a data reporting means editor 610, and a data collection means editor 710, Fig. 2 A, as a menu option under the gateway editor 210.

[00147] The present invention also includes a gateway process.

[00148] Referring now to FIG. 12, a flow chart of the gateway process is shown. A system operation, search process or element container or process container is shown in transit 21100 passing through a gateway 21200. The container, operation or process interacts with the gateway 21300, uploading, downloading and exchanging information with the container, operation or process. The gateway stores container information 21400 and the container registers are altered 21500. The container registers also interact with the search interface 21600. The gateways report the register information or make it available for collection by the data reporting and collection means (Fig. 8) operating on the communication device 26 to provide the information to the analysis engine 21800, which stores 90100, analyzes and instructs the execution engine 21900, which processes and instructions are also stored 90100 by the execution engine upon receipt.

[00149] All operations in Fig. 12 utilize the communication device 26 whenever necessary or expeditious.

[00150] Referring now to FIG. 13 A, a drawing of nested containers 100 prior to the container modification process on a network 201 is shown. (Note: The same container numbering scheme is used in Fig. 13 A, 13 B, 13 C, 13 D and in 2 B.) Information containers 505 and 909, residing within container 908, operating under the rules governing container interaction within that container 908 downloaded to container 505 and 909 from gateway 9081 upon their entrance to container 908, which rules had been downloaded from execution engine 500 acting under the direction of analysis engine 400, and under the rules programmed

into their own registers 404120, 909120, compare the specified (by those rules) set of registers 404120, 909120, i.e., time and space, and determine a container 404 encapsulated within 505 would be more appropriately encapsulated within container 909.

[00151] Referring now to FIG. 13 B a drawing of nested containers during a container modification process on a network 201 is shown. Container 404 is moved to reside with container 909. As the container 404 exits container 505, the gateway of container 505, being gateway 5051, operating under the rules governing container interaction with a gateway 5051 upon egress or egress as programmed in the gateway editor 210 and modified by the execution engine 500 executing the instructions of the analysis engine 400, or any greater logical analysis engine 408 providing execution instructions to an execution engine 508 operating in a larger encompassing container 108 entering through that container's gateway 208 or an independent gateway 707, or sub-analysis engine operating at any gateway level, records the register information of container 404. The gateway 5051 reports the transaction to the gateway 9081 of container 908, being the next higher logical container. Gateway 9081 holds in gateway storage 205 the information until collected by one or more data collection processes 700, or reported to one or more data reporting processes 600, serving one or more analysis engines 400 residing independently on the system 10 or an analysis engine at higher logical container 303. The analysis engine 400, comparing reports of user hierarchically graded usage under the operations of the search engine 320 and the search interface 300, on information container 808 after receiving reports from the data reporting means of container 404 being moved to container 909 determines, i.e., that the number of time and space relevant containers residing within container 909 is sufficient to warrant an action, and directs the execution engine 500 to copy container 909, nested within container 908, to a

third information container 808. As the copy instruction from execution engine 500 transits the gateway of container 908, the gateway 9081 records the instruction. The copy instruction interacts with the registers 909120 of container 909 regarding the rules governing its copying to another location. Once approved by the governing rules of registers 909120 appended to container 909, container 909 is duplicated. As the duplicate container 909 exits the container 908, the gateway records the register information 909120 of container 909, and the registers 909120 of container 909 are altered by special instructions from gateway 9081 under the rules residing in gateway 9081 regarding ingress and egress and the rules residing in the registers 909120 of container 909 regarding alteration by gateways upon ingress and egress. Passing through independent gateway 707, the register information 909120 is recorded, and awaits data collection or reporting 700, 600. As container 909 enters container 808, the gateway records the register information 909120 of container 909, the registers 909120 of 909 are altered by special instructions from gateway 8081, operating under the rules as described in the paragraph above, and container 909 takes up residence within container 808.

[00152] Referring now to FIG. 13 C, a drawing of nested containers after the container modification process on a network 201 process is shown. Container 909, now also logically residing within container 808, commences to interact with other containers 606 in 808 under the rules governing container interaction within container 808 as received from gateway 8081 upon transiting that gateway, and under the rules of registers 606120, 909120 of the interacting containers 606, 909, operating under the rules as described in the paragraph above. Through data collection and reporting 700, 600, analysis engine is appraised of container's 909 new duplicate residence. I.e., operating under the registers of space relevance, a body of law pertaining to Boston Municipal tax law may be housed in a

container holding Massachusetts tax law, but it would be more appropriately located in a container holding Boston tax law, with only a pointer to that location residing in the Massachusetts tax law container. In this example, such an analysis could be accomplished by comparison of zip code information in the space registers, or logical rules-based analysis, with "state" being a larger set than "city". Or, i.e., operating under the registers of time relevance, the curve of time relevance for a concert might follow an ascending curve for the months prior, hit a brief plateau, and then reach a precipitous decline, at which time certain pertinent information only might be moved to an archival container of city events or rock concerts of that year. In this example, once the curve is mapped into a register, that map would cause an increasing frequency of pointers to that container in other containers or gateways, or inclusion of that container in other containers, as the analysis engine compares that curve with increasing user inquiry.

[00153] Referring now to Fig. 13 D, a flowchart of the reconstruction process is shown.

[00154] Information containers 505 and 909, residing within container 908, operating under the rules governing container interaction within that container 908 downloaded 888103 to container 505 and 909 from gateway 9081 upon their entrance to container 908, which rules had been downloaded 888102 from execution engine 500 acting under the direction 888101 of analysis engine 400, and under the rules programmed into their own registers 404120, 909120, compare 888104 the specified (by those rules) set of registers 404120, 909120, i.e., time and space, and determine 888105 a container 404 encapsulated within 505 would be more appropriately encapsulated within container 909.

[00155] Container 404 is moved 888106 to reside with container 909. As the container 404 exits container 505, the gateway of container 505, being gateway 5051, operating under the rules governing container interaction with a gateway 5051 upon egress or egress as programmed in the gateway editor 210 and modified 888108 by the execution engine 500 executing the instructions of the analysis engine 400, or any greater logical analysis engine 408 providing execution instructions 888107 to an execution engine 508 operating in a larger encompassing container 108 entering through that container's gateway 208 or an independent gateway 707, or sub-analysis engine operating at any gateway level, records 888109 the register information of container 404, and alters the register information of container 404. The gateway 5051 reports 888110 the transaction to the gateway 9081 of container 908, being the next higher logical container. Gateway 9081 holds 888111 in gateway storage 205 the information until collected by one or more data collection processes 700, or reported to one or more data reporting processes 600, serving 888112 one or more analysis engines 400 residing independently on the system 10 or an analysis engine at higher logical container 303. The analysis engine 400, comparing 888114 reports of user hierarchically graded usage on information container 808 under the operations of the search engine 320 and the search interface 300, after receiving 888113 reports from the data reporting means of container 404 being moved to container 909, determines 888115, i.e., that the number of time and space relevant containers residing within container 909 is sufficient to warrant an action, and directs 888115 the execution engine 500 to copy container 909, nested within container 908, to a third information container 808. As the copy instruction from execution engine 500 transits the gateway of container 908, the gateway 9081 records 888116 the instruction. The copy instruction interacts 888117 with the registers 909120 of

container 909 regarding the rules governing its copying to another location. Once approved 888118 by the governing rules of registers 909120 appended to container 909, container 909 is duplicated 888118. As the duplicate container 909 exits the container 908, the gateway records 888119 the register information 909120 of container 909, and the registers 909120 of container 909 are altered 888120 by special instructions from gateway 9081 under the rules residing in gateway 9081 regarding ingress and egress and the rules residing in the registers 909120 of container 909 regarding alteration by gateways upon ingress and egress. Passing through independent gateway 707, the register information 909120 is recorded 888121, and awaits 888122 data collection or reporting 700, 600. As container 909 enters container 808, the gateway records 888123 the register information 909120 of container 909, the registers 909120 of 909 are altered 888124 by special instructions from gateway 8081, operating under the rules as described in the paragraph above, and container 909 takes up residence 888125 within container 808.

[00156] Container 909, now also logically residing (in addition to its original container residence) within container 808, commences to interact 888126 with other containers 606 in 808 under the rules governing container interaction within container 808 as received from gateway 8081 upon transiting that gateway, and under the rules of registers 606120, 909120 of the interacting containers 606, 909, operating under the rules as described in the paragraph above. Through data collection and reporting 700, 600, analysis engine is appraised 888127 of container's 909 new duplicate residence.

[00157] Referring now to Fig. 14, the screen interface of the container editor is shown. This interface is a process wherein input is received by the user using the main menu 78 or drop down menu 1419, or using an input device to “drag and drop” or click, causing the

system 10 to acquire 1409, edit 1410 or create 1411 a file 1407, container 1408 or digital content 01, to search for 1412, acquire 1413, edit 1414 or create 1415, print 1416, or containerize 1417 a container 100, to select 1402, (or by clicking on register), search 1403, acquire 1404, edit 1405, or create a register 1406 to append or detach registers 120 to those containers, to set register values in those registers 120, to utilize the register editor 125 through 1405 to create new registers, or to 1418 add, detach, acquire a gateway 200 to append or detach to those containers, and utilize the gateway editor 210 through 1418. (See detailed description referring to Fig. 5)

**[00158]** Referring now to Fig. 15, the screen interface of the gateway editor is shown. This interface is a process wherein input is received by the user using the main menu 1501 or drop down menu 1513, or using an input device to “drag and drop” or click, causing the system 10 to search for 1507, acquire 1508, edit 1509 create 1510, print 1511 or containerize 1512 gateways, and causing the system 10 to establish rules by which an individual gateway governs the transiting 1502, entering 1503, exiting 1504 of containers and the interaction of containers within its domain 1505, and external of its domain.1506. (See detailed description referring to Fig. 11).

**[00159]** Referring now to Fig. 16, the screen interface of the search interface. This interface is a process wherein input is received by the user using the main menu 1625 or drop down menu 1624, or using an input device to “drag and drop” or click, or by entering text, causing the system 10 to select 1615, search for 1616, acquire 1617, edit 1618 create 1619, print 1620, containerize 1621 (by accessing the container editor 110) or insert 1622 digital search strings into the search box 1623 in order to submit that string to the search engine 320, or causing the system 10 to select 1602, search for 1603, acquire 1604, edit 1605, create



1612, containerize 1613 (by accessing the container editor 110), or insert 1614 search keys (templates that comprise search scope in geographic range, container level, and specific key words or digital strings), or containerized searches (containers 110), into the search box 1623 in order to submit that string to the search engine 320, or causing the system 10 to set a search range by geographic range 1607, container level 1608, or acquire 1609, edit 1610 or create 1611 a scope template. (templates that comprise search scope in geographic range and, container level.) (See detailed description referring to Fig. 6).

[00160] Referring now to Fig. 17, a drawing showing, on an input device or computer screen 24, in any generic (dashed lines) software application program, a drop-down menu link 1403 on a drop down menu 1402 dropping down from a main menu 1401, and a free-floating button link 1404, is shown. When input is received at 1402 or 1403, the system 10 makes available to the user the containerization process or container editor 110. When input is received at drop-down menu link 1405 or a button link 1406, the system 10 makes available to the user the means to enter and interact with this system 10 or this network 201 in any of their aspects. The interfaces 1403, 1404 show a process wherein input is received causing the system 10 to encapsulate content or access the container editor 110. The link also allows the user to encapsulate the page or file on which he is currently working, without selecting content, and if so desired, without accessing the container editor. The interfaces 1405, 1406 show a process wherein input is received causing the system 10 to access or interact with the system 10 or the network 201.

[00161] The present invention also includes a search engine 320. Once the key word(s), phrase or digital segment is entered into the search interface 300, or an offered

selection chosen on the menu, it is utilized by the search engine 320 to locate the desired site or data.

[00162] The search engine employed may be any industry standard search engine such as Verity “Topic”, or Personal Library Software, as used in Dow Jones News Retrieval, or Internet search engines such as Webcrawler, Yahoo, Excite, Infoseek, Alexa or any Internet search engine, or any new engines to be developed capable of searching for and locating digital segments, whether text, audio, video or graphic.

[00163] The present invention also includes an analysis engine 400. Utilizing rules-based analysis, the analysis engine determines the class of storage medium upon which containers reside, the subsets and supersets by which and in which containers encompass and reside within one another, the routes of access to those containers, the historically successful search parameters by which those containers are accessed based upon the identity of the user accessing the containers, and the grade of access chosen by the user in accessing that container 100.

[00164] Utilizing a pre-programmed sequence of compilation, and inductive, deductive and derivative analysis, the analysis engine manufactures instructions based upon the analysis of the information submitted by the gateways and the search interface, and submits those instructions to the appropriate execution engine 500 in order to create new information containers, content assemblages, storage schemes, access routes, search templates, and gateway instructions, and others, and to provide informed search options through the search interface to the inquiring user.

[00165] The present invention also includes an engine editor 510, that provides a system administrator with a means of editing the operating principles of that search engine,

and search template loading in the search interface 300, a reporting and collection means editor 610, 710, governing data reporting 600 and data collection 700 at the gateways 200 as defined by the gateway editor 210 and the register editor 125, a container editor 110 for creating and modifying containers and appending registers to containers, a register editor 125 for creating and modifying container registers and establishing and adjusting the values therein, container gateways 200 with their own storage 205, information containers 100 for holding information and container registers for holding information about specific containers and their history on the network.

[00166] The present invention also includes an execution engine 300. Based upon instructions received from the analysis engine 400 utilizing the communication device 26, the execution engine 500 provides search phrases to the search interface 300 based upon initially received inquiries, relocates containers including their programs, data and registers to other directories, drives, computers, networks on other classes of storage mediums, i.e., tape drive, optical drive, CD-ROM, deletes, copies, moves containers to nest within or encompass other containers on other directories, drives, computers, networks to nest within other containers, alters the class of storage medium upon which containers reside, the subsets and supersets by which and in which containers encompass and reside within one another, the routes of access to those containers, and the historically successful search parameters by which those containers are accessed based upon the identity of the user accessing the container and the grade of access chosen by the user in accessing that container.

[00167] The execution engine 400 fulfills the instructions of the analysis search engine 500, to create new information containers, content sub and superset assemblages, storage schemes, access routes, search templates, gateway 200 instructions and other system

functions. The execution engine includes an editor 510 that provides a system manager with a means of editing the operating principles of that search engine, governing data reporting, data collection 700, search template loading, gateway instructions, and other functions.

[00168] The present invention also includes flat or relational databases 900, used where, and as required.

[00169] The present invention also includes a communication device 26 supporting all operations on a network wide basis.

[00170] The present invention also includes a search engine 300 to locate the desired site or data. The present invention also includes databases 900, flat or relational, to serve the other components of the system as needed and where needed.

[00171] The present invention also includes editors, by which the user may alter the governing aspects of the system. Editors include, but are not limited to, a container editor 110, a register editor 125, a gateway editor 210, an engine editor 510, a reporting means editor 610, a search interface 300, and a collection means editor 710.

[00172] The present invention also includes specific screen interfaces for the editors, as described in Fig. 14, Fig. 15. and Fig. 16.

[00173] The present invention also includes a means for this system 10 and network 201 or container editor 110 to be accessed from a menu or button selection within any program, as described in Fig. 17.

[00174] While the present invention has been described with reference to certain preferred embodiments, those skilled in the art will recognize that various modifications may be provided. For example, both analysis engine and execution engine may be duplicated or modified for distribution at various locations and hierarchical positions in the gateway and

container system throughout the network and designed to work in concert. Also, the physical computing infrastructure may be mainframe, mini, client server or other with various network and distributed computing designs, including digitally supported or based physical or public media, and the components of the system 10, as described in Fig. 1 may be physically distributed through space. Even the contents of a single container may be logically referenced but be physically distributed through the network and reside at multiple storage locations. The whole system may be hierarchically nested within other systems to the nth degree. Whole systems may also be encapsulated within containers. A single container may also encompass a single physical media, such as a CD-ROM disk, programmed with the container, gateway and register design. Gateways may be strategically placed on containers at ingress and/or egress points or may be placed strategically throughout the system for optimal collection and reporting output and gateway system control. Also, the loop of gateway data collection and reporting, analysis engine analysis, instruction, and gateway modification, and execution engine operations may be infinitely nested, from the smallest container of two sub-containers to whole networks holding millions of containers and thousands of levels, with analysis itself nested within the multiple levels. Gateways may be established at both logical and physical junctures such as a satellite uplink point. Also, the provision to establish a unique network identity might be designed to include as of yet unknown computer networks as they arise. The analysis and execution engines may operate on a rules-based, fuzzy logic, artificial intelligence, neural net, or other system not yet devised. Other variations upon and modifications to the preferred embodiments are provided for by the present invention, which is limited only by the following claims. Also, the classification scheme of nested containers, while designated by the container creators, may transform, be utilized otherwise, or be wholly

discarded according to usage. Also, hardware configurations, such as the use of RAM or hard drives for storage or lasers for communication may assume myriad forms without altering the essential operation of this invention.

WHAT IS CLAIMED IS:

1. A method comprising:  
receiving a search query;  
searching container registers encapsulated and logically defined in a plurality of containers to identify one or more containers responsive to the search query; and  
providing a list characterizing the identified containers.
2. A method as in claim 1, when the received search query comprises a labeled data tree having at least one parent-child relationship.
3. A method as in claim 1, further comprising: providing information identifying containers that have previously been used to respond to one or more processed queries that are substantially similar to the search query.
4. A method as in claim 1, wherein the provided information is stored in one or more search templates.
5. A method as in claim 1, further comprising: providing information identifying substantially similar search phrases, search templates, or labeled data trees that have previously been used to respond to one or more processed queries that are substantially similar to the search query.

6. A method as in claim 5, further comprising:  
receiving a selection of one of the substantially similar search phrases; and  
providing a list of previously identified containers associated with the selected search phrase.
7. A method as in claim 1, wherein the list provides a title of each identified container and a short description of its contents.
8. A method as in claim 1, further comprising: receiving a container search level parameter; and wherein the searching content and container registers only searches within container levels associated with the container search level parameter.
9. A method as in claim 1, further comprising: receiving a container search level parameter; and wherein the list of identified containers only comprises containers associated with the container search level parameter.
10. A method as in claim 1, wherein the searching further comprises: encapsulating the search query into a search container.
11. A method as in claim 10, wherein the searching further comprises:  
receiving, by a gateway, the search container;  
storing, by the gateway, data contained within a register of the search container; and



determining whether any registers of containers accessible via the gateway are associated with the register of the search container.

12. A method as in claim 11, further comprising:

generating a new gateway; and

associating the container with the new gateway.

13. A method as in claim 11, further comprising: periodically aggregating the contents of registers in a plurality of gateways to characterize a plurality of containers coupled thereto.

14. A method as in claim 11, wherein the contents of the registers in each of the plurality of gateways comprise at least one metric chosen from a group comprising: frequency of access of the gateway, grade of access of the gateway, description of users that have accessed the gateway, an identity of containers that have accessed the gateway, parameters associated with the gateway register, and historically accumulated register data.

15. A method as in claim 11, further comprising: monitoring transactions involving one or more gateways or containers.

16. A method as in claim 15, further comprising: generating new containers based on the monitored transactions.

17. A method as in claim 15, wherein the transactions are based on each instance a gateway or container passes through another gateway or container.

18. A method comprising:  
receiving a search query;  
polling a plurality of gateways to identify registers encapsulated therein, the registers relating to one or more containers logically defining data contained therein associated with the search query, wherein the containers are coupled to the gateways; and  
providing a list characterizing the identified containers.

19. A method comprising:  
reporting, by a plurality of gateways, registers or register values encapsulated therein, the registers relating to one or more containers coupled to the gateways logically defining data contained therein;  
updating a centralized index based on the reporting; and  
modifying at least one of a register, a register value, content of a container, stored information in a gateway, and an access route to any of the containers based on the updating.

20. A method as in claim 19, wherein the reporting occurs asynchronously or periodically after at least one interaction of container through the registers in which at least one of the following is altered: a register, a register value, the contents of a container, stored information in a gateway, and an access route to any of the containers.

21. A computer program product, tangibly embodied on computer-readable media, comprising instructions operable to cause a data processing apparatus to:

receive a search query;

search content and container registers encapsulated and logically defined in a plurality of containers to identify one or more containers associated with the search query; and

provide a list characterizing the identified containers..

22. A computer program product, tangibly embodied on computer-readable media, comprising instructions operable to cause a data processing apparatus to:

receive a search query;

poll a plurality of gateways to identify registers encapsulated therein, the registers relating to one or more containers logically defining data contained therein associated with the search query, wherein the containers are coupled to the gateways; and

provide a list characterizing the identified containers.

23. An apparatus comprising:

means for receiving a search query;

means for searching content and container registers encapsulated and logically defined in a plurality of containers to identify one or more containers associated with the search query; and

means for providing a list characterizing the identified containers.

24. An apparatus comprising:

means for receiving a search query;

means for polling a plurality of gateways to identify registers encapsulated therein, the registers relating to one or more containers logically defining data contained therein associated with the search query, wherein the containers are coupled to the gateways; and

means for providing a list characterizing the identified containers.

25. A method comprising:

receiving historical data from a plurality container registers encapsulated and logically defined in a plurality of containers, the historical data associated with interactions of the containers with other containers via those registers; and

modifying at least one of a register, a register value, content of a container, stored information in a gateway associated with a container, or an access route to any of the containers associated with any of a gateway based on the polling.

26. A method as in claim 25, when the obtained historical data comprises a labeled data tree having at least one parent-child relationship.

27. An apparatus comprising:

a processor; and

a memory unit operable to encode instructions operable to cause the processor to:

poll a container registers encapsulated and logically defined in a plurality of containers to obtain historical data associated with interactions of the containers with other containers; and

modify at least one container register or encapsulated content of at least one container based on the obtained historical data.

28. A method comprising:

polling gateways to obtain historical data associated with interactions of a plurality of containers with other containers, the containers encapsulating and logically defining container registers storing, the gateways intercepting data associated with the containers during the interactions; and

modify at least one of the container registers, content encapsulated within at least one of the containers based on the obtained historical data, or information stored in at least one gateway based on the polling.

29. An apparatus comprising:

a processor; and

a memory unit operable to encode instructions operable to cause the processor to:

poll gateways to obtain historical data associated with interactions of a plurality of containers with other containers, the containers encapsulating and logically defining container registers storing, the gateways intercepting data associated with the containers during the interactions; and

modify at least one of the container registers, content encapsulated within at least one of the containers, or information stored in at least one gateway based on the obtained historical data.

30. A method comprising:
- receiving a search query;
  - searching container registers encapsulated and logically defined in a plurality of containers to identify one or more search query templates; and
  - providing a list characterizing the identified one or more search query templates to formulate subsequent search queries.

ABSTRACT

A search query may be run against a plurality of container registers encapsulated and logically defined in a plurality of containers to identify one or more container registers responsive to the search query. Thereafter, a list characterizing the identified containers may be provided. Related methods, apparatus, computer program products, and computer systems are also described.

10570207.DOC

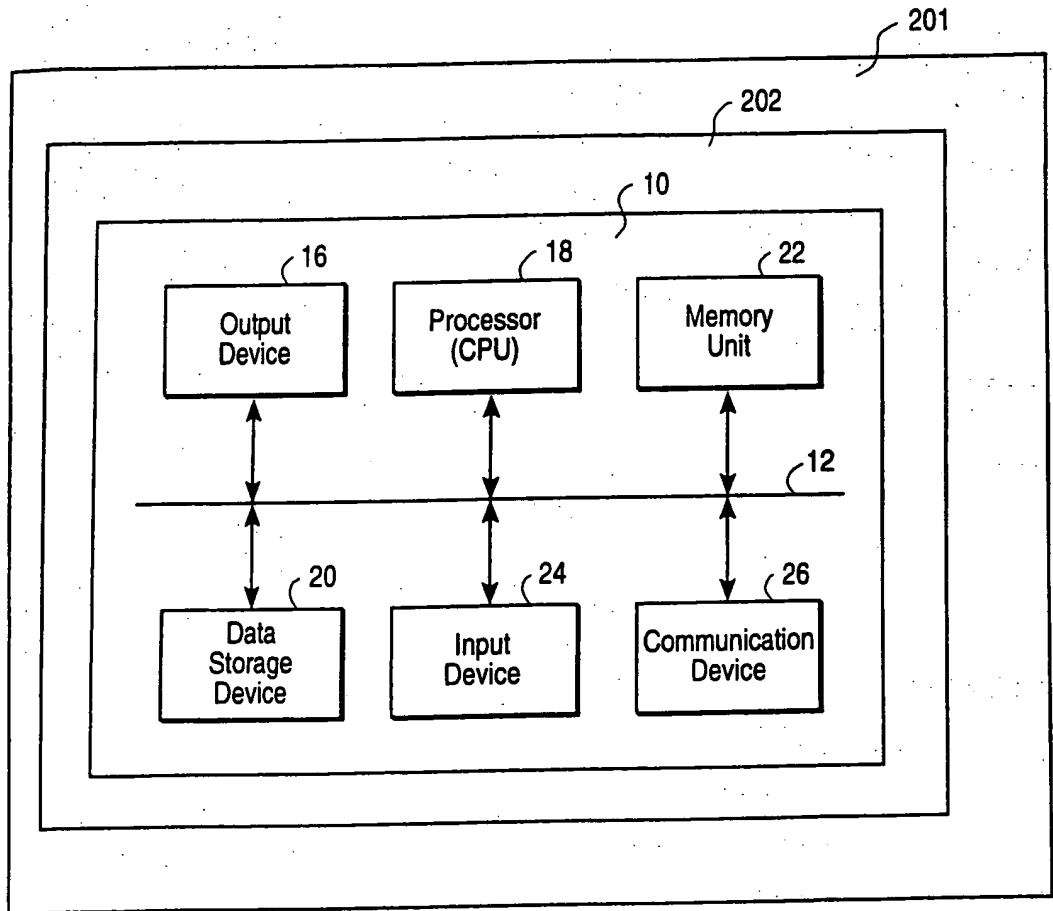


FIG. 1



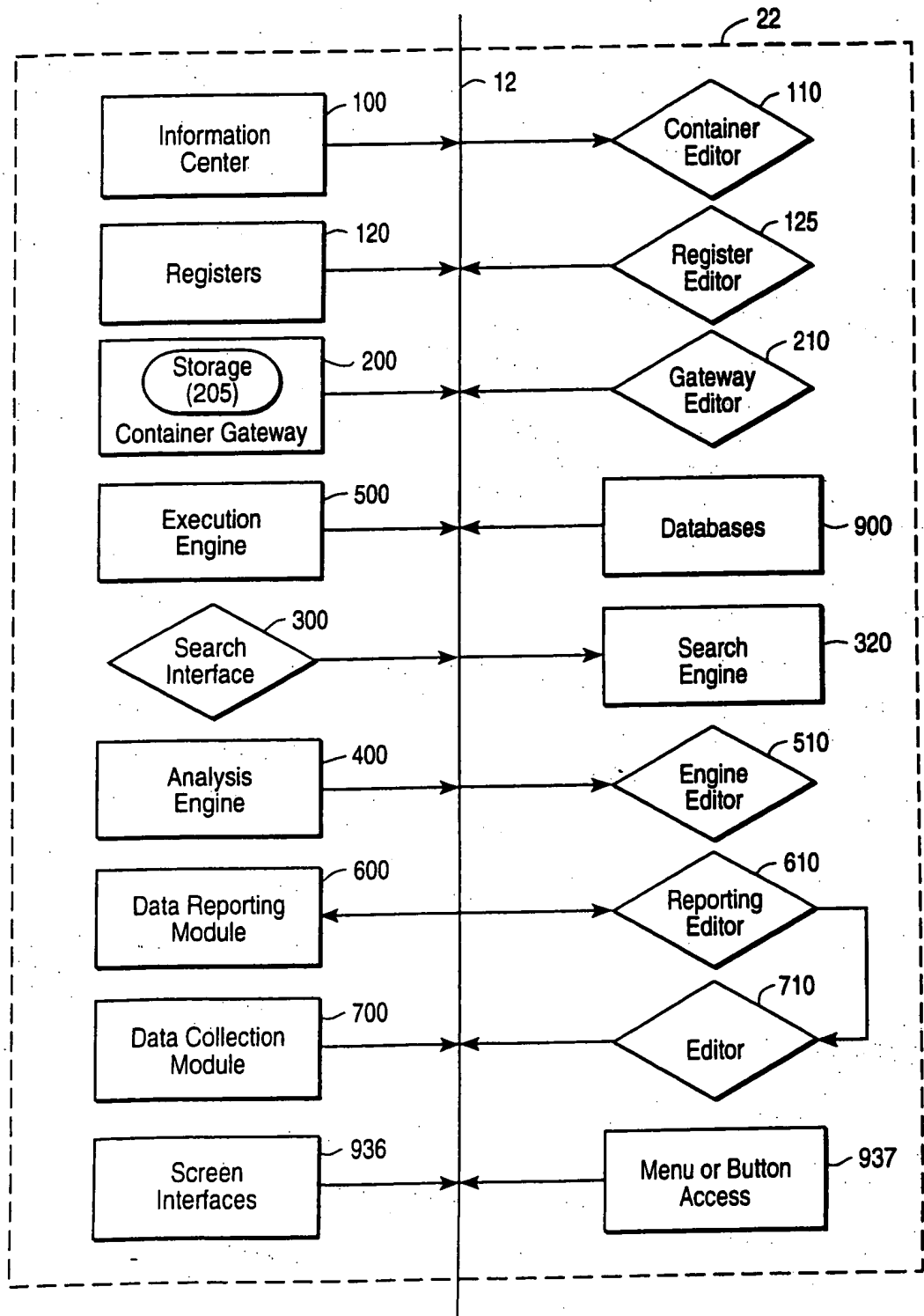


FIG. 2A

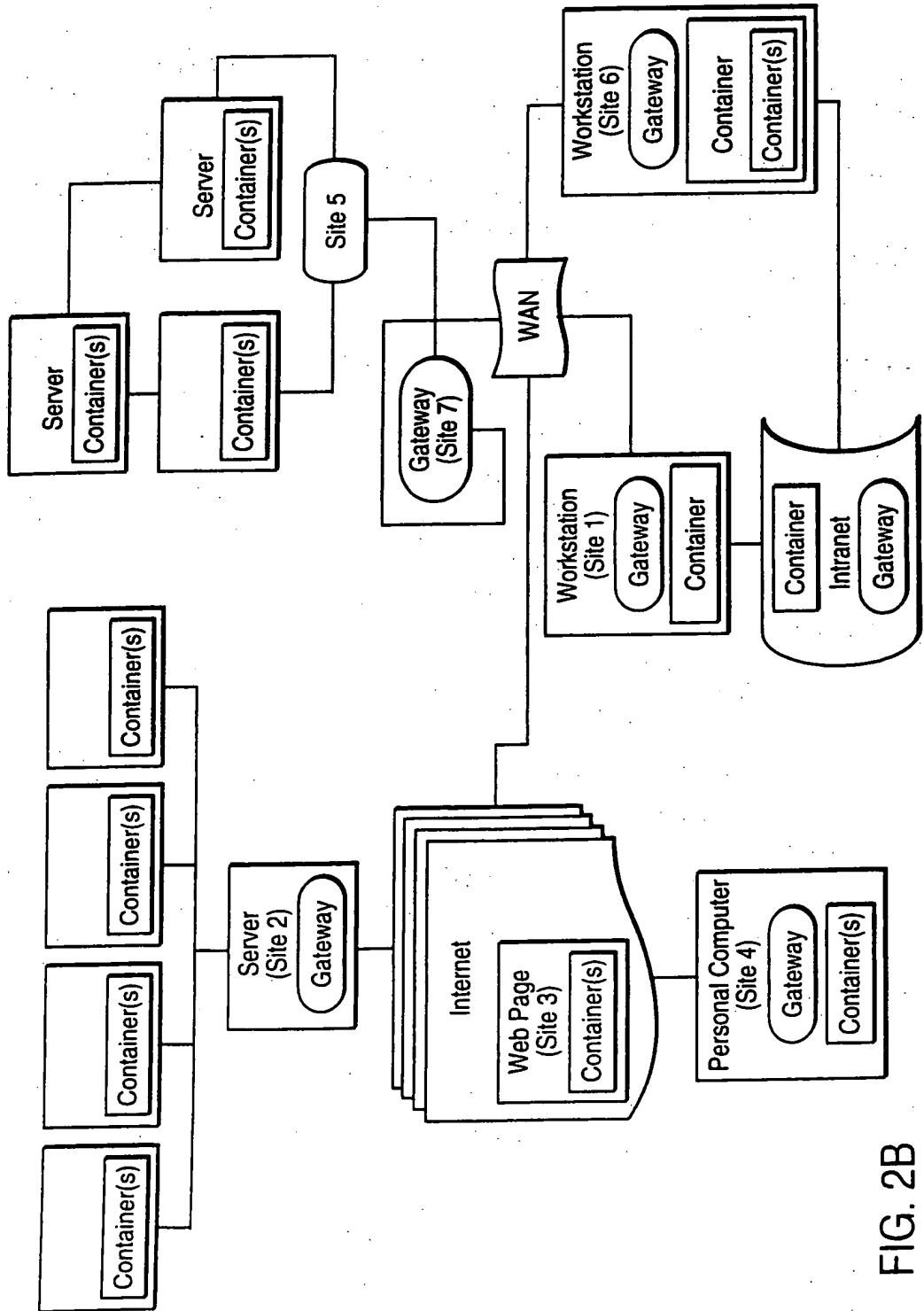


FIG. 2B

Applicant(s): Michael De Angelo  
SYSTEM AND METHOD FOR CREATING AND  
MANIPULATING INFORMATION CONTAINERS WITH  
DYNAMIC REGISTERS

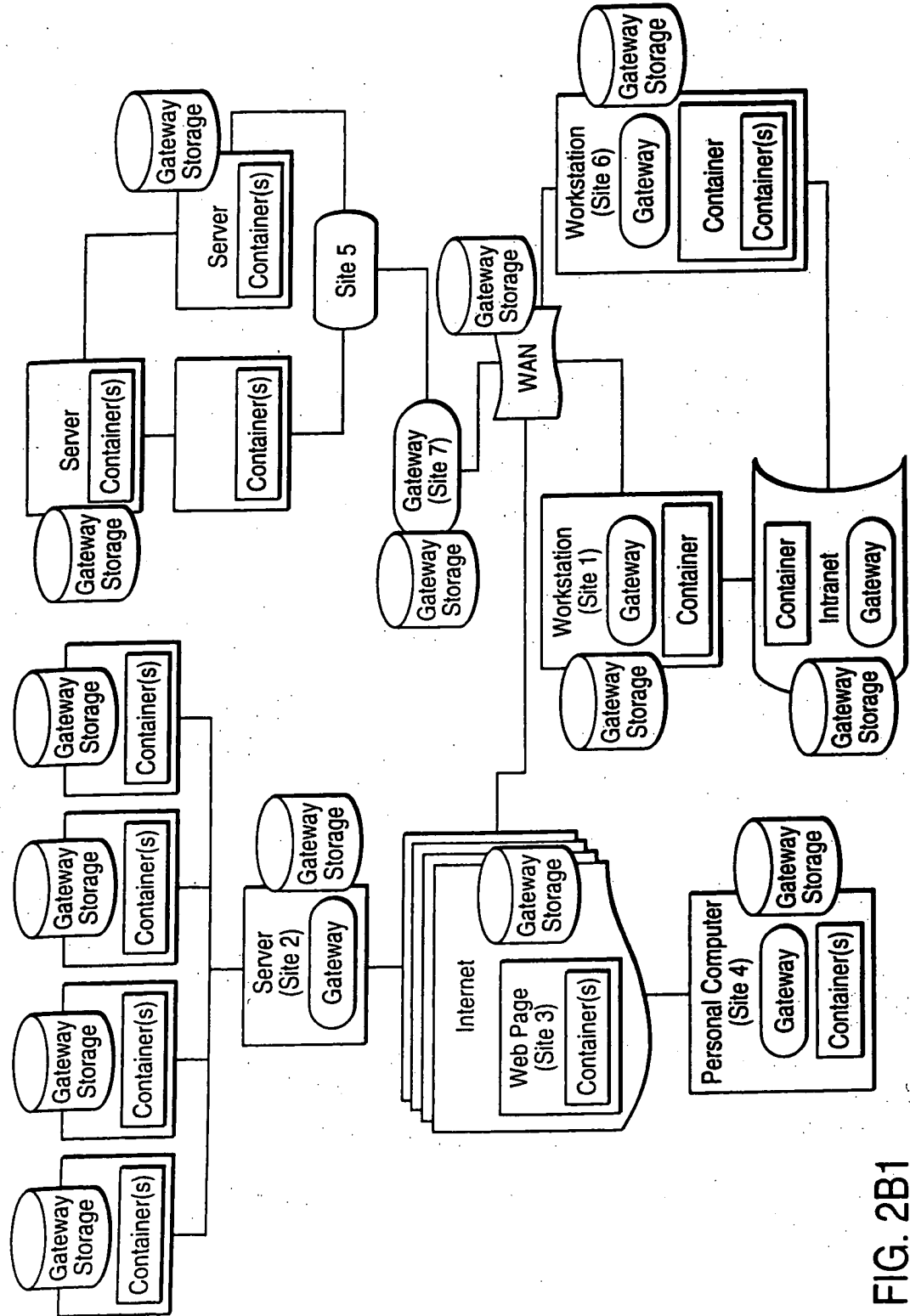


FIG. 2B1

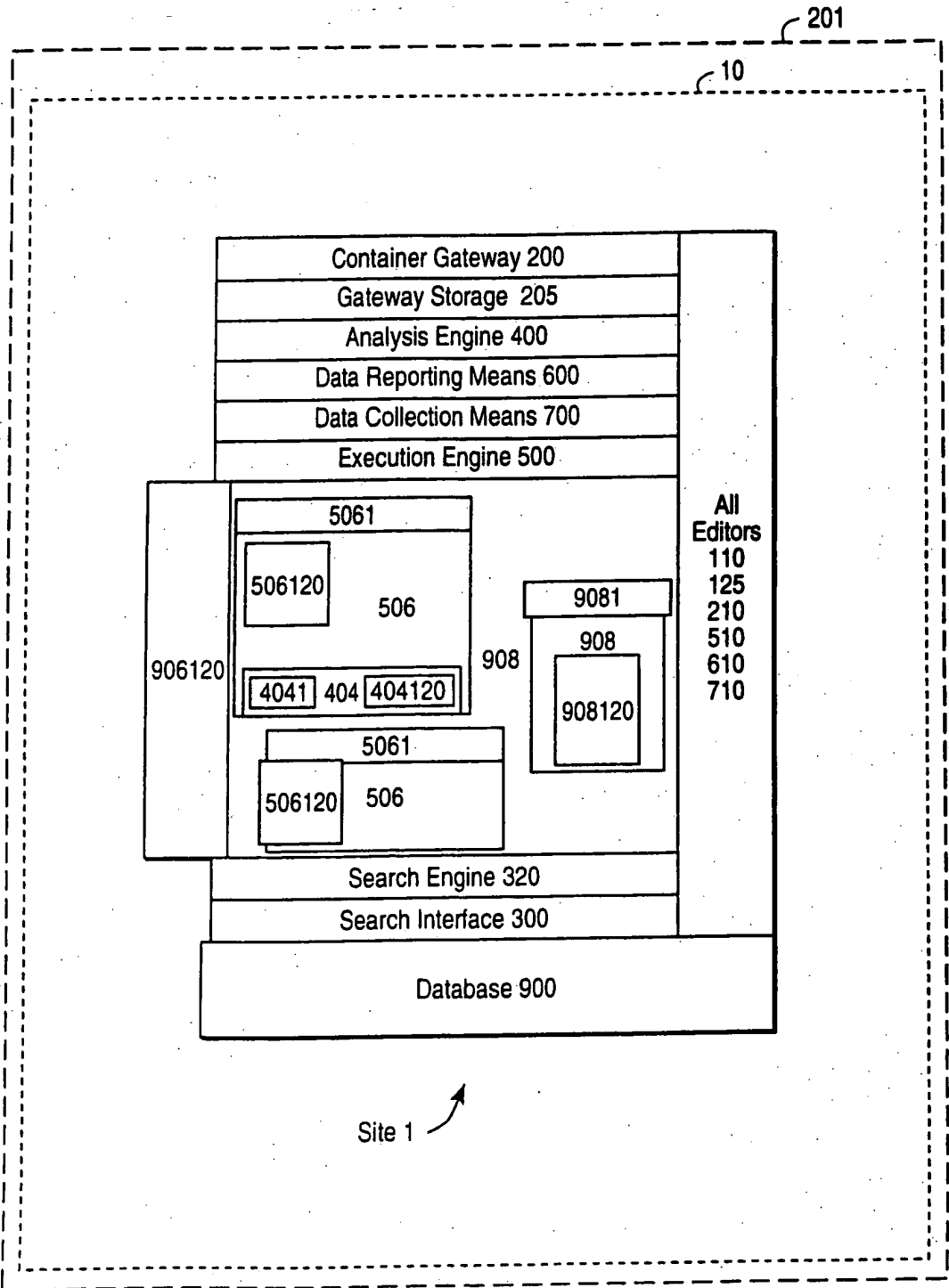


FIG. 2C

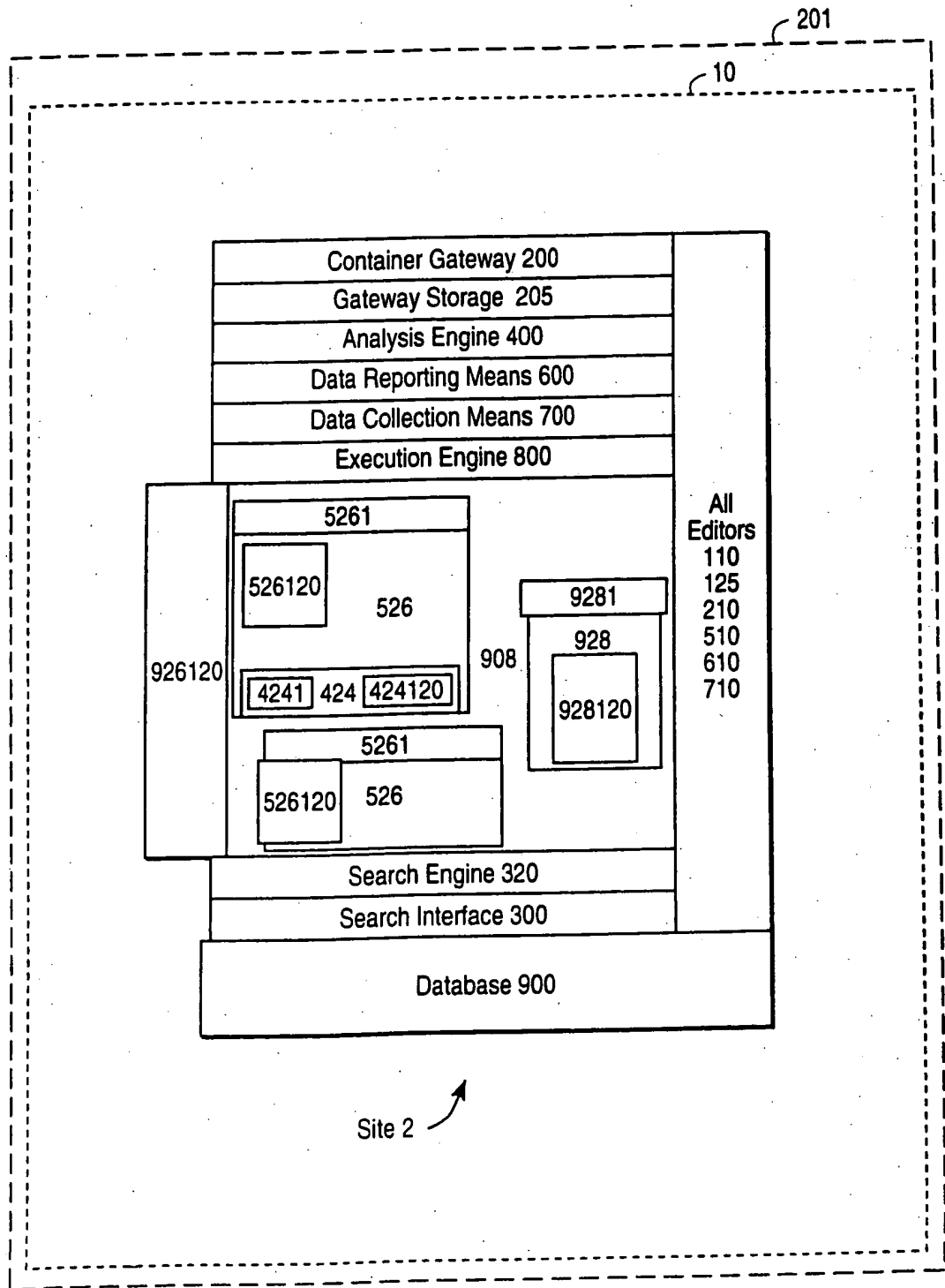


FIG. 2D

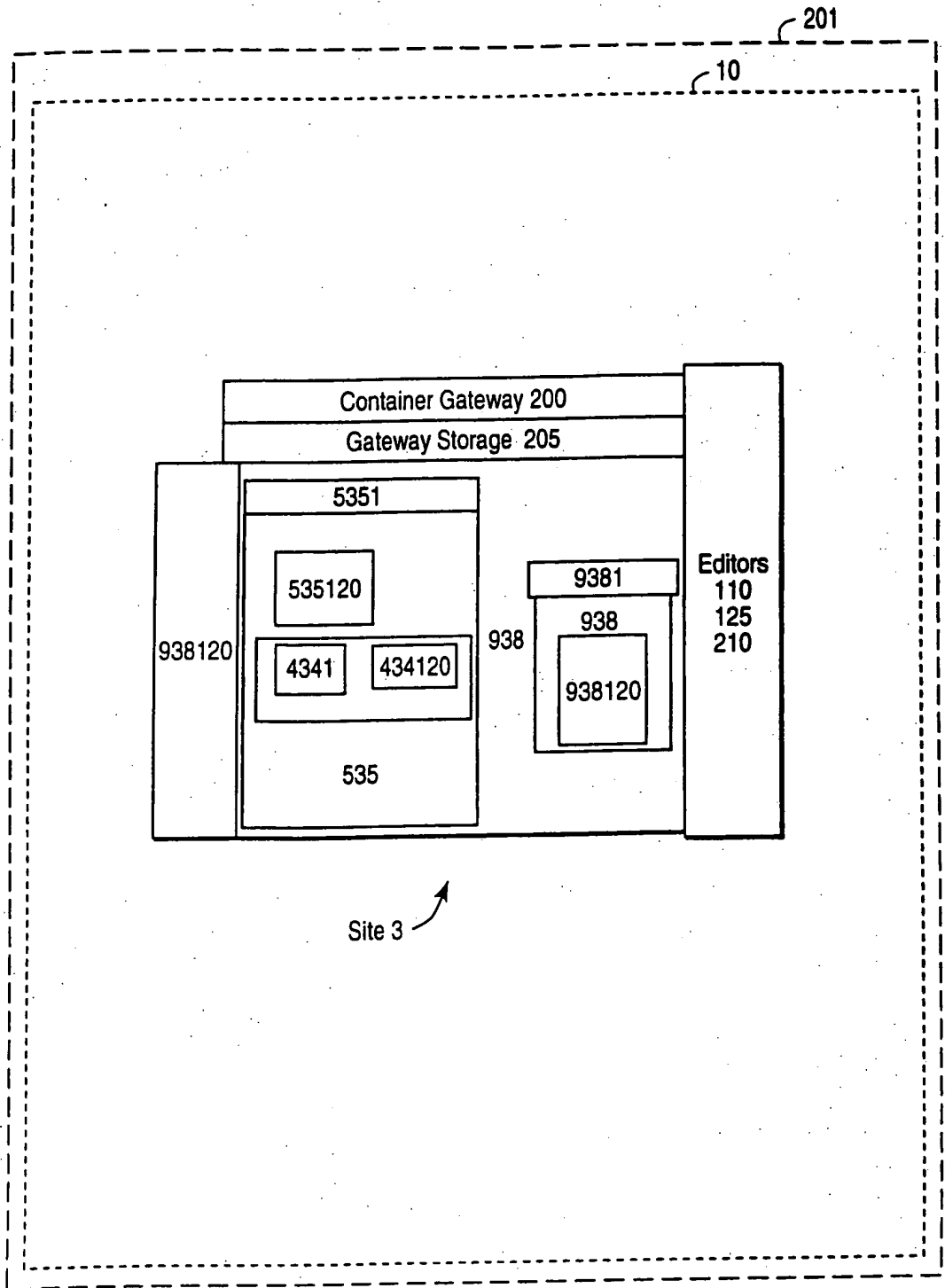


FIG. 2E

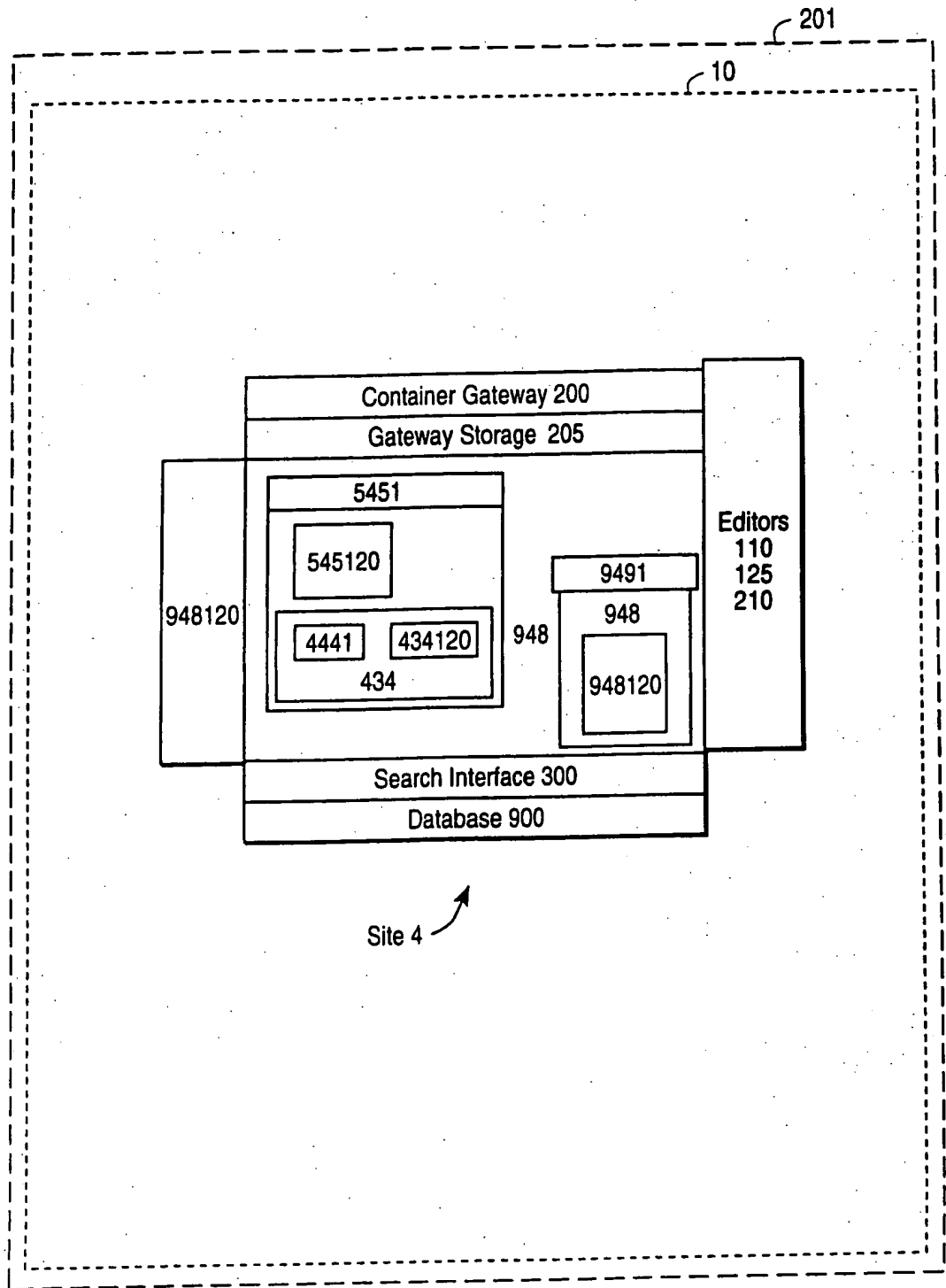


FIG. 2F

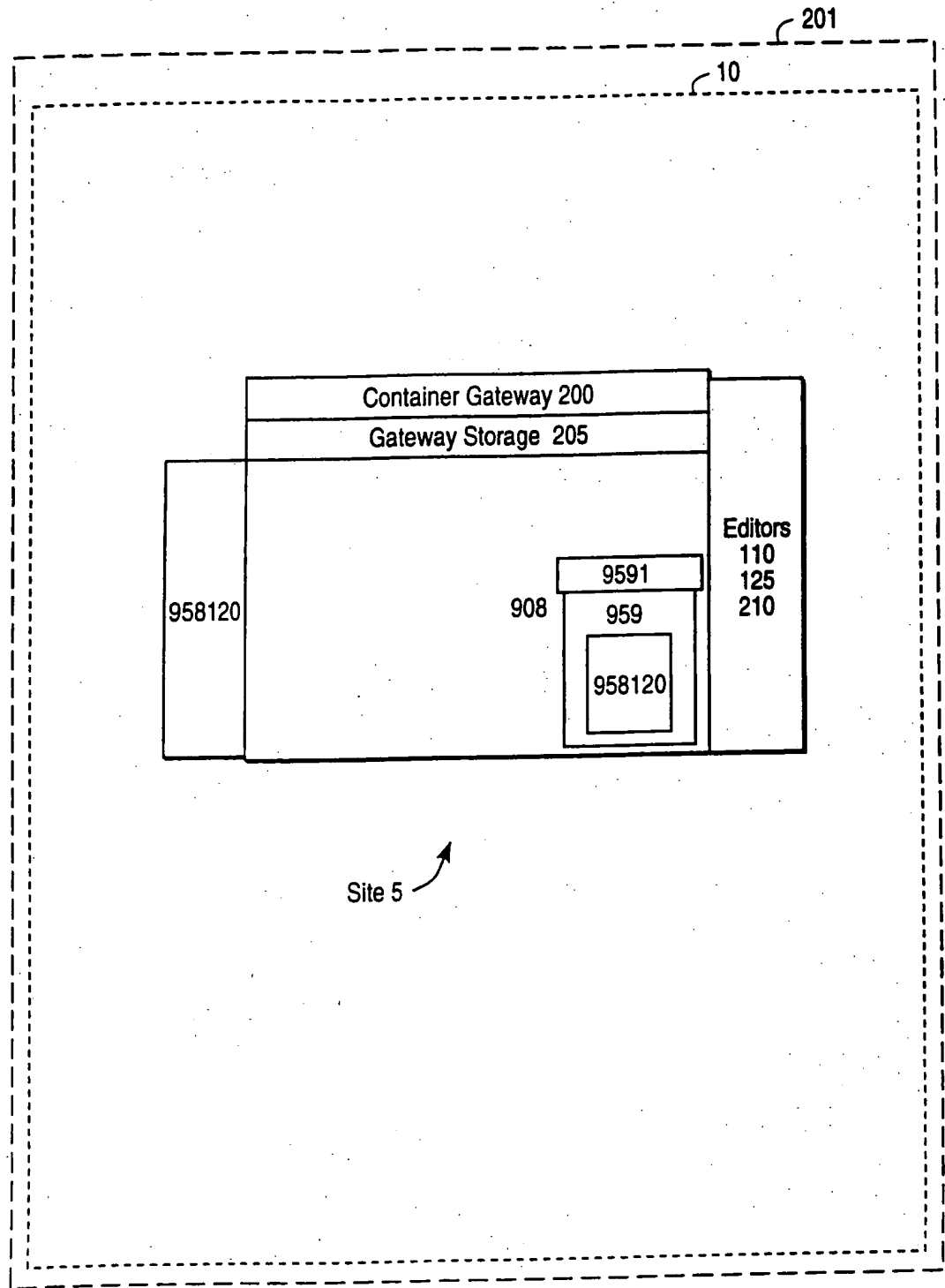


FIG. 2G



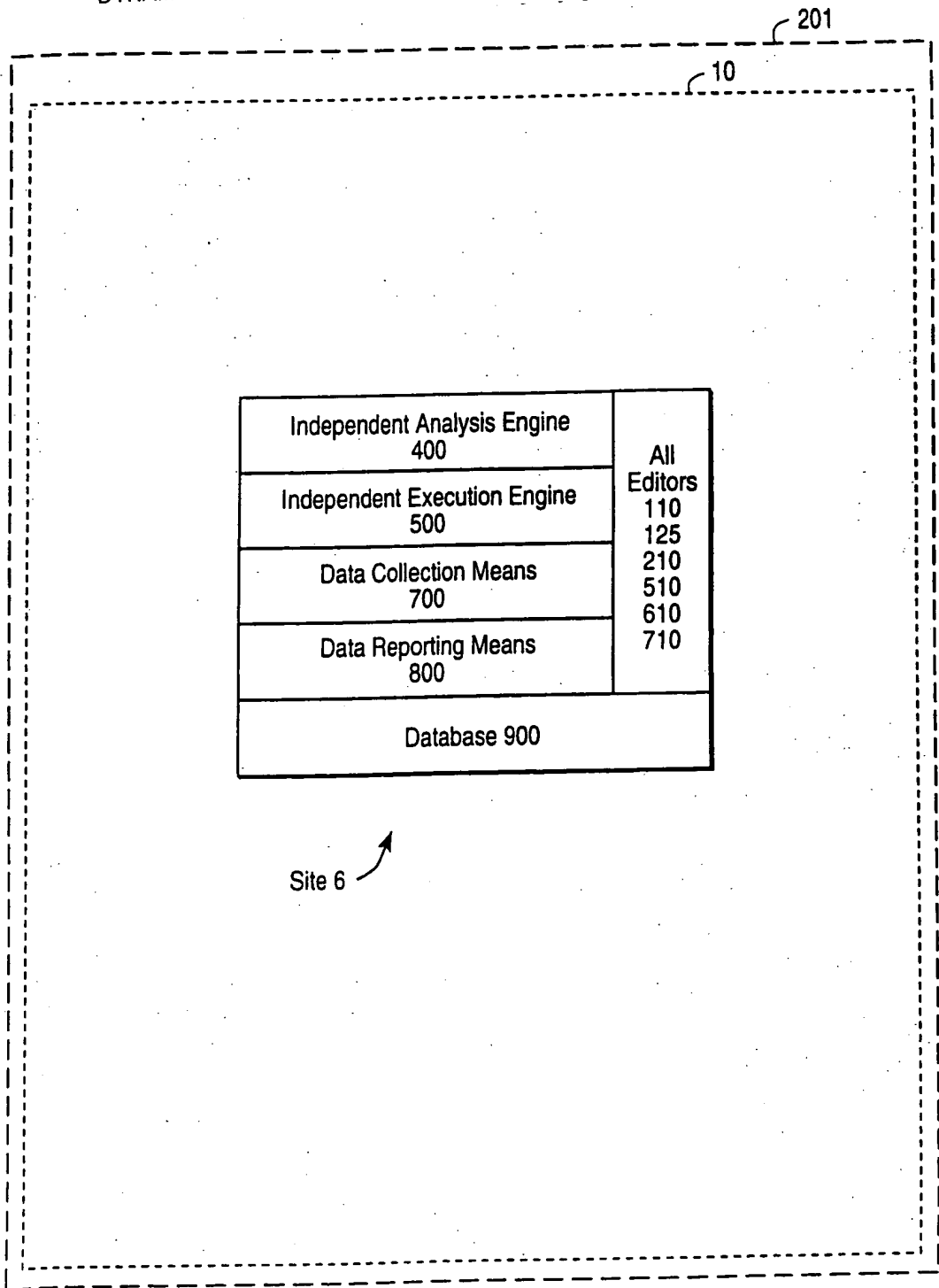


FIG. 2H

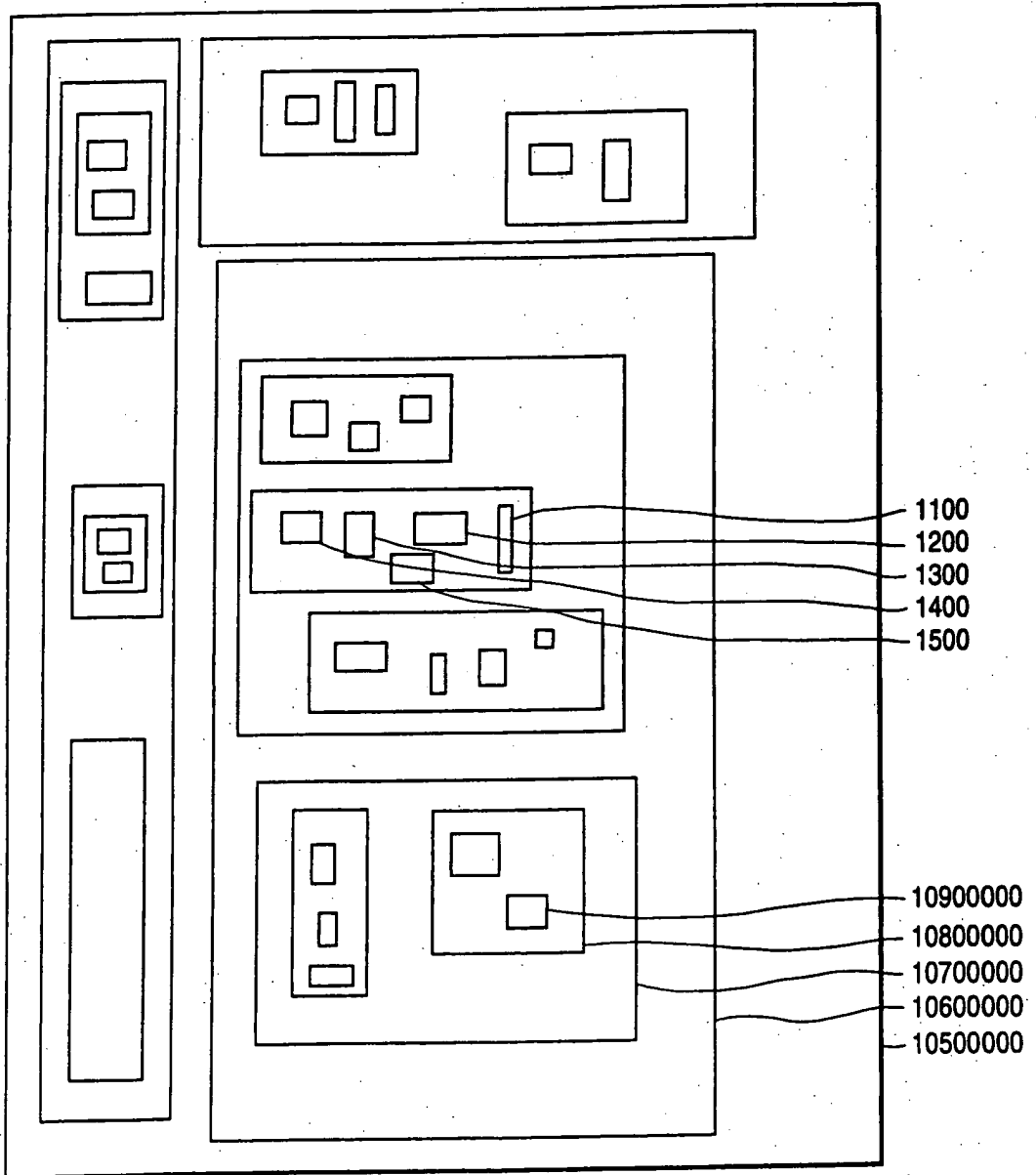


FIG. 3A

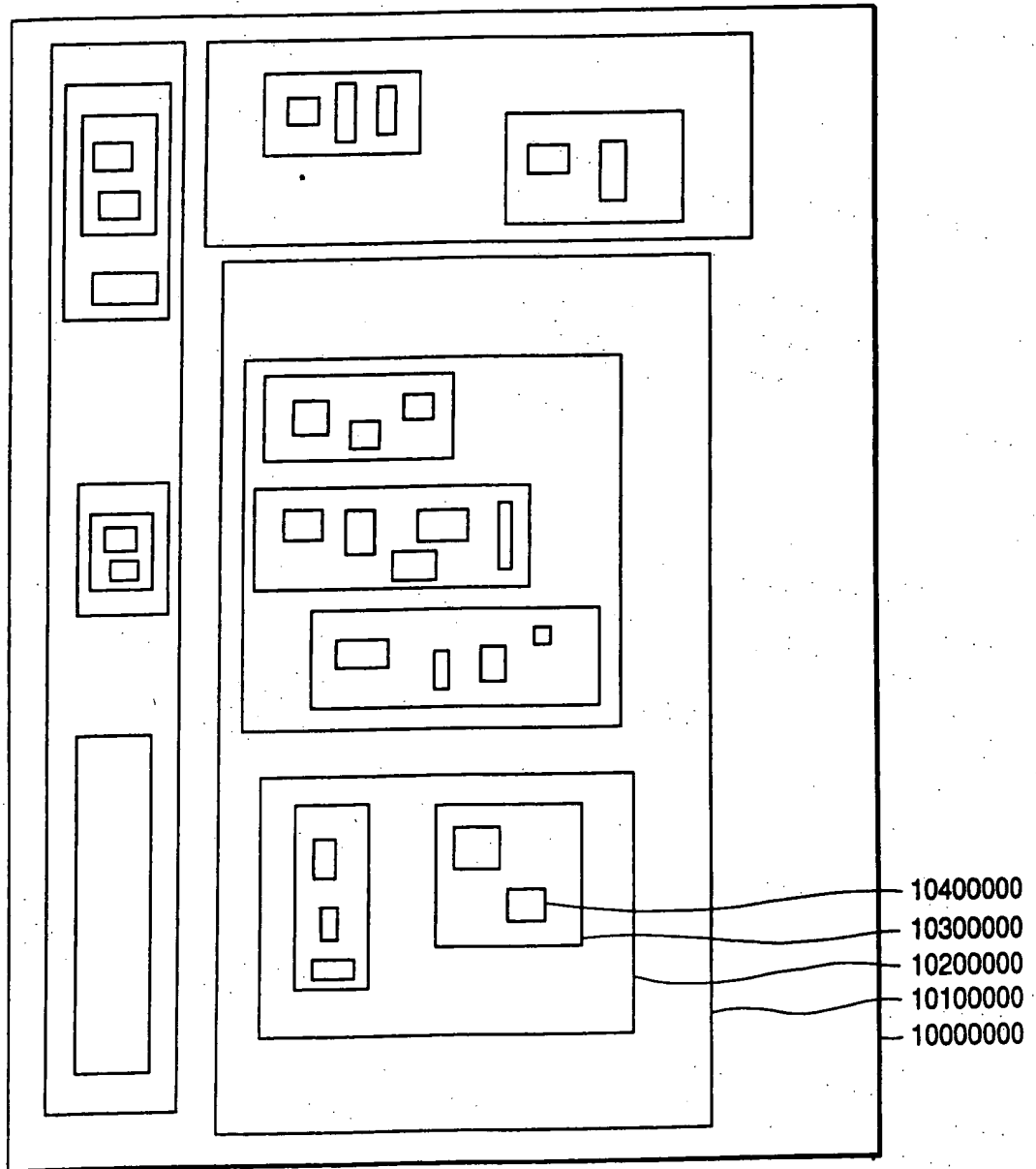


FIG. 3B

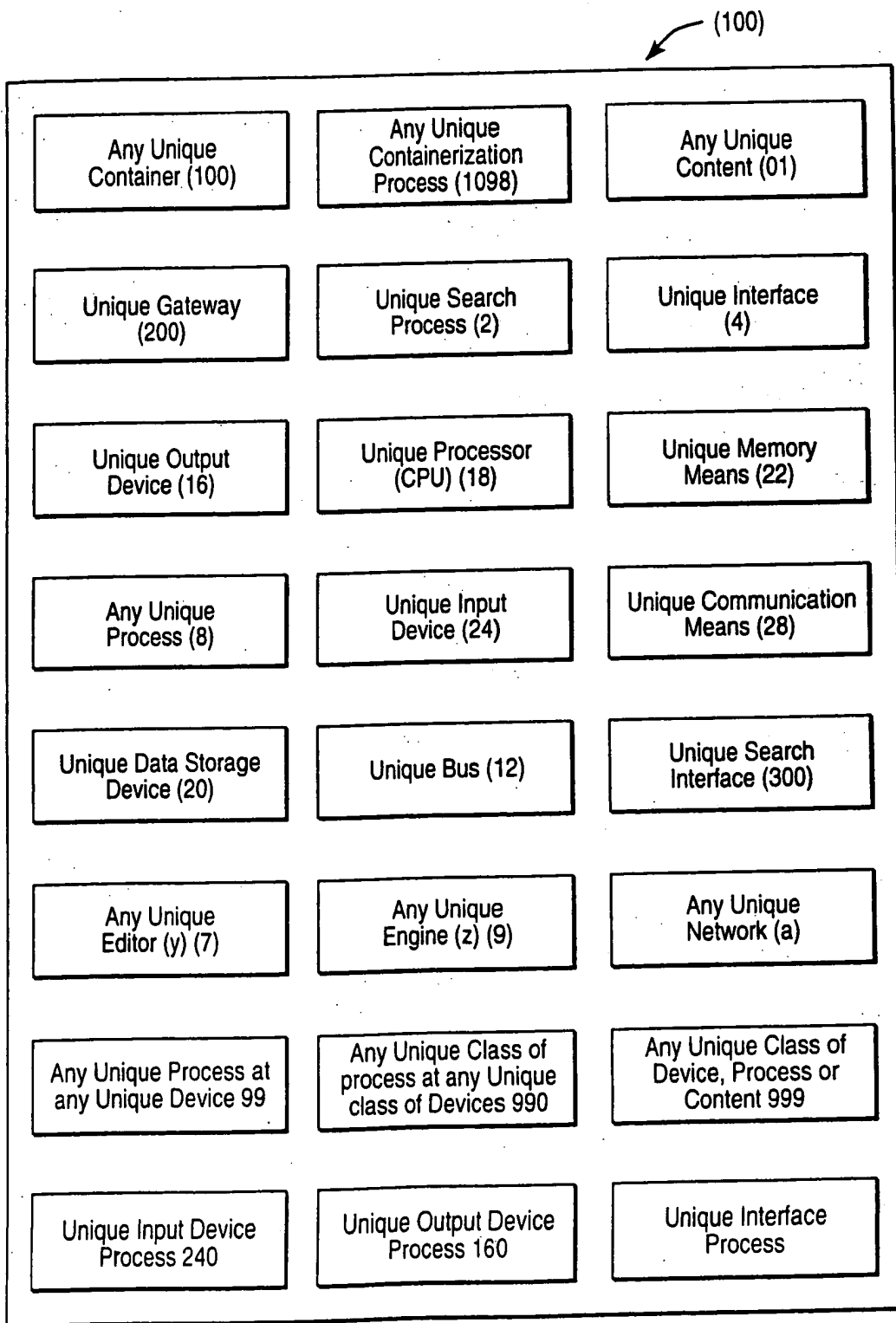


FIG. 3C

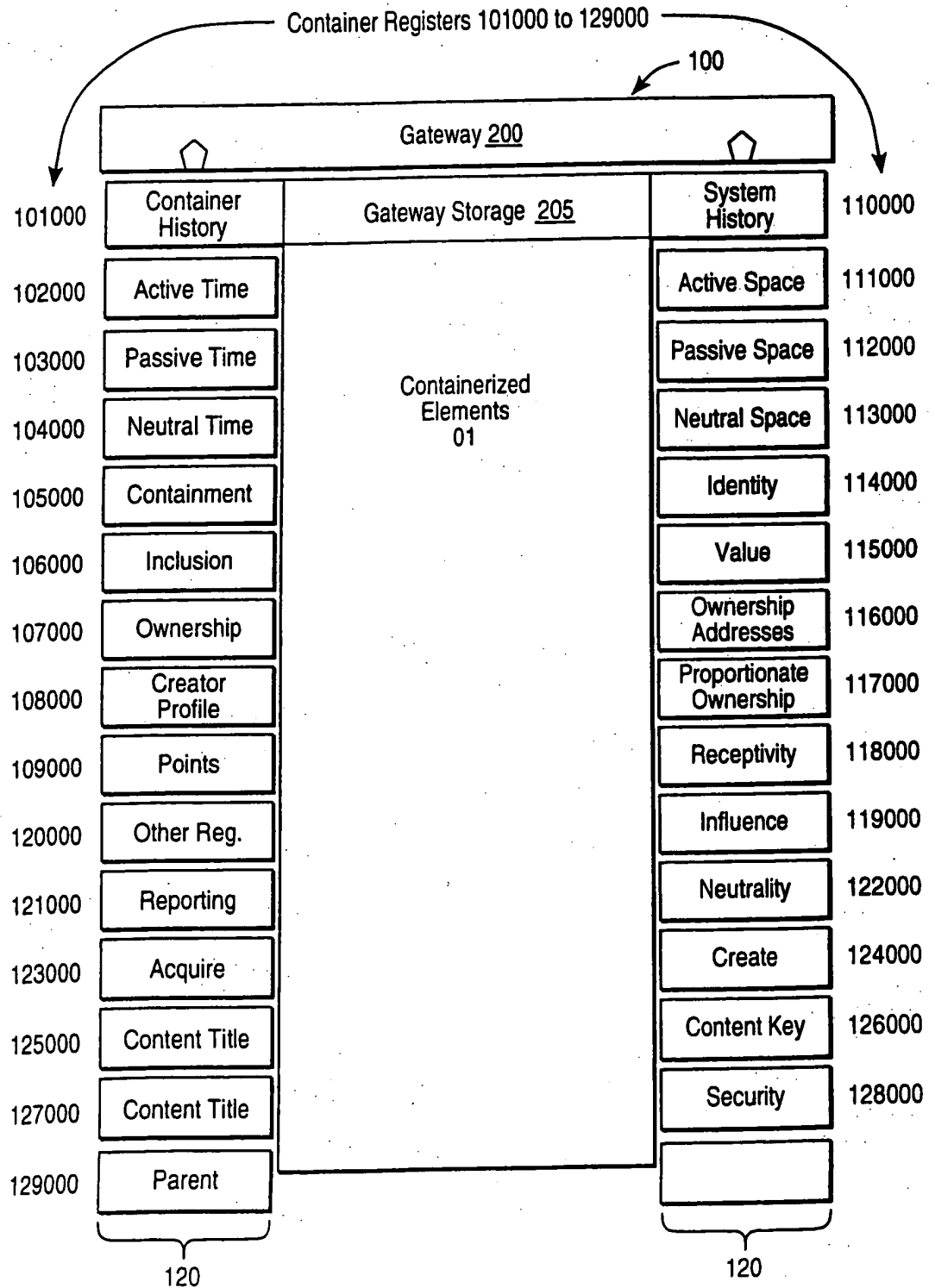


FIG. 4

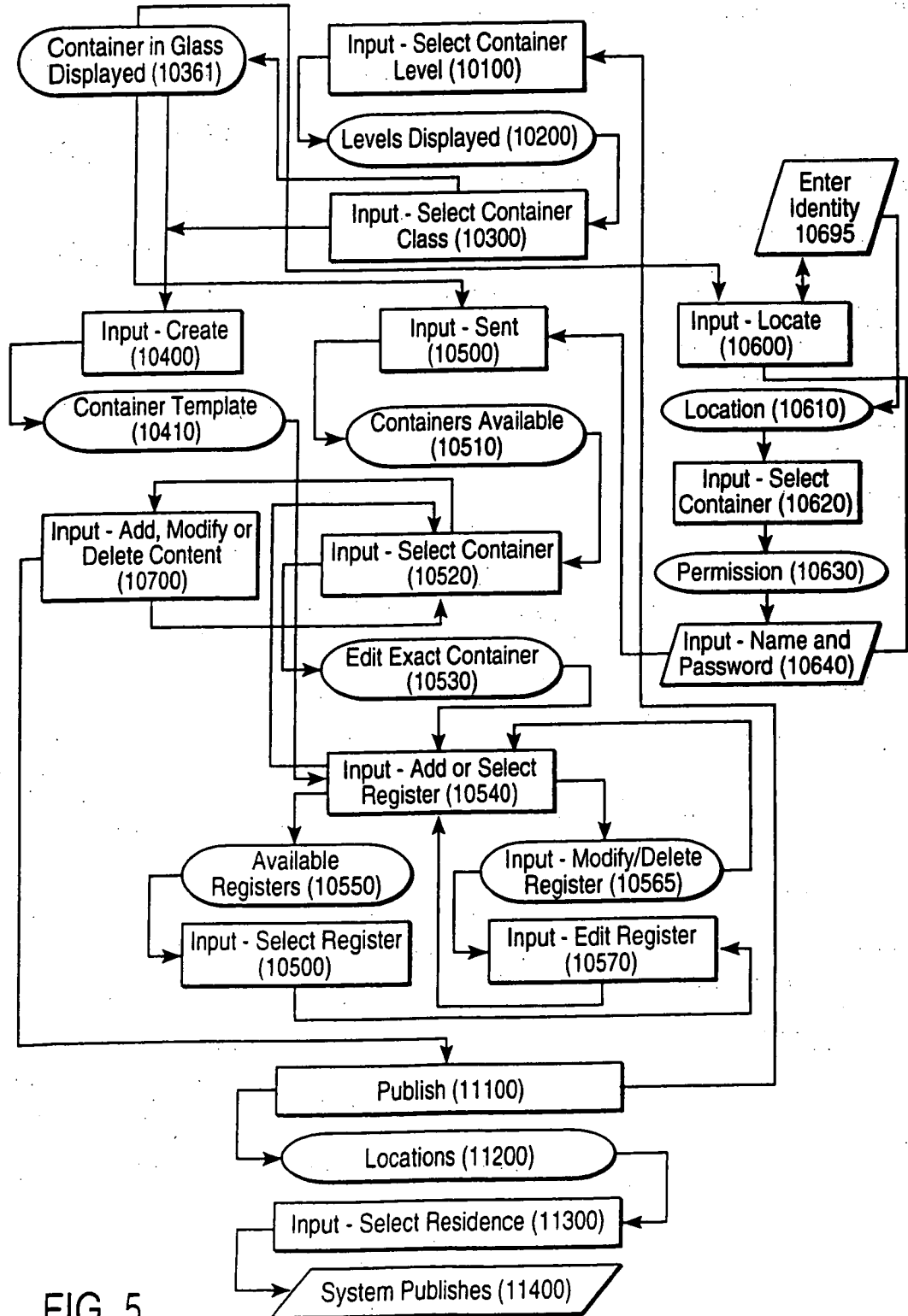


FIG. 5

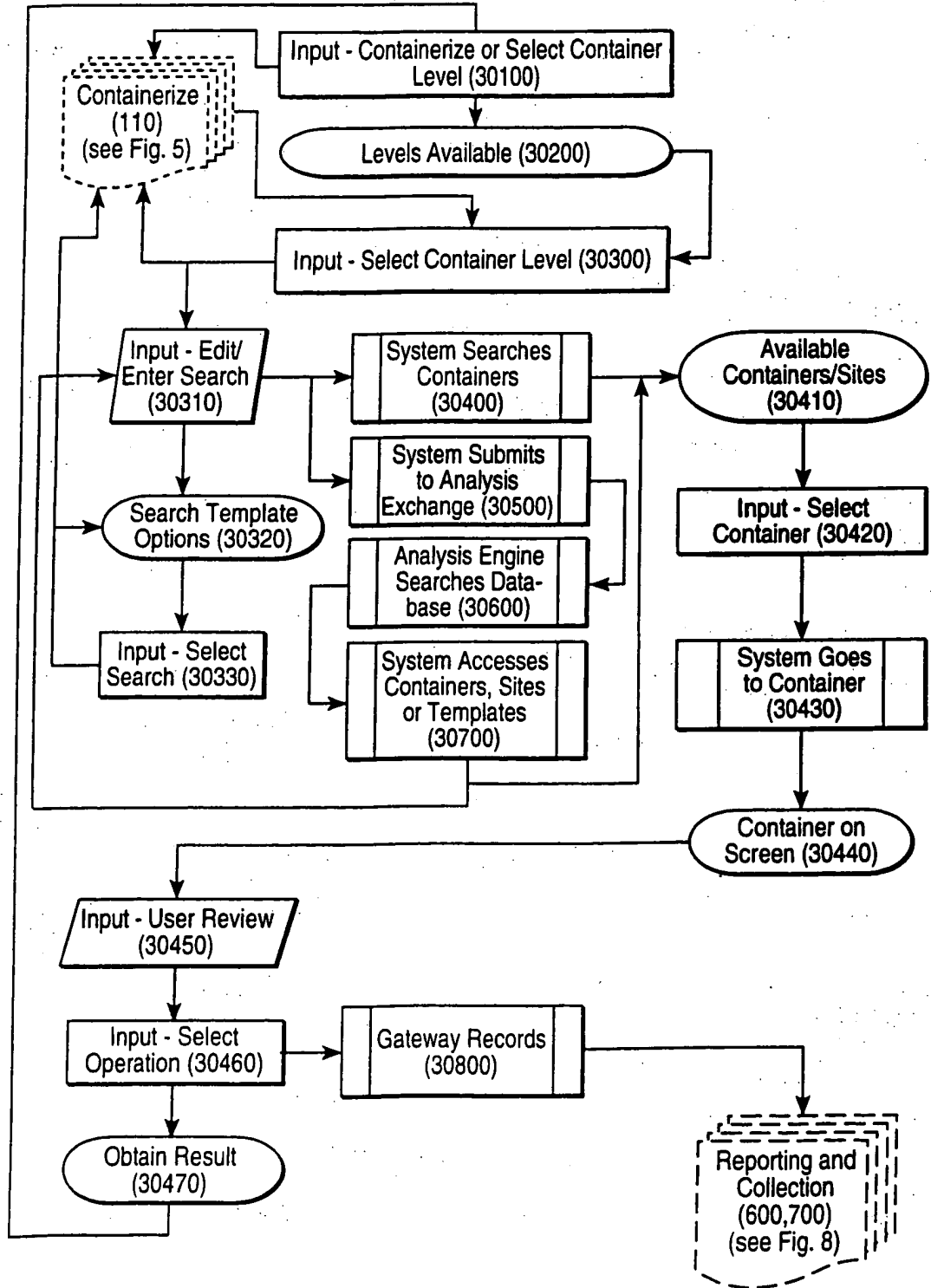


FIG. 6

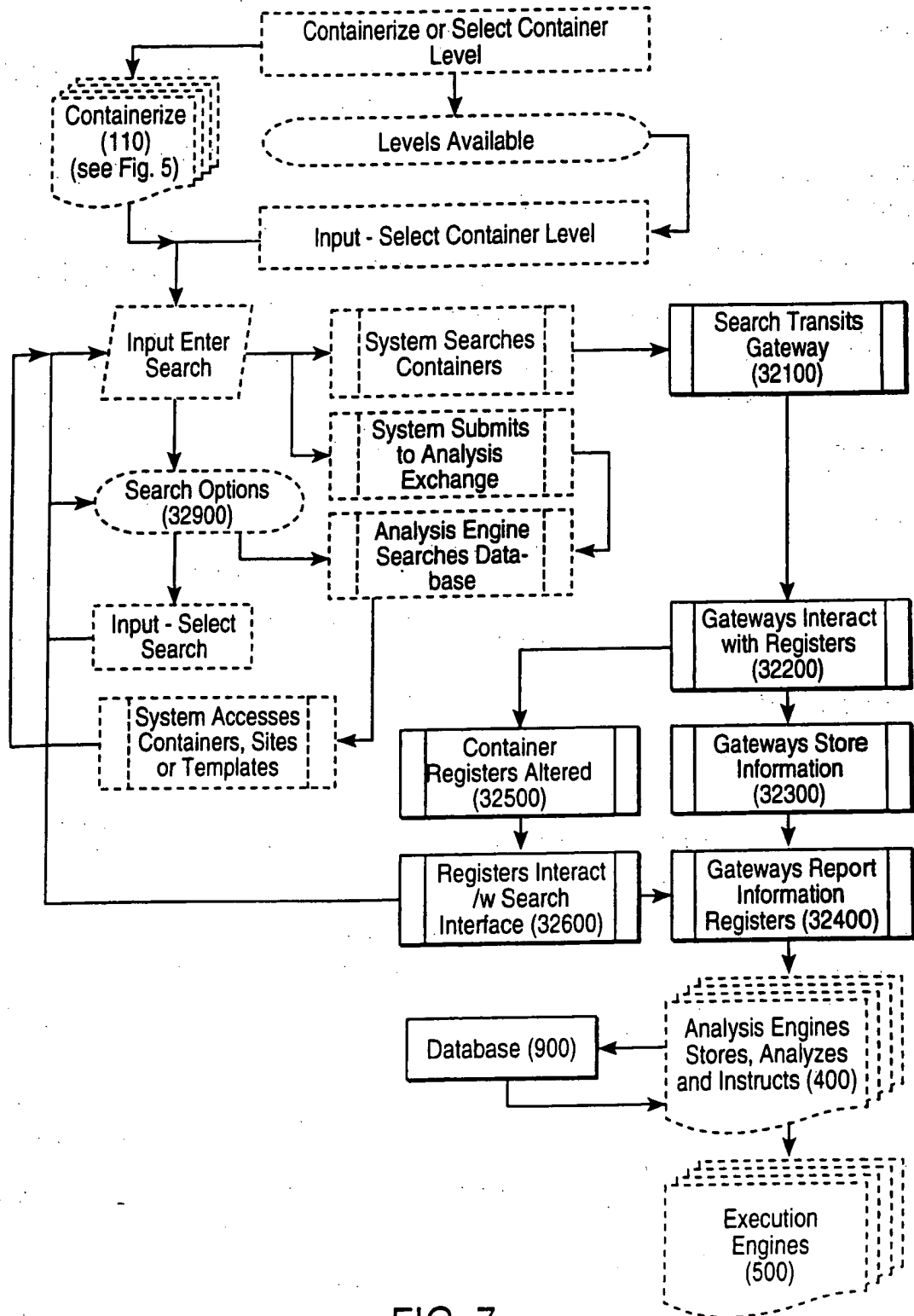


FIG. 7



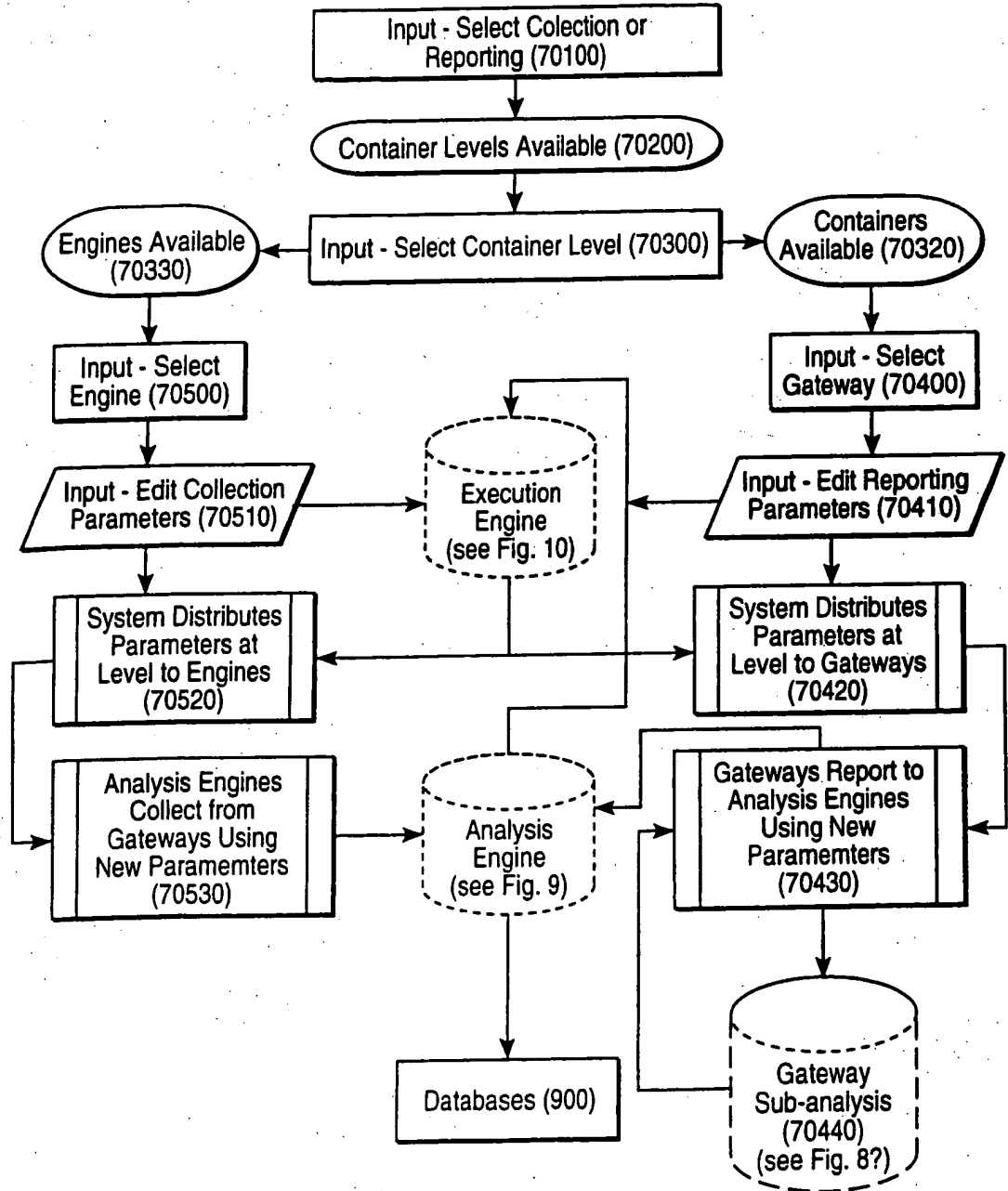


FIG. 8

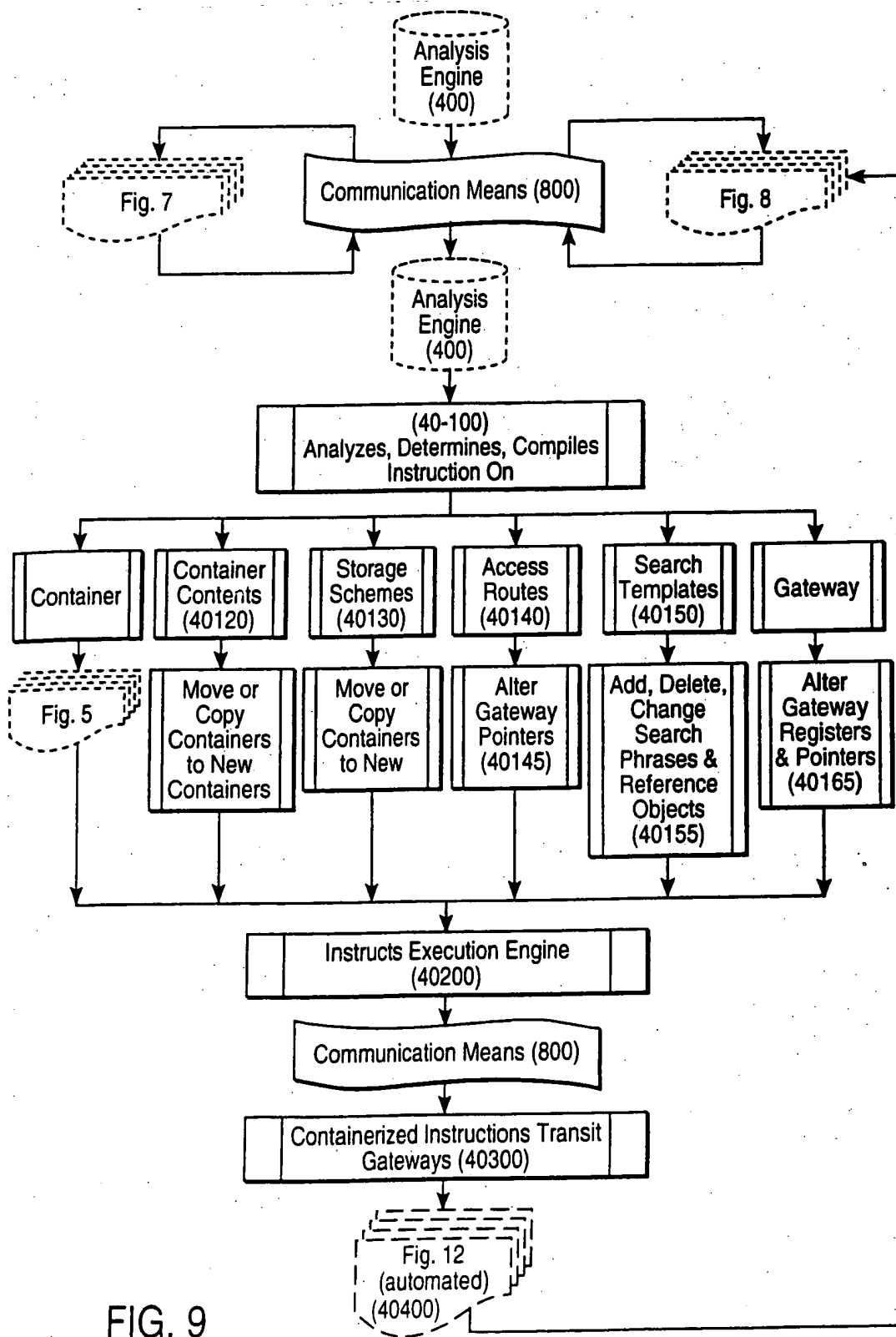


FIG. 9

Applicant(s): Michael De Angelo  
SYSTEM AND METHOD FOR CREATING AND  
MANIPULATING INFORMATION CONTAINERS WITH  
DYNAMIC REGISTERS

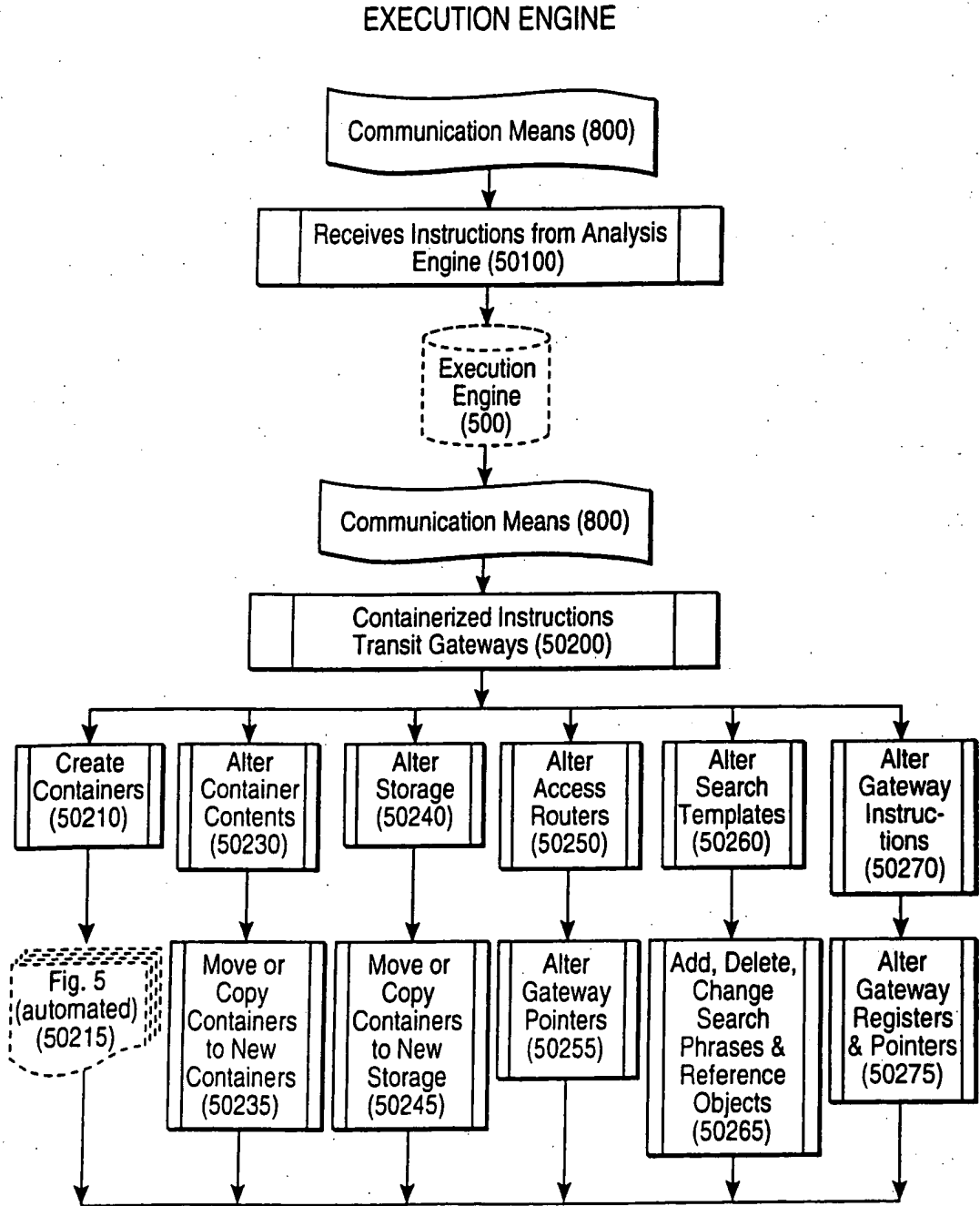


FIG. 10

### GATEWAY EDITOR

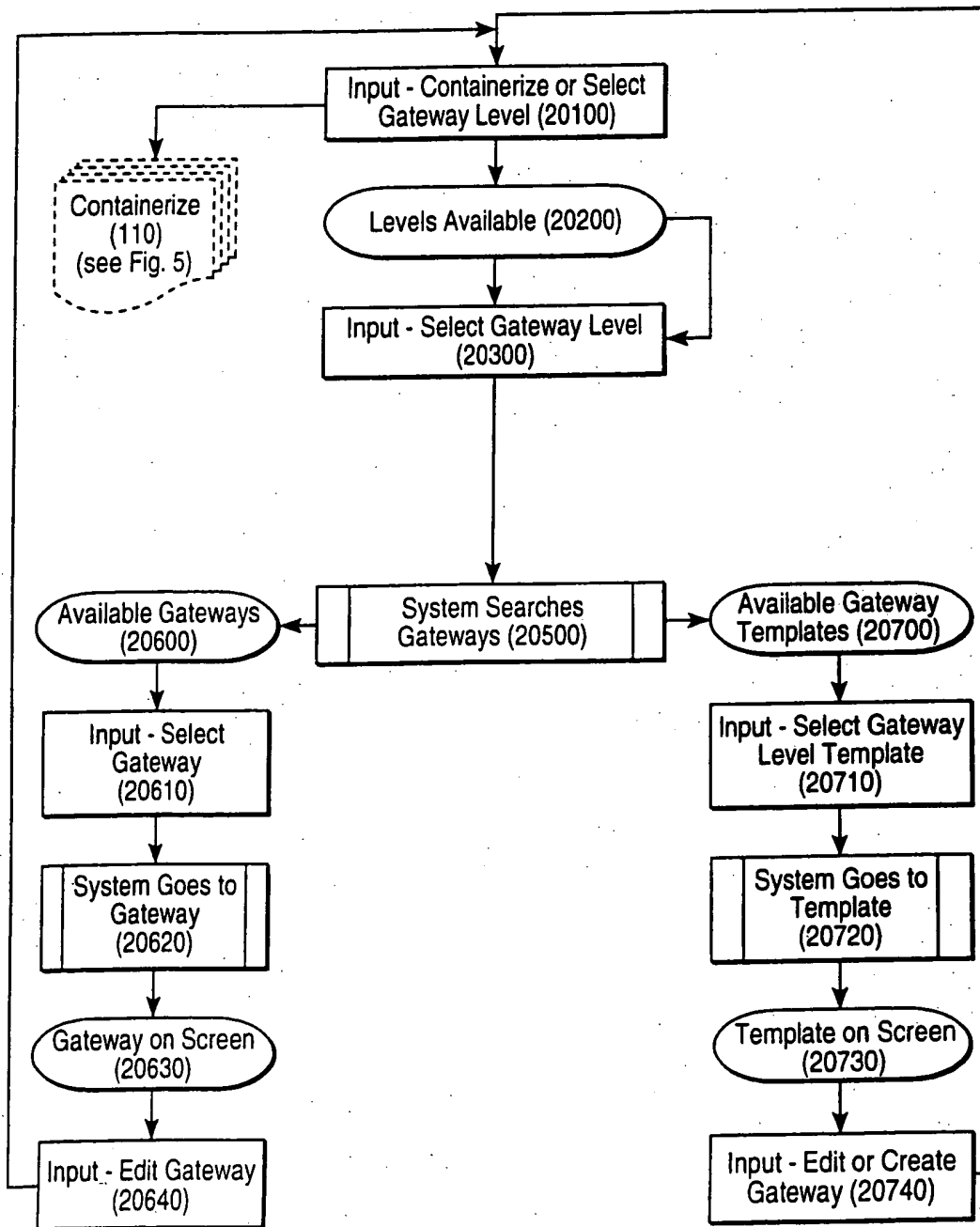


FIG. 11

Applicant(s): Michael De Angelo  
SYSTEM AND METHOD FOR CREATING AND  
MANIPULATING INFORMATION CONTAINERS WITH  
DYNAMIC REGISTERS

### GATEWAY PROCESS

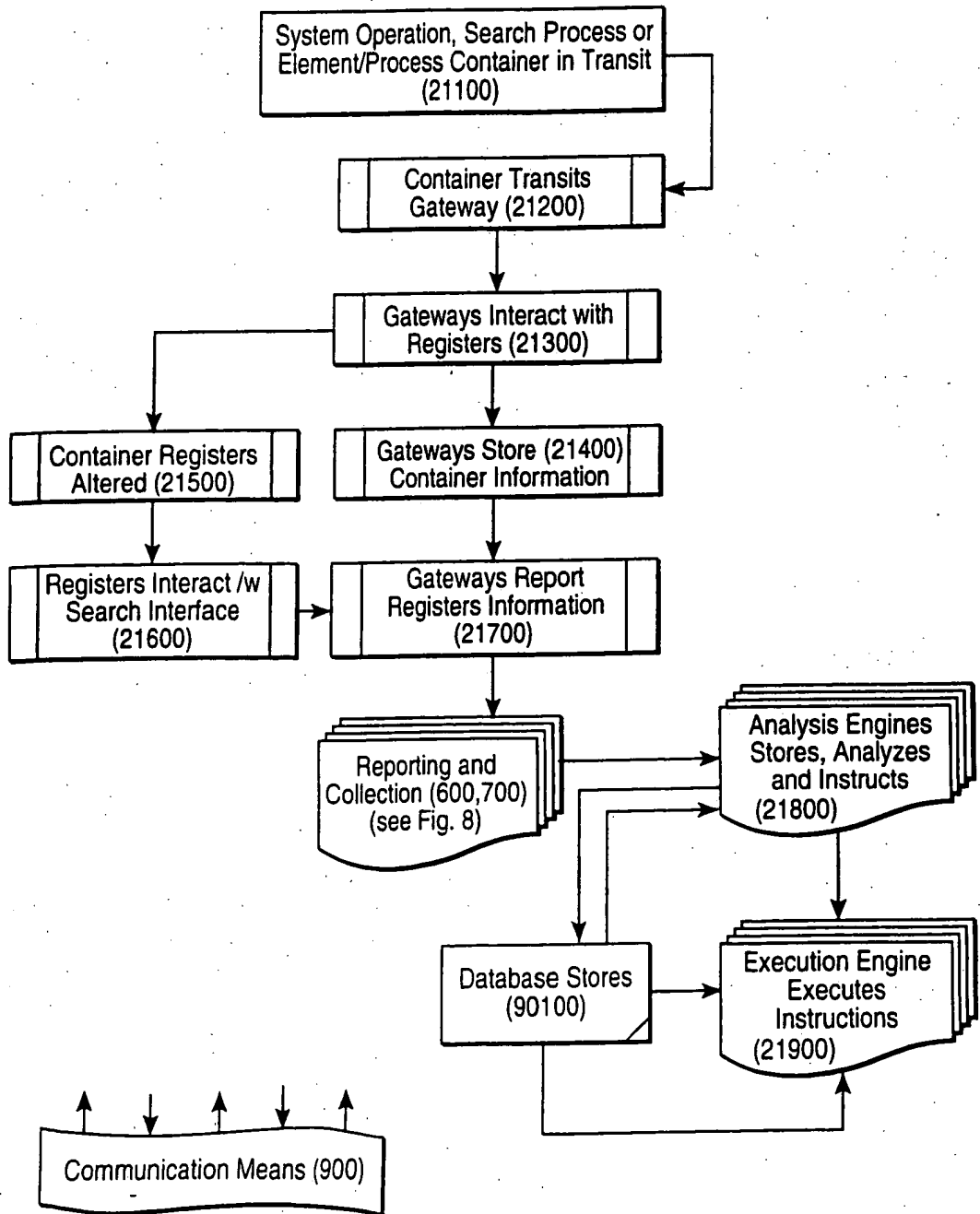


FIG. 12

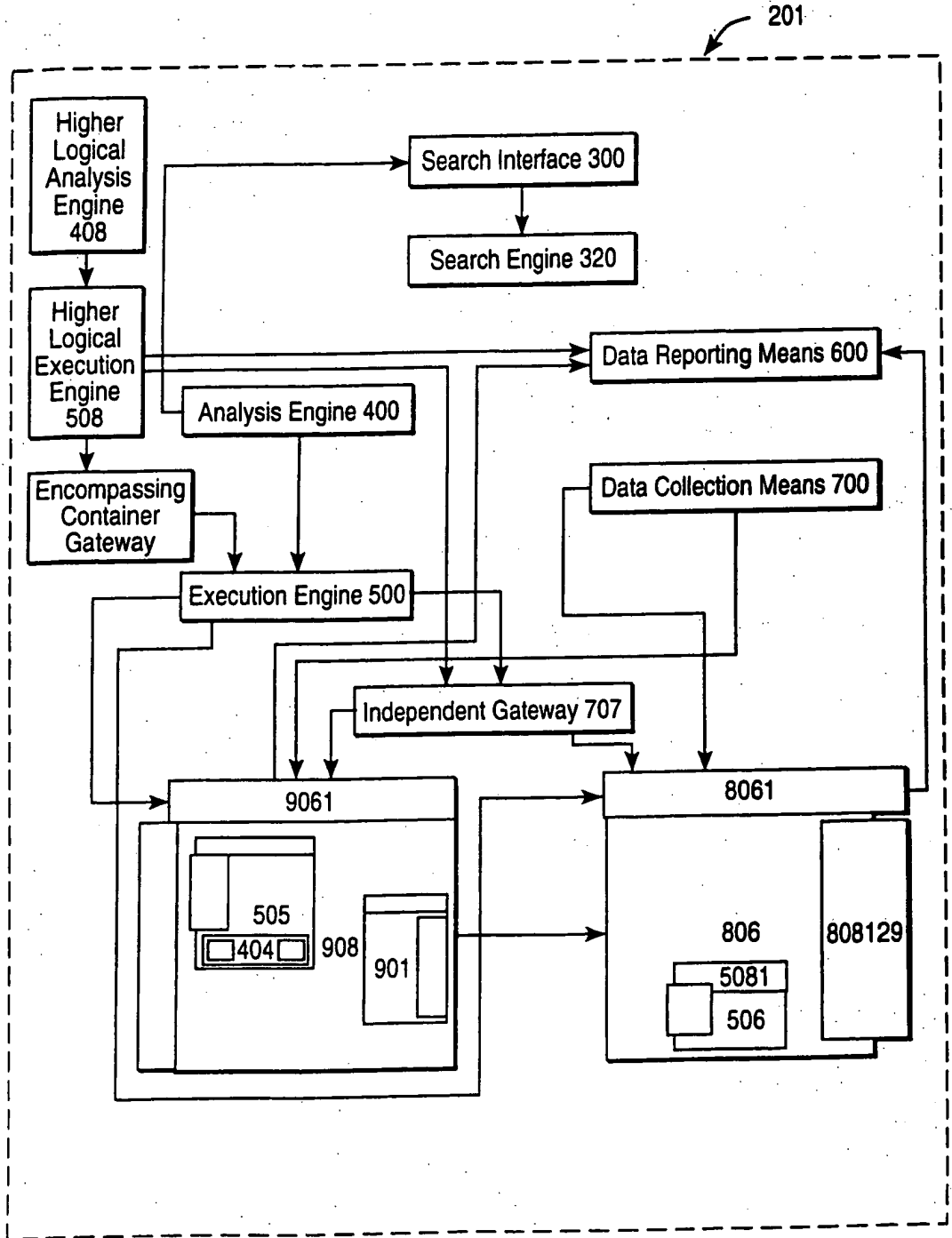


FIG. 13A

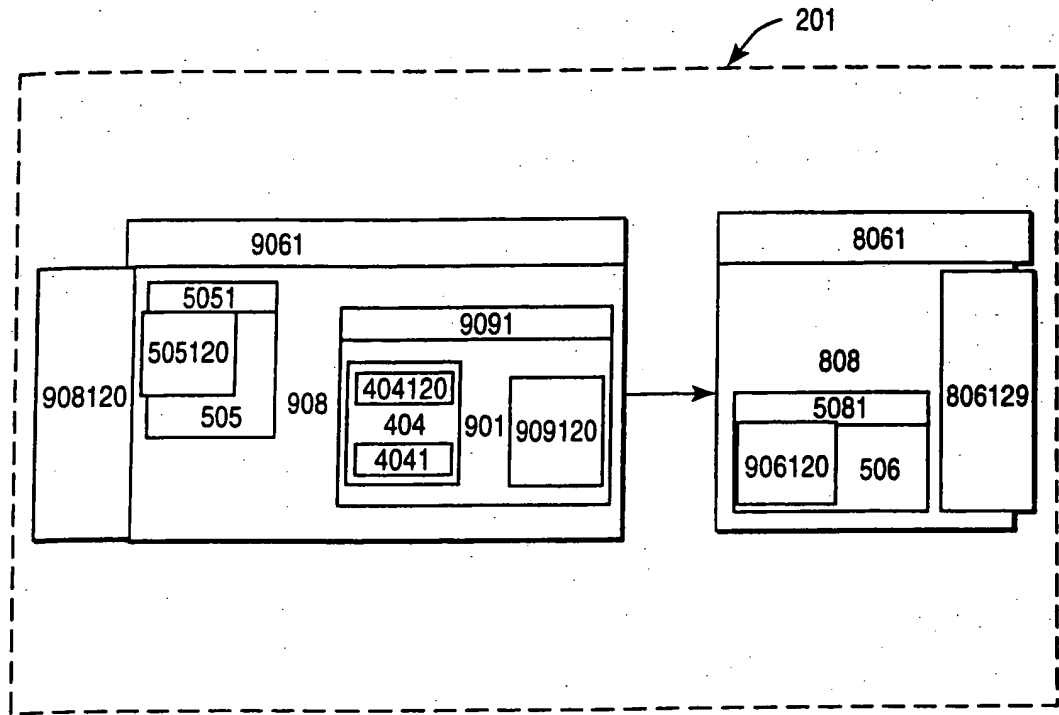


FIG. 13B

Applicant(s): Michael De Angelo  
SYSTEM AND METHOD FOR CREATING AND  
MANIPULATING INFORMATION CONTAINERS WITH  
DYNAMIC REGISTERS

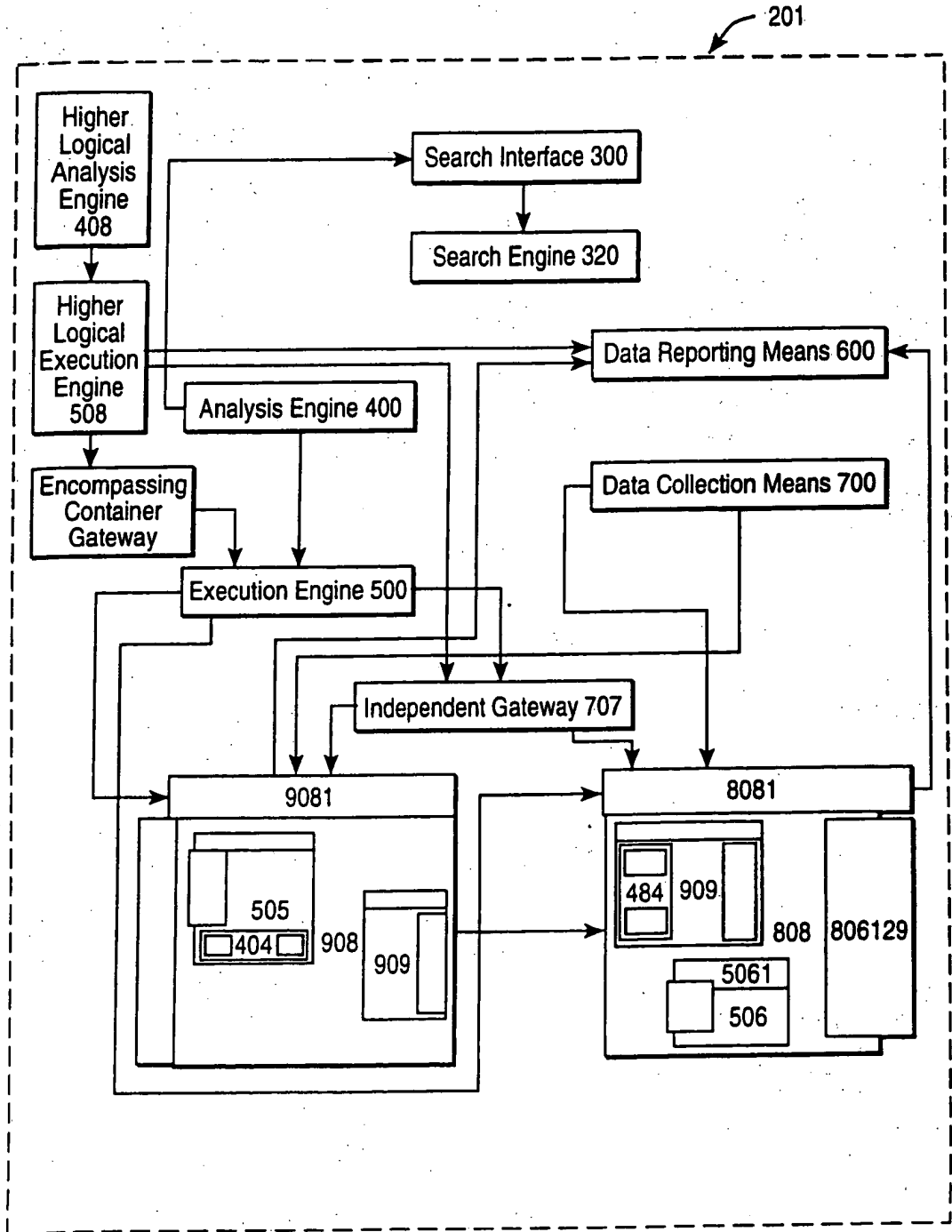


FIG. 13C



Applicant(s): Michael De Angelo  
SYSTEM AND METHOD FOR CREATING AND  
MANIPULATING INFORMATION CONTAINERS WITH  
DYNAMIC REGISTERS

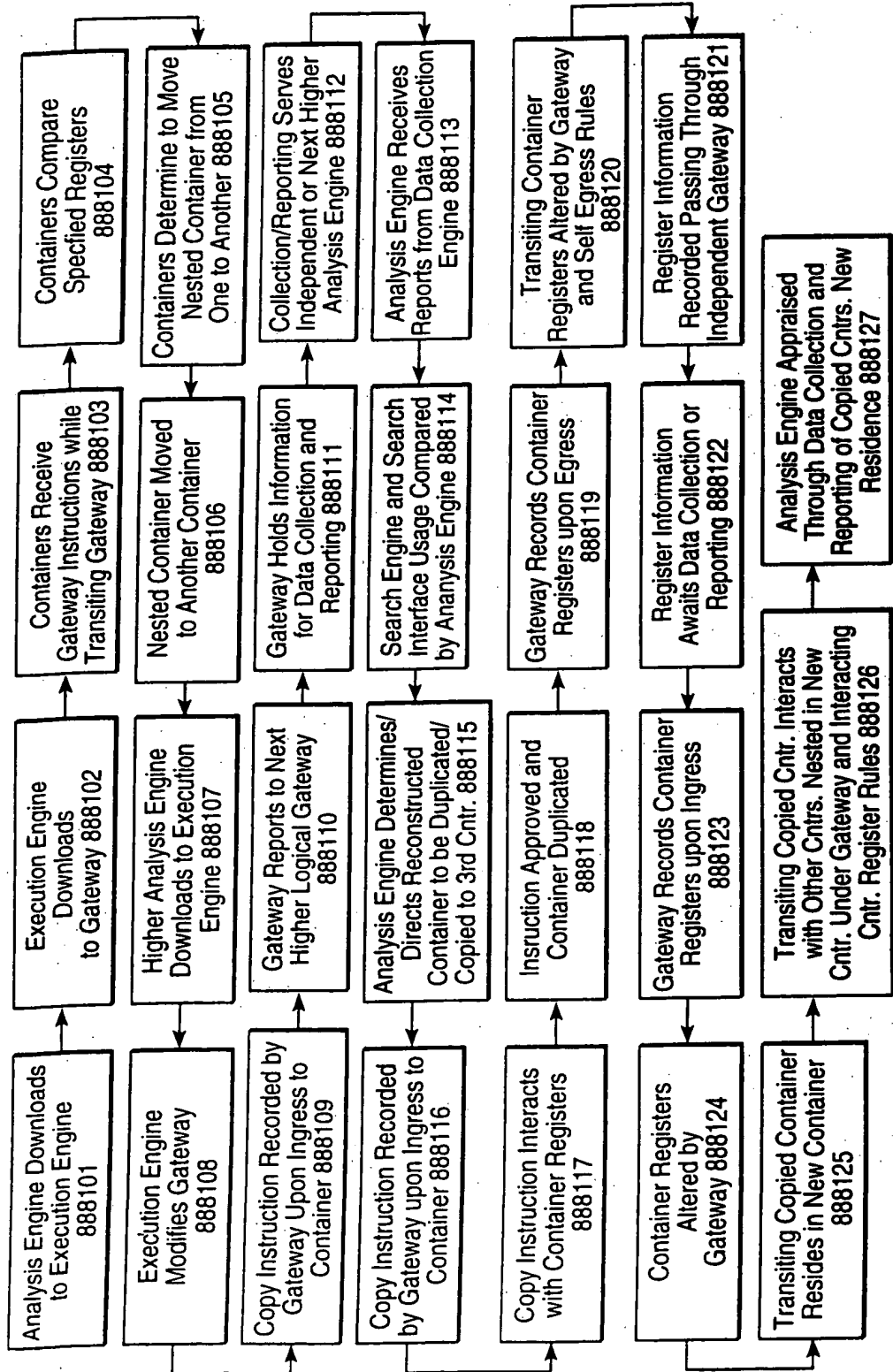


FIG. 13D

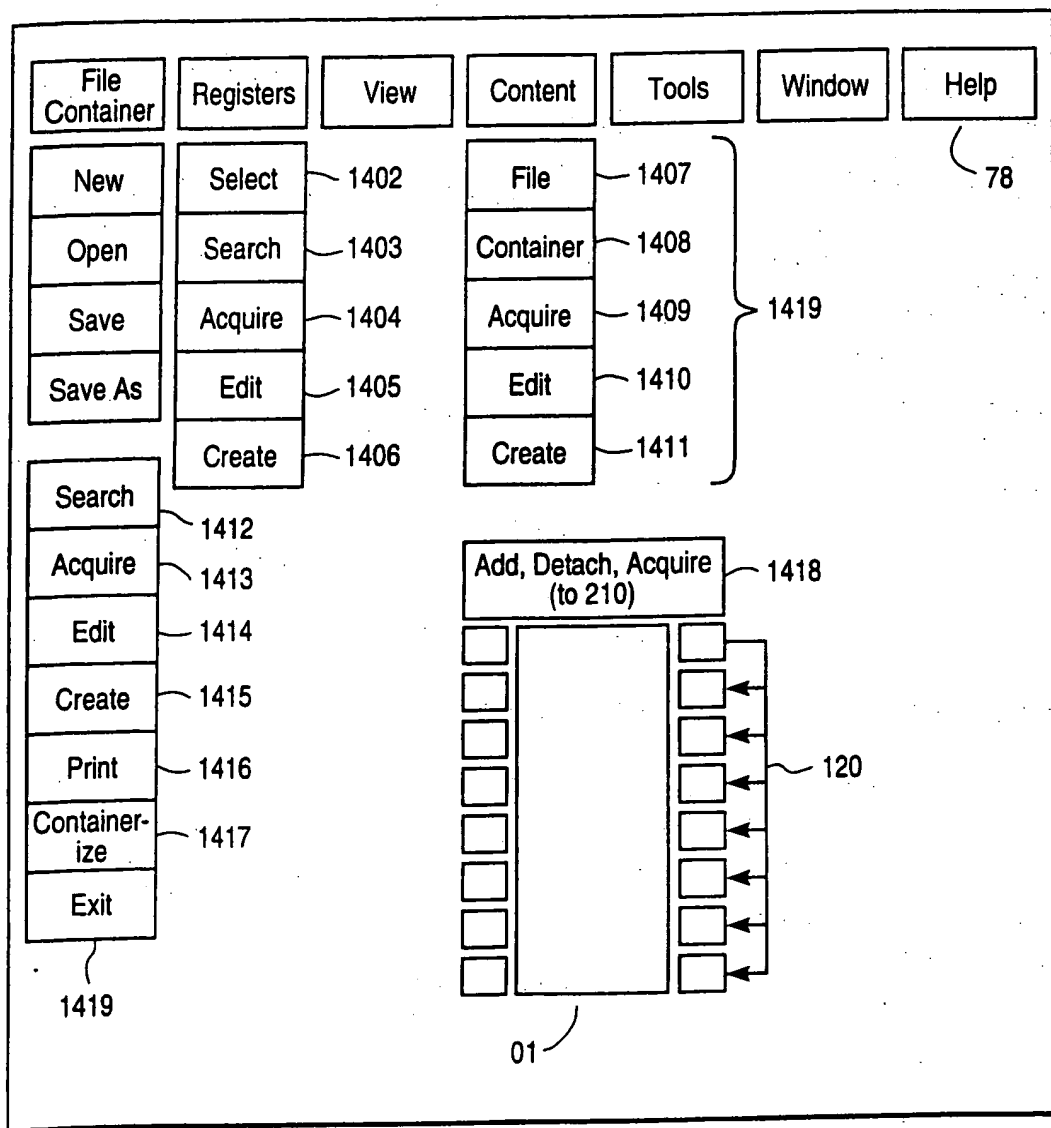


FIG. 14

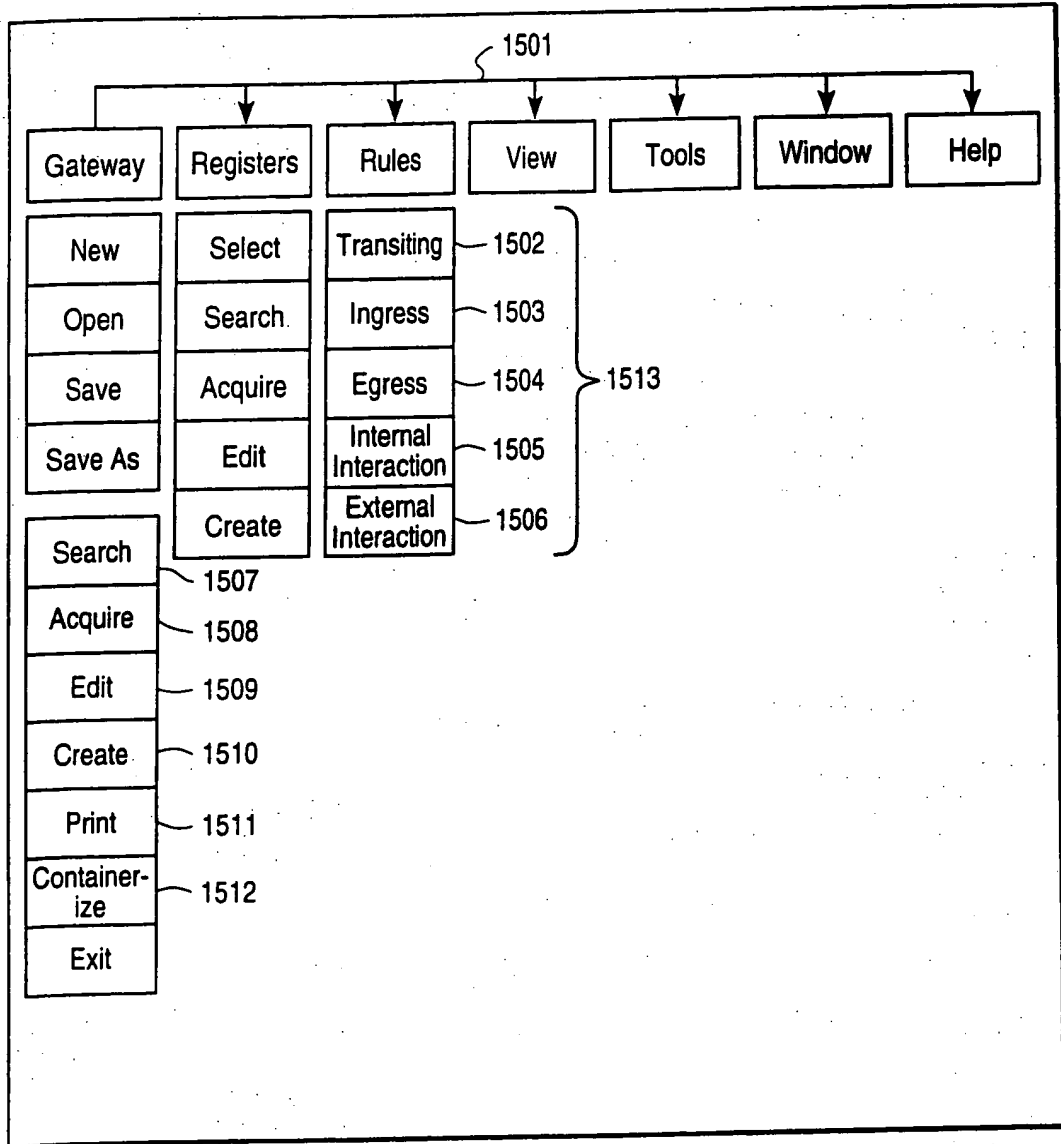


FIG. 15

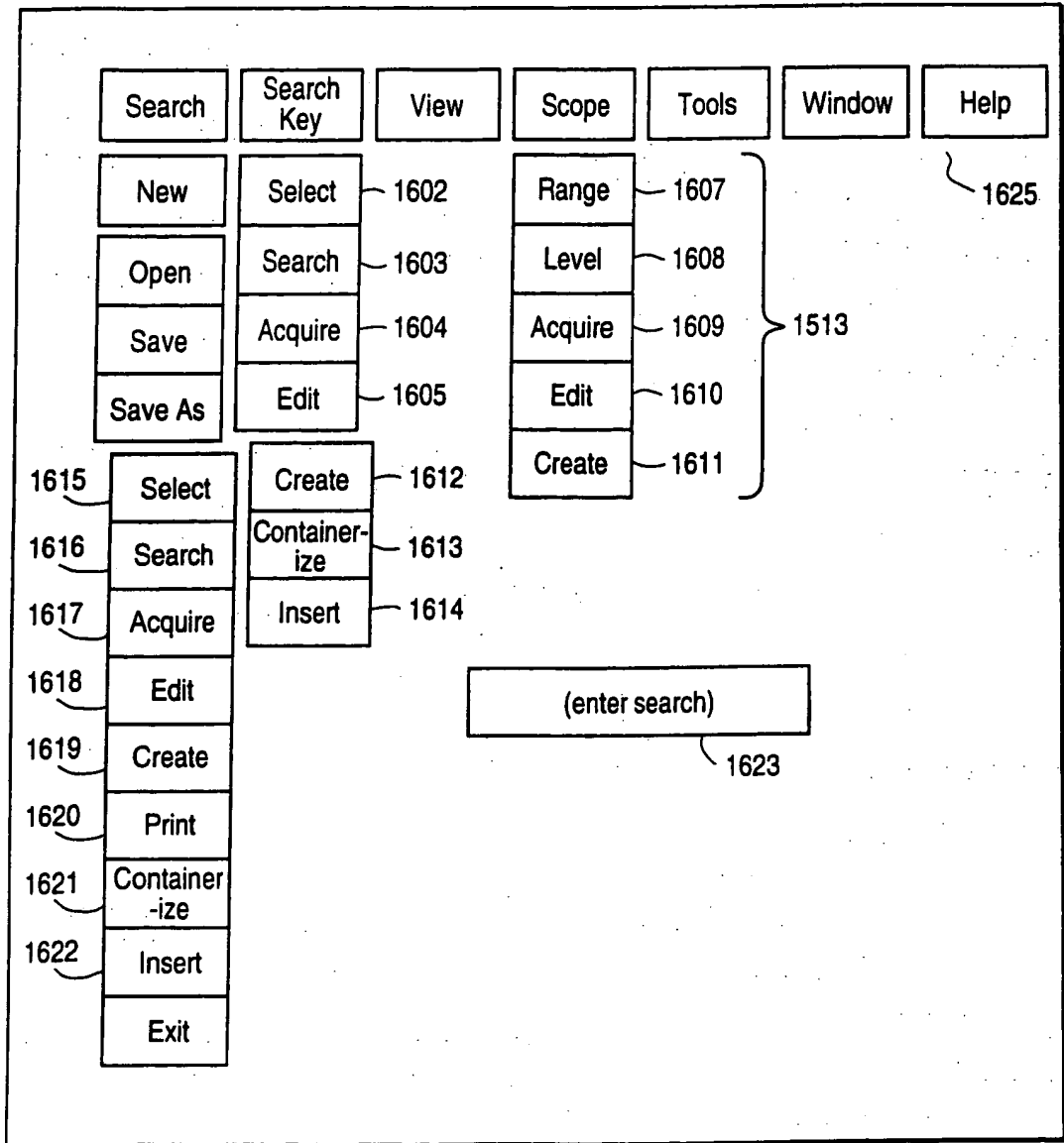


FIG. 16

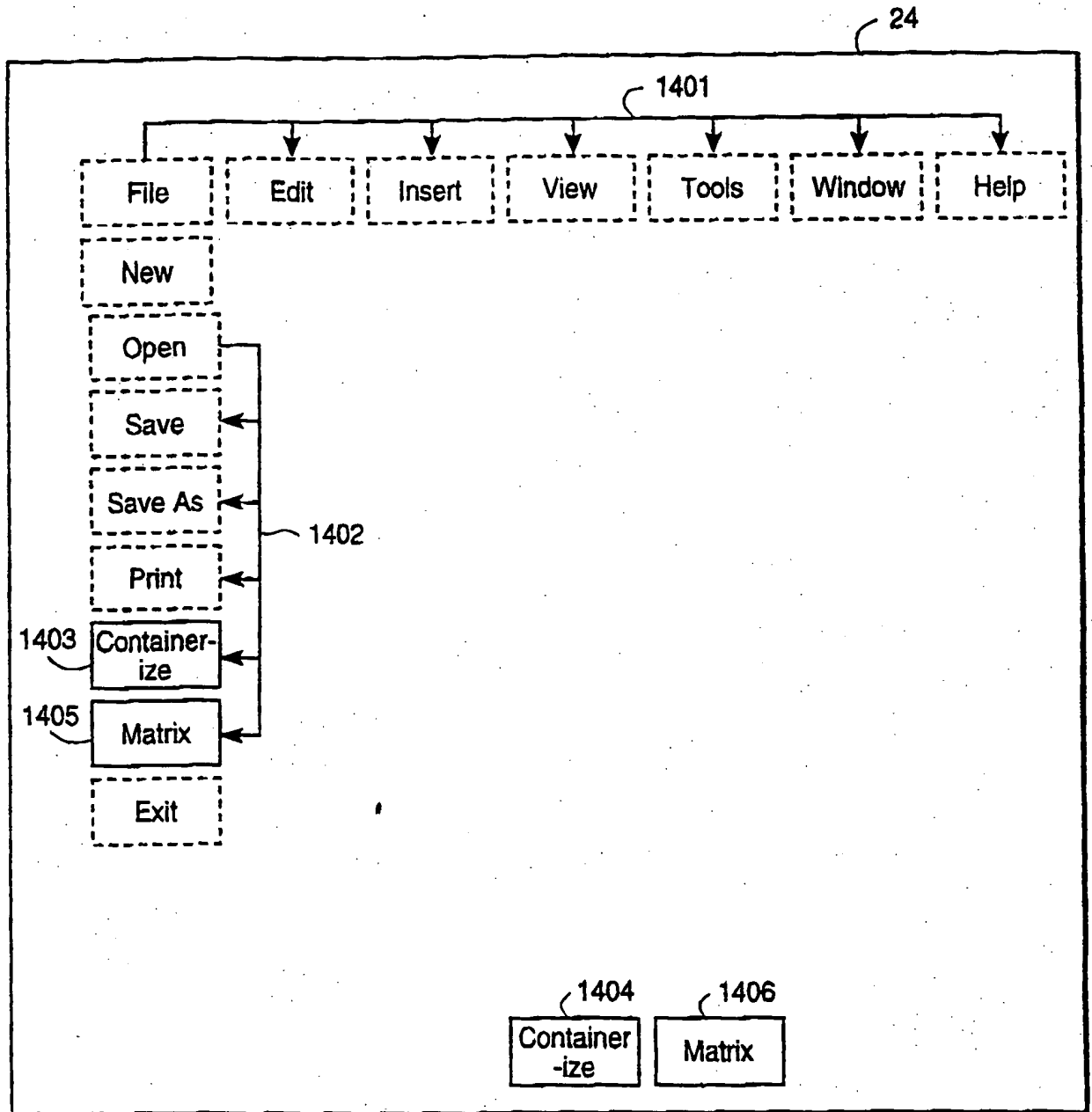


FIG. 17

PATENT APPLICATION SERIAL NO \_\_\_\_\_

U.S. DEPARTMENT OF COMMERCE  
PATENT AND TRADEMARK OFFICE  
FEE RECORD SHEET

11/21/2005 SHASSEN1 00000077 11280700

01 FC:2011	150.00	OP
02 FC:2111	250.00	OP
03 FC:2311	100.00	OP
04 FC:2202	250.00	OP
05 FC:2201	900.00	OP

PTO-1556  
(5/87)

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

**PATENT APPLICATION FEE DETERMINATION RECORD**

Substitute for Form PTO-875 Effective December 8, 2004

Application or Docket Number  
 11280700

**APPLICATION AS FILED - PART I**

FOR	NUMBER FILED (Column 1)	NUMBER EXTRA (Column 2)
BASIC FEE (37 CFR 1.16(e), (f), or (g))	N/A	N/A
SEARCH FEE (37 CFR 1.16(h), (i), or (j))	N/A	N/A
EXAMINATION FEE (37 CFR 1.16(k), (l), or (m))	N/A	N/A
TOTAL CLAIMS (37 CFR 1.16(n))	30	minus 20 = 10
INDEPENDENT CLAIMS (37 CFR 1.16(o))	12	minus 3 = 9
APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is \$250 (\$125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).	
MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))		

SMALL ENTITY

RATE (\$)	FEE (\$)
N/A	150.00
N/A	\$250
N/A	\$100
X\$ 25 =	250
X100 =	900
+180=	
TOTAL	1650

OR OTHER THAN SMALL ENTITY

RATE (\$)	FEE (\$)
N/A	300.00
N/A	\$500
N/A	\$200
X\$50 =	
X200 =	
+360=	
TOTAL	

\* If the difference in column 1 is less than zero, enter "0" in column 2.

**APPLICATION AS AMENDED - PART II**

AMENDMENT A	CLAIMS REMAINING AFTER AMENDMENT	HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA
	Total (37 CFR 1.16(n))	Minus	**
Independent (37 CFR 1.16(o))	Minus	***	=
Application Size Fee (37 CFR 1.16(s))			
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))			

SMALL ENTITY

RATE (\$)	ADDITIONAL FEE (\$)
X\$ 25 =	
X100 =	
+180=	
TOTAL ADD'L FEE	

OR OTHER THAN SMALL ENTITY

RATE (\$)	ADDITIONAL FEE (\$)
X\$50 =	
X200 =	
+360=	
TOTAL ADD'L FEE	

AMENDMENT B	CLAIMS REMAINING AFTER AMENDMENT	HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA
	Total (37 CFR 1.16(n))	Minus	**
Independent (37 CFR 1.16(o))	Minus	***	=
Application Size Fee (37 CFR 1.16(s))			
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))			

SMALL ENTITY

RATE (\$)	ADDITIONAL FEE (\$)
X\$ 25 =	
X100 =	
+180=	
TOTAL ADD'L FEE	

OR OTHER THAN SMALL ENTITY

RATE (\$)	ADDITIONAL FEE (\$)
X\$50 =	
X200 =	
+360=	
TOTAL ADD'L FEE	

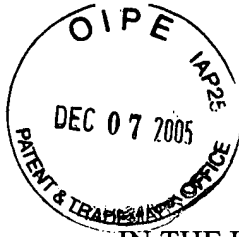
\* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.

\*\* If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".

\*\*\* If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".

The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.



*JFW*

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant :	Michael De Angelo	Art Unit :	Unknown
Serial No. :	11/280,700	Examiner :	Unknown
Filed :	November 14, 2005		
Title :	SYSTEM AND METHOD FOR CREATING AND MANIPULATING INFORMATION CONTAINERS WITH DYNAMIC REGISTER		

Mail Stop Amendment  
 Commissioner for Patents  
 P.O. Box 1450  
 Alexandria, VA 22313-1450

PRELIMINARY AMENDMENT

Prior to examination, please amend the application as indicated on the following pages.

Amendments to the Specification begin on page 2 of this paper.

Remarks/Arguments begin on page 3 of this paper.

CERTIFICATE OF MAILING BY FIRST CLASS MAIL

I hereby certify under 37 CFR §1.8(a) that this correspondence is being deposited with the United States Postal Service as first class mail with sufficient postage on the date indicated below and is addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

Date of Deposit December 5, 2005

Signature *Kelly M. Smith*

Kelly M. Smith  
 Typed or Printed Name of Person Signing Certificate



Applicant : Michael De Angelo  
Serial No. : 11/280,700  
Filed : November 14, 2005  
Page : 2 of 3

Attorney's Docket No.: 17776-002US4  
Preliminary Amendment

Amendments to the Specification:

Please add a new paragraph after the paragraph beginning at page 12, line 12:

Fig. 3B is a graphical representation for a second embodiment of a container having a plurality of containers nested within that container.

Applicant : Michael De Angelo  
Serial No. : 11/280,700  
Filed : November 14, 2005  
Page : 3 of 3

Attorney's Docket No.: 17776-002US4  
Preliminary Amendment

REMARKS

The specification has been amended to add a brief description for Fig. 3B (for support, see, inter alia, specification page 23, lines 20-23 and page 24, lines 1-6). No new matter is added..

Please apply any applicable charges or credits to Deposit Account No. 06 1050.

Respectfully submitted,



Date: December 5, 2005

---

Carl A. Kukkonen, III  
Reg. No. 42,773

Fish & Richardson P.C.  
**Customer No.: 20985**  
12390 El Camino Real  
San Diego, California 92130  
Telephone: (858) 678-5070  
Facsimile: (858) 678-5099

10575607.doc

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

**PATENT APPLICATION FEE DETERMINATION RECORD**

Substitute for Form PTO-875 Effective December 8, 2004

Application or Docket Number

11280700

**APPLICATION AS FILED - PART I**

(Column 1) (Column 2)

FOR	NUMBER FILED	NUMBER EXTRA
BASIC FEE (37 CFR 1.16(a), (b), or (c))	N/A	N/A
SEARCH FEE (37 CFR 1.16(a), (b), or (c))	N/A	N/A
EXAMINATION FEE (37 CFR 1.16(a), (b), or (c))	N/A	N/A
TOTAL CLAIMS (37 CFR 1.16(a))	80 minus 20 =	10
INDEPENDENT CLAIMS (37 CFR 1.16(b))	12 minus 3 =	9
APPLICATION SIZE FEE (37 CFR 1.16(d))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is \$250 (\$125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).	
MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))		

**SMALL ENTITY**

OR

**OTHER THAN SMALL ENTITY**

RATE (\$)	FEE (\$)
N/A	150.00
N/A	\$250
N/A	\$100
X\$ 25 =	250
X100 =	900
+180 =	
TOTAL	1650

OR

RATE (\$)	FEE (\$)
N/A	300.00
N/A	\$500
N/A	\$200
X\$50 =	
X200 =	
+360 =	
TOTAL	

\* If the difference in column 1 is less than zero, enter "0" in column 2.

**APPLICATION AS AMENDED - PART II**

(Column 1) (Column 2) (Column 3)

12/7/05

AMENDMENT A	CLAIMS REMAINING AFTER AMENDMENT	HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA
Total (37 CFR 1.16(a))	Minus	**	=
Independent (37 CFR 1.16(b))	Minus	**	=
Application Size Fee (37 CFR 1.16(s))			
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))			

**SMALL ENTITY**

OR

**OTHER THAN SMALL ENTITY**

RATE (\$)	ADDITIONAL FEE (\$)
X\$ 25 =	
X100 =	
+180 =	
TOTAL ADD'L FEE	

OR

RATE (\$)	ADDITIONAL FEE (\$)
X\$50 =	
X200 =	
+360 =	
TOTAL ADD'L FEE	

AMENDMENT B

CLAIMS REMAINING AFTER AMENDMENT	HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA
Total (37 CFR 1.16(a))	Minus	**
Independent (37 CFR 1.16(b))	Minus	**
Application Size Fee (37 CFR 1.16(s))		
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))		

RATE (\$)	ADDITIONAL FEE (\$)
X\$ 25 =	
X100 =	
+180 =	
TOTAL ADD'L FEE	

OR

RATE (\$)	ADDITIONAL FEE (\$)
X\$50 =	
X200 =	
+360 =	
TOTAL ADD'L FEE	

- \* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.
- \*\* If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".
- \*\*\* If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".

The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

Information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1460, Alexandria, VA 22313-1450.

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

11280700



## UNITED STATES PATENT AND TRADEMARK OFFICE

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

APPLICATION NUMBER	FILING OR 371 (c) DATE	FIRST NAMED APPLICANT	ATTORNEY DOCKET NUMBER
11/280,700	11/14/2005	Michael De Angelo	17776-002US4/

CONFIRMATION NO. 9680

20985  
 FISH & RICHARDSON, PC  
 P.O. BOX 1022  
 MINNEAPOLIS, MN 55440-1022

FORMALITIES  
 LETTER

Date Mailed: 12/14/2005

## NOTICE TO FILE MISSING PARTS OF NONPROVISIONAL APPLICATION

FILED UNDER 37 CFR 1.53(b)

*Filing Date Granted***Items Required To Avoid Abandonment:**

An application number and filing date have been accorded to this application. The item(s) indicated below, however, are missing. Applicant is given **TWO MONTHS** from the date of this Notice within which to file all required items and pay any fees required below to avoid abandonment. Extensions of time may be obtained by filing a petition accompanied by the extension fee under the provisions of 37 CFR 1.136(a).

- The oath or declaration is missing. *A properly signed oath or declaration in compliance with 37 CFR 1.63, identifying the application by the above Application Number and Filing Date, is required.*  
*Note: If a petition under 37 CFR 1.47 is being filed, an oath or declaration in compliance with 37 CFR 1.63 signed by all available joint inventors, or if no inventor is available by a party with sufficient proprietary interest, is required.*

The application is informal since it does not comply with the regulations for the reason(s) indicated below.

The required item(s) identified below must be timely submitted to avoid abandonment:

- A substitute specification in compliance with 37 CFR 1.52, 1.121(b)(3), and 1.125, is required. The substitute specification must be accompanied by a marked up copy as set forth in 37 CFR 1.125(c) and a statement that the specification contains no new matter (see 37 CFR 1.125(b)). The specification, claims, or abstract page(s) submitted is not acceptable and cannot be scanned or properly stored because:
  - Since a preliminary amendment was present on the filing date of the application and such amendment is part of the original disclosure of the application, the substitute specification must include all of the desired changes made in the preliminary amendment. See 37 CFR 1.115 and 1.215.

Applicant is cautioned that correction of the above items may cause the specification and drawings page count to exceed 100 pages. If the specification and drawings exceed 100 pages, applicant will need to submit the required application size fee.

The applicant needs to satisfy supplemental fees problems indicated below.

The required item(s) identified below must be timely submitted to avoid abandonment:

- To avoid abandonment, a surcharge (for late submission of filing fee, search fee, examination fee or oath or declaration) as set forth in 37 CFR 1.16(f) of \$65 for a small entity in compliance with 37 CFR 1.27, must be submitted with the missing items identified in this letter.

**SUMMARY OF FEES DUE:**

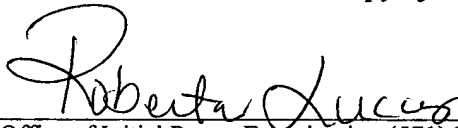
Total additional fee(s) required for this application is **\$65** for a Small Entity

- **\$65** Surcharge.

Replies should be mailed to:    Mail Stop Missing Parts  
  Commissioner for Patents  
  P.O. Box 1450  
  Alexandria VA 22313-1450

---

*A copy of this notice **MUST** be returned with the reply.*



Office of Initial Patent Examination (571) 272-4000, or 1-800-PTO-9199, or 1-800-972-6382  
PART 3 - OFFICE COPY



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant : Michael De Angelo Art Unit : 2161
Serial No. : 11/280,700 Examiner : Unknown
Filed : November 14, 2005
Title : SYSTEM AND METHOD FOR CREATING AND MANIPULATING INFORMATION CONTAINERS WITH DYNAMIC REGISTER

MAIL STOP MISSING PARTS

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

RESPONSE TO NOTICE TO FILE MISSING PARTS OF APPLICATION

In response to the Notice to File Missing Parts of Application under 37 CFR §1.53(b) mailed December 14, 2005 (copy enclosed), applicant claims small entity status (see 37 CFR 1.27) and submits herewith the following:

- Payment of the surcharge of \$65 for late filing of the basic filing fee and/or declaration;
A Combined Declaration and Power of Attorney in compliance with 37 CFR §1.63;
Other: Substitute specification in compliance with 37 CFR 1.52, 1.121 (b)(3), and 1.125 (includes marked version and clean version). No new matter has been added; and
A check in the total amount of \$65 is attached;

CERTIFICATE OF MAILING BY FIRST CLASS MAIL

I hereby certify under 37 CFR §1.8(a) that this correspondence is being deposited with the United States Postal Service as first class mail with sufficient postage on the date indicated below and is addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

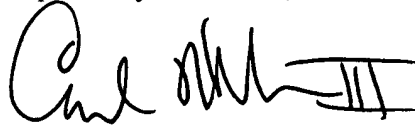
February 14, 2006
Date of Deposit
Veronica Whalen
Signature
Veronica Whalen
Typed or Printed Name of Person Signing Certificate

Applicant : Michael De Angelo  
Serial No. : 11/280,700  
Filed : November 14, 2005  
Page : 2 of 2

Attorney's Docket No.: 17776-002US4

It is understood that this perfects the application and no additional papers or filing fees are required. Please apply any other charges or credits to Deposit Account No. 06-1050.

Respectfully submitted,



Date: 2/14/06

\_\_\_\_\_  
Carl A. Kukkonen, III  
Reg. No. 42,773

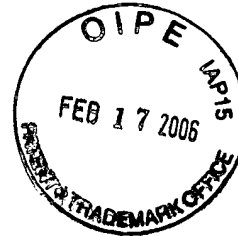
Fish & Richardson P.C.  
12390 El Camino Real  
San Diego, California 92130  
Telephone: (858) 678-5070  
Facsimile: (858) 678-5099

10592212.doc


**UNITED STATES PATENT AND TRADEMARK OFFICE**

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

APPLICATION NUMBER	FILING OR 371 (c) DATE	FIRST NAMED APPLICANT	ATTORNEY DOCKET NUMBER
11/280,700	11/14/2005	Michael De Angelo	17776-002US4/

 20985  
 FISH & RICHARDSON, PC  
 P.O. BOX 1022  
 MINNEAPOLIS, MN 55440-1022

**CONFIRMATION NO. 9680**  
**FORMALITIES**  
**LETTER**

Date Mailed: 12/14/2005

**NOTICE TO FILE MISSING PARTS OF NONPROVISIONAL APPLICATION**

 02/21/2006 SSITHIB1 00000019 11280700  
 01 FC:2051

65.00 00

**FILED UNDER 37 CFR 1.53(b)**
*Filing Date Granted*
**Items Required To Avoid Abandonment:**

An application number and filing date have been accorded to this application. The item(s) indicated below, however, are missing. Applicant is given **TWO MONTHS** from the date of this Notice within which to file all required items and pay any fees required below to avoid abandonment. Extensions of time may be obtained by filing a petition accompanied by the extension fee under the provisions of 37 CFR 1.136(a).

- The oath or declaration is missing. *A properly signed oath or declaration in compliance with 37 CFR 1.63, identifying the application by the above Application Number and Filing Date, is required.*  
*Note: If a petition under 37 CFR 1.47 is being filed, an oath or declaration in compliance with 37 CFR 1.63 signed by all available joint inventors, or if no inventor is available by a party with sufficient proprietary interest, is required.*

The application is informal since it does not comply with the regulations for the reason(s) indicated below.

The required item(s) identified below must be timely submitted to avoid abandonment:

- A substitute specification in compliance with 37 CFR 1.52, 1.121(b)(3), and 1.125, is required. The substitute specification must be accompanied by a marked up copy as set forth in 37 CFR 1.125(c) and a statement that the specification contains no new matter (see 37 CFR 1.125(b)). The specification, claims, or abstract page(s) submitted is not acceptable and cannot be scanned or properly stored because:
  - Since a preliminary amendment was present on the filing date of the application and such amendment is part of the original disclosure of the application, the substitute specification must include all of the desired changes made in the preliminary amendment. See 37 CFR 1.115 and 1.215.

Applicant is cautioned that correction of the above items may cause the specification and drawings page count to exceed 100 pages. If the specification and drawings exceed 100 pages, applicant will need to submit the required application size fee.

The applicant needs to satisfy supplemental fees problems indicated below.



The required item(s) identified below must be timely submitted to avoid abandonment:

- To avoid abandonment, a surcharge (for late submission of filing fee, search fee, examination fee or oath or declaration) as set forth in 37 CFR 1.16(f) of \$65 for a small entity in compliance with 37 CFR 1.27, must be submitted with the missing items identified in this letter.

**SUMMARY OF FEES DUE:**


Total additional fee(s) required for this application is **\$65** for a Small Entity

- **\$65** Surcharge.

Replies should be mailed to:    Mail Stop Missing Parts  
  Commissioner for Patents  
  P.O. Box 1450  
  Alexandria VA 22313-1450

---

*A copy of this notice **MUST** be returned with the reply.*



Office of Initial Patent Examination (571) 272-4000, or 1-800-PTO-9199, or 1-800-972-6382  
PART 2 - COPY TO BE RETURNED WITH RESPONSE



# MARKED VERSION



**SYSTEM AND METHOD FOR CREATING AND MANIPULATING  
INFORMATION CONTAINERS WITH DYNAMIC REGISTERS**

**CROSS-REFERENCE TO RELATED APPLICATIONS**

[0001] The present application is a continuation of U.S. Patent Application No. 09/284,113, entitled System And Method For Creating And Manipulating Information Containers With Dynamic Registers, filed on April 7, 1999, which is incorporated herein in its entirety, and claims the benefit of PCT/US99/01988 filed January 28, 1999 and of U.S. Patent Application No. 60/073,209, filed January 30, 1998.

**BACKGROUND OF THE INVENTION**

[0002] 1. Field of the Invention

The present invention relates generally to computer systems in a multi-user mainframe or mini computer system, a client server network, or in local, wide area or public networks, and in particular, to computer networks for creating and manipulating information containers with dynamic interactive registers in a computer, media or publishing network, in order to manufacture information on, upgrade the utility of, and develop intelligence in, a computer network by offering the means to create and manipulate information containers with dynamic registers.

[0003] 2. Description of the Related Art

In the present day, querying and usage of information resources on a computer network is accomplished by individuals directing a search effort by submitting key words or phrases to be compared to those key words or phrases contained in the content or description of that information resource, with indices and contents residing in a fixed location unchanging except by human input. Similarly, the class of storage medium upon which information resides, its class and subclass organizational structures, and its routes of access all remain fundamentally unaltered by ongoing user queries and usage. Only the direct and intended

intervention of the owner of the information content or computer hosting site changes these parameters, normally accomplished manually by programmers or systems operators at their own discretion or the discretion of the site owner.

[0004] There exists currently in the art a limited means of interfacing a computer user with the information available on computer networks such as the world wide web. Primarily, these means are search engines. Search engines query thousands or tens of thousands of index pages per second to suggest the location of information while the user waits. While factual information can be accessed, the more complex, particular or subtle the inquiry, the more branches and sub-branches need to be explored in a time consuming fashion in order to have any chance of success. Further, there are no such automatic devices that reconstruct the information into more useful groupings or makes it more accessible according to factors attached to the content by the content creator such as the space or time relevancy of its content, or factors attached to the content by the system's compilation and analysis of the accumulated biography of that specific content's readership.

[0005] The utility of wide area and public computer networks is thus greatly limited by the static information model and infrastructure upon which those networks operate.

[0006] One problem is that on a wide area or public network, specific content such as a document remains inert, except by the direct intervention of users, and is modified neither by patterns or history of usage on the network, or the existence of other content on the network.

[0007] Another problem is that content does not reside in an information infrastructure conducive to reconstruction by expert rule-based, fuzzy logic, or artificial

intelligence based systems. Neither the intelligence of other information users nor the expert intelligence of an observant network computer system can be utilized in constructing, or reconstructing information resources. Where content resides in a fixed location and structure, “information” becomes something defined by the mind of the information provider rather than the mind of the information user, where the actual construction and utility of information exists. Information remains, like raw ore, in an unrefined state.

**[0008]** Another problem is that the class of storage medium upon which data resides cannot be system or user managed and altered according to the actual recorded and analyzed hierarchically graded usage of any given information resource residing on that storage medium except by statistical analysis of universal, undefined “hits” or visits to that page or site.

**[0009]** Another problem is that information resource groupings remain fixed on the given storage medium location according to the original installation by the resource author, not altered according to the actual recorded and analyzed hierarchically graded usage of that given information resource. Content itself remains inert, with no possibility of evolution.

**[0010]** A further problem with the prior art is that neither the search templates generated by those more knowledgeable in a given field of inquiry, nor the search strategies historically determined to be successful, or system-constructed according to analyses of search strategies historically determined to be successful, are available to inquiring users. A search template is here defined as one or more text phrases, graphics, video or audio bits, alone or in any defined outline or relational format designed to accomplish an inquiry. Internet or wide area network search may return dozens of briefs to a keyword or key phrase

inquiry sometimes requiring the time-consuming examination of multiple information resources or locations, with no historical relation to the success of any given search strategy.

[0011] A further problem is that there is limited means to add to, subtract from, or alter the information content of documents, databases, or sites without communicating with the owners or operators of those information resources, e.g., contacting, obtaining permission, negotiating and manually altering, adding or subtracting content. Additionally, once so altered, there is not a means to derive a proportionate value, and thereby a proportionate royalty as the information is used.

[0012] A final problem is that the physical residence of a body of data or its cyberspace location may not serve its largest body of users in the most expedient manner of access. Neither the expert intelligence of other information users nor the expert intelligence of an observant computer system is presently utilized by inherent network intelligence to analyze, re-design and construct access routes to information medium except by statistical analysis of universal, undefined "hits" or visits to that page or site.

[0013] Therefore, there is a need for a system and methods for creating and manipulating information containers with dynamic interactive registers defining more comprehensive information about contained content in a computer, media or publishing network, in order to manufacture information on, upgrade the utility of, and develop intelligence in, a computer network by providing a searching user the means to utilize the searches of other users or the historically determined and compiled searches of the system, a means to containerize information with multiple registers governing the interaction of that container, a means to re-classify the storage medium and location of information resources

resident on the network, a means to allow the reconstruction of content into more useful formations, and a means to reconstruct the access routes to that information.

### **SUMMARY OF THE INVENTION**

**[0014]** The present invention is a system and methods for manufacturing information on, upgrading the utility of, and developing intelligence in, a computer or digital network, local, wide area, public, corporate, or digital-based, supported, or enhanced physical media form or public or published media, or other by offering the means to create and manipulate information containers with dynamic registers.

**[0015]** The system of the present invention comprises an input device, an output device, a processor, a memory unit, a data storage device, and a means of communicating with other computers, network of computers, or digital-based, supported or enhanced physical media forms or public or published media. These components are preferably coupled by a bus and configured for multi-media presentation, but may also be distributed throughout a network according to the requirements of highest and best use.

**[0016]** The memory unit advantageously includes an information container made interactive with dynamic registers, a container editor, a search interface, a search engine, a search engine editor, system-wide hierarchical container gateways interacting with dynamic container registers, a gateway editor, a register editor, a data collection means with editor, a data reporting means with editor, an analysis engine with editor, an executing engine with editor, databases, and a means of communicating with other computers as above. These

components may reside in a distributed fashion in any configuration on multiple computer systems or networks.

**[0017]** The present invention advantageously provides a container editor for creating containers, containerizing storing information in containers and defining and altering container registers. A container is an interactive nestable logical domain configurable as both subset and superset, including a minimum set of attributes coded into dynamic interactive evolving registers, containing any information component, digital code, file, search string, set, database, network, event or process, and maintaining a unique network-wide lifelong identity.

**[0018]** The container editor allows the authoring user to create containers and encapsulate any information component in a container with registers, establishing a unique network lifelong identity, characteristics, and parameters and rules of interaction. The authoring user defines and sets the register with a starting counter and/or mathematical description by utilizing menus and simple graphing tools or other tools appropriate to that particular register. The registers determine the interaction of that container with other containers, system components, system gateways, events and processes on the computer network.

**[0019]** Containers and registers, upon creation, may be universal or class-specific. The editor provides the means to create system-defined registers as well as the means to create other registers. The editor enables the register values to be set by the user or by the system, in which case the register value may be fixed or alterable by the user upon creation. Register values are evolving or non-evolving for the duration of the life of the container on



the system. Evolving registers may change through time, space, interaction, system history and other means.

[0020] System-defined registers comprise: (1) an historical container register, logging the history of the interaction of that container with other containers, events and processes on the network, (2) an historical system register, logging the history of pertinent critical and processes on the network, (3) a point register accumulating points based upon a hierarchically rated history of usage, (4) an identity register maintaining a unique network wide identification and access location for a given container, (5) a brokerage register maintaining a record of ownership percentage and economic values, and others.

[0021] The present invention also includes user-defined registers. User defined registers may be created wholly by the user and assigned a starting value, or simply assigned value by the user when that register is pre-existent in the system or acquired from another user, and then appended to any information container, or detached from any container.

[0022] Exemplary user-defined registers comprise (1) a report register, setting trigger levels for report sequences, content determination and delivery target, (2) a triple time register, consisting of a range, map, graph, list, curve or other representation designating time relevance, actively, assigning the time characteristics by which that container will act upon another container or process, passively, assigning the time characteristics by which that container be acted upon by another container or process, and neutrally, assigning the time characteristics by which that container will interact with another container or process, (3) a triple space register, consisting of a range, map, graph, list, curve or other representation designating the domain and determinants of space relevance, actively, assigning the space characteristics by which that content will act upon another container or process, passively,

assigning the space, characteristics by which that content will be acted upon by another container or process, and neutrally, assigning the space characteristics by which that container will interact with another container or process, (4) a domain of influence register, determining the set, class and range of containers upon which that container will act, (5) a domain of receptivity register, determining the set, class and range of containers allowed to act upon that container, (6) a domain of neutrality register, determining the set, class and range of containers with which that container will interact, (7) a domain of containment register, determining the set, class and range of containers which that container may logically encompass, (8) a domain of inclusion register, determining the set, class and range of containers by which that container might be encompassed, (9) an ownership register, recording the original ownership of that containers, (10) a proportionate ownership register, determining the proportionate ownership of that containers, (11) a creator profile register, describing the creator or creators of that container, (12) an ownership address register, maintaining the address of the creator or creators of that container, (13) a value register, assigning a monetary or credit value to that container, and (14) other registers created by users or the system.

**[0023]** Containers are nestable and configurable as both subset and superset and may be designated hierarchically according to inclusive range, such as image component, image, image file, image collection, image database, or if text, text fragment, sentence, paragraph, page, document, document collection, document, database, document library, or any arrangement wherein containers are defined as increasingly inclusive sets of sets of digital components.

[0024] The present invention also includes, structurally integrated into each container, or strategically placed within a network at container transit points, unique gateways, nestable in a hierarchical or set and class network scheme. Gateways gather and store container register information according to system-defined, system-generated, or user determined rules as containers exit and enter one another, governing how containers system processes or system components interact within the domain of that container, or after exiting and entering that container, and governing how containers, system components and system processes interact with that unique gateway, including how data collection and reporting is managed at that gateway. The gateways record the register information of internally nested sub and superset containers, transient containers and search templates, including the grade of access requested, and, acting as an agent of an analysis engine and execution engine, govern the traffic and interaction of those containers and searches with the information resource of which they are the gateway and other gateways. The gateways' record of internally nested and transient container registers, and its own interaction with those containers, is made available, according to a rules-based determination, to the process of the analysis engine by the data collection and/or data reporting means.

[0025] The present invention also includes a means of data storage at any given gateway.

[0026] The present invention also includes a data collection means, residing anywhere on the network, or located at one or more hierarchical levels of nestable container gateways for gathering information from other gateways and analysis engines according to system, system-generated or user determined rules. The data collection means manages the gathering of data regarding network-wide user choices, usage and information about

information, by collecting it from container and gateway registers as those containers and gateways pass through one another. Such statistics as frequency, pattern, and range of time, space and logical class is collected as directed by the analysis engine, and made that data available to the analysis engine by advancing it directly to the analysis engine, or incrementally, to the next greater hierarchically inclusive collection level. The rules of data collection may be manually set or altered by the system manager, or set by the system and altered by the system in its evolutionary capacity.

**[0027]** The present invention also includes a data reporting means, located at one or more hierarchical levels of nestable container gateways for submitting information to other gateways and analysis engines according to system, system-generated or user determined rules. The data reporting means manages the sending of data from the registers, gateways and search templates in a frequency, pattern, and range of time, space and logical class as directed by the analysis engine, and makes that data available to the analysis engine by advancing it directly to the analysis engine, or incrementally to the next greater hierarchically inclusive reporting level. The rules of data collection may be manually set or altered by the system manager, or set by the system and altered by the system in its evolutionary capacity. The data reporting means may be established to work in concert, in redundancy, or in contiguous or interwoven threads of hierarchically nested containers.

**[0028]** The present invention also includes an analysis engine that receives, reports and collects information regarding the interaction of user searches with gateways and container registers, as well as container registers with other container registers, and container registers with gateways. The analysis engine analyzes the information submitted by the gateways and instructs the execution engine to create new information containers, content

assemblages, storage schemes, access routes, search templates, and gateway instructions. The analysis engine includes an editor that provides a system manager with a means of editing the operating principles of that engine, governing data reporting, data collection, search template loading, gateway instructions, and other.

**[0029]** The present invention also includes an execution engine, fulfilling the instructions of the analysis engine, to create new information containers, content sun and superset assemblages, storage schemes, access routes, search templates, and gateway instructions. The execution engine includes an editor that provides a system manager with a means of editing the operating principles of that engine, governing data reporting, data collection, search template loading, gateway instructions, and other.

**[0030]** The present invention also includes a search interface or browser. The search interface provides a means for a searching user to submit, record and access search streams or phrases generated historically by himself, other users, or the system. Search streams or phrases of other users are those that have been historically determined by the system to have the highest probability of utility to the searching user. Search streams or phrases generated by the system are those that have been constructed by the system through the analysis engine based upon the same criteria.

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

**[0031]** FIG. 1 is a block diagram of a first and preferred embodiment of a system constructed according to the present invention.

**[0032]** FIG. 2 A is block diagram of a preferred embodiment of the memory unit.

[0033] FIG. 2 B is an exemplary embodiment of a computer network showing computer servers, personal computers, workstations, Internet, Wide Area Networks, Intranets in relationship with containers and gateways.

[0034] FIG. 2B1 is an exemplary embodiment of a computer network showing computer servers, personal computers, workstations, Internet, Wide Area Networks, Intranets in relationship with containers and gateways and exemplary locations of gateway storage in proximity to one or more of the various sites.

[0035] FIGS. 2C through 2H are exemplary embodiments in block diagram form of computer network components showing a possible placement of nested containers, computer servers, gateways, and the software components named in Fig. 2 A on a network.

[0036] FIG. 3A is a graphical representation for one embodiment of a container having a plurality of containers nested within that container.

[0037] Fig. 3B is a graphical representation for a second embodiment of a container having a plurality of containers nested within that container.

[0037][0038] FIG. 3C is a drawing showing elements that might be logically encapsulated by a container. FIG. 4 is a drawing of an information container showing a gateway and registers logically encapsulating containerized elements.

[0038][0039] FIG. 5 is a flowchart showing a preferred method for the containerization process and container editor operating on the communication device.

[0039][0040] FIG. 6 is a flowchart showing a preferred method for searching for containers within a node.

[0040][0041] FIG. 7 is a flowchart further showing a preferred method for searching for containers over one or more gateways.

[0041][0042] FIG. 8 is a flowchart showing a method for performing the data collection and reporting on containers.

[0042][0043] FIG. 9 is a flowchart showing the operation of the analysis engine.

[0043][0044] FIG. 10 is a flowchart showing the operation of the execution engine.

[0044][0045] FIG. 11 is a flowchart showing the operation of the gateway editor.

[0045][0046] FIG. 12 is a flowchart showing the operation of the gateway process.

[0046][0047] FIG. 13A is a drawing showing an example of nested containers, gateways, registers, analysis engines and an execution engine prior to container reconstruction as depicted in 13 B, 13 C and 13 D.

[0047][0048] FIG. 13B is a drawing showing the reconstructed nested containers of Figure 13A.

[0048][0049] FIG. 13C is a drawing showing further reconstruction of nested containers, with a container relocated to reside within another container.

[0049][0050] FIG. 13D is a drawing showing a flowchart of the reconstruction process

[0050][0051] FIG. 14 is a drawing showing the screen interface of the container editor.

[0051][0052] FIG. 15 is a drawing showing the screen interface of the gateway editor.

[0052][0053] FIG. 16 is a drawing showing the screen interface of the search interface.

[0053][0054] FIG. 17 is a drawing of a generic application program showing a drop-down menu link, and a button link to the containerization process or container editor.

**DESCRIPTION OF THE PREFERRED EMBODIMENT****[0054][0055] THE SYSTEM**

**[0055][0056]** Referring now to FIG. 1, a preferred embodiment of a system 10 for creating and manipulating information containers with dynamic interactive registers in a computer, media, or publishing network 201 in order to manufacture information on, upgrade the utility of, and develop intelligence in that network 201, is shown. The system 10 preferably comprises an input device 24, an output device 16, a processor 18, a memory unit 22, a data storage device 20, and a communication device 26 operating on a network 201. The input device 24, an output device 16, a processor 18, a memory unit 22, a data storage device 20, are preferably coupled together by a bus 12 in a von Neumann architecture. Those skilled in the art will realize that these components 24, 16, 18, 22, 20, and 26 may be coupled together according to various other computer architectures including any physical distribution of components linked together by the communication device 26 without departing from the spirit or scope of the present invention, and may be infinitely nested or chained, both as computer systems within a network 202, and as networks within networks 201.

**[0056][0057]** The output device 16 preferably comprises a computer monitor for displaying high-resolution graphics and speakers for outputting high fidelity audio signals. The output device 16 is used to display various user interfaces 110, 125, 210, 300, 510, 610, 710, as will be described below, for searching for and containerizing information, and editing the container gateways, containers, container registers, the data reporting means and the data collection means, and the search, analysis and execution engines. The author uses the input device 24 to manipulate icons, text, charts or graphs, or to select objects or text, in the



process of packaging, searching or editing in a conventional manner such as in the Macintosh of Windows operating systems.

~~10057~~10058 The processor 18 preferably executes programmed instruction steps, generates commands, stores data and analyzes data configurations according to programmed instruction steps that are stored in the memory unit 22 and in the data storage device 20. The processor 22 is preferably a microprocessor such as the Motorola 680(x)0, the Intel 80(x)86 or Pentium, Pentium II, and successors, or processors made by AMD, or Cyrix CPU of the any class.

~~10058~~10059 The memory unit 22 is preferably a predetermined amount of dynamic random access memory, a read-only memory, or both. The memory unit 22 stores data, operating systems, and programmed instructions steps, and manages the operations of all hardware and software components in the system 10 and on the network 201, utilizing the communication device 26 whenever necessary or expeditious to link multiple computer systems 202 within the network 201.

~~10059~~10060 The data storage device 20 is preferably a disk storage device for storing data and programmed instruction steps. In the exemplary embodiment, the data storage device 20 is a hard disk drive. Historical recordings of network usage are stored on distributed and centralized data storage devices 20.

~~10060~~10061 The preferred embodiment of the input device 24 comprises a keyboard, microphone, and mouse type controller. Data and commands to the system 10 are input through the input device 24.

~~10061~~10062 The present invention also includes a communication device 26. The communication device 26 underlies and sustains the operations of, referring now also to Fig 2

the analysis 400 and execution 500 engines, the data reporting 600 and collection 700 means, the container editor 110, the search interface 300, and the search engine 320, providing the means to search, access, move, copy, utilize or otherwise perform operations with and on data. The communication device 26 utilizes one or more of the following technologies: modem, infrared, microwave, laser, photons, electrons, wave phenomena, cellular carrier, satellite, laser, router hub, direct cabling, physical transport, radio, broadcast or cable TV or other to communicate with other computers, digital-supported television, computer networks, or digital-based or supported public or published media, or physical media forms, on any a local, wide area, public, or any computer-based computer supported, or computer interfaced network, including but not limited to the Internet. It also allows for the functioning and distribution of any container 100 or container component herein described to reside anywhere on any computer system in any configuration on that local, wide area, public, or corporate computer-based or computer related network, or digital-based or supported media form.

10062|10063| Referring now to Figure 2 A, a preferred embodiment of the memory unit 22 is shown. The memory unit includes: an interactive information container 100, a container editor 110, container registers 120, a container register editor 125, system-wide hierarchical container gateways 200, gateway storage 205, gateway editors 210, engine editors 510, a search interface 300, search engine 320, analysis engine 400, execution engine 500, a data reporting module, 600, a data reporting editor 610, a data collection module 700, a data collection editor 710, screen interfaces (GUI's) 936, menu or access buttons from generic computer programs 937, and databases 900, all residing in memory optimized between a data storage means 20 such as magnetic, optical, laser, or other fixed storage, and a memory means 22 such as RAM. The memory unit 22 functions by operating on

communications network 12 with a communication device 26 on multiple computer systems 202 within the network 201. These components will be described first briefly in the following paragraphs, then in more detail with reference to Figures 3 A through 17.

~~10063~~~~10064~~ Those skilled in the art will realize that these components might also be stored in contiguous blocks of memory, and that software components or portions thereof may reside in the memory unit 22 or the data storage means 20.

~~10064~~~~10065~~ The present invention includes information containers 100 as noted above. The information container 100 is a logically defined data enclosure which encapsulates any element or digital segment (text, graphic, photograph, audio, video, or other), or set of digital segments, or referring now to FIG. 3 C, any system component or process, or other containers or sets of containers. A container 100 at minimum includes in its construction a logically encapsulated portion of cyberspace, a register and a gateway. A container 100 at minimum encapsulates a single digital bit, a single natural number or the logical description of another container, and at maximum all defined cyberspace, existing, growing and to be discovered, including but not limited to all containers, defined and to be defined in cyberspace. A container 100 contains the code to enable it to interact with the components enumerated in 2 A, and to reconstruct itself internally and manage itself on the network 201.

~~10065~~~~10066~~ The container 100 also includes container registers 120. Container registers 120 are interactive dynamic values appended to the logical enclosure of an information container 100, and serve to govern the interaction of that container 100 with other containers 100, container gateways 200 and the system 10, and to record the historical interaction of that container 100 on the system 10. Container registers 120 may be values

alone or contain code to establish certain parameters in interaction with other containers 100 or gateways 200.

~~10066~~10067 The present invention also includes container gateways 200. Container gateways 200 are logically defined gateways residing both on containers 100 and independently in the system 10. Gateways 200 govern the interactions of containers 100 within their domain, and alter the registers 120 of transiting containers 100 upon ingress and egress.

~~10067~~10068 The present invention also includes container gateway storage 205 to hold the data collected from registers 120 of transient containers 100 in order to make it available to the data collection means 700 and the data reporting means 600, and to store the rules governing the operations of its particular gateway 200, governing transiting containers upon ingress and egress, and governing the interactive behavior of containers 100 within the container 100 to which that gateway 200 is attached. Gateway storage 205 may be located on gateways 200 themselves, containers 100 or anywhere on the network 202, 201, including but not limited to Internet, Intranet, LAN, WAN, according to best analysis and use.

~~10068~~10069 The memory unit 22 also includes an execution engine 500 to perform the functions on the system 10 as directed by the analysis engine after its analysis of data from the data reporting means 600, the data collection means 700, and the search interface 300.

~~10069~~10070 The memory unit 22 also includes a search interface 300, by which the user enters, selects or edits search phrases or digital strings to be used by the search engine 320 to locate containers 100.

[0070][0071] The memory unit 22 also includes an analysis engine 400 which performs rules based or other analysis upon the data collected from the search interface 300 and the data collection 700 and data reporting 600 means.

[0071][0072] The memory unit 22 also includes a data reporting means 600, by which means the information collected by gateways 200 from transient containers 100 is sent to the analysis engine 400.

[0072][0073] The memory unit 22 also includes a data collection means 700, by which means the analysis engine 400 gathers the information collected by gateways 200 from transient containers 100.

[0073][0074] The memory unit 22 also includes a container editor 110 for creating, selecting, acquiring, modifying and appending registers 120 and gateways 200 to containers 100, for creating, selecting, acquiring, and modifying containers, and for selecting content 01 to encapsulate.

[0074][0075] The memory unit 22 also includes a register editor 125, for creating, selecting, acquiring and modifying container registers 120 and establishing and adjusting the values therein.

[0075][0076] The memory unit 22 also includes a gateway editor 210, by which means the user determines the rules governing the interaction of a given gateway 210 with the registers 120 of transient containers 100, governing transiting containers upon ingress and egress, and governing the interactive behavior of containers within the container to which that gateway is attached.

~~10076~~~~10077~~ The memory unit 22 also includes databases 900, by which means the analysis engine 400, the execution engine 500, the gateways 100, the editors 110, 125, 210, 510, 610, 710, and the search interface 300, store information for later use.

~~10077~~~~10078~~ The memory unit 22 present invention also includes a search engine 320 by which means the user is able to locate containers 100 and, referring now to Fig. 4, containerized elements 01.

~~10078~~~~10079~~ The memory unit 22 present invention also includes an engine editor 510, by which means the user establishes the rules and operating procedures for the analysis engine 400 and the execution engine 500.

~~10079~~~~10080~~ The memory unit 22 present invention also includes a reporting means editor 610, by which means the user establishes the rules and schedule under which the information collected by gateways 200 from transient containers 100 will be sent to the analysis engine 400.

~~10080~~~~10081~~ The memory unit 22 present invention also includes a collection means editor 710, by which means the user establishes the rules and schedule under which the analysis engine 400 will gathers the information collected by gateways 200 from transient containers 100.

~~10081~~~~10082~~ The memory unit 22 present invention also includes screen interfaces (GUI's) 936, specifically designed to simplify and enhance the operations of the container editor 110, the gateway editor 210, and the search interface 300.

~~10082~~~~10083~~ The present invention also includes a menu or button access 937, by which a user utilizing any generic computer program may access the system 10 or the container editor 110 from a menu selection(s) or button(s) within that program.

[0083][0084] The present invention also includes a computer, media or publishing network 201, comprising computers, digital devices and digital media 202 and a communication device 26, within which the components enumerated in Fig. 2 A interact, compiling, analyzing, and altering containers 100 and the network 201 according to information gathered from container registers 120.

[0084][0085] The memory unit 22 also includes one or more computers 202, by which means the components of Fig 1 sustain the operations described in Fig. 2 A.

[0085][0086] The memory unit 22 also includes flat or relational databases 900, used where, and as required. Databases are used to store search phrases, search templates, system history for the analysis engine and execution engine, container levels and container, sites and digital elements, or any and all storage required to operate the system.

[0086][0087] Referring now to FIG. 2 B, a drawing of a computer network 201 as a system 10, showing a possible placement of nested containers 100, computer servers, gateways 200, on the sites described below. (Note: Fig. 2 B utilizes in parts the same numbering scheme as Fig. 13 A, 13 B, 13 C, 13 D and as Fig. 2 A.) In FIG. 2 B various exemplary sites are shown, any or all of which might interact dynamically within the system. Site 1 shows a single workstation with a container and gateway connected to an Intranet. (Individual containers may be a floppy or CD-Rom to be downloaded or inserted.) Site 2 shows a server with a gateway in relationship to various containers.. Site 3 shows an Internet web page with a container residing on it. Site 4 shows a personal computer with containers and a gateway connected to the Internet. Site 5 shows a configuration of multiple servers and containers on a Wide Area Network.. Site 6 shows a workstations with a gateway and containers within a container connected to a Wide Area Network. Site 7 shows an

independent gateway, capable of acting as a data collection and data reporting site as it gathers data from the registers of transiting containers, and as an agent of the execution engine as it alters the registers of transient containers. A container 100 contains the code to enable it to interact with the components enumerated in 2A, and to reconstruct itself internally and manage itself on the network 201. The code resides in and with the container in its registers and gateway definitions and controls. Additional system code resides in all sites to manage the individual and collective operation and oversight of the components enumerated in 2A, with the specific components distributed amongst the sites according to the requirements of optimization.

~~10087~~[0088] Referring now to Fig. 2 B 1 various exemplary sites are shown as described above in Fig. 2 B, with the addition of possible location of one or more gateway storage 205 locations.

~~10088~~[0089] Referring now to Figures 2 C through 2 H, various exemplary sites with one or more of the logical components of the system 10 in relationship are shown. Site 1 comprises an interactive information container 100, a container editor 110, container registers 120, a container register editor 125, system-wide hierarchical container gateways 200, gateway storage 205, gateway editors 210, engine editors 510, a search interface 300, search engine 320, analysis engine 400, execution engine 500, a data reporting means 600, a data reporting means editor 610, a data collection means 700, a data collection means editor 710, and databases 900, all residing on data storage means 20, utilizing the memory unit to function 22, operating on communications network 12 with a communication device 26.



~~100891~~10090 Site 2 comprises an interactive information container 100, a container editor 110, container registers 120, a container register editor 125, system-wide hierarchical container gateways 200, gateway storage 205, gateway editors 210, engine editors 510, search engine 320, analysis engine 400, execution engine 500, a data reporting means 600, a data reporting means editor 610, a data collection means 700, a data collection means editor 710, and databases 900, all residing on data storage means 20, utilizing the memory unit to function 22, operating on communications network 12 with a communication device 26.

~~100901~~10091 Site 3 comprises an interactive information container 100, a container editor 110, container registers 120, a container register editor 125, hierarchical container gateways 200, gateway storage 205, gateway editors 210, and databases 900, all residing on data storage means 20, utilizing the memory unit to function 22, operating on communications network 12 with a communication device 26.

~~100911~~10092 Site 4 comprises an interactive information container 100, a container editor 110, container registers 120, a container register editor 125, hierarchical container gateways 200, gateway storage 205, gateway editors 210, a search interface 300, and databases 900, all residing on data storage means 20, utilizing the memory unit to function 22, operating on communications network 12 with a communication device 26.

~~100921~~10093 Site 5 comprises an interactive information container 100, container registers 120, a container register editor 125, hierarchical container gateways 200, gateway storage 205, and databases 900, all residing on data storage means 20, accessed and utilized by non-resident memory unit 22, operating on communications network 12 with a communication device 26.

1009310094 Site 6 includes an independent analysis engine 400, execution engine 500, data collection means 700, and data reporting means 600 gateway editors 210, engine editors 510, a data reporting means editor 610, a data collection means 700, a data collection means editor 710, and databases 900, all residing on data storage means 20, utilizing the memory unit to function 22, operating on communications network 12 with a communication device 26.

1009410095 Referring now to FIG. 3 A and FIG. 3 B, a block diagram of several nested information containers is shown, including examples of elements, e.g., code 1100, text 1200, audio 1300, video 1400, photograph 1500, graphic images 1600, and examples of possible container level classifications in increasing size, e.g., element 10900000, document 10800000, database 10700000, warehouse 10600000, domain 10500000, and continuing increasingly larger on Fig 3 (B), subject 10400000, field 10300000, master field 10200000, species 10100000. Containers may be infinitely nested and assigned any class, super class or sub class scheme and description by the creator of the container to govern nesting within that container. In addition to digital elements, containers may also include system process and components, including containerization itself.

1009510096 Referring now to FIG. 3 C, a block diagram of an information container system is shown, listing, without any relationship indicated, some of the possible system components and processes, or sets thereof, that may be encapsulated as elements 01 in an information container 100. An information container 100 may include one or more of the following: any unique, container 100, gateway 200, output device 16, input device 24, output device process 160, input device process 240, data storage device 20, data storage device process 2000, processor 18, bus 12, content 01, search process 02, interface 04, memory unit

22, communication device 26, search interface 300, search process 98, network 201, class of device, process or content 999, class of process at any unique class of device 990, process at any unique device 99, editor 110, 125, 210, 510, 610, 710, engine 320, 400, 500, containerization process 1098, or process 08.

~~0096~~~~0097~~ Any container may include (n) other containers, to infinity. The use of value evolving container registers 120 in conjunction with gateways 200, data reporting modules 600, data collection modules 700, the analysis engine 400, and the execution engine 500 provides the information container 100 with extensive knowledge of the use, operation of its internal contents, prior to, during and after those contents' residence within that container 100, and extensive knowledge of the use, operation and contents of the system 10 external to itself, and allows the container 100 to establish and evolve its own identity and course of interaction on the system 10. Further, containers 100, as logical enclosures, can exist and operate independent of their digital contents, whether encapsulating audio, video, text, graphic, or other.

~~0097~~~~0098~~ Referring now to FIG. 4, a block diagram of an information container 100 is shown. The information container 100 is a logically defined data enclosure which encapsulates any element, digital segment (text, graphic, photograph, audio, video, or other), set of digital segments as described above with reference to FIG. 3 (C), any system component or process, or other containers or sets of containers. The container 100 comprises the containerized elements 01, registers 120 and a gateway 200.

~~0098~~~~0099~~ Registers 120 appended to an information container 110 are unique in that they operate independently of the encapsulated contents, providing rules of interaction, history of interaction, identity and interactive life to that container 100 through the duration

of its existence on a network 201, without requiring reference to, or interaction with, its specific contents. They enable a container 100 to establish an identity independent of its contents. Additionally, registers 120 are unique in that their internal values evolve through interaction with other containers 100, gateways 200, the analysis engine 400, the execution engine 500, and the choices made by the users in the search interface 300, the container editor 110, the register editor 125, the gateway editor 210, the engine editor 510. Registers 120 are also unique in that they can interact with any register of a similar definition on any container 100 residing on the network 201, independent of that container's contents. Registers 120, once constructed, may be copied and appended to other containers 100 with their internal values reset, to form new containers. Register values, when collected at gateways 200 and made available to the analysis engine 400 through the data collection means 700 and the data reporting means 600, provide an entirely new layer of network observation and analysis and operational control through the execution engine 500. Registers 120 accomplish not only a real time information about information system, but also a real time information about information usage on a network. Further, because the user base of a network determines usage, the system 10, in gathering information about information usage, is observing the choices of the human mind. When these choices are submitted to the analysis of a rules-based or other analysis engine 400, the system 10 becomes capable of becoming progressively more responsive to the need of the user base, in effect, learning to become more useful by utilizing the execution engine 500 to create system-wide changes by altering the rules of gateway 200 interaction and thereby altering the registers 120 of transient containers 100 and establishing a complete evolutionary cycle of enhanced utility.

[0099][00100] Further, in establishing the pre-defined registers as described in the following four paragraphs, the following unique aspects of information about information are utilized for the first time: 1) the dynamic governance of information according to its utility through time, in active, passive and neutral aspects, as explained below; 2) the dynamic governance of information according to its utility through space in active, passive and neutral aspects, as explained below; 3) the dynamic governance of information according to its ownership, as explained below; 4) the dynamic governance of information according to its unique history of interaction as an identity on a network, as explained below; 5) the dynamic governance of information according to the history of the system on which it exists, as explained below; 6) the dynamic governance of information according to established rules of interaction, in active, passive and neutral aspects, as explained below; 7) the dynamic governance of information according to the profile of its creator, as explained below; 8) the dynamic governance of information according to the value established by its ongoing usage, as explained below; 9) the dynamic governance of information according to its distributed ownership, as explained below; 10) the dynamic governance of information according to what class of information it might be incorporated into, and according to what class of information container it might incorporate, as explained below; 11) the dynamic governance of information according to self-reporting, as explained below.

[00100][00101] Referring now to Fig 4, registers 120 may be (1) pre-defined, (2) created by the user or acquired by the user, or (3) system-defined or system-created. Pre-defined registers 120 are those immediately available for selection by the user within a given container editor as part of that container editor, in order that the user may append any of those registers 120 to a container 100 and define values for those registers 120 where required.

Registers 120 created by the user are those conceived and created by a specific user or user group and made immediately available for selection by the user or user group in conjunction with any of a wide number of container editors, in order that the user may append any of those registers 120 to a container 100 and define values for those registers 120 where required. Registers 120 acquired by the user are those registers existing network-wide 201, created by the user base, that might be located and acquired by the user in order that the user may append any of those registers 120 to a container 100 and define values for those registers 120 where required. System-defined registers are those registers whose values are set and/or controlled by the system 10. System-created registers are those registers created by the system 10.

~~1001011~~1001021 Registers 120 are user or user-base created or system-created values or ranges made available by the system 10 to attach to a unique container, and hold system-set, user-set, or system-evolved values. Values may be numeric, may describe domains of time or space, or may provide information about the container 100, the user, or the system 10. Registers 120 may be active, passive or interactive and may evolve with system use. Pre-defined registers include, but are not limited to, system history 110000, container history 101000, active time 102000, passive time 103000, neutral time 104000, active space 111000, passive space 112000, neutral space 113000, containment 105000, inclusion 106000, identity 114000, value 115000, ownership 107000, ownership addresses 116000, proportionate ownership 117000, creator profile 108000, receptivity 118000, influence 119000, points 109000, others 120000, reporting 121000, neutrality 122000, acquire 123000, create 124000, content title 125000, content key phrase(s) 126000, and content description 127000, security 12800, and parent rules 129000.

~~100102~~~~100103~~ Pre-defined registers comprise an historical container register 101000, logging the history of the interaction of that container 100 with other containers, events and processes on the network 201, an historical system register 110000, logging the history of pertinent critical and processes on the network, a point register 109000 accumulating points based upon a hierarchically rated history of usage, an identity register 114000 maintaining a unique network wide identification and access location for a given container specifying a unique time and place of origin and original residence, a proportionate ownership register 117000 maintaining a record of ownership percentage and economic values, and others 120000.

~~100103~~~~100104~~ User-defined registers include a report register 121000 setting trigger levels for report sequences, content determination and delivery target, three time registers, consisting of a range, map, graph, list, curve or other designating time relevance, 102000 assigning the time characteristics by which that container will act upon another container or process, 103000 assigning the time characteristics by which that container be acted upon by another container or process, and 104000 assigning the time characteristics by which that container will interact with another container or process, three space registers, consisting of a range, map, graph, list, curve or other designating the domain and determinants of space relevance, 111000 assigning the space characteristics by which that content will act upon another container or process, 112000 assigning the space, characteristics by which that content will be acted upon by another container or process, and 113000 assigning the space characteristics by which that container will interact with another container or process, a domain of influence register 119000, determining the set, class and range of containers upon which that container will act, a domain of receptivity register

118000, determining the set, class and range of containers allowed to act upon that container, a domain of neutrality register 122000, determining the set, class and range of containers with which that container will interact, a domain of containment register 105000, determining the set, class and range of containers which that container may logically encompass, a domain of inclusion 106000 register, determining the set, class and range of containers by which that container might be encapsulated, an ownership register 107000, recording the original ownership of that containers, a creator profile register 108000, describing the creator or creators of that container, an ownership address register 116000, maintaining the address of the creator or creators of that container, a value register 115000, assigning a monetary or credit value to that container, other registers 120000 created by users or the system, a reporting register 121000, determining the content, scheduling and recipients of information about that container, a neutrality register 122000, an acquire register 123000, enabling the user to search and utilize other registers residing on the network, a create register 124000, enabling the user to construct a new register, a content title register 125000, naming the contents of the container, a content key register, 126000, identifying the container contents with a key phrase generated by the user and/or the system based upon successful usage of that phrase in conjunction with the utilization of the information within that container 100, a content description register 127000, identifying the container contents with additional description, a security register 128000, controlling container security, and a parent container register 129000, storing the rules governing container interaction as dictated by the parent (encapsulating) container.

[00104][00105] The container also includes a gateway 200 and gateway storage 205.



[00105][00106] Gateways 200 are logically defined passageways residing both on containers 100 and independently in the system 10. Gateways 200 govern the interactions of containers 100 encapsulated within their domain by reading and storing register 120 information of containers entering and exiting that container 100.

[00106][00107] The present invention also includes container gateway storage 205. Gateway storage 205 stores information regarding the residence, absence, transience, and alteration of encapsulated and encapsulating containers 100, and their attached registers 120, holding the data collected from registers 120 of transient containers 100 in order to make it available to the data collection means 700 and the data reporting means 600, and storing the rules governing the operations of its particular gateway 200.

[00107][00108] Referring now to FIG. 5, a flow chart of the preferred method for creating a container 100 is shown.

[00108][00109] Input is received from the user selecting a container level through use of a drop-down menu 10100. A menu of all possible container classes within the subset and superset scheme of multiple hierarchically nested containers, i.e.; element, document, file, database, warehouse, domain, and more, is displayed on the output device 10200. Input is received from the user selecting a class 10300.

[00109][00110] A graphic representation of a container in that class, with registers common to all containers as well as registers unique to its class is displayed 10301.

[00110][00111] Input is received from the user choosing to “create” 10400, “edit” 10500, or “locate” 10600.

[00111][00112] When the input of “create” 10400 is received from the user, a container template in that class appears 10410. Input from the user is then received adding or

selecting a register 10540 to append to that container template. When input is received from the user adding a register, a list of registers that might be added to that class of container is made available to select 10550. Input is received from the user selecting a register 10560 and editing it 10570. The menu returns to “add or select” 10540.

100112|100113| If the input of “locate” 10600 is received from the user, the system prompts the user to enter the identity of the container or class of containers 10605. The system locates the container(s) 10610. Input is received from the user selecting a container 10620. The system prompts the user for a security code for permission to access the container for template use, or to alter its registers, or to alter its content 10630. . Input is received from the user entering a name and password providing access to one of the security levels 10640. Input is received from the user editing the container accordingly by transition to step 10500 and performing the steps for editing.

100113|100114| If the input of “edit” 10500 is received, a list of containers available to edit at that level is shown 10510. Input is received from the user selecting a container 10520. That container appears, available to edit 10530. Input is received from the user selecting “add” or “select” registers 10540 by the user clicking on the graphically depicted register, or from a drop down menu. Input is received from the user selecting the register to edit 10560. Input is received from the user selecting “modify” or “delete” for that register 10565. If input is received from the user to “delete,” that register is severed from the container. If input is received from the user to “modify”, the register editor 10570 screen appropriate to that register appears, i.e., an x-y type graph to define a curve of relevant active time, in which the user manipulates the x-y termini, scale and curve, or a global map in which Input is received from the user selecting the locale of active space, whether zip code, city,

county, state, country, continent, plant or other. When input is received from the user saving the definition, the screen returns to the main container screen to make another selection available. . Input is received from the user defining as many registers as he chooses. One of the registers may be named "new register." Input is received from the user selecting the new register, and if chosen by the user, defining a wholly unique and new kind of register by the user entering input into the register editor 125.

[00114][00115] When the input is received from the user choosing to add a register, a list of registers that might be added to that class of container are made available to select 10550. Input is received from the user selecting a register 10560 and editing it 10570. The menu returns to "add or select" 10540, and in turn to Input – Select Container.

[00115][00116] Input may then be received from the user choosing to add, modify, or delete the container contents 10700. Once the registers are defined, input is received from the user indicating completion and the interface reverts to the container editor. When input is received from the user choosing "select component" (to select the component to containerize) from the main menu bar 10700, a window appears allowing the user to select any file, component, or other container. If for example, the user were creating a warehouse container, and wishes to incorporate several databases into that container, input would then be received from the user selecting "database." The program would prompt the user for the location (directory) of that database or container. If the requested selection is not containerized, input may then be received from the user choosing to containerize the element at that time, after which the program returns to "select component." Once input is received from the user defining the database location, the program logically encases the directory or directories in the defined container. The above procedure may be repeated as many times as

desired to include multiple databases within a single container. While logical simplicity would dictate that all containers within a container be of the same subset, it would be possible for input to be received from the user choosing containers of any subset to include in the container. When input is received from the user choosing "finished," the container is created with a unique network identity, preferably through some combination of exact time and digital device serial number, or centralized numbering system, or other means. The container 100 contains all digital code, including data and program software from the selected items or containers.

[00116][00117] Input may then be received from the user to publish the container 11100 at a user-identified or system suggested location 11200 to be selected 11400.

[00117][00118] Input is received from the user to "publish", from the main menu bar 11100. Input is received from the user choosing to leave the container where it was created, move or copy it to another drive, directory, computer, or network the user designates, or select the location from location options offered by the system 11200, or submit, or duplicate and submit, the container to the analysis engine 400 for intelligent inclusion in other containers, thus allowing the system to publish the container as instructed or choose the residence of the container 11400.

[00118][00119] If input is received from the user to choosing to "move," or "copy" a browse function allows the user to name the new location or browse a list of possible locations. If input is received from the user choosing to "submit," a browser function allows the user to name the analysis search engine 310 or browse a list of possible analyses engines. When input is received from the user choosing the residence of the container 11300, the program restores the search interface screen.

[00119][00120] Referring now to FIG. 6, a flow chart of the method for searching for containers 100.

[00120][00121] When input is received from the user selecting “search interface” from the main title bar, the search interface screen appears. The user is given the choice of containerizing selected content or requesting that container levels be displayed 30100. From a drop down menu another menu appears allowing input to be received from the user selecting the container level 30200. Input is received from the user selecting the container level (from the smallest component to the whole system) 30300.

[00121][00122] Input is received 30310 from the user selecting the phrases, containers or components, which then are re-submitted to the same process, until the input is received from the user selecting a specific site or container.

[00122][00123] The search phrase, whether containerized or not, is submitted simultaneously to the search engine 30400 and the analysis engine 30500.

[00123][00124] The screen then reports in a selection menu, the number of applicable sites found by the search engine 30410, the number of historically proven applicable sites found by the analysis engine 30410, the number of historically proven applicable containers at the selected container level or any container level found by the analysis engine 30410, and the number of historically proven new search phrases or digital segments found by the analysis engine 30320. Input is received from the user selecting one of the named sets above 30330. If input is received from the user choosing the search engine, the search interface lists the applicable site titles with a brief description 30410. If input is received from the user choosing the site list of the analysis, the search interface lists the applicable site titles with a brief description 30410. If input is received from the user

choosing the container list of the analysis engine, the search interface lists the applicable container titles with a brief description 30410. If input is received from the user selecting a container 30420, the system offers the means to view titles and descriptions of sub-containers at any chosen class level. . If input is received from the user choosing the phrase list of the analysis engine, the search interface lists the applicable phrases or digital segments with a brief description 30320. The search and search result cycle repeats until input is received from the user choosing to go to an individual container or site.

[00124][00125] Input is received from the user entering text or any digital string describing his search objectives into a text or search box. When input is received from the user submitting the search string, the system provides the option of containerizing the search through the container editor 110. Once the search container 101 is created, the system restores the search interface 300 screen the user.

[00125][00126] Input is received from the user selecting “search”, “supported search” or “both” from another drop-down menu and from submitting the search. When input is received from the user selecting “search” 30310, the search phrase is submitted to the search engine 30400, which searches both content and the appropriate container registers, as pre-indexed in the search engine, and returns a list of appropriate locations, components or containers. When input is received from the selecting “supported search”, the search phrase is submitted to the analysis engine search support, which returns a list, in a drop-down menu, of search phrases or individual containers, for any and all container levels, used by other users or created by the system and known to be historically successful for the described effort and the described searching user, as per the results of the analysis search engine. Input is received from the user selecting a new search phrase or specific container from the drop

down menu 30330. When input is received from the user choosing a new search phrase, that phrase is also submitted to the analysis engine 30500 which returns a list of pre-compiled historically proven sites, components or containers associated with that search phrase 30320. Input is received from the user choosing a selection 30420 and the system calls up that specific site, container or component. If input is received from the user selecting a specific site, container or component at any time during the search process, that element is called up by the system 30440.

~~00126~~[00127] Input is received from the user choosing to containerize a search or select a container level in which to search 30100. When input is received from the user choosing to containerize the search, the software moves to the container editor as described in Fig. 5, and then returns the user to the search interface screen. Input is received from the user selecting to search a specific container level or the whole network. The system shows the available levels 30200. Input is received from the user selecting a container level 30300, and entering the text or digital component comprising the search string 30310. The system searches the containers 30400 while simultaneously submitting the search string to the analysis engine 30500. While the system is accessing containers, sites or templates 30700, the analysis engine 30500 inquires of the appropriate database 30600 to access historically successful containers, sites or search templates corresponding to the search request 30700, which is then shown on another portion or option of the search interface, either as available containers or sites 30410 or as search template options 30320. On one portion or option of the search interface screen the corresponding containers or sites are listed and/or previewed for selection 30410. Input is received from the user selecting the container to access 30420. The system accesses that container 30430 and shows it on the screen 30440 for user review.

Input is received from the user selecting an operation, i.e., preview, read, purchase, move, copy, lease, in any composed schedule with operations assigned specific values 30460, and the system obtains the specified result 30470. The selection of the operation including any interaction with any uniquely defined container 100 is recorded 30800 by the container gateway (Fig. 2 A, 200), stored in the gateway storage 205 and made available to the analysis engine (Fig. 9) by the data collection and reporting means (Fig. 8). Reporting and collection occurs on a regular basis according to user determined times or rules. The analysis engine compiles and analyzes selections according to various rules-based systems applicable to the particular container area of residence in cyberspace.

[00127][00128] Input is received from the user selecting the container or site 30410, proceeding as described above, or selecting a search template 30330, and editing it to re-enter the search 30310. All operations on Fig. 6 utilize the communication device 26 whenever necessary or expeditious.

[00128][00129] Referring now to FIG. 7, a flow chart of the search process is shown. Steps in FIG. 7 repeated from FIG. 6 are given the same reference number as in FIG. 6 for convenience and ease of understanding. Fig. 7 commences with “SEARCH TRANSITS GATEWAY 32100”, continuing from Fig. 6, “SYSTEM SEARCHES CONTAINERS 30400”. The submitted search 32100 transits the gateway 200. The gateway 200 interacts with the container registers 32200. The gateways 200 store the information downloaded from the registers 32300, and the container registers are altered 32500. The container registers 120 then interact with the registers 120 of the encapsulated search, which registers, and the values set within, have been constructed and appended to the search through the search interface 32600. Values are exchanged and compared and operations performed under the rules



governing both interacting containers 100, and the rules governing the search container 100 and any gateway 200. The search engine 320, operating under the principles and means of search engines presently existing as described elsewhere, then provides to the search interface 32600 a list of containers 100 meeting the requirements of the search and its appended registers, as well as additional search options 32900. The gateway 200 reports and makes available for collection to the analysis engine 400 the information obtained from the interaction 32400. On a periodic basis defined by the user or a rules-based system, the analysis engine 400 (Fig. 9) stores in databases 900, analyzes and instructs the execution engine 500, and the execution engine 500 executes changes in the system components as defined below (Fig. 10). All operations on Fig. 7 utilize the communication device 26 whenever necessary or expeditious.

[00129][00130] On the remaining figures, shapes referring to other figures, to operations external to the scope of the present figures, or to the subject of the present drawing, are indicated with dashed lines, and are shown only to place the described operations in the context of continuous and continual operations external to the drawing.

[00130][00131] Referring now to FIG. 8, a flow chart of the preferred process for collecting and reporting information on containers is shown. The data reporting 600 and data collection 700 means utilizes subroutines within the analysis engines 400 and gateways 200 to submit and collect register information and sub level analysis to other analysis engines 400 or other gateways 200 of a higher (larger) logical set in a set pattern and frequency defined by the administrator.

[00131][00132] Input is received from the user selecting "data reporting" 70100 from the "edit gateway" drop-down menu. Container levels are displayed 70200. Input is

received from the user selecting container level 70300. A menu of all possible gateways 70320 and analysis engines 70330 residing on gateways on the above defined container class appears, depicted graphically as a tree of analysis engines and gateways at that container level. Input is received from the user selecting "source" from "source or destination." Input is received from the user 70400 selecting a container, containers, or class of container by clicking on the graphically depicted container(s) or container level on a display device. Input is received from the user 70410 selecting "destination" from "source or destination" Input is received from the user 70500 selecting an analysis engine, analysis engines, or class of analysis engine by clicking on the graphically depicted analysis engine(s) or analysis engine level on a display device. A time scheduler is displayed. Input is received from the user 70510 selecting the reporting frequency for the selected gateways to report data to the selected engines. The data from the gateways is thenceforth continuously moved or copied to the analysis engines by the system 10 utilizing the execution engine 500 according to the defined schedule, rules and pattern 70420, 70520.

[00132][00133] Input is received from the user selecting "choose container level" 70300 from the gateway editor drop-down menu. A menu 70320 appears listing the classes of containers on the system within the defined subset and superset scheme of multiple hierarchically nested containers, i.e.; element, document, file, database, warehouse, domain, appears. Input is received from the user selecting the class of containers. A graphic representation of that container level throughout the system appears. Input 70300 is received from the user selecting individual containers or all the containers in that class.

[00133][00134] From the gateway editor drop-down menu input 70100 is received from the user selecting "data collecting" A menu of all possible gateways and

analysis engines residing on gateways on the above defined container class appears, depicted graphically as a tree of analysis engines, and gateways at that container level. Input 70510 is received from the user selecting “source” from “source or destination.” Input is received from the user selecting a container, containers, or class of container by clicking on the graphically depicted container(s) or container level. Input 70510 is received from the user selecting “destination” from “source or destination.” Input 70510 is received from the user selecting an analysis engine, analysis engines, or class of analysis engine by clicking on the graphically depicted analysis engine(s) or analysis engine level. A time scheduler appears. Input 70510 is received from the user selecting the collecting frequency for the selected engines to collect data from the selected gateways. The data from the gateways is thenceforth continuously moved or copied to the analysis engines by the system 10 utilizing the execution engine 500 according to the defined schedule, rules and pattern.

[00134][00135] \_\_\_\_\_ The data collection 700 means, utilizing the communication device 26 and an execution engine 500, comprises one or more subroutines or agents programmed to travel through the network collecting the accumulated data and analyses from selected analysis engines, gateways or selected subset level of analysis engines or gateways (as above) in a pattern and frequency defined by the gateway administrator at a given container level. Input 70510 is received from the user or administrator, defining the collection and reporting of data, thus controlling permission within his gateway, and being subject to permission levels defined by others beyond his gateway.

[00135][00136] \_\_\_\_\_ Input is received from the user or gateway administrator selecting collection or reporting 70100 and the system shows the container levels available 70200. Input is received from the user selecting a container level 70300. Input is received

from the user selecting “gateway” 70400 or “engine” 70500. The system shows gateways 70320 or engines 70330 associated with that level. Input is received from the user editing the reporting parameters associated with a gateway or a class of gateways 70410 or an engine or class of engines 70510. Input is received from the user selecting the collecting frequency for the chosen engines. When input is received from the user choosing to user save the definition, the screen returns to the main container screen, step 70100 to make another selection available. Input is received from the user choosing to repeat the cycle, choosing “destination” to describe the destination analysis engines and the data collecting frequency from those destination analysis engines. The data collection means 700 collects the accumulated gateway information in a pattern and frequency defined by the gateway administrator or user at a given container level.

[00136][00137] The system utilizing the execution engine (see Fig. 10) distributes the new parameters to the gateways 70420 or engines 70520 by the communication device 26. Using the new parameters the gateways report to the analysis engines 70430 after, in some cases, conducting sub-analysis 70440, or using sub-analysis 70440 to submit directly to specified gateways under certain conditions and parameters, and the analysis engines collect from the gateways 70530. The analysis engine uploads, downloads and utilizes information to databases 900 to conducts its analysis.

[00137][00138] The invention includes an analysis engine 400. Through the data reporting 600 means and data collection 700 the analysis engine 400 receives data and sub-analysis from the search interface and the gateways. Data includes, for each gateway 200, the frequency and grade of access, the description of the user accessing, the identity of

the container 100 accessing, the register parameters, and the historically accumulated register data.

~~00138~~~~00139~~ Referring now to FIG. 9, a flow chart of the operation of the analysis engine 400 is shown. Analysis engines 400 may reside at any gateway or anywhere in the system 10. The analysis engine 400, operating under its own programmed sequence, utilizing the communication device 26, works, by means of programmed rules of logical, mathematical, statistical or other analysis upon gateway and register information, in continuous interaction with the search process 410 and the data collection and reporting process 420 to analyze, determine and compile instructions 40100 on container construction 40110 to containerize in an automated process 40115, on container contents 40120 to move, copy or delete containers 40125, on storage schemes 40130 to move or copy containers to new storage 40135, on access routes 40140 to alter gateway pointers to sought information 40145, on search templates 40150 to add, delete or change search phrases and the referenced objects indicated by those search phrases 40155 and on gateway instructions 40160 to alter gateway registers and pointers 40165.

~~00139~~~~00140~~ Thus, analyses might include, but are not limited to, the physical locus of the users accessing, the demographic classification of the users accessing, the access frequency for a given container, the range or curve of time relevance affecting a container, the range or region of space relevance affecting a container 100, the number or number of a specific type of container 100 transiting a gateway 200, the hierarchically graded usage of containers 100 or container contents 01 compared with the demographic of those users accessing the container, the hierarchically graded usage of containers 100 or container contents 01 compared with search phrases entered into the search interface 300, the

hierarchically graded usage of containers 100 or container contents 01 compared with search phrases entered into the search interface 300 compared with the demographic of the users accessing, the number of pertinent containers nested within a given container 100. Once an analysis is accomplished, the result is compared to pre-programmed rules triggering instruction sets (such as moving a container to nest within another container).

~~100140~~~~100141~~ Instructions are then sent to the execution engine 40200, which utilizes the communication device 26 to execute the instructions derived from the analyses. These containerized instructions transit the gateways 40300 and are utilized in the gateway process (Fig. 12)

~~100141~~~~100142~~ Referring now to FIG. 10, a flow chart of the operation of the execution engine is shown. The execution engine 400, operating under its own programmed sequence in response to the instructions from the analysis engine 50100, utilizing the communication device 26, works in continuous process as its containerized execution instructions transit the gateways 50200 to create containers 50210 in an automated containerization process 50215, alter container contents 50230 by moving or copying containers to new containers 50235, to alter storage 50240 by moving or copying containers to new storage 50245, to alter access routes 50250 by altering gateway pointers 50255, to alter search templates 50260 by adding, changing and deleting search phrases and the referenced objects indicated by those search phrases 50265, to alter gateway instructions 50270 by altering gateway registers and pointers 50275. The execution works in a continuous loop with the gateway process 50300, the data collection and reporting process 50400 and the analysis engine process 50300.

{00142}{00143} The invention includes gateways 200. Gateways may be placed and reside anywhere on the network where containers transit. Gateways also reside on any or all containers. The gateway reads and stores the chosen register information from transient containers entering or exiting its logical boundaries. The resident analysis search engine, if any, performs the specified level of analysis. Data and analysis is both held for the collection means according to the pattern and timing specified in the data reporting 600 editor and submitted according to the pattern and timing specified in the data collection means editor 700.

{00143}{00144} The gateways are network-wide, hierarchical, and nestable, and reside with a container encompassing any component, digital code, file, search string, set, database, network, event or process and maintaining a unique lifelong network wide identity and unique in all the universe historical identity, or may be strategically placed at such container transit points to gather and store register information attached to any such container, according to system-defined, system-generated, or user determined rules residing in its registers defining the behavior of those containers and components as they exit and enter one another, or interact with one another or any system process or system component within the logical domain of that container, or after exiting and entering that container, or defining how they interact with that unique gateway.

{00144}{00145} Gateway's registers comprise both system-defined and user-defined registers, alterable by author, duration, location, network-wide history, individual container history and/or interaction with other containers, gateways, networks or media, and evolve according to that gateway's history on a computer network, or according to the network history of events and processes, or according to that information component's

interaction with other information containers, components, system components, network events or processes.

~~100145~~100146 Referring now to FIG. 11, a flow chart of the gateway editor is shown. From the main title bar input is received from the user selecting “containerize” or “gateway level” 20100. When input is received from the user selecting “containerize” the system enters the container editor process 110. When input is received from the user selecting “gateway,” the system shows the gateway levels available 20200. A menu of all possible gateways within the subset and superset scheme of defined multiple hierarchically nested gateways appears. Input is received from the user selecting the gateway level 20300. The system searches the gateways 20500 to locate the available gateway templates 20700 and the available gateways 20600. Input is received from the user selecting the gateway 20610 or gateway level template 20720. The system goes to the gateway 20620 or to the template 20720. A graphic representation of the chosen gateway 20630 or template 20730 appears. Input is received from the user to edit 20640 or create a gateway 20740. Once completed, input may be received from the user selecting “analysis level” from the gateway 200 drop-down menu, to select the level of analysis in a multi-level analysis sequence to be accomplished at the local level by a gateway-resident analysis engine. The user accesses the container editor to containerize (Fig. 5). Input is received from the user selecting the registers by clicking on the graphically depicted register, or from a drop down menu. Input is received from the user setting the registers as described elsewhere in (“container registers”). Input is received from the user selecting or defining the rules governing the interaction of that gateway with transient containers. Input is received from the user selecting or defining the rules governing the interaction of containers existing within the logical domain of the



container 100 to which that gateway is attached. The user publishes the gateway (Fig. 5). Input is received from the user selecting "residence" from the main menu bar. ). Input is received from the user choosing to leave the gateway where it was created, move it to container on another drive, directory, computer, or network. If the user chooses "move," a browse function allows the user to name the new location or browse a list of possible locations. Once input is received from the user choosing the residence of the gateway, the program restores the search interface screen.

[00146][00147] The invention includes a data reporting means editor 610, and a data collection means editor 710, Fig. 2 A, as a menu option under the gateway editor 210.

[00147][00148] The present invention also includes a gateway process.

[00148][00149] Referring now to FIG. 12, a flow chart of the gateway process is shown. A system operation, search process or element container or process container is shown in transit 21100 passing through a gateway 21200. The container, operation or process interacts with the gateway 21300, uploading, downloading and exchanging information with the container, operation or process. The gateway stores container information 21400 and the container registers are altered 21500. The container registers also interact with the search interface 21600. The gateways report the register information or make it available for collection by the data reporting and collection means (Fig. 8) operating on the communication device 26 to provide the information to the analysis engine 21800, which stores 90100, analyzes and instructs the execution engine 21900, which processes and instructions are also stored 90100 by the execution engine upon receipt.

[00149][00150] All operations in Fig. 12 utilize the communication device 26 whenever necessary or expeditious.

[00150][00151] Referring now to FIG. 13 A, a drawing of nested containers 100 prior to the container modification process on a network 201 is shown. (Note: The same container numbering scheme is used in Fig. 13 A, 13 B, 13 C, 13 D and in 2 B.) Information containers 505 and 909, residing within container 908, operating under the rules governing container interaction within that container 908 downloaded to container 505 and 909 from gateway 9081 upon their entrance to container 908, which rules had been downloaded from execution engine 500 acting under the direction of analysis engine 400, and under the rules programmed into their own registers 404120, 909120, compare the specified (by those rules) set of registers 404120, 909120, i.e., time and space, and determine a container 404 encapsulated within 505 would be more appropriately encapsulated within container 909.

[00151][00152] Referring now to FIG. 13 B a drawing of nested containers during a container modification process on a network 201 is shown. Container 404 is moved to reside with container 909. As the container 404 exits container 505, the gateway of container 505, being gateway 5051, operating under the rules governing container interaction with a gateway 5051 upon egress or egress as programmed in the gateway editor 210 and modified by the execution engine 500 executing the instructions of the analysis engine 400, or any greater logical analysis engine 408 providing execution instructions to an execution engine 508 operating in a larger encompassing container 108 entering through that container's gateway 208 or an independent gateway 707, or sub-analysis engine operating at any gateway level, records the register information of container 404. The gateway 5051 reports the transaction to the gateway 9081 of container 908, being the next higher logical container. Gateway 9081 holds in gateway storage 205 the information until collected by one or more data collection processes 700, or reported to one or more data reporting processes

600, serving one or more analysis engines 400 residing independently on the system 10 or an analysis engine at higher logical container 303. The analysis engine 400, comparing reports of user hierarchically graded usage under the operations of the search engine 320 and the search interface 300, on information container 808 after receiving reports from the data reporting means of container 404 being moved to container 909 determines, i.e., that the number of time and space relevant containers residing within container 909 is sufficient to warrant an action, and directs the execution engine 500 to copy container 909, nested within container 908, to a third information container 808. As the copy instruction from execution engine 500 transits the gateway of container 908, the gateway 9081 records the instruction. The copy instruction interacts with the registers 909120 of container 909 regarding the rules governing its copying to another location. Once approved by the governing rules of registers 909120 appended to container 909, container 909 is duplicated. As the duplicate container 909 exits the container 908, the gateway records the register information 909120 of container 909, and the registers 909120 of container 909 are altered by special instructions from gateway 9081 under the rules residing in gateway 9081 regarding ingress and egress and the rules residing in the registers 909120 of container 909 regarding alteration by gateways upon ingress and egress. Passing through independent gateway 707, the register information 909120 is recorded, and awaits data collection or reporting 700, 600. As container 909 enters container 808, the gateway records the register information 909120 of container 909, the registers 909120 of 909 are altered by special instructions from gateway 8081, operating under the rules as described in the paragraph above, and container 909 takes up residence within container 808.

[00152][00153] Referring now to FIG. 13 C, a drawing of nested containers after the container modification process on a network 201 process is shown. Container 909, now also logically residing within container 808, commences to interact with other containers 606 in 808 under the rules governing container interaction within container 808 as received from gateway 8081 upon transiting that gateway, and under the rules of registers 606120, 909120 of the interacting containers 606, 909, operating under the rules as described in the paragraph above. Through data collection and reporting 700, 600, analysis engine is appraised of container's 909 new duplicate residence. I.e., operating under the registers of space relevance, a body of law pertaining to Boston Municipal tax law may be housed in a container holding Massachusetts tax law, but it would be more appropriately located in a container holding Boston tax law, with only a pointer to that location residing in the Massachusetts tax law container. In this example, such an analysis could be accomplished by comparison of zip code information in the space registers, or logical rules-based analysis, with "state" being a larger set than "city". Or, i.e., operating under the registers of time relevance, the curve of time relevance for a concert might follow an ascending curve for the months prior, hit a brief plateau, and then reach a precipitous decline, at which time certain pertinent information only might be moved to an archival container of city events or rock concerts of that year. In this example, once the curve is mapped into a register, that map would cause an increasing frequency of pointers to that container in other containers or gateways, or inclusion of that container in other containers, as the analysis engine compares that curve with increasing user inquiry.

[00153][00154] Referring now to Fig. 13 D, a flowchart of the reconstruction process is shown.

~~100154~~~~100155~~ Information containers 505 and 909, residing within container 908, operating under the rules governing container interaction within that container 908 downloaded 888103 to container 505 and 909 from gateway 9081 upon their entrance to container 908, which rules had been downloaded 888102 from execution engine 500 acting under the direction 888101 of analysis engine 400, and under the rules programmed into their own registers 404120, 909120, compare 888104 the specified (by those rules) set of registers 404120, 909120, i.e., time and space, and determine 888105 a container 404 encapsulated within 505 would be more appropriately encapsulated within container 909.

~~100155~~~~100156~~ Container 404 is moved 888106 to reside with container 909. As the container 404 exits container 505, the gateway of container 505, being gateway 5051, operating under the rules governing container interaction with a gateway 5051 upon egress or egress as programmed in the gateway editor 210 and modified 888108 by the execution engine 500 executing the instructions of the analysis engine 400, or any greater logical analysis engine 408 providing execution instructions 888107 to an execution engine 508 operating in a larger encompassing container 108 entering through that container's gateway 208 or an independent gateway 707, or sub-analysis engine operating at any gateway level, records 888109 the register information of container 404, and alters the register information of container 404. The gateway 5051 reports 888110 the transaction to the gateway 9081 of container 908, being the next higher logical container. Gateway 9081 holds 888111 in gateway storage 205 the information until collected by one or more data collection processes 700, or reported to one or more data reporting processes 600, serving 888112 one or more analysis engines 400 residing independently on the system 10 or an analysis engine at higher logical container 303. The analysis engine 400, comparing 888114 reports of user

hierarchically graded usage on information container 808 under the operations of the search engine 320 and the search interface 300, after receiving 888113 reports from the data reporting means of container 404 being moved to container 909, determines 888115, i.e., that the number of time and space relevant containers residing within container 909 is sufficient to warrant an action, and directs 888115 the execution engine 500 to copy container 909, nested within container 908, to a third information container 808. As the copy instruction from execution engine 500 transits the gateway of container 908, the gateway 9081 records 888116 the instruction. The copy instruction interacts 888117 with the registers 909120 of container 909 regarding the rules governing its copying to another location. Once approved 888118 by the governing rules of registers 909120 appended to container 909, container 909 is duplicated 888118. As the duplicate container 909 exits the container 908, the gateway records 888119 the register information 909120 of container 909, and the registers 909120 of container 909 are altered 888120 by special instructions from gateway 9081 under the rules residing in gateway 9081 regarding ingress and egress and the rules residing in the registers 909120 of container 909 regarding alteration by gateways upon ingress and egress. Passing through independent gateway 707, the register information 909120 is recorded 888121, and awaits 888122 data collection or reporting 700, 600. As container 909 enters container 808, the gateway records 888123 the register information 909120 of container 909, the registers 909120 of 909 are altered 888124 by special instructions from gateway 8081, operating under the rules as described in the paragraph above, and container 909 takes up residence 888125 within container 808.

1001561001571 Container 909, now also logically residing (in addition to its original container residence) within container 808, commences to interact 888126 with other

containers 606 in 808 under the rules governing container interaction within container 808 as received from gateway 8081 upon transiting that gateway, and under the rules of registers 606120, 909120 of the interacting containers 606, 909, operating under the rules as described in the paragraph above. Through data collection and reporting 700, 600, analysis engine is appraised 888127 of container's 909 new duplicate residence.

~~100157~~[00158] Referring now to Fig. 14, the screen interface of the container editor is shown. This interface is a process wherein input is received by the user using the main menu 78 or drop down menu 1419, or using an input device to “drag and drop” or click, causing the system 10 to acquire 1409, edit 1410 or create 1411 a file 1407, container 1408 or digital content 01, to search for 1412, acquire 1413, edit 1414 or create 1415, print 1416, or containerize 1417 a container 100, to select 1402, (or by clicking on register), search 1403, acquire 1404, edit 1405, or create a register 1406 to append or detach registers 120 to those containers, to set register values in those registers 120, to utilize the register editor 125 through 1405 to create new registers, or to 1418 add, detach, acquire a gateway 200 to append or detach to those containers, and utilize the gateway editor 210 through 1418. (See detailed description referring to Fig. 5)

~~100158~~[00159] Referring now to Fig. 15, the screen interface of the gateway editor is shown. This interface is a process wherein input is received by the user using the main menu 1501 or drop down menu 1513, or using an input device to “drag and drop” or click, causing the system 10 to search for 1507, acquire 1508, edit 1509 create 1510, print 1511 or containerize 1512 gateways, and causing the system 10 to establish rules by which an individual gateway governs the transiting 1502, entering 1503, exiting 1504 of containers and

the interaction of containers within its domain 1505, and external of its domain.1506. (See detailed description referring to Fig. 11).

~~100159~~100160 Referring now to Fig. 16, the screen interface of the search interface. This interface is a process wherein input is received by the user using the main menu 1625 or drop down menu 1624, or using an input device to “drag and drop” or click, or by entering text, causing the system 10 to select 1615, search for 1616, acquire 1617, edit 1618 create 1619, print 1620, containerize 1621 (by accessing the container editor 110) or insert 1622 digital search strings into the search box 1623 in order to submit that string to the search engine 320, or causing the system 10 to select 1602, search for 1603, acquire 1604, edit 1605, create 1612, containerize 1613 (by accessing the container editor 110), or insert 1614 search keys (templates that comprise search scope in geographic range, container level, and specific key words or digital strings), or containerized searches (containers 110), into the search box 1623 in order to submit that string to the search engine 320 , or causing the system 10 to set a search range by geographic range 1607, container level 1608, or acquire 1609, edit 1610 or create 1611 a scope template. (templates that comprise search scope in geographic range and, container level.) (See detailed description referring to Fig. 6).

~~100160~~100161 Referring now to Fig. 17, a drawing showing, on an input device or computer screen 24, in any generic (dashed lines) software application program, a drop-down menu link 1403 on a drop down menu 1402 dropping down from a main menu 1401, and a free-floating button link 1404, is shown. When input is received at 1402 or 1403, the system 10 makes available to the user the containerization process or container editor 110. When input is received at drop-down menu link 1405 or a button link 1406, the system 10 makes available to the user the means to enter and interact with this system 10 or this



network 201 in any of their aspects. The interfaces 1403, 1404 show a process wherein input is received causing the system 10 to encapsulate content or access the container editor 110. The link also allows the user to encapsulate the page or file on which he is currently working, without selecting content, and if so desired, without accessing the container editor. The interfaces 1405, 1406 show a process wherein input is received causing the system 10 to access or interact with the system 10 or the network 201.

[00161][00162] The present invention also includes a search engine 320. Once the key word(s), phrase or digital segment is entered into the search interface 300, or an offered selection chosen on the menu, it is utilized by the search engine 320 to locate the desired site or data.

[00162][00163] The search engine employed may be any industry standard search engine such as Verity "Topic", or Personal Library Software, as used in Dow Jones News Retrieval, or Internet search engines such as Webcrawler, Yahoo, Excite, Infoseek, Alexa or any Internet search engine, or any new engines to be developed capable of searching for and locating digital segments, whether text, audio, video or graphic.

[00163][00164] The present invention also includes an analysis engine 400. Utilizing rules-based analysis, the analysis engine determines the class of storage medium upon which containers reside, the subsets and supersets by which and in which containers encompass and reside within one another, the routes of access to those containers, the historically successful search parameters by which those containers are accessed based upon the identity of the user accessing the containers, and the grade of access chosen by the user in accessing that container 100.

[00164][00165] Utilizing a pre-programmed sequence of compilation, and inductive, deductive and derivative analysis, the analysis engine manufactures instructions based upon the analysis of the information submitted by the gateways and the search interface, and submits those instructions to the appropriate execution engine 500 in order to create new information containers, content assemblages, storage schemes, access routes, search templates, and gateway instructions, and others, and to provide informed search options through the search interface to the inquiring user.

[00165][00166] The present invention also includes an engine editor 510, that provides a system administrator with a means of editing the operating principles of that search engine, and search template loading in the search interface 300, a reporting and collection means editor 610, 710, governing data reporting 600 and data collection 700 at the gateways 200 as defined by the gateway editor 210 and the register editor 125, a container editor 110 for creating and modifying containers and appending registers to containers, a register editor 125 for creating and modifying container registers and establishing and adjusting the values therein, container gateways 200 with their own storage 205, information containers 100 for holding information and container registers for holding information about specific containers and their history on the network.

[00166][00167] The present invention also includes an execution engine 300. Based upon instructions received from the analysis engine 400 utilizing the communication device 26, the execution engine 500 provides search phrases to the search interface 300 based upon initially received inquiries, relocates containers including their programs, data and registers to other directories, drives, computers, networks on other classes of storage mediums, i.e., tape drive, optical drive, CD-ROM, deletes, copies, moves containers to nest

within or encompass other containers on other directories, drives, computers, networks to nest within other containers, alters the class of storage medium upon which containers reside, the subsets and supersets by which and in which containers encompass and reside within one another, the routes of access to those containers, and the historically successful search parameters by which those containers are accessed based upon the identity of the user accessing the container and the grade of access chosen by the user in accessing that container.

[00167][00168] The execution engine 400 fulfills the instructions of the analysis search engine 500, to create new information containers, content sub and superset assemblages, storage schemes, access routes, search templates, gateway 200 instructions and other system functions. The execution engine includes an editor 510 that provides a system manager with a means of editing the operating principles of that search engine, governing data reporting, data collection 700, search template loading, gateway instructions, and other functions.

[00168][00169] The present invention also includes flat or relational databases 900, used where, and as required.

[00169][00170] The present invention also includes a communication device 26 supporting all operations on a network wide basis.

[00170][00171] The present invention also includes a search engine 300 to locate the desired site or data. The present invention also includes databases 900, flat or relational, to serve the other components of the system as needed and where needed.

[00171][00172] The present invention also includes editors, by which the user may alter the governing aspects of the system. Editors include, but are not limited to, a

container editor 110, a register editor 125, a gateway editor 210, an engine editor 510, a reporting means editor 610, a search interface 300, and a collection means editor 710.

[00172][00173] The present invention also includes specific screen interfaces for the editors, as described in Fig. 14, Fig. 15. and Fig. 16.

[00173][00174] The present invention also includes a means for this system 10 and network 201 or container editor 110 to be accessed from a menu or button selection within any program, as described in Fig. 17.

[00174][00175] While the present invention has been described with reference to certain preferred embodiments, those skilled in the art will recognize that various modifications may be provided. For example, both analysis engine and execution engine may be duplicated or modified for distribution at various locations and hierarchical positions in the gateway and container system throughout the network and designed to work in concert. Also, the physical computing infrastructure may be mainframe, mini, client server or other with various network and distributed computing designs, including digitally supported or based physical or public media, and the components of the system 10, as described in Fig. 1 may be physically distributed through space. Even the contents of a single container may be logically referenced but be physically distributed through the network and reside at multiple storage locations. The whole system may be hierarchically nested within other systems to the nth degree. Whole systems may also be encapsulated within containers. A single container may also encompass a single physical media, such as a CD-ROM disk, programmed with the container, gateway and register design. Gateways may be strategically placed on containers at ingress and/or egress points or may be placed strategically throughout the system for optimal collection and reporting output and gateway system control. Also, the loop of

gateway data collection and reporting, analysis engine analysis, instruction, and gateway modification, and execution engine operations may be infinitely nested, from the smallest container of two sub-containers to whole networks holding millions of containers and thousands of levels, with analysis itself nested within the multiple levels. Gateways may be established at both logical and physical junctures such as a satellite uplink point. Also, the provision to establish a unique network identity might be designed to include as of yet unknown computer networks as they arise. The analysis and execution engines may operate on a rules-based, fuzzy logic, artificial intelligence, neural net, or other system not yet devised. Other variations upon and modifications to the preferred embodiments are provided for by the present invention, which is limited only by the following claims. Also, the classification scheme of nested containers, while designated by the container creators, may transform, be utilized otherwise, or be wholly discarded according to usage. Also, hardware configurations, such as the use of RAM or hard drives for storage or lasers for communication may assume myriad forms without altering the essential operation of this invention.

WHAT IS CLAIMED IS:

1. A method comprising:  
receiving a search query;  
searching container registers encapsulated and logically defined in a plurality of containers to identify one or more containers responsive to the search query; and  
providing a list characterizing the identified containers.
2. A method as in claim 1, when the received search query comprises a labeled data tree having at least one parent-child relationship.
3. A method as in claim 1, further comprising: providing information identifying containers that have previously been used to respond to one or more processed queries that are substantially similar to the search query.
4. A method as in claim 1, wherein the provided information is stored in one or more search templates.
5. A method as in claim 1, further comprising: providing information identifying substantially similar search phrases, search templates, or labeled data trees that have previously been used to respond to one or more processed queries that are substantially similar to the search query.

6. A method as in claim 5, further comprising:  
receiving a selection of one of the substantially similar search phrases; and  
providing a list of previously identified containers associated with the selected search phrase.
7. A method as in claim 1, wherein the list provides a title of each identified container and a short description of its contents.
8. A method as in claim 1, further comprising: receiving a container search level parameter; and wherein the searching content and container registers only searches within container levels associated with the container search level parameter.
9. A method as in claim 1, further comprising: receiving a container search level parameter; and wherein the list of identified containers only comprises containers associated with the container search level parameter.
10. A method as in claim 1, wherein the searching further comprises: encapsulating the search query into a search container.
11. A method as in claim 10, wherein the searching further comprises:  
receiving, by a gateway, the search container;  
storing, by the gateway, data contained within a register of the search container; and

determining whether any registers of containers accessible via the gateway are associated with the register of the search container.

12. A method as in claim 11, further comprising:

generating a new gateway; and

associating the container with the new gateway.

13. A method as in claim 11, further comprising: periodically aggregating the contents of registers in a plurality of gateways to characterize a plurality of containers coupled thereto.

14. A method as in claim 11, wherein the contents of the registers in each of the plurality of gateways comprise at least one metric chosen from a group comprising: frequency of access of the gateway, grade of access of the gateway, description of users that have accessed the gateway, an identity of containers that have accessed the gateway, parameters associated with the gateway register, and historically accumulated register data.

15. A method as in claim 11, further comprising: monitoring transactions involving one or more gateways or containers.

16. A method as in claim 15, further comprising: generating new containers based on the monitored transactions.



17. A method as in claim 15, wherein the transactions are based on each instance a gateway or container passes through another gateway or container.

18. A method comprising:  
receiving a search query;  
polling a plurality of gateways to identify registers encapsulated therein, the registers relating to one or more containers logically defining data contained therein associated with the search query, wherein the containers are coupled to the gateways; and  
providing a list characterizing the identified containers.

19. A method comprising:  
reporting, by a plurality of gateways, registers or register values encapsulated therein, the registers relating to one or more containers coupled to the gateways logically defining data contained therein;  
updating a centralized index based on the reporting; and  
modifying at least one of a register, a register value, content of a container, stored information in a gateway, and an access route to any of the containers based on the updating.

20. A method as in claim 19, wherein the reporting occurs asynchronously or periodically after at least one interaction of container through the registers in which at least one of the following is altered: a register, a register value, the contents of a container, stored information in a gateway, and an access route to any of the containers.

21. A computer program product, tangibly embodied on computer-readable media, comprising instructions operable to cause a data processing apparatus to:

- receive a search query;
- search content and container registers encapsulated and logically defined in a plurality of containers to identify one or more containers associated with the search query; and
- provide a list characterizing the identified containers..

22. A computer program product, tangibly embodied on computer-readable media, comprising instructions operable to cause a data processing apparatus to:

- receive a search query;
- poll a plurality of gateways to identify registers encapsulated therein, the registers relating to one or more containers logically defining data contained therein associated with the search query, wherein the containers are coupled to the gateways; and
- provide a list characterizing the identified containers.

23. An apparatus comprising:

- means for receiving a search query;
- means for searching content and container registers encapsulated and logically defined in a plurality of containers to identify one or more containers associated with the search query; and
- means for providing a list characterizing the identified containers.

24. An apparatus comprising:

means for receiving a search query;

means for polling a plurality of gateways to identify registers encapsulated therein, the registers relating to one or more containers logically defining data contained therein associated with the search query, wherein the containers are coupled to the gateways; and

means for providing a list characterizing the identified containers.

25. A method comprising:

receiving historical data from a plurality container registers encapsulated and logically defined in a plurality of containers, the historical data associated with interactions of the containers with other containers via those registers; and

modifying at least one of a register, a register value, content of a container, stored information in a gateway associated with a container, or an access route to any of the containers associated with any of a gateway based on the polling.

26. A method as in claim 25, when the obtained historical data comprises a labeled data tree having at least one parent-child relationship.

27. An apparatus comprising:

a processor; and

a memory unit operable to encode instructions operable to cause the processor to:

poll a container registers encapsulated and logically defined in a plurality of containers to obtain historical data associated with interactions of the containers with other containers; and

modify at least one container register or encapsulated content of at least one container based on the obtained historical data.

28. A method comprising:

polling gateways to obtain historical data associated with interactions of a plurality of containers with other containers, the containers encapsulating and logically defining container registers storing, the gateways intercepting data associated with the containers during the interactions; and

modify at least one of the container registers, content encapsulated within at least one of the containers based on the obtained historical data, or information stored in at least one gateway based on the polling.

29. An apparatus comprising:

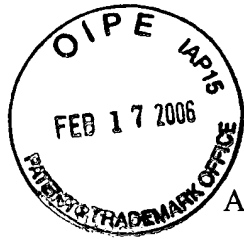
a processor; and

a memory unit operable to encode instructions operable to cause the processor to:

poll gateways to obtain historical data associated with interactions of a plurality of containers with other containers, the containers encapsulating and logically defining container registers storing, the gateways intercepting data associated with the containers during the interactions; and

modify at least one of the container registers, content encapsulated within at least one of the containers, or information stored in at least one gateway based on the obtained historical data.

30. A method comprising:
- receiving a search query;
  - searching container registers encapsulated and logically defined in a plurality of containers to identify one or more search query templates; and
  - providing a list characterizing the identified one or more search query templates to formulate subsequent search queries.



ABSTRACT

A search query may be run against a plurality of container registers encapsulated and logically defined in a plurality of containers to identify one or more container registers responsive to the search query. Thereafter, a list characterizing the identified containers may be provided. Related methods, apparatus, computer program products, and computer systems are also described.

10592228.DOC~~10570207.DOC~~



**CLEAN  
(UNMARKED)  
VERSION**



**SYSTEM AND METHOD FOR CREATING AND MANIPULATING  
INFORMATION CONTAINERS WITH DYNAMIC REGISTERS**

**CROSS-REFERENCE TO RELATED APPLICATIONS**

[0001] The present application is a continuation of U.S. Patent Application No. 09/284,113, entitled System And Method For Creating And Manipulating Information Containers With Dynamic Registers, filed on April 7, 1999, which is incorporated herein in its entirety, and claims the benefit of PCT/US99/01988 filed January 28, 1999 and of U.S. Patent Application No. 60/073,209, filed January 30, 1998.

**BACKGROUND OF THE INVENTION**

[0002] 1. Field of the Invention

The present invention relates generally to computer systems in a multi-user mainframe or mini computer system, a client server network, or in local, wide area or public networks, and in particular, to computer networks for creating and manipulating information containers with dynamic interactive registers in a computer, media or publishing network, in order to manufacture information on, upgrade the utility of, and develop intelligence in, a computer network by offering the means to create and manipulate information containers with dynamic registers.

[0003] 2. Description of the Related Art

In the present day, querying and usage of information resources on a computer network is accomplished by individuals directing a search effort by submitting key words or phrases to be compared to those key words or phrases contained in the content or description of that information resource, with indices and contents residing in a fixed location unchanging except by human input. Similarly, the class of storage medium upon which information resides, its class and subclass organizational structures, and its routes of access all remain fundamentally unaltered by ongoing user queries and usage. Only the direct and intended



intervention of the owner of the information content or computer hosting site changes these parameters, normally accomplished manually by programmers or systems operators at their own discretion or the discretion of the site owner.

[0004] There exists currently in the art a limited means of interfacing a computer user with the information available on computer networks such as the world wide web. Primarily, these means are search engines. Search engines query thousands or tens of thousands of index pages per second to suggest the location of information while the user waits. While factual information can be accessed, the more complex, particular or subtle the inquiry, the more branches and sub-branches need to be explored in a time consuming fashion in order to have any chance of success. Further, there are no such automatic devices that reconstruct the information into more useful groupings or makes it more accessible according to factors attached to the content by the content creator such as the space or time relevancy of its content, or factors attached to the content by the system's compilation and analysis of the accumulated biography of that specific content's readership.

[0005] The utility of wide area and public computer networks is thus greatly limited by the static information model and infrastructure upon which those networks operate.

[0006] One problem is that on a wide area or public network, specific content such as a document remains inert, except by the direct intervention of users, and is modified neither by patterns or history of usage on the network, or the existence of other content on the network.

[0007] Another problem is that content does not reside in an information infrastructure conducive to reconstruction by expert rule-based, fuzzy logic, or artificial

intelligence based systems. Neither the intelligence of other information users nor the expert intelligence of an observant network computer system can be utilized in constructing, or reconstructing information resources. Where content resides in a fixed location and structure, “information” becomes something defined by the mind of the information provider rather than the mind of the information user, where the actual construction and utility of information exists. Information remains, like raw ore, in an unrefined state.

**[0008]** Another problem is that the class of storage medium upon which data resides cannot be system or user managed and altered according to the actual recorded and analyzed hierarchically graded usage of any given information resource residing on that storage medium except by statistical analysis of universal, undefined “hits” or visits to that page or site.

**[0009]** Another problem is that information resource groupings remain fixed on the given storage medium location according to the original installation by the resource author, not altered according to the actual recorded and analyzed hierarchically graded usage of that given information resource. Content itself remains inert, with no possibility of evolution.

**[0010]** A further problem with the prior art is that neither the search templates generated by those more knowledgeable in a given field of inquiry, nor the search strategies historically determined to be successful, or system-constructed according to analyses of search strategies historically determined to be successful, are available to inquiring users. A search template is here defined as one or more text phrases, graphics, video or audio bits, alone or in any defined outline or relational format designed to accomplish an inquiry. Internet or wide area network search may return dozens of briefs to a keyword or key phrase

inquiry sometimes requiring the time-consuming examination of multiple information resources or locations, with no historical relation to the success of any given search strategy.

**[0011]** A further problem is that there is limited means to add to, subtract from, or alter the information content of documents, databases, or sites without communicating with the owners or operators of those information resources, e.g., contacting, obtaining permission, negotiating and manually altering, adding or subtracting content. Additionally, once so altered, there is not a means to derive a proportionate value, and thereby a proportionate royalty as the information is used.

**[0012]** A final problem is that the physical residence of a body of data or its cyberspace location may not serve its largest body of users in the most expedient manner of access. Neither the expert intelligence of other information users nor the expert intelligence of an observant computer system is presently utilized by inherent network intelligence to analyze, re-design and construct access routes to information medium except by statistical analysis of universal, undefined “hits” or visits to that page or site.

**[0013]** Therefore, there is a need for a system and methods for creating and manipulating information containers with dynamic interactive registers defining more comprehensive information about contained content in a computer, media or publishing network, in order to manufacture information on, upgrade the utility of, and develop intelligence in, a computer network by providing a searching user the means to utilize the searches of other users or the historically determined and compiled searches of the system, a means to containerize information with multiple registers governing the interaction of that container, a means to re-classify the storage medium and location of information resources

resident on the network, a means to allow the reconstruction of content into more useful formations, and a means to reconstruct the access routes to that information.

### **SUMMARY OF THE INVENTION**

**[0014]** The present invention is a system and methods for manufacturing information on, upgrading the utility of, and developing intelligence in, a computer or digital network, local, wide area, public, corporate, or digital-based, supported, or enhanced physical media form or public or published media, or other by offering the means to create and manipulate information containers with dynamic registers.

**[0015]** The system of the present invention comprises an input device, an output device, a processor, a memory unit, a data storage device, and a means of communicating with other computers, network of computers, or digital-based, supported or enhanced physical media forms or public or published media. These components are preferably coupled by a bus and configured for multi-media presentation, but may also be distributed throughout a network according to the requirements of highest and best use.

**[0016]** The memory unit advantageously includes an information container made interactive with dynamic registers, a container editor, a search interface, a search engine, a search engine editor, system-wide hierarchical container gateways interacting with dynamic container registers, a gateway editor, a register editor, a data collection means with editor, a data reporting means with editor, an analysis engine with editor, an executing engine with editor, databases, and a means of communicating with other computers as above. These

components may reside in a distributed fashion in any configuration on multiple computer systems or networks.

**[0017]** The present invention advantageously provides a container editor for creating containers, containerizing storing information in containers and defining and altering container registers. A container is an interactive nestable logical domain configurable as both subset and superset, including a minimum set of attributes coded into dynamic interactive evolving registers, containing any information component, digital code, file, search string, set, database, network, event or process, and maintaining a unique network-wide lifelong identity.

**[0018]** The container editor allows the authoring user to create containers and encapsulate any information component in a container with registers, establishing a unique network lifelong identity, characteristics, and parameters and rules of interaction. The authoring user defines and sets the register with a starting counter and/or mathematical description by utilizing menus and simple graphing tools or other tools appropriate to that particular register. The registers determine the interaction of that container with other containers, system components, system gateways, events and processes on the computer network.

**[0019]** Containers and registers, upon creation, may be universal or class-specific. The editor provides the means to create system-defined registers as well as the means to create other registers. The editor enables the register values to be set by the user or by the system, in which case the register value may be fixed or alterable by the user upon creation. Register values are evolving or non-evolving for the duration of the life of the container on

the system. Evolving registers may change through time, space, interaction, system history and other means.

[0020] System-defined registers comprise: (1) an historical container register, logging the history of the interaction of that container with other containers, events and processes on the network, (2) an historical system register, logging the history of pertinent critical and processes on the network, (3) a point register accumulating points based upon a hierarchically rated history of usage, (4) an identity register maintaining a unique network wide identification and access location for a given container, (5) a brokerage register maintaining a record of ownership percentage and economic values, and others.

[0021] The present invention also includes user-defined registers. User defined registers may be created wholly by the user and assigned a starting value, or simply assigned value by the user when that register is pre-existent in the system or acquired from another user, and then appended to any information container, or detached from any container.

[0022] Exemplary user-defined registers comprise (1) a report register, setting trigger levels for report sequences, content determination and delivery target, (2) a triple time register, consisting of a range, map, graph, list, curve or other representation designating time relevance, actively, assigning the time characteristics by which that container will act upon another container or process, passively, assigning the time characteristics by which that container be acted upon by another container or process, and neutrally, assigning the time characteristics by which that container will interact with another container or process, (3) a triple space register, consisting of a range, map, graph, list, curve or other representation designating the domain and determinants of space relevance, actively, assigning the space characteristics by which that content will act upon another container or process, passively,

assigning the space, characteristics by which that content will be acted upon by another container or process, and neutrally, assigning the space characteristics by which that container will interact with another container or process, (4) a domain of influence register, determining the set, class and range of containers upon which that container will act, (5) a domain of receptivity register, determining the set, class and range of containers allowed to act upon that container, (6) a domain of neutrality register, determining the set, class and range of containers with which that container will interact, (7) a domain of containment register, determining the set, class and range of containers which that container may logically encompass, (8) a domain of inclusion register, determining the set, class and range of containers by which that container might be encompassed, (9) an ownership register, recording the original ownership of that containers, (10) a proportionate ownership register, determining the proportionate ownership of that containers, (11) a creator profile register, describing the creator or creators of that container, (12) an ownership address register, maintaining the address of the creator or creators of that container, (13) a value register, assigning a monetary or credit value to that container, and (14) other registers created by users or the system.

**[0023]** Containers are nestable and configurable as both subset and superset and may be designated hierarchically according to inclusive range, such as image component, image, image file, image collection, image database, or if text, text fragment, sentence, paragraph, page, document, document collection, document, database, document library, or any arrangement wherein containers are defined as increasingly inclusive sets of sets of digital components.

**[0024]** The present invention also includes, structurally integrated into each container, or strategically placed within a network at container transit points, unique gateways, nestable in a hierarchical or set and class network scheme. Gateways gather and store container register information according to system-defined, system-generated, or user determined rules as containers exit and enter one another, governing how containers system processes or system components interact within the domain of that container, or after exiting and entering that container, and governing how containers, system components and system processes interact with that unique gateway, including how data collection and reporting is managed at that gateway. The gateways record the register information of internally nested sub and superset containers, transient containers and search templates, including the grade of access requested, and, acting as an agent of an analysis engine and execution engine, govern the traffic and interaction of those containers and searches with the information resource of which they are the gateway and other gateways. The gateways' record of internally nested and transient container registers, and its own interaction with those containers, is made available, according to a rules-based determination, to the process of the analysis engine by the data collection and/or data reporting means.

**[0025]** The present invention also includes a means of data storage at any given gateway.

**[0026]** The present invention also includes a data collection means, residing anywhere on the network, or located at one or more hierarchical levels of nestable container gateways for gathering information from other gateways and analysis engines according to system, system-generated or user determined rules. The data collection means manages the gathering of data regarding network-wide user choices, usage and information about



information, by collecting it from container and gateway registers as those containers and gateways pass through one another. Such statistics as frequency, pattern, and range of time, space and logical class is collected as directed by the analysis engine, and made that data available to the analysis engine by advancing it directly to the analysis engine, or incrementally, to the next greater hierarchically inclusive collection level. The rules of data collection may be manually set or altered by the system manager, or set by the system and altered by the system in its evolutionary capacity.

**[0027]** The present invention also includes a data reporting means, located at one or more hierarchical levels of nestable container gateways for submitting information to other gateways and analysis engines according to system, system-generated or user determined rules. The data reporting means manages the sending of data from the registers, gateways and search templates in a frequency, pattern, and range of time, space and logical class as directed by the analysis engine, and makes that data available to the analysis engine by advancing it directly to the analysis engine, or incrementally to the next greater hierarchically inclusive reporting level. The rules of data collection may be manually set or altered by the system manager, or set by the system and altered by the system in its evolutionary capacity. The data reporting means may be established to work in concert, in redundancy, or in contiguous or interwoven threads of hierarchically nested containers.

**[0028]** The present invention also includes an analysis engine that receives, reports and collects information regarding the interaction of user searches with gateways and container registers, as well as container registers with other container registers, and container registers with gateways. The analysis engine analyzes the information submitted by the gateways and instructs the execution engine to create new information containers, content

assemblages, storage schemes, access routes, search templates, and gateway instructions. The analysis engine includes an editor that provides a system manager with a means of editing the operating principles of that engine, governing data reporting, data collection, search template loading, gateway instructions, and other.

[0029] The present invention also includes an execution engine, fulfilling the instructions of the analysis engine, to create new information containers, content sun and superset assemblages, storage schemes, access routes, search templates, and gateway instructions. The execution engine includes an editor that provides a system manager with a means of editing the operating principles of that engine, governing data reporting, data collection, search template loading, gateway instructions, and other.

[0030] The present invention also includes a search interface or browser. The search interface provides a means for a searching user to submit, record and access search streams or phrases generated historically by himself, other users, or the system. Search streams or phrases of other users are those that have been historically determined by the system to have the highest probability of utility to the searching user. Search streams or phrases generated by the system are those that have been constructed by the system through the analysis engine based upon the same criteria.

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

[0031] FIG. 1 is a block diagram of a first and preferred embodiment of a system constructed according to the present invention.

[0032] FIG. 2 A is block diagram of a preferred embodiment of the memory unit.

[0033] FIG. 2 B is an exemplary embodiment of a computer network showing computer servers, personal computers, workstations, Internet, Wide Area Networks, Intranets in relationship with containers and gateways.

[0034] FIG. 2B1 is an exemplary embodiment of a computer network showing computer servers, personal computers, workstations, Internet, Wide Area Networks, Intranets in relationship with containers and gateways and exemplary locations of gateway storage in proximity to one or more of the various sites.

[0035] FIGS. 2C through 2H are exemplary embodiments in block diagram form of computer network components showing a possible placement of nested containers, computer servers, gateways, and the software components named in Fig. 2 A on a network.

[0036] FIG. 3A is a graphical representation for one embodiment of a container having a plurality of containers nested within that container.

[0037] Fig. 3B is a graphical representation for a second embodiment of a container having a plurality of containers nested within that container.

[0038] FIG. 3C is a drawing showing elements that might be logically encapsulated by a container. FIG. 4 is a drawing of an information container showing a gateway and registers logically encapsulating containerized elements.

[0039] FIG. 5 is a flowchart showing a preferred method for the containerization process and container editor operating on the communication device.

[0040] FIG. 6 is a flowchart showing a preferred method for searching for containers within a node.

[0041] FIG. 7 is a flowchart further showing a preferred method for searching for containers over one or more gateways.

[0042] FIG. 8 is a flowchart showing a method for performing the data collection and reporting on containers.

[0043] FIG. 9 is a flowchart showing the operation of the analysis engine.

[0044] FIG. 10 is a flowchart showing the operation of the execution engine.

[0045] FIG. 11 is a flowchart showing the operation of the gateway editor.

[0046] FIG. 12 is a flowchart showing the operation of the gateway process.

[0047] FIG. 13A is a drawing showing an example of nested containers, gateways, registers, analysis engines and an execution engine prior to container reconstruction as depicted in 13 B, 13 C and 13 D.

[0048] FIG. 13B is a drawing showing the reconstructed nested containers of Figure 13A.

[0049] FIG. 13C is a drawing showing further reconstruction of nested containers, with a container relocated to reside within another container.

[0050] FIG. 13D is a drawing showing a flowchart of the reconstruction process

[0051] FIG. 14 is a drawing showing the screen interface of the container editor.

[0052] FIG. 15 is a drawing showing the screen interface of the gateway editor.

[0053] FIG. 16 is a drawing showing the screen interface of the search interface.

[0054] FIG. 17 is a drawing of a generic application program showing a drop-down menu link, and a button link to the containerization process or container editor.

## **DESCRIPTION OF THE PREFERRED EMBODIMENT**

[0055] THE SYSTEM

[0056] Referring now to FIG. 1, a preferred embodiment of a system 10 for creating and manipulating information containers with dynamic interactive registers in a computer, media, or publishing network 201 in order to manufacture information on, upgrade the utility of, and develop intelligence in that network 201, is shown. The system 10 preferably comprises an input device 24, an output device 16, a processor 18, a memory unit 22, a data storage device 20, and a communication device 26 operating on a network 201. The input device 24, an output device 16, a processor 18, a memory unit 22, a data storage device 20, are preferably coupled together by a bus 12 in a von Neumann architecture. Those skilled in the art will realize that these components 24, 16, 18, 22, 20, and 26 may be coupled together according to various other computer architectures including any physical distribution of components linked together by the communication device 26 without departing from the spirit or scope of the present invention, and may be infinitely nested or chained, both as computer systems within a network 202, and as networks within networks 201.

[0057] The output device 16 preferably comprises a computer monitor for displaying high-resolution graphics and speakers for outputting high fidelity audio signals. The output device 16 is used to display various user interfaces 110, 125, 210, 300, 510, 610, 710, as will be described below, for searching for and containerizing information, and editing the container gateways, containers, container registers, the data reporting means and the data collection means, and the search, analysis and execution engines. The author uses the input device 24 to manipulate icons, text, charts or graphs, or to select objects or text, in the process of packaging, searching or editing in a conventional manner such as in the Macintosh of Windows operating systems.

**[0058]** The processor 18 preferably executes programmed instruction steps, generates commands, stores data and analyzes data configurations according to programmed instruction steps that are stored in the memory unit 22 and in the data storage device 20. The processor 22 is preferably a microprocessor such as the Motorola 680(x)0, the Intel 80(x)86 or Pentium, Pentium II, and successors, or processors made by AMD, or Cyrix CPU of the any class.

**[0059]** The memory unit 22 is preferably a predetermined amount of dynamic random access memory, a read-only memory, or both. The memory unit 22 stores data, operating systems, and programmed instructions steps, and manages the operations of all hardware and software components in the system 10 and on the network 201, utilizing the communication device 26 whenever necessary or expeditious to link multiple computer systems 202 within the network 201.

**[0060]** The data storage device 20 is preferably a disk storage device for storing data and programmed instruction steps. In the exemplary embodiment, the data storage device 20 is a hard disk drive. Historical recordings of network usage are stored on distributed and centralized data storage devices 20.

**[0061]** The preferred embodiment of the input device 24 comprises a keyboard, microphone, and mouse type controller. Data and commands to the system 10 are input through the input device 24.

**[0062]** The present invention also includes a communication device 26. The communication device 26 underlies and sustains the operations of, referring now also to Fig 2 the analysis 400 and execution 500 engines, the data reporting 600 and collection 700 means, the container editor 110, the search interface 300, and the search engine 320, providing the

means to search, access, move, copy, utilize or otherwise perform operations with and on data. The communication device 26 utilizes one or more of the following technologies: modem, infrared, microwave, laser, photons, electrons, wave phenomena, cellular carrier, satellite, laser, router hub, direct cabling, physical transport, radio, broadcast or cable TV or other to communicate with other computers, digital-supported television, computer networks, or digital-based or supported public or published media, or physical media forms, on any a local, wide area, public, or any computer-based computer supported, or computer interfaced network, including but not limited to the Internet. It also allows for the functioning and distribution of any container 100 or container component herein described to reside anywhere on any computer system in any configuration on that local, wide area, public, or corporate computer-based or computer related network, or digital-based or supported media form.

[0063] Referring now to Figure 2 A, a preferred embodiment of the memory unit 22 is shown. The memory unit includes: an interactive information container 100, a container editor 110, container registers 120, a container register editor 125, system-wide hierarchical container gateways 200, gateway storage 205, gateway editors 210, engine editors 510, a search interface 300, search engine 320, analysis engine 400, execution engine 500, a data reporting module, 600, a data reporting editor 610, a data collection module 700, a data collection editor 710, screen interfaces (GUI's) 936, menu or access buttons from generic computer programs 937, and databases 900, all residing in memory optimized between a data storage means 20 such as magnetic, optical, laser, or other fixed storage, and a memory means 22 such as RAM. The memory unit 22 functions by operating on communications network 12 with a communication device 26 on multiple computer systems 202 within the

network 201. These components will be described first briefly in the following paragraphs, then in more detail with reference to Figures 3 A through 17.

**[0064]** Those skilled in the art will realize that these components might also be stored in contiguous blocks of memory, and that software components or portions thereof may reside in the memory unit 22 or the data storage means 20.

**[0065]** The present invention includes information containers 100 as noted above. The information container 100 is a logically defined data enclosure which encapsulates any element or digital segment (text, graphic, photograph, audio, video, or other), or set of digital segments, or referring now to FIG. 3 C, any system component or process, or other containers or sets of containers. A container 100 at minimum includes in its construction a logically encapsulated portion of cyberspace, a register and a gateway. A container 100 at minimum encapsulates a single digital bit, a single natural number or the logical description of another container, and at maximum all defined cyberspace, existing, growing and to be discovered, including but not limited to all containers, defined and to be defined in cyberspace. A container 100 contains the code to enable it to interact with the components enumerated in 2 A, and to reconstruct itself internally and manage itself on the network 201.

**[0066]** The container 100 also includes container registers 120. Container registers 120 are interactive dynamic values appended to the logical enclosure of an information container 100, and serve to govern the interaction of that container 100 with other containers 100, container gateways 200 and the system 10, and to record the historical interaction of that container 100 on the system 10. Container registers 120 may be values alone or contain code to establish certain parameters in interaction with other containers 100 or gateways 200.



[0067] The present invention also includes container gateways 200. Container gateways 200 are logically defined gateways residing both on containers 100 and independently in the system 10. Gateways 200 govern the interactions of containers 100 within their domain, and alter the registers 120 of transiting containers 100 upon ingress and egress.

[0068] The present invention also includes container gateway storage 205 to hold the data collected from registers 120 of transient containers 100 in order to make it available to the data collection means 700 and the data reporting means 600, and to store the rules governing the operations of its particular gateway 200, governing transiting containers upon ingress and egress, and governing the interactive behavior of containers 100 within the container 100 to which that gateway 200 is attached. Gateway storage 205 may be located on gateways 200 themselves, containers 100 or anywhere on the network 202, 201, including but not limited to Internet, Intranet, LAN, WAN, according to best analysis and use.

[0069] The memory unit 22 also includes an execution engine 500 to perform the functions on the system 10 as directed by the analysis engine after its analysis of data from the data reporting means 600, the data collection means 700, and the search interface 300.

[0070] The memory unit 22 also includes a search interface 300, by which the user enters, selects or edits search phrases or digital strings to be used by the search engine 320 to locate containers 100.

[0071] The memory unit 22 also includes an analysis engine 400 which performs rules based or other analysis upon the data collected from the search interface 300 and the data collection 700 and data reporting 600 means.

[0072] The memory unit 22 also includes a data reporting means 600, by which means the information collected by gateways 200 from transient containers 100 is sent to the analysis engine 400.

[0073] The memory unit 22 also includes a data collection means 700, by which means the analysis engine 400 gathers the information collected by gateways 200 from transient containers 100.

[0074] The memory unit 22 also includes a container editor 110 for creating, selecting, acquiring, modifying and appending registers 120 and gateways 200 to containers 100, for creating, selecting, acquiring, and modifying containers, and for selecting content 01 to encapsulate.

[0075] The memory unit 22 also includes a register editor 125, for creating, selecting, acquiring and modifying container registers 120 and establishing and adjusting the values therein.

[0076] The memory unit 22 also includes a gateway editor 210, by which means the user determines the rules governing the interaction of a given gateway 210 with the registers 120 of transient containers 100, governing transiting containers upon ingress and egress, and governing the interactive behavior of containers within the container to which that gateway is attached.

[0077] The memory unit 22 also includes databases 900, by which means the analysis engine 400, the execution engine 500, the gateways 100, the editors 110, 125, 210, 510, 610, 710, and the search interface 300, store information for later use.

**[0078]** The memory unit 22 present invention also includes a search engine 320 by which means the user is able to locate containers 100 and, referring now to Fig. 4, containerized elements 01.

**[0079]** The memory unit 22 present invention also includes an engine editor 510, by which means the user establishes the rules and operating procedures for the analysis engine 400 and the execution engine 500.

**[0080]** The memory unit 22 present invention also includes a reporting means editor 610, by which means the user establishes the rules and schedule under which the information collected by gateways 200 from transient containers 100 will be sent to the analysis engine 400.

**[0081]** The memory unit 22 present invention also includes a collection means editor 710, by which means the user establishes the rules and schedule under which the analysis engine 400 will gathers the information collected by gateways 200 from transient containers 100.

**[0082]** The memory unit 22 present invention also includes screen interfaces (GUI's) 936, specifically designed to simplify and enhance the operations of the container editor 110, the gateway editor 210, and the search interface 300.

**[0083]** The present invention also includes a menu or button access 937, by which a user utilizing any generic computer program may access the system 10 or the container editor 110 from a menu selection(s) or button(s) within that program.

**[0084]** The present invention also includes a computer, media or publishing network 201, comprising computers, digital devices and digital media 202 and a communication device 26, within which the components enumerated in Fig. 2 A interact,

compiling, analyzing, and altering containers 100 and the network 201 according to information gathered from container registers 120.

**[0085]** The memory unit 22 also includes one or more computers 202, by which means the components of Fig 1 sustain the operations described in Fig. 2 A.

**[0086]** The memory unit 22 also includes flat or relational databases 900, used where, and as required. Databases are used to store search phrases, search templates, system history for the analysis engine and execution engine, container levels and container, sites and digital elements, or any and all storage required to operate the system.

**[0087]** Referring now to FIG. 2 B, a drawing of a computer network 201 as a system 10, showing a possible placement of nested containers 100, computer servers, gateways 200, on the sites described below. (Note: Fig. 2 B utilizes in parts the same numbering scheme as Fig. 13 A, 13 B, 13 C, 13 D and as Fig. 2 A.) In FIG. 2 B various exemplary sites are shown, any or all of which might interact dynamically within the system. Site 1 shows a single workstation with a container and gateway connected to an Intranet. (Individual containers may be a floppy or CD-Rom to be downloaded or inserted.) Site 2 shows a server with a gateway in relationship to various containers.. Site 3 shows an Internet web page with a container residing on it. Site 4 shows a personal computer with containers and a gateway connected to the Internet. Site 5 shows a configuration of multiple servers and containers on a Wide Area Network.. Site 6 shows a workstations with a gateway and containers within a container connected to a Wide Area Network. Site 7 shows an independent gateway, capable of acting as a data collection and data reporting site as it gathers data from the registers of transiting containers, and as an agent of the execution engine as it alters the registers of transient containers. A container 100 contains the code to

enable it to interact with the components enumerated in 2A, and to reconstruct itself internally and manage itself on the network 201. The code resides in and with the container in its registers and gateway definitions and controls. Additional system code resides in all sites to manage the individual and collective operation and oversight of the components enumerated in 2A, with the specific components distributed amongst the sites according to the requirements of optimization.

[0088] Referring now to Fig. 2 B 1 various exemplary sites are shown as described above in Fig. 2 B, with the addition of possible location of one or more gateway storage 205 locations.

[0089] Referring now to Figures 2 C through 2 H, various exemplary sites with one or more of the logical components of the system 10 in relationship are shown. Site 1 comprises an interactive information container 100, a container editor 110, container registers 120, a container register editor 125, system-wide hierarchical container gateways 200, gateway storage 205, gateway editors 210, engine editors 510, a search interface 300, search engine 320, analysis engine 400, execution engine 500, a data reporting means 600, a data reporting means editor 610, a data collection means 700, a data collection means editor 710, and databases 900, all residing on data storage means 20, utilizing the memory unit to function 22, operating on communications network 12 with a communication device 26.

[0090] Site 2 comprises an interactive information container 100, a container editor 110, container registers 120, a container register editor 125, system-wide hierarchical container gateways 200, gateway storage 205, gateway editors 210, engine editors 510, search engine 320, analysis engine 400, execution engine 500, a data reporting means 600,

a data reporting means editor 610, a data collection means 700, a data collection means editor 710, and databases 900, all residing on data storage means 20, utilizing the memory unit to function 22, operating on communications network 12 with a communication device 26.

**[0091]** Site 3 comprises an interactive information container 100, a container editor 110, container registers 120, a container register editor 125, hierarchical container gateways 200, gateway storage 205, gateway editors 210, and databases 900, all residing on data storage means 20, utilizing the memory unit to function 22, operating on communications network 12 with a communication device 26.

**[0092]** Site 4 comprises an interactive information container 100, a container editor 110, container registers 120, a container register editor 125, hierarchical container gateways 200, gateway storage 205, gateway editors 210, a search interface 300, and databases 900, all residing on data storage means 20, utilizing the memory unit to function 22, operating on communications network 12 with a communication device 26.

**[0093]** Site 5 comprises an interactive information container 100, container registers 120, a container register editor 125, hierarchical container gateways 200, gateway storage 205, and databases 900, all residing on data storage means 20, accessed and utilized by non-resident memory unit 22, operating on communications network 12 with a communication device 26.

**[0094]** Site 6 includes an independent analysis engine 400, execution engine 500, data collection means 700, and data reporting means 600 gateway editors 210, engine editors 510, a data reporting means editor 610, a data collection means 700, a data collection means editor 710, and databases 900, all residing on data storage means 20, utilizing the memory

unit to function 22, operating on communications network 12 with a communication device 26.

[0095] Referring now to FIG. 3 A and FIG. 3 B, a block diagram of several nested information containers is shown, including examples of elements, e.g., code 1100, text 1200, audio 1300, video 1400, photograph 1500, graphic images 1600, and examples of possible container level classifications in increasing size, e.g., element 10900000, document 10800000, database 10700000, warehouse 10600000, domain 10500000, and continuing increasingly larger on Fig 3 (B), subject 10400000, field 10300000, master field 10200000, species 10100000. Containers may be infinitely nested and assigned any class, super class or sub class scheme and description by the creator of the container to govern nesting within that container. In addition to digital elements, containers may also include system process and components, including containerization itself.

[0096] Referring now to FIG. 3 C, a block diagram of an information container system is shown, listing, without any relationship indicated, some of the possible system components and processes, or sets thereof, that may be encapsulated as elements 01 in an information container 100. An information container 100 may include one or more of the following: any unique, container 100, gateway 200, output device 16, input device 24, output device process 160, input device process 240, data storage device 20, data storage device process 2000, processor 18, bus 12, content 01, search process 02, interface 04, memory unit 22, communication device 26, search interface 300, search process 98, network 201, class of device, process or content 999, class of process at any unique class of device 990, process at any unique device 99, editor 110, 125, 210, 510, 610, 710, engine 320, 400, 500, containerization process 1098, or process 08.

**[0097]** Any container may include (n) other containers, to infinity. The use of value evolving container registers 120 in conjunction with gateways 200, data reporting modules 600, data collection modules 700, the analysis engine 400, and the execution engine 500 provides the information container 100 with extensive knowledge of the use, operation of its internal contents, prior to, during and after those contents' residence within that container 100, and extensive knowledge of the use, operation and contents of the system 10 external to itself, and allows the container 100 to establish and evolve its own identity and course of interaction on the system 10. Further, containers 100, as logical enclosures, can exist and operate independent of their digital contents, whether encapsulating audio, video, text, graphic, or other.

**[0098]** Referring now to FIG. 4, a block diagram of an information container 100 is shown. The information container 100 is a logically defined data enclosure which encapsulates any element, digital segment (text, graphic, photograph, audio, video, or other), set of digital segments as described above with reference to FIG. 3 (C), any system component or process, or other containers or sets of containers. The container 100 comprises the containerized elements 01, registers 120 and a gateway 200.

**[0099]** Registers 120 appended to an information container 110 are unique in that they operate independently of the encapsulated contents, providing rules of interaction, history of interaction, identity and interactive life to that container 100 through the duration of its existence on a network 201, without requiring reference to, or interaction with, its specific contents. They enable a container 100 to establish an identity independent of its contents. Additionally, registers 120 are unique in that their internal values evolve through interaction with other containers 100, gateways 200, the analysis engine 400, the execution



engine 500, and the choices made by the users in the search interface 300, the container editor 110, the register editor 125, the gateway editor 210, the engine editor 510. Registers 120 are also unique in that they can interact with any register of a similar definition on any container 100 residing on the network 201, independent of that container's contents. Registers 120, once constructed, may be copied and appended to other containers 100 with their internal values reset, to form new containers. Register values, when collected at gateways 200 and made available to the analysis engine 400 through the data collection means 700 and the data reporting means 600, provide an entirely new layer of network observation and analysis and operational control through the execution engine 500. Registers 120 accomplish not only a real time information about information system, but also a real time information about information usage on a network. Further, because the user base of a network determines usage, the system 10, in gathering information about information usage, is observing the choices of the human mind. When these choices are submitted to the analysis of a rules-based or other analysis engine 400, the system 10 becomes capable of becoming progressively more responsive to the need of the user base, in effect, learning to become more useful by utilizing the execution engine 500 to create system-wide changes by altering the rules of gateway 200 interaction and thereby altering the registers 120 of transient containers 100 and establishing a complete evolutionary cycle of enhanced utility.

**[00100]** Further, in establishing the pre-defined registers as described in the following four paragraphs, the following unique aspects of information about information are utilized for the first time: 1) the dynamic governance of information according to its utility through time, in active, passive and neutral aspects, as explained below; 2) the dynamic governance of information according to its utility through space in active, passive and neutral

aspects, as explained below; 3) the dynamic governance of information according to its ownership, as explained below; 4) the dynamic governance of information according to its unique history of interaction as an identity on a network, as explained below; 5) the dynamic governance of information according to the history of the system on which it exists, as explained below; 6) the dynamic governance of information according to established rules of interaction, in active, passive and neutral aspects, as explained below; 7) the dynamic governance of information according to the profile of its creator, as explained below; 8) the dynamic governance of information according to the value established by its ongoing usage, as explained below; 9) the dynamic governance of information according to its distributed ownership, as explained below; 10) the dynamic governance of information according to what class of information it might be incorporated into, and according to what class of information container it might incorporate, as explained below; 11) the dynamic governance of information according to self-reporting, as explained below.

**[00101]** Referring now to Fig 4, registers 120 may be (1) pre-defined, (2) created by the user or acquired by the user, or (3) system-defined or system-created. Pre-defined registers 120 are those immediately available for selection by the user within a given container editor as part of that container editor, in order that the user may append any of those registers 120 to a container 100 and define values for those registers 120 where required. Registers 120 created by the user are those conceived and created by a specific user or user group and made immediately available for selection by the user or user group in conjunction with any of a wide number of container editors, in order that the user may append any of those registers 120 to a container 100 and define values for those registers 120 where required. Registers 120 acquired by the user are those registers existing network-wide 201,

created by the user base, that might be located and acquired by the user in order that the user may append any of those registers 120 to a container 100 and define values for those registers 120 where required. System-defined registers are those registers whose values are set and/or controlled by the system 10. System-created registers are those registers created by the system 10.

**[00102]** Registers 120 are user or user-base created or system-created values or ranges made available by the system 10 to attach to a unique container, and hold system-set, user-set, or system-evolved values. Values may be numeric, may describe domains of time or space, or may provide information about the container 100, the user, or the system 10.

Registers 120 may be active, passive or interactive and may evolve with system use. Pre-defined registers include, but are not limited to, system history 110000, container history 101000, active time 102000, passive time 103000, neutral time 104000, active space 111000, passive space 112000, neutral space 113000, containment 105000, inclusion 106000, identity 114000, value 115000, ownership 107000, ownership addresses 116000, proportionate ownership 117000, creator profile 108000, receptivity 118000, influence 119000, points 109000, others 120000, reporting 121000, neutrality 122000, acquire 123000, create 124000, content title 125000, content key phrase(s) 126000, and content description 127000, security 12800, and parent rules 129000.

**[00103]** Pre-defined registers comprise an historical container register 101000, logging the history of the interaction of that container 100 with other containers, events and processes on the network 201, an historical system register 110000, logging the history of pertinent critical and processes on the network, a point register 109000 accumulating points based upon a hierarchically rated history of usage, an identity register 114000 maintaining a

unique network wide identification and access location for a given container specifying a unique time and place of origin and original residence, a proportionate ownership register 117000 maintaining a record of ownership percentage and economic values, and others 120000.

[00104] User-defined registers include a report register 121000 setting trigger levels for report sequences, content determination and delivery target, three time registers, consisting of a range, map, graph, list, curve or other designating time relevance, 102000 assigning the time characteristics by which that container will act upon another container or process, 103000 assigning the time characteristics by which that container be acted upon by another container or process, and 104000 assigning the time characteristics by which that container will interact with another container or process, three space registers, consisting of a range, map, graph, list, curve or other designating the domain and determinants of space relevance, 111000 assigning the space characteristics by which that content will act upon another container or process, 112000 assigning the space, characteristics by which that content will be acted upon by another container or process, and 113000 assigning the space characteristics by which that container will interact with another container or process, a domain of influence register 119000, determining the set, class and range of containers upon which that container will act, a domain of receptivity register 118000, determining the set, class and range of containers allowed to act upon that container, a domain of neutrality register 122000, determining the set, class and range of containers with which that container will interact, a domain of containment register 105000, determining the set, class and range of containers which that container may logically encompass, a domain of inclusion 106000 register, determining the set, class and range of containers by which that container might be

encapsulated, an ownership register 107000, recording the original ownership of that containers, a creator profile register 108000, describing the creator or creators of that container, an ownership address register 116000, maintaining the address of the creator or creators of that container, a value register 115000, assigning a monetary or credit value to that container, other registers 120000 created by users or the system, a reporting register 121000, determining the content, scheduling and recipients of information about that container, a neutrality register 122000, an acquire register 123000, enabling the user to search and utilize other registers residing on the network, a create register 124000, enabling the user to construct a new register, a content title register 125000, naming the contents of the container, a content key register, 126000, identifying the container contents with a key phrase generated by the user and/or the system based upon successful usage of that phrase in conjunction with the utilization of the information within that container 100, a content description register 127000, identifying the container contents with additional description, a security register 128000, controlling container security, and a parent container register 129000, storing the rules governing container interaction as dictated by the parent (encapsulating) container.

**[00105]** The container also includes a gateway 200 and gateway storage 205.

**[00106]** Gateways 200 are logically defined passageways residing both on containers 100 and independently in the system 10. Gateways 200 govern the interactions of containers 100 encapsulated within their domain by reading and storing register 120 information of containers entering and exiting that container 100.

**[00107]** The present invention also includes container gateway storage 205. Gateway storage 205 stores information regarding the residence, absence, transience, and alteration of encapsulated and encapsulating containers 100, and their attached registers 120,

holding the data collected from registers 120 of transient containers 100 in order to make it available to the data collection means 700 and the data reporting means 600, and storing the rules governing the operations of its particular gateway 200.

**[00108]** Referring now to FIG. 5, a flow chart of the preferred method for creating a container 100 is shown.

**[00109]** Input is received from the user selecting a container level through use of a drop-down menu 10100. A menu of all possible container classes within the subset and superset scheme of multiple hierarchically nested containers, i.e.; element, document, file, database, warehouse, domain, and more, is displayed on the output device 10200. Input is received from the user selecting a class 10300.

**[00110]** A graphic representation of a container in that class, with registers common to all containers as well as registers unique to its class is displayed 10301.

**[00111]** Input is received from the user choosing to “create” 10400, “edit” 10500, or “locate” 10600.

**[00112]** When the input of “create” 10400 is received from the user, a container template in that class appears 10410. Input from the user is then received adding or selecting a register 10540 to append to that container template. When input is received from the user adding a register, a list of registers that might be added to that class of container is made available to select 10550. Input is received from the user selecting a register 10560 and editing it 10570. The menu returns to “add or select” 10540.

**[00113]** If the input of “locate” 10600 is received from the user, the system prompts the user to enter the identity of the container or class of containers 10605. The system locates the container(s) 10610. Input is received from the user selecting a container

10620. The system prompts the user for a security code for permission to access the container for template use, or to alter its registers, or to alter its content 10630. . Input is received from the user entering a name and password providing access to one of the security levels 10640. Input is received from the user editing the container accordingly by transition to step 10500 and performing the steps for editing.

[00114] If the input of “edit” 10500 is received, a list of containers available to edit at that level is shown 10510. Input is received from the user selecting a container 10520. That container appears, available to edit 10530. Input is received from the user selecting “add” or “select” registers 10540 by the user clicking on the graphically depicted register, or from a drop down menu. Input is received from the user selecting the register to edit 10560. Input is received from the user selecting “modify” or “delete” for that register 10565. If input is received from the user to “delete,” that register is severed from the container. If input is received from the user to “modify”, the register editor 10570 screen appropriate to that register appears, i.e., an x-y type graph to define a curve of relevant active time, in which the user manipulates the x-y termini, scale and curve, or a global map in which Input is received from the user selecting the locale of active space, whether zip code, city, county, state, country, continent, plant or other. When input is received from the user saving the definition, the screen returns to the main container screen to make another selection available. . Input is received from the user defining as many registers as he chooses. One of the registers may be named "new register." Input is received from the user selecting the new register, and if chosen by the user, defining a wholly unique and new kind of register by the user entering input into the register editor 125.

[00115] When the input is received from the user choosing to add a register, a list of registers that might be added to that class of container are made available to select 10550. Input is received from the user selecting a register 10560 and editing it 10570. The menu returns to “add or select” 10540, and in turn to Input – Select Container.

[00116] Input may then be received from the user choosing to add, modify, or delete the container contents 10700. Once the registers are defined, input is received from the user indicating completion and the interface reverts to the container editor. When input is received from the user choosing "select component" (to select the component to containerize) from the main menu bar 10700, a window appears allowing the user to select any file, component, or other container. If for example, the user were creating a warehouse container, and wishes to incorporate several databases into that container, input would then be received from the user selecting "database." The program would prompt the user for the location (directory) of that database or container. If the requested selection is not containerized, input may then be received from the user choosing to containerize the element at that time, after which the program returns to "select component." Once input is received from the user defining the database location, the program logically encases the directory or directories in the defined container. The above procedure may be repeated as many times as desired to include multiple databases within a single container. While logical simplicity would dictate that all containers within a container be of the same subset, it would be possible for input to be received from the user choosing containers of any subset to include in the container. When input is received from the user choosing "finished," the container is created with a unique network identity, preferably through some combination of exact time and digital device serial number, or centralized numbering system, or other means. The



container 100 contains all digital code, including data and program software from the selected items or containers.

**[00117]** Input may then be received from the user to publish the container 11100 at a user-identified or system suggested location 11200 to be selected 11400.

**[00118]** Input is received from the user to "publish", from the main menu bar 11100. Input is received from the user choosing to leave the container where it was created, move or copy it to another drive, directory, computer, or network the user designates, or select the location from location options offered by the system 11200, or submit, or duplicate and submit, the container to the analysis engine 400 for intelligent inclusion in other containers, thus allowing the system to publish the container as instructed or choose the residence of the container 11400.

**[00119]** If input is received from the user to choosing to "move," or "copy" a browse function allows the user to name the new location or browse a list of possible locations. If input is received from the user choosing to "submit," a browser function allows the user to name the analysis search engine 310 or browse a list of possible analyses engines. When input is received from the user choosing the residence of the container 11300, the program restores the search interface screen.

**[00120]** Referring now to FIG. 6, a flow chart of the method for searching for containers 100.

**[00121]** When input is received from the user selecting "search interface" from the main title bar, the search interface screen appears. The user is given the choice of containerizing selected content or requesting that container levels be displayed 30100. From a drop down menu another menu appears allowing input to be received from the user

selecting the container level 30200. Input is received from the user selecting the container level (from the smallest component to the whole system) 30300.

**[00122]** Input is received 30310 from the user selecting the phrases, containers or components, which then are re-submitted to the same process, until the input is received from the user selecting a specific site or container.

**[00123]** The search phrase, whether containerized or not, is submitted simultaneously to the search engine 30400 and the analysis engine 30500.

**[00124]** The screen then reports in a selection menu, the number of applicable sites found by the search engine 30410, the number of historically proven applicable sites found by the analysis engine 30410, the number of historically proven applicable containers at the selected container level or any container level found by the analysis engine 30410, and the number of historically proven new search phrases or digital segments found by the analysis engine 30320. Input is received from the user selecting one of the named sets above 30330. If input is received from the user choosing the search engine, the search interface lists the applicable site titles with a brief description 30410. If input is received from the user choosing the site list of the analysis, the search interface lists the applicable site titles with a brief description 30410. If input is received from the user choosing the container list of the analysis engine, the search interface lists the applicable container titles with a brief description 30410. If input is received from the user selecting a container 30420, the system offers the means to view titles and descriptions of sub-containers at any chosen class level. . If input is received from the user choosing the phrase list of the analysis engine, the search interface lists the applicable phrases or digital segments with a brief description 30320. The

search and search result cycle repeats until input is received from the user choosing to go to an individual container or site.

**[00125]** Input is received from the user entering text or any digital string describing his search objectives into a text or search box. When input is received from the user submitting the search string, the system provides the option of containerizing the search through the container editor 110. Once the search container 101 is created, the system restores the search interface 300 screen the user.

**[00126]** Input is received from the user selecting “search”, “supported search” or “both” from another drop-down menu and from submitting the search. When input is received from the user selecting “search” 30310, the search phrase is submitted to the search engine 30400, which searches both content and the appropriate container registers, as pre-indexed in the search engine, and returns a list of appropriate locations, components or containers. When input is received from the selecting “supported search”, the search phrase is submitted to the analysis engine search support, which returns a list, in a drop-down menu, of search phrases or individual containers, for any and all container levels, used by other users or created by the system and known to be historically successful for the described effort and the described searching user, as per the results of the analysis search engine. Input is received from the user selecting a new search phrase or specific container from the drop down menu 30330. When input is received from the user choosing a new search phrase, that phrase is also submitted to the analysis engine 30500 which returns a list of pre-compiled historically proven sites, components or containers associated with that search phrase 30320. Input is received from the user choosing a selection 30420 and the system calls up that specific site, container or component. If input is received from the user selecting a specific

site, container or component at any time during the search process, that element is called up by the system 30440.

[00127] Input is received from the user choosing to containerize a search or select a container level in which to search 30100. When input is received from the user choosing to containerize the search, the software moves to the container editor as described in Fig. 5, and then returns the user to the search interface screen. Input is received from the user selecting to search a specific container level or the whole network. The system shows the available levels 30200. Input is received from the user selecting a container level 30300, and entering the text or digital component comprising the search string 30310. The system searches the containers 30400 while simultaneously submitting the search string to the analysis engine 30500. While the system is accessing containers, sites or templates 30700, the analysis engine 30500 inquires of the appropriate database 30600 to access historically successful containers, sites or search templates corresponding to the search request 30700, which is then shown on another portion or option of the search interface, either as available containers or sites 30410 or as search template options 30320. On one portion or option of the search interface screen the corresponding containers or sites are listed and/or previewed for selection 30410. Input is received from the user selecting the container to access 30420. The system accesses that container 30430 and shows it on the screen 30440 for user review. Input is received from the user selecting an operation, i.e., preview, read, purchase, move, copy, lease, in any composed schedule with operations assigned specific values 30460, and the system obtains the specified result 30470. The selection of the operation including any interaction with any uniquely defined container 100 is recorded 30800 by the container gateway (Fig. 2 A, 200), stored in the gateway storage 205 and made available to the analysis engine (Fig. 9) by the data

collection and reporting means (Fig. 8). Reporting and collection occurs on a regular basis according to user determined times or rules. The analysis engine compiles and analyzes selections according to various rules-based systems applicable to the particular container area of residence in cyberspace.

**[00128]** Input is received from the user selecting the container or site 30410, proceeding as described above, or selecting a search template 30330, and editing it to re-enter the search 30310. All operations on Fig. 6 utilize the communication device 26 whenever necessary or expeditious.

**[00129]** Referring now to FIG. 7, a flow chart of the search process is shown. Steps in FIG. 7 repeated from FIG. 6 are given the same reference number as in FIG. 6 for convenience and ease of understanding. Fig. 7 commences with “SEARCH TRANSITS GATEWAY 32100”, continuing from Fig. 6, “SYSTEM SEARCHES CONTAINERS 30400”. The submitted search 32100 transits the gateway 200. The gateway 200 interacts with the container registers 32200. The gateways 200 store the information downloaded from the registers 32300, and the container registers are altered 32500. The container registers 120 then interact with the registers 120 of the encapsulated search, which registers, and the values set within, have been constructed and appended to the search through the search interface 32600. Values are exchanged and compared and operations performed under the rules governing both interacting containers 100, and the rules governing the search container 100 and any gateway 200. The search engine 320, operating under the principles and means of search engines presently existing as described elsewhere, then provides to the search interface 32600 a list of containers 100 meeting the requirements of the search and its appended registers, as well as additional search options 32900. The gateway 200 reports and makes

available for collection to the analysis engine 400 the information obtained from the interaction 32400. On a periodic basis defined by the user or a rules-based system, the analysis engine 400 (Fig. 9) stores in databases 900, analyzes and instructs the execution engine 500, and the execution engine 500 executes changes in the system components as defined below (Fig. 10). All operations on Fig. 7 utilize the communication device 26 whenever necessary or expeditious.

[00130] On the remaining figures, shapes referring to other figures, to operations external to the scope of the present figures, or to the subject of the present drawing, are indicated with dashed lines, and are shown only to place the described operations in the context of continuous and continual operations external to the drawing.

[00131] Referring now to FIG. 8, a flow chart of the preferred process for collecting and reporting information on containers is shown. The data reporting 600 and data collection 700 means utilizes subroutines within the analysis engines 400 and gateways 200 to submit and collect register information and sub level analysis to other analysis engines 400 or other gateways 200 of a higher (larger) logical set in a set pattern and frequency defined by the administrator.

[00132] Input is received from the user selecting "data reporting" 70100 from the "edit gateway" drop-down menu. Container levels are displayed 70200. Input is received from the user selecting container level 70300. A menu of all possible gateways 70320 and analysis engines 70330 residing on gateways on the above defined container class appears, depicted graphically as a tree of analysis engines and gateways at that container level. Input is received from the user selecting "source" from "source or destination." Input is received from the user 70400 selecting a container, containers, or class of container by clicking on the

graphically depicted container(s) or container level on a display device. Input is received from the user 70410 selecting “destination” from “source or destination” Input is received from the user 70500 selecting an analysis engine, analysis engines, or class of analysis engine by clicking on the graphically depicted analysis engine(s) or analysis engine level on a display device. A time scheduler is displayed. Input is received from the user 70510 selecting the reporting frequency for the selected gateways to report data to the selected engines. The data from the gateways is thenceforth continuously moved or copied to the analysis engines by the system 10 utilizing the execution engine 500 according to the defined schedule, rules and pattern 70420, 70520.

**[00133]** Input is received from the user selecting "choose container level" 70300 from the gateway editor drop-down menu. A menu 70320 appears listing the classes of containers on the system within the defined subset and superset scheme of multiple hierarchically nested containers, i.e.; element, document, file, database, warehouse, domain, appears. Input is received from the user selecting the class of containers. A graphic representation of that container level throughout the system appears. Input 70300 is received from the user selecting individual containers or all the containers in that class.

**[00134]** From the gateway editor drop-down menu input 70100 is received from the user selecting "data collecting" A menu of all possible gateways and analysis engines residing on gateways on the above defined container class appears, depicted graphically as a tree of analysis engines, and gateways at that container level. Input 70510 is received from the user selecting “source” from “source or destination.” Input is received from the user selecting a container, containers, or class of container by clicking on the graphically depicted container(s) or container level. Input 70510 is received from the user selecting “destination”

from “source or destination.” Input 70510 is received from the user selecting an analysis engine, analysis engines, or class of analysis engine by clicking on the graphically depicted analysis engine(s) or analysis engine level. A time scheduler appears. Input 70510 is received from the user selecting the collecting frequency for the selected engines to collect data from the selected gateways. The data from the gateways is thenceforth continuously moved or copied to the analysis engines by the system 10 utilizing the execution engine 500 according to the defined schedule, rules and pattern.

**[00135]** The data collection 700 means, utilizing the communication device 26 and an execution engine 500, comprises one or more subroutines or agents programmed to travel through the network collecting the accumulated data and analyses from selected analysis engines, gateways or selected subset level of analysis engines or gateways (as above) in a pattern and frequency defined by the gateway administrator at a given container level. Input 70510 is received from the user or administrator, defining the collection and reporting of data, thus controlling permission within his gateway, and being subject to permission levels defined by others beyond his gateway.

**[00136]** Input is received from the user or gateway administrator selecting collection or reporting 70100 and the system shows the container levels available 70200. Input is received from the user selecting a container level 70300. Input is received from the user selecting “gateway” 70400 or “engine” 70500. The system shows gateways 70320 or engines 70330 associated with that level. Input is received from the user editing the reporting parameters associated with a gateway or a class of gateways 70410 or an engine or class of engines 70510. Input is received from the user selecting the collecting frequency for the chosen engines. When input is received from the user choosing to user save the definition,



the screen returns to the main container screen, step 70100 to make another selection available. Input is received from the user choosing to repeat the cycle, choosing “destination” to describe the destination analysis engines and the data collecting frequency from those destination analysis engines. The data collection means 700 collects the accumulated gateway information in a pattern and frequency defined by the gateway administrator or user at a given container level.

[00137] The system utilizing the execution engine (see Fig. 10) distributes the new parameters to the gateways 70420 or engines 70520 by the communication device 26. Using the new parameters the gateways report to the analysis engines 70430 after, in some cases, conducting sub-analysis 70440, or using sub-analysis 70440 to submit directly to specified gateways under certain conditions and parameters, and the analysis engines collect from the gateways 70530. The analysis engine uploads, downloads and utilizes information to databases 900 to conducts its analysis.

[00138] The invention includes an analysis engine 400. Through the data reporting 600 means and data collection 700 the analysis engine 400 receives data and sub-analysis from the search interface and the gateways. Data includes, for each gateway 200, the frequency and grade of access, the description of the user accessing, the identity of the container 100 accessing, the register parameters, and the historically accumulated register data.

[00139] Referring now to FIG. 9, a flow chart of the operation of the analysis engine 400 is shown. Analysis engines 400 may reside at any gateway or anywhere in the system 10. The analysis engine 400, operating under its own programmed sequence, utilizing the communication device 26, works, by means of programmed rules of logical,

mathematical, statistical or other analysis upon gateway and register information, in continuous interaction with the search process 410 and the data collection and reporting process 420 to analyze, determine and compile instructions 40100 on container construction 40110 to containerize in an automated process 40115, on container contents 40120 to move, copy or delete containers 40125, on storage schemes 40130 to move or copy containers to new storage 40135, on access routes 40140 to alter gateway pointers to sought information 40145, on search templates 40150 to add, delete or change search phrases and the referenced objects indicated by those search phrases 40155 and on gateway instructions 40160 to alter gateway registers and pointers 40165.

[00140] Thus, analyses might include, but are not limited to, the physical locus of the users accessing, the demographic classification of the users accessing, the access frequency for a given container, the range or curve of time relevance affecting a container, the range or region of space relevance affecting a container 100, the number or number of a specific type of container 100 transiting a gateway 200, the hierarchically graded usage of containers 100 or container contents 01 compared with the demographic of those users accessing the container, the hierarchically graded usage of containers 100 or container contents 01 compared with search phrases entered into the search interface 300, the hierarchically graded usage of containers 100 or container contents 01 compared with search phrases entered into the search interface 300 compared with the demographic of the users accessing, the number of pertinent containers nested within a given container 100. Once an analysis is accomplished, the result is compared to pre-programmed rules triggering instruction sets (such as moving a container to nest within another container).

[00141] Instructions are then sent to the execution engine 40200, which utilizes the communication device 26 to execute the instructions derived from the analyses. These containerized instructions transit the gateways 40300 and are utilized in the gateway process (Fig. 12)

[00142] Referring now to FIG. 10, a flow chart of the operation of the execution engine is shown. The execution engine 400, operating under its own programmed sequence in response to the instructions from the analysis engine 50100, utilizing the communication device 26, works in continuous process as its containerized execution instructions transit the gateways 50200 to create containers 50210 in an automated containerization process 50215, alter container contents 50230 by moving or copying containers to new containers 50235, to alter storage 50240 by moving or copying containers to new storage 50245, to alter access routes 50250 by altering gateway pointers 50255, to alter search templates 50260 by adding, changing and deleting search phrases and the referenced objects indicated by those search phrases 50265, to alter gateway instructions 50270 by altering gateway registers and pointers 50275. The execution works in a continuous loop with the gateway process 50300, the data collection and reporting process 50400 and the analysis engine process 50300.

[00143] The invention includes gateways 200. Gateways may be placed and reside anywhere on the network where containers transit. Gateways also reside on any or all containers. The gateway reads and stores the chosen register information from transient containers entering or exiting its logical boundaries. The resident analysis search engine, if any, performs the specified level of analysis. Data and analysis is both held for the collection means according to the pattern and timing specified in the data reporting 600 editor and

submitted according to the pattern and timing specified in the data collection means editor 700.

**[00144]** The gateways are network-wide, hierarchical, and nestable, and reside with a container encompassing any component, digital code, file, search string, set, database, network, event or process and maintaining a unique lifelong network wide identity and unique in all the universe historical identity, or may be strategically placed at such container transit points to gather and store register information attached to any such container, according to system-defined, system-generated, or user determined rules residing in its registers defining the behavior of those containers and components as they exit and enter one another, or interact with one another or any system process or system component within the logical domain of that container, or after exiting and entering that container, or defining how they interact with that unique gateway.

**[00145]** Gateway's registers comprise both system-defined and user-defined registers, alterable by author, duration, location, network-wide history, individual container history and/or interaction with other containers, gateways, networks or media, and evolve according to that gateway's history on a computer network, or according to the network history of events and processes, or according to that information component's interaction with other information containers, components, system components, network events or processes.

**[00146]** Referring now to FIG. 11, a flow chart of the gateway editor is shown. From the main title bar input is received from the user selecting "containerize" or "gateway level" 20100. When input is received from the user selecting "containerize" the system enters the container editor process 110. When input is received from the user selecting

“gateway,” the system shows the gateway levels available 20200. A menu of all possible gateways within the subset and superset scheme of defined multiple hierarchically nested gateways appears. Input is received from the user selecting the gateway level 20300. The system searches the gateways 20500 to locate the available gateway templates 20700 and the available gateways 20600. Input is received from the user selecting the gateway 20610 or gateway level template 20720. The system goes to the gateway 20620 or to the template 20720. A graphic representation of the chosen gateway 20630 or template 20730 appears. Input is received from the user to edit 20640 or create a gateway 20740. Once completed, input may be received from the user selecting "analysis level" from the gateway 200 drop-down menu, to select the level of analysis in a multi-level analysis sequence to be accomplished at the local level by a gateway-resident analysis engine. The user accesses the container editor to containerize (Fig. 5). Input is received from the user selecting the registers by clicking on the graphically depicted register, or from a drop down menu. Input is received from the user setting the registers as described elsewhere in (“container registers”). Input is received from the user selecting or defining the rules governing the interaction of that gateway with transient containers. Input is received from the user selecting or defining the rules governing the interaction of containers existing within the logical domain of the container 100 to which that gateway is attached. The user publishes the gateway (Fig. 5). Input is received from the user selecting "residence" from the main menu bar. ). Input is received from the user choosing to leave the gateway where it was created, move it to container on another drive, directory, computer, or network. If the user chooses "move," a browse function allows the user to name the new location or browse a list of possible

locations. Once input is received from the user choosing the residence of the gateway, the program restores the search interface screen.

**[00147]** The invention includes a data reporting means editor 610, and a data collection means editor 710, Fig. 2 A, as a menu option under the gateway editor 210.

**[00148]** The present invention also includes a gateway process.

**[00149]** Referring now to FIG. 12, a flow chart of the gateway process is shown. A system operation, search process or element container or process container is shown in transit 21100 passing through a gateway 21200. The container, operation or process interacts with the gateway 21300, uploading, downloading and exchanging information with the container, operation or process. The gateway stores container information 21400 and the container registers are altered 21500. The container registers also interact with the search interface 21600. The gateways report the register information or make it available for collection by the data reporting and collection means (Fig. 8) operating on the communication device 26 to provide the information to the analysis engine 21800, which stores 90100, analyzes and instructs the execution engine 21900, which processes and instructions are also stored 90100 by the execution engine upon receipt.

**[00150]** All operations in Fig. 12 utilize the communication device 26 whenever necessary or expeditious.

**[00151]** Referring now to FIG. 13 A, a drawing of nested containers 100 prior to the container modification process on a network 201 is shown. (Note: The same container numbering scheme is used in Fig. 13 A, 13 B, 13 C, 13 D and in 2 B.) Information containers 505 and 909, residing within container 908, operating under the rules governing container interaction within that container 908 downloaded to container 505 and 909 from gateway

9081 upon their entrance to container 908, which rules had been downloaded from execution engine 500 acting under the direction of analysis engine 400, and under the rules programmed into their own registers 404120, 909120, compare the specified (by those rules) set of registers 404120, 909120, i.e., time and space, and determine a container 404 encapsulated within 505 would be more appropriately encapsulated within container 909.

**[00152]** Referring now to FIG. 13 B a drawing of nested containers during a container modification process on a network 201 is shown. Container 404 is moved to reside with container 909. As the container 404 exits container 505, the gateway of container 505, being gateway 5051, operating under the rules governing container interaction with a gateway 5051 upon egress or egress as programmed in the gateway editor 210 and modified by the execution engine 500 executing the instructions of the analysis engine 400, or any greater logical analysis engine 408 providing execution instructions to an execution engine 508 operating in a larger encompassing container 108 entering through that container's gateway 208 or an independent gateway 707, or sub-analysis engine operating at any gateway level, records the register information of container 404. The gateway 5051 reports the transaction to the gateway 9081 of container 908, being the next higher logical container. Gateway 9081 holds in gateway storage 205 the information until collected by one or more data collection processes 700, or reported to one or more data reporting processes 600, serving one or more analysis engines 400 residing independently on the system 10 or an analysis engine at higher logical container 303. The analysis engine 400, comparing reports of user hierarchically graded usage under the operations of the search engine 320 and the search interface 300, on information container 808 after receiving reports from the data reporting means of container 404 being moved to container 909 determines, i.e., that the number of time and space

relevant containers residing within container 909 is sufficient to warrant an action, and directs the execution engine 500 to copy container 909, nested within container 908, to a third information container 808. As the copy instruction from execution engine 500 transits the gateway of container 908, the gateway 9081 records the instruction. The copy instruction interacts with the registers 909120 of container 909 regarding the rules governing its copying to another location. Once approved by the governing rules of registers 909120 appended to container 909, container 909 is duplicated. As the duplicate container 909 exits the container 908, the gateway records the register information 909120 of container 909, and the registers 909120 of container 909 are altered by special instructions from gateway 9081 under the rules residing in gateway 9081 regarding ingress and egress and the rules residing in the registers 909120 of container 909 regarding alteration by gateways upon ingress and egress. Passing through independent gateway 707, the register information 909120 is recorded, and awaits data collection or reporting 700, 600. As container 909 enters container 808, the gateway records the register information 909120 of container 909, the registers 909120 of 909 are altered by special instructions from gateway 8081, operating under the rules as described in the paragraph above, and container 909 takes up residence within container 808.

[00153] Referring now to FIG. 13 C, a drawing of nested containers after the container modification process on a network 201 process is shown. Container 909, now also logically residing within container 808, commences to interact with other containers 606 in 808 under the rules governing container interaction within container 808 as received from gateway 8081 upon transiting that gateway, and under the rules of registers 606120, 909120 of the interacting containers 606, 909, operating under the rules as described in the paragraph above. Through data collection and reporting 700, 600, analysis engine is appraised of



container's 909 new duplicate residence. I.e., operating under the registers of space relevance, a body of law pertaining to Boston Municipal tax law may be housed in a container holding Massachusetts tax law, but it would be more appropriately located in a container holding Boston tax law, with only a pointer to that location residing in the Massachusetts tax law container. In this example, such an analysis could be accomplished by comparison of zip code information in the space registers, or logical rules-based analysis, with "state" being a larger set than "city". Or, i.e., operating under the registers of time relevance, the curve of time relevance for a concert might follow an ascending curve for the months prior, hit a brief plateau, and then reach a precipitous decline, at which time certain pertinent information only might be moved to an archival container of city events or rock concerts of that year. In this example, once the curve is mapped into a register, that map would cause an increasing frequency of pointers to that container in other containers or gateways, or inclusion of that container in other containers, as the analysis engine compares that curve with increasing user inquiry.

**[00154]** Referring now to Fig. 13 D, a flowchart of the reconstruction process is shown.

**[00155]** Information containers 505 and 909, residing within container 908, operating under the rules governing container interaction within that container 908 downloaded 888103 to container 505 and 909 from gateway 9081 upon their entrance to container 908, which rules had been downloaded 888102 from execution engine 500 acting under the direction 888101 of analysis engine 400, and under the rules programmed into their own registers 404120, 909120, compare 888104 the specified (by those rules) set of registers

404120, 909120, i.e., time and space, and determine 888105 a container 404 encapsulated within 505 would be more appropriately encapsulated within container 909.

[00156] Container 404 is moved 888106 to reside with container 909. As the container 404 exits container 505, the gateway of container 505, being gateway 5051, operating under the rules governing container interaction with a gateway 5051 upon egress or egress as programmed in the gateway editor 210 and modified 888108 by the execution engine 500 executing the instructions of the analysis engine 400, or any greater logical analysis engine 408 providing execution instructions 888107 to an execution engine 508 operating in a larger encompassing container 108 entering through that container's gateway 208 or an independent gateway 707, or sub-analysis engine operating at any gateway level, records 888109 the register information of container 404, and alters the register information of container 404. The gateway 5051 reports 888110 the transaction to the gateway 9081 of container 908, being the next higher logical container. Gateway 9081 holds 888111 in gateway storage 205 the information until collected by one or more data collection processes 700, or reported to one or more data reporting processes 600, serving 888112 one or more analysis engines 400 residing independently on the system 10 or an analysis engine at higher logical container 303. The analysis engine 400, comparing 888114 reports of user hierarchically graded usage on information container 808 under the operations of the search engine 320 and the search interface 300, after receiving 888113 reports from the data reporting means of container 404 being moved to container 909, determines 888115, i.e., that the number of time and space relevant containers residing within container 909 is sufficient to warrant an action, and directs 888115 the execution engine 500 to copy container 909, nested within container 908, to a third information container 808. As the copy instruction

from execution engine 500 transits the gateway of container 908, the gateway 9081 records 888116 the instruction. The copy instruction interacts 888117 with the registers 909120 of container 909 regarding the rules governing its copying to another location. Once approved 888118 by the governing rules of registers 909120 appended to container 909, container 909 is duplicated 888118. As the duplicate container 909 exits the container 908, the gateway records 888119 the register information 909120 of container 909, and the registers 909120 of container 909 are altered 888120 by special instructions from gateway 9081 under the rules residing in gateway 9081 regarding ingress and egress and the rules residing in the registers 909120 of container 909 regarding alteration by gateways upon ingress and egress. Passing through independent gateway 707, the register information 909120 is recorded 888121, and awaits 888122 data collection or reporting 700, 600. As container 909 enters container 808, the gateway records 888123 the register information 909120 of container 909, the registers 909120 of 909 are altered 888124 by special instructions from gateway 8081, operating under the rules as described in the paragraph above, and container 909 takes up residence 888125 within container 808.

[00157] Container 909, now also logically residing (in addition to its original container residence) within container 808, commences to interact 888126 with other containers 606 in 808 under the rules governing container interaction within container 808 as received from gateway 8081 upon transiting that gateway, and under the rules of registers 606120, 909120 of the interacting containers 606, 909, operating under the rules as described in the paragraph above. Through data collection and reporting 700, 600, analysis engine is appraised 888127 of container's 909 new duplicate residence.

[00158] Referring now to Fig. 14, the screen interface of the container editor is shown. This interface is a process wherein input is received by the user using the main menu 78 or drop down menu 1419, or using an input device to “drag and drop” or click, causing the system 10 to acquire 1409, edit 1410 or create 1411 a file 1407, container 1408 or digital content 01, to search for 1412, acquire 1413, edit 1414 or create 1415, print 1416, or containerize 1417 a container 100, to select 1402, (or by clicking on register), search 1403, acquire 1404, edit 1405, or create a register 1406 to append or detach registers 120 to those containers, to set register values in those registers 120, to utilize the register editor 125 through 1405 to create new registers, or to 1418 add, detach, acquire a gateway 200 to append or detach to those containers, and utilize the gateway editor 210 through 1418. (See detailed description referring to Fig. 5)

[00159] Referring now to Fig. 15, the screen interface of the gateway editor is shown. This interface is a process wherein input is received by the user using the main menu 1501 or drop down menu 1513, or using an input device to “drag and drop” or click, causing the system 10 to search for 1507, acquire 1508, edit 1509 create 1510, print 1511 or containerize 1512 gateways, and causing the system 10 to establish rules by which an individual gateway governs the transiting 1502, entering 1503, exiting 1504 of containers and the interaction of containers within its domain 1505, and external of its domain.1506. (See detailed description referring to Fig. 11).

[00160] Referring now to Fig. 16, the screen interface of the search interface. This interface is a process wherein input is received by the user using the main menu 1625 or drop down menu 1624, or using an input device to “drag and drop” or click, or by entering text, causing the system 10 to select 1615, search for 1616, acquire 1617, edit 1618 create 1619,

print 1620, containerize 1621 (by accessing the container editor 110) or insert 1622 digital search strings into the search box 1623 in order to submit that string to the search engine 320, or causing the system 10 to select 1602, search for 1603, acquire 1604, edit 1605, create 1612, containerize 1613 (by accessing the container editor 110), or insert 1614 search keys (templates that comprise search scope in geographic range, container level, and specific key words or digital strings), or containerized searches (containers 110), into the search box 1623 in order to submit that string to the search engine 320, or causing the system 10 to set a search range by geographic range 1607, container level 1608, or acquire 1609, edit 1610 or create 1611 a scope template. (templates that comprise search scope in geographic range and, container level.) (See detailed description referring to Fig. 6).

**[00161]** Referring now to Fig. 17, a drawing showing, on an input device or computer screen 24, in any generic (dashed lines) software application program, a drop-down menu link 1403 on a drop down menu 1402 dropping down from a main menu 1401, and a free-floating button link 1404, is shown. When input is received at 1402 or 1403, the system 10 makes available to the user the containerization process or container editor 110. When input is received at drop-down menu link 1405 or a button link 1406, the system 10 makes available to the user the means to enter and interact with this system 10 or this network 201 in any of their aspects. The interfaces 1403, 1404 show a process wherein input is received causing the system 10 to encapsulate content or access the container editor 110. The link also allows the user to encapsulate the page or file on which he is currently working, without selecting content, and if so desired, without accessing the container editor. The interfaces 1405, 1406 show a process wherein input is received causing the system 10 to access or interact with the system 10 or the network 201.

**[00162]** The present invention also includes a search engine 320. Once the key word(s), phrase or digital segment is entered into the search interface 300, or an offered selection chosen on the menu, it is utilized by the search engine 320 to locate the desired site or data.

**[00163]** The search engine employed may be any industry standard search engine such as Verity “Topic”, or Personal Library Software, as used in Dow Jones News Retrieval, or Internet search engines such as Webcrawler, Yahoo, Excite, Infoseek, Alexa or any Internet search engine, or any new engines to be developed capable of searching for and locating digital segments, whether text, audio, video or graphic.

**[00164]** The present invention also includes an analysis engine 400. Utilizing rules-based analysis, the analysis engine determines the class of storage medium upon which containers reside, the subsets and supersets by which and in which containers encompass and reside within one another, the routes of access to those containers, the historically successful search parameters by which those containers are accessed based upon the identity of the user accessing the containers, and the grade of access chosen by the user in accessing that container 100.

**[00165]** Utilizing a pre-programmed sequence of compilation, and inductive, deductive and derivative analysis, the analysis engine manufactures instructions based upon the analysis of the information submitted by the gateways and the search interface, and submits those instructions to the appropriate execution engine 500 in order to create new information containers, content assemblages, storage schemes, access routes, search templates, and gateway instructions, and others, and to provide informed search options through the search interface to the inquiring user.

[00166] The present invention also includes an engine editor 510, that provides a system administrator with a means of editing the operating principles of that search engine, and search template loading in the search interface 300, a reporting and collection means editor 610, 710, governing data reporting 600 and data collection 700 at the gateways 200 as defined by the gateway editor 210 and the register editor 125, a container editor 110 for creating and modifying containers and appending registers to containers, a register editor 125 for creating and modifying container registers and establishing and adjusting the values therein, container gateways 200 with their own storage 205, information containers 100 for holding information and container registers for holding information about specific containers and their history on the network.

[00167] The present invention also includes an execution engine 300. Based upon instructions received from the analysis engine 400 utilizing the communication device 26, the execution engine 500 provides search phrases to the search interface 300 based upon initially received inquiries, relocates containers including their programs, data and registers to other directories, drives, computers, networks on other classes of storage mediums, i.e., tape drive, optical drive, CD-ROM, deletes, copies, moves containers to nest within or encompass other containers on other directories, drives, computers, networks to nest within other containers, alters the class of storage medium upon which containers reside, the subsets and supersets by which and in which containers encompass and reside within one another, the routes of access to those containers, and the historically successful search parameters by which those containers are accessed based upon the identity of the user accessing the container and the grade of access chosen by the user in accessing that container.

[00168] The execution engine 400 fulfills the instructions of the analysis search engine 500, to create new information containers, content sub and superset assemblages, storage schemes, access routes, search templates, gateway 200 instructions and other system functions. The execution engine includes an editor 510 that provides a system manager with a means of editing the operating principles of that search engine, governing data reporting, data collection 700, search template loading, gateway instructions, and other functions.

[00169] The present invention also includes flat or relational databases 900, used where, and as required.

[00170] The present invention also includes a communication device 26 supporting all operations on a network wide basis.

[00171] The present invention also includes a search engine 300 to locate the desired site or data. The present invention also includes databases 900, flat or relational, to serve the other components of the system as needed and where needed.

[00172] The present invention also includes editors, by which the user may alter the governing aspects of the system. Editors include, but are not limited to, a container editor 110, a register editor 125, a gateway editor 210, an engine editor 510, a reporting means editor 610, a search interface 300, and a collection means editor 710.

[00173] The present invention also includes specific screen interfaces for the editors, as described in Fig. 14, Fig. 15. and Fig. 16.

[00174] The present invention also includes a means for this system 10 and network 201 or container editor 110 to be accessed from a menu or button selection within any program, as described in Fig. 17.



[00175] While the present invention has been described with reference to certain preferred embodiments, those skilled in the art will recognize that various modifications may be provided. For example, both analysis engine and execution engine may be duplicated or modified for distribution at various locations and hierarchical positions in the gateway and container system throughout the network and designed to work in concert. Also, the physical computing infrastructure may be mainframe, mini, client server or other with various network and distributed computing designs, including digitally supported or based physical or public media, and the components of the system 10, as described in Fig. 1 may be physically distributed through space. Even the contents of a single container may be logically referenced but be physically distributed through the network and reside at multiple storage locations. The whole system may be hierarchically nested within other systems to the nth degree. Whole systems may also be encapsulated within containers. A single container may also encompass a single physical media, such as a CD-ROM disk, programmed with the container, gateway and register design. Gateways may be strategically placed on containers at ingress and/or egress points or may be placed strategically throughout the system for optimal collection and reporting output and gateway system control. Also, the loop of gateway data collection and reporting, analysis engine analysis, instruction, and gateway modification, and execution engine operations may be infinitely nested, from the smallest container of two sub-containers to whole networks holding millions of containers and thousands of levels, with analysis itself nested within the multiple levels. Gateways may be established at both logical and physical junctures such as a satellite uplink point. Also, the provision to establish a unique network identity might be designed to include as of yet unknown computer networks as they arise. The analysis and execution engines may operate on a rules-based, fuzzy logic, artificial

intelligence, neural net, or other system not yet devised. Other variations upon and modifications to the preferred embodiments are provided for by the present invention, which is limited only by the following claims. Also, the classification scheme of nested containers, while designated by the container creators, may transform, be utilized otherwise, or be wholly discarded according to usage. Also, hardware configurations, such as the use of RAM or hard drives for storage or lasers for communication may assume myriad forms without altering the essential operation of this invention.

WHAT IS CLAIMED IS:

1. A method comprising:  
receiving a search query;  
searching container registers encapsulated and logically defined in a plurality of containers to identify one or more containers responsive to the search query; and  
providing a list characterizing the identified containers.
2. A method as in claim 1, when the received search query comprises a labeled data tree having at least one parent-child relationship.
3. A method as in claim 1, further comprising: providing information identifying containers that have previously been used to respond to one or more processed queries that are substantially similar to the search query.
4. A method as in claim 1, wherein the provided information is stored in one or more search templates.
5. A method as in claim 1, further comprising: providing information identifying substantially similar search phrases, search templates, or labeled data trees that have previously been used to respond to one or more processed queries that are substantially similar to the search query.

6. A method as in claim 5, further comprising:  
receiving a selection of one of the substantially similar search phrases; and  
providing a list of previously identified containers associated with the selected search phrase.
7. A method as in claim 1, wherein the list provides a title of each identified container and a short description of its contents.
8. A method as in claim 1, further comprising: receiving a container search level parameter; and wherein the searching content and container registers only searches within container levels associated with the container search level parameter.
9. A method as in claim 1, further comprising: receiving a container search level parameter; and wherein the list of identified containers only comprises containers associated with the container search level parameter.
10. A method as in claim 1, wherein the searching further comprises: encapsulating the search query into a search container.
11. A method as in claim 10, wherein the searching further comprises:  
receiving, by a gateway, the search container;  
storing, by the gateway, data contained within a register of the search container; and

determining whether any registers of containers accessible via the gateway are associated with the register of the search container.

12. A method as in claim 11, further comprising:

generating a new gateway; and

associating the container with the new gateway.

13. A method as in claim 11, further comprising: periodically aggregating the contents of registers in a plurality of gateways to characterize a plurality of containers coupled thereto.

14. A method as in claim 11, wherein the contents of the registers in each of the plurality of gateways comprise at least one metric chosen from a group comprising: frequency of access of the gateway, grade of access of the gateway, description of users that have accessed the gateway, an identity of containers that have accessed the gateway, parameters associated with the gateway register, and historically accumulated register data.

15. A method as in claim 11, further comprising: monitoring transactions involving one or more gateways or containers.

16. A method as in claim 15, further comprising: generating new containers based on the monitored transactions.

17. A method as in claim 15, wherein the transactions are based on each instance a gateway or container passes through another gateway or container.

18. A method comprising:  
receiving a search query;  
polling a plurality of gateways to identify registers encapsulated therein, the registers relating to one or more containers logically defining data contained therein associated with the search query, wherein the containers are coupled to the gateways; and  
providing a list characterizing the identified containers.

19. A method comprising:  
reporting, by a plurality of gateways, registers or register values encapsulated therein, the registers relating to one or more containers coupled to the gateways logically defining data contained therein;  
updating a centralized index based on the reporting; and  
modifying at least one of a register, a register value, content of a container, stored information in a gateway, and an access route to any of the containers based on the updating.

20. A method as in claim 19, wherein the reporting occurs asynchronously or periodically after at least one interaction of container through the registers in which at least one of the following is altered: a register, a register value, the contents of a container, stored information in a gateway, and an access route to any of the containers.

21. A computer program product, tangibly embodied on computer-readable media, comprising instructions operable to cause a data processing apparatus to:

receive a search query;

search content and container registers encapsulated and logically defined in a plurality of containers to identify one or more containers associated with the search query; and

provide a list characterizing the identified containers..

22. A computer program product, tangibly embodied on computer-readable media, comprising instructions operable to cause a data processing apparatus to:

receive a search query;

poll a plurality of gateways to identify registers encapsulated therein, the registers relating to one or more containers logically defining data contained therein associated with the search query, wherein the containers are coupled to the gateways; and

provide a list characterizing the identified containers.

23. An apparatus comprising:

means for receiving a search query;

means for searching content and container registers encapsulated and logically defined in a plurality of containers to identify one or more containers associated with the search query; and

means for providing a list characterizing the identified containers.

24. An apparatus comprising:

means for receiving a search query;

means for polling a plurality of gateways to identify registers encapsulated therein, the registers relating to one or more containers logically defining data contained therein associated with the search query, wherein the containers are coupled to the gateways; and

means for providing a list characterizing the identified containers.

25. A method comprising:

receiving historical data from a plurality container registers encapsulated and logically defined in a plurality of containers, the historical data associated with interactions of the containers with other containers via those registers; and

modifying at least one of a register, a register value, content of a container, stored information in a gateway associated with a container, or an access route to any of the containers associated with any of a gateway based on the polling.

26. A method as in claim 25, when the obtained historical data comprises a labeled data tree having at least one parent-child relationship.

27. An apparatus comprising:

a processor; and

a memory unit operable to encode instructions operable to cause the processor to:

poll a container registers encapsulated and logically defined in a plurality of containers to obtain historical data associated with interactions of the containers with other containers; and



modify at least one container register or encapsulated content of at least one container based on the obtained historical data.

28. A method comprising:

polling gateways to obtain historical data associated with interactions of a plurality of containers with other containers, the containers encapsulating and logically defining container registers storing, the gateways intercepting data associated with the containers during the interactions; and

modify at least one of the container registers, content encapsulated within at least one of the containers based on the obtained historical data, or information stored in at least one gateway based on the polling.

29. An apparatus comprising:

a processor; and

a memory unit operable to encode instructions operable to cause the processor to:

poll gateways to obtain historical data associated with interactions of a plurality of containers with other containers, the containers encapsulating and logically defining container registers storing, the gateways intercepting data associated with the containers during the interactions; and

modify at least one of the container registers, content encapsulated within at least one of the containers, or information stored in at least one gateway based on the obtained historical data.

30. A method comprising:
- receiving a search query;
  - searching container registers encapsulated and logically defined in a plurality of containers to identify one or more search query templates; and
  - providing a list characterizing the identified one or more search query templates to formulate subsequent search queries.



### ABSTRACT

A search query may be run against a plurality of container registers encapsulated and logically defined in a plurality of containers to identify one or more container registers responsive to the search query. Thereafter, a list characterizing the identified containers may be provided. Related methods, apparatus, computer program products, and computer systems are also described.

10592228.DOC



Attorney's Docket No. 17776-002US4

COMBINED DECLARATION AND POWER OF ATTORNEY

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name.

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled SYSTEM AND METHOD FOR CREATING AND MANIPULATING INFORMATION CONTAINERS WITH DYNAMIC REGISTER, the specification of which:

- is attached hereto.
[X] was filed on November 14, 2005 as Application Serial No. 11/280,700 and was amended on
was described and claimed in PCT International Application No. and as amended under PCT Article 19 on

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose all information I know to be material to patentability in accordance with Title 37, Code of Federal Regulations, §1.56.

I hereby claim the benefit under Title 35, United States Code, §119(e)(1) of any United States provisional application(s) listed below:

Table with 3 columns: U.S. Serial No., Filing Date, Status. Row 1: 60/073,209, January 30, 1998, Status.

I hereby claim the benefit under Title 35, United States Code, §120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, §112, I acknowledge the duty to disclose all information I know to be material to patentability as defined in Title 37, Code of Federal Regulations, §1.56(a) which became available between the filing date of the prior application and the national or PCT international filing date of this application:

Table with 3 columns: U.S. Serial No., Filing Date, Status. Row 1: 09/284,113, April 1999, Pending.

I hereby claim foreign priority benefits under Title 35, United States Code, §119 of any foreign application(s) for patent or inventor's certificate or of any PCT international application(s) designating at least one country other than the United States of America listed below and have also identified below any foreign application for patent or inventor's certificate or any PCT international application(s) designating at least one country other than the United States of America filed by me on the same subject matter having a filing date before that of the application(s) of which priority is claimed:

Table with 4 columns: Country, Application No., Filing Date, Priority Claimed. Row 1: PCT, PCT/AUS99/01988, January 28 1999, [X] Yes [ ] No.

BEST AVAILABLE COPY

Attorney's Docket No. 1:776-002US4

I hereby appoint all registered practitioners associated with Customer No. 26181 to prosecute this application and to transact all business in the Patent and Trademark Office connected therewith, and request that all correspondence be addressed to:

Customer No. 26181

Direct all telephone calls to CARL A. KUUKONEN, III at telephone number (858) 678-5030.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true, and further that these statements were made with full knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patents issued thereon.

Full Name of Inventor: MICHAEL DE ANGELO

Inventor's Signature:

*Michael De Angelo*

Date:

*Feb 13, 2006*

Residence Address:

3700 Andreas Hills Drive Palm Springs, CA 92264

Citizenship:

United States

Post Office Address:

3700 Andreas Hills Drive Palm Springs, CA 92264

10584582.doc

TEST AVAILAB

*JPW*



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant : Michael De Angelo  
Serial No. : 11/280,700  
Filed : November 14, 2005  
Title : SYSTEM AND METHOD FOR CREATING AND MANIPULATING  
INFORMATION CONTAINERS WITH DYNAMIC REGISTER

Art Unit : 2161  
Examiner : Unknown

**MAIL STOP AMENDMENT**

Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

INFORMATION DISCLOSURE STATEMENT

Applicants request consideration of the references listed on the attached PTO-1449 form. Under 37 C.F.R. § 1.98 (a)(2)(ii), only copies of foreign patent documents and/or non-patent literature are enclosed. Copies of any listed U.S. patents or U.S. patent application publications can be provided upon request.

This statement is being filed within three months of the filing date of the application or before the receipt of a first Office Action on the merits. Please apply any charges or credits to Deposit Account No. 06-1050.

Respectfully submitted,

Carl A. Kukkonen, III  
Reg. No. 42,773

Date: May 1, 2006

Fish & Richardson P.C.  
12390 El Camino Real  
San Diego, California 92130  
Telephone: (858) 678-5070  
Facsimile: (858) 678-5099  
10625442.doc

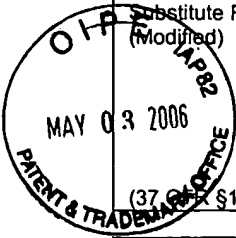
CERTIFICATE OF MAILING BY FIRST CLASS MAIL

I hereby certify under 37 CFR §1.8(a) that this correspondence is being deposited with the United States Postal Service as first class mail with sufficient postage on the date indicated below and is addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

\_\_\_\_\_  
Date of Deposit May 1, 2006

\_\_\_\_\_  
Signature *Teresa Salazar-Fischer*

\_\_\_\_\_  
Typed or Printed Name of Person Signing Certificate  
Teresa Salazar-Fischer



Substitute Form PTO-1449 (Modified) <b>Information Disclosure Statement                  by Applicant</b> (Use several sheets if necessary) (37 C.F.R. § 1.98(b))	U.S. Department of Commerce Patent and Trademark Office	Attorney's Docket No. 17776-002US4	Application No. 11/280,700
	Applicant Michael De Angelo		
	Filing Date November 14, 2005	Group Art Unit 2161	

U.S. Patent Documents							
Examiner Initial	Desig. ID	Document Number	Publication Date	Patentee	Class	Subclass	Filing Date If Appropriate
	AA	5,664,208	09/02/97	Pavley, et al.			
	AB	5,768,510	06/16/98	Gish			
	AC	5,815,665A	09/98	Teper, et al.			
	AD	5,848,246	12/08/98	Gish			
	AE	6,016,495	01/00	Mc Keehan, et al.			
	AF	6,075,791 A	06/00	Chiussi, et al.			
	AG	6,154,782 A	11/00	Kawaguchi, et al.			
	AH	6,173,280 B1	01/01	Ramkumar, et al.			
	AI	6,198,738B1	03/01	Chang, et al.			
	AJ	6,351,745	02/02	Itakura, et al.			

Foreign Patent Documents or Published Foreign Patent Applications								
Examiner Initial	Desig. ID	Document Number	Publication Date	Country or Patent Office	Class	Subclass	Translation	
							Yes	No
	AK	WO 98/02831	01/22/98	PCT				
	AL	WO 99/39285	08/05/99	PCT				

Other Documents (include Author, Title, Date, and Place of Publication)		
Examiner Initial	Desig. ID	Document
	AM	none

Examiner Signature	Date Considered
EXAMINER: Initials citation considered. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.	



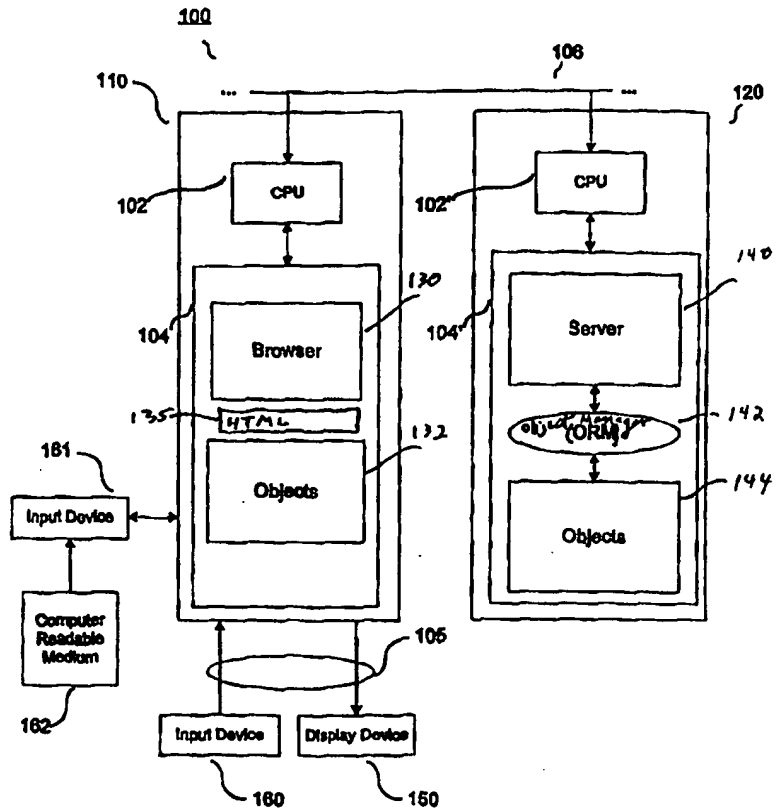
INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

<p>(51) International Patent Classification <sup>6</sup> : <b>G06F 17/30</b></p>	<p><b>A1</b></p>	<p>(11) International Publication Number: <b>WO 98/02831</b> (43) International Publication Date: 22 January 1998 (22.01.98)</p>
<p>(21) International Application Number: PCT/US97/11885 (22) International Filing Date: 10 July 1997 (10.07.97) (30) Priority Data: 08/678,680 11 July 1996 (11.07.96) US (71) Applicant: TANDEM COMPUTERS INCORPORATED [US/US]; 10435 N. Tantau Avenue, Loc 200-16, Cupertino, CA 95014 (US). (72) Inventors: ERLENKOETTER, Ansgar; Auf der Heide 44, D-61267 Neu-Anspach (DE). DE BORST, Jeroen, Peter; Alte Mauergasse 5, D-61348 Bad Homberg (US). BONHAM, Peter, Douglas; Am Alten Bach 19, D-61352 Bad Homberg (DE). (74) Agents: GRANATELLI, Lawrence, W. et al.; Graham &amp; James L.L.P., 600 Hansen Way, Palo Alto, CA 94304 (US).</p>	<p>(81) Designated States: JP, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).</p> <p><b>Published</b> <i>With international search report. Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.</i></p>	

(54) Title: HYPERMEDIA OBJECT MANAGEMENT

(57) Abstract

A method and apparatus that uses a hypermedia approach to managing distributed objects. A first embodiment of the present invention uses the World Wide Web hypermedia system. A user initializes browser software that allows the user to browse and change various attributes of objects in the system. The browser communicates with a server that includes an http adapter and a gateway. The gateway can access objects in the system and generate HTML code in accordance with the objects. One embodiment of the present invention uses hierarchical tree-oriented objects. These objects are "self-describing" (also called "introspective"). The server queries the objects in response to the queries from the browser and each queried object responds with information about itself. In another preferred embodiment, the server initiates queries of the objects and retains this information for use in responding to later queries from the browser.





**FOR THE PURPOSES OF INFORMATION ONLY**

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon	KR	Republic of Korea	PL	Poland		
CN	China	KZ	Kazakhstan	PT	Portugal		
CU	Cuba	LC	Saint Lucia	RO	Romania		
CZ	Czech Republic	LI	Liechtenstein	RU	Russian Federation		
DE	Germany	LK	Sri Lanka	SD	Sudan		
DK	Denmark	LR	Liberia	SE	Sweden		
EE	Estonia			SG	Singapore		

HYPERMEDIA OBJECT MANAGEMENT

5

**FIELD OF THE INVENTION**

This application relates to object oriented programming and, in particular, to management of distributed objects via the World Wide Web.

10

**BACKGROUND OF THE INVENTION**

The past several years have seen an explosive growth of the use of distributed objects. Now, a single system may be composed of objects obtained from different vendors and having different interfaces. Such objects are called "heterogeneous objects." Thus, a system can be formed of a large and rapidly changing number of heterogeneous objects. Such a system requires a flexible and adaptive approach for system and application management. Conventionally, a heterogeneous system is managed by way of object-specific presentation facilities, i.e., by way of a user front-end that was written for each type of heterogeneous object. Such an approach is, however, too expensive in both development time and maintenance and administrative costs. In addition, conventional object management is often achieved through a single management center. Use of a single center is not efficient when a large number of objects need to be managed.

15

20

**SUMMARY OF THE INVENTION**

The present invention overcomes the problems and disadvantages of the prior art by using a hypermedia approach to object management. In this approach, each object is akin to a hypermedia document. The described embodiment of the present invention uses the World Wide Web hypermedia system. In a preferred embodiment of the present invention, a user initializes browser software that allows the user to browse and change various attributes of objects in the system. The browser communicates with a server that includes an http adapter and a gateway. The gateway can access objects in the system and generate HTML code in accordance with the objects.

25

30

A described embodiment of the present invention uses hierarchical tree-oriented objects. In a first embodiment, these objects are "self-describing"

35

5 (also called "introspective"). The server queries the objects in response to the queries from the browser and each queried object responds with information about itself. In another preferred embodiment, the server initiates queries of the objects and retains this information for use in responding to later queries from the browser.

10 In accordance with the purpose of the invention, as embodied and broadly described herein the invention is a system for managing objects, including a first server, comprising: a first receiver portion configured to receive a request in a hypermedia format; a first translator portion configured to convert the hypermedia request to an object request; a sender portion  
15 configured to send the object request to an object manager; a second receiver portion configured to receive a response from the object manager; and a second translator portion configured to convert the object manager response to the hypermedia format.

20 In further accordance with the purpose of this invention, as embodied and broadly described herein the invention is a method for browsing objects, where a browser communicates with a server, comprising the steps, performed by the browser, of: sending an initial URL to the server; receiving first data from the server, where the first data specifies an object corresponding to the URL; sending user-entered data associated with the object to the server; and  
25 receiving second data from the server, where the second data specifies a second object corresponding to the user-entered data.

30 Advantages of the invention will be set forth in part in the description which follows and in part will be obvious from the description or may be learned by practice of the invention. The advantages of the invention will be realized and attained by means of the elements and combinations particularly pointed out in the appended claims and equivalents.

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

35 The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate several embodiments of the invention and, together with the description, serve to explain the principles of the invention.

5 Fig. 1 is a block diagram of a computer system in accordance with a preferred embodiment of the present invention.

Fig. 2 is another block diagram of a computer system in accordance with a preferred embodiment of the present invention.

10 Fig. 3 is a diagram of data sent between a browser, server, and object manager in accordance with the embodiment of Fig. 1.

Fig. 4 is a diagram of a format in which objects are organized.

Fig. 5 shows another example of a page displayed by the browser.

Figs. 6(a) and 6(b) show an example of HTML that causes the browser to display a portion of the page of Fig. 5.

15 Figs. 7(a) through 7(c) show further examples of HTML that result in the portions of page of Fig. 5.

Figs. 8(a) and 8(b) show several examples of ORM (Object Resource Management) requests made by the server to the object manager and the resulting responses from the object manager.

20 Fig. 9 shows another page displayed by the browser.

Fig. 10 shows layers of functions available to the object manager.

### **DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS**

Reference will now be made in detail to a preferred embodiment of the invention, an example of which is illustrated in the accompanying drawings. 25 Wherever possible, the same reference numbers will be used throughout the drawings to refer to the same or like parts.

#### **I. System Overview**

30 Fig. 1 is a block diagram of a computer system 100 in accordance with a preferred embodiment of the present invention. Computer system 100 includes a first computer 110 and a second computer 120. First computer 110 and second computer 120 are connected together via line 106, which can be, for example, a LAN, a WAN, or an internet connection. Line 106 can also represent a wireless connection, such as a cellular network connection.

35 First computer 110 includes a CPU 102; a memory 104; input/output lines 105; an input device 160, such as a keyboard or mouse; and a display

5 device 150, such as a display terminal. First computer 110 also includes an  
input device 161 that reads computer instructions stored on computer readable  
medium 162. These instructions are the instructions of e.g., browser software  
130. Memory 104 of first computer 110 includes browser software 130,  
Hypertext Markup Language (HTML) 135, and objects 132. A person of  
10 ordinary skill in the art will understand that memory 104 also contains  
additional information, such as application programs, operating systems, data,  
etc., which are not shown in the figure for the sake of clarity.

Second computer 120 includes a CPU 102' and a memory 104'.  
Memory 104' of second computer 120 includes server software 140, an object  
15 manager (ORM) 142, and objects 144. HTML 135 in the memory of first  
computer 110 was downloaded over line 106 from server 140 of second  
computer 120. A person of ordinary skill in the art will understand that  
memory 104' also contains additional information, such as application  
programs, operating systems, data, etc., which are not shown in the figure for  
20 the sake of clarity. Server 140, object manager 142, and objects 144 can also  
be located in memory 104 of first computer 110.

It will be understood by a person of ordinary skill in the art that  
computer system 100 can also include numerous elements not shown in the  
Figure for the sake of clarity, such as disk drives, keyboards, display devices,  
25 network connections, additional memory, additional CPUs, LANs, input/output  
lines, etc.

The following paragraphs provide a general discussion of the World  
Wide Web ("the Web"). The Web is built around a network of "server"  
computers, such as second computer 120, which exchange requests and data  
30 with each other using the hypertext transfer protocol ("http"). A human  
designer designs the layout of a Web page, which is then specified using HTML  
("Hypertext Markup Language"). Several versions of HTML are currently in  
existence. Examples include HTML versions 2.0 and 3.0, as specified by the  
WWW Consortium of MIT. The HTML used in the described embodiment of  
35 the invention includes frames, forms, and tables, as are known to persons of

5 ordinary skill in the art.

A user views a Web page using one of a number of commercially available "browser" programs. The browser submits an appropriate http request to establish a communications link with a Web server of the network. A typical http request references a Web page by its unique Uniform Resource Locator ("URL"). A URL identifies the Web server hosting that Web page, so that an http request for access to the Web page can be routed to the appropriate Web server for handling. Web pages can also be linked graphically to each other.

Fig. 2 is an additional block diagram of a computer system in accordance with a preferred embodiment of the present invention. Browser 130 communicates with server 140. Server 140 includes an http adapter 202 and a management gateway 204. Http Adapter 202 handles communication via the known http protocol. Management gateway 204 communicates with object manager 142. Server 140 communicates with one or more objects 132, 144 using a request/response (RR) protocol, such as the ORM (Object Resource Management) protocol, which is discussed below. Note that objects 132 and 144 can be located on the same or different physical computers or machines. Server 140 also communicates with external interface 206, which communicates using the known SNMP and CMIP protocols. Server 140 also communicates with external gateway 208, which communicates using the known SNMP and CMIP protocols. The system can contain more than one servers 140 and more objects than are shown in Fig. 4.

Fig. 3 is a diagram of data sent between a browser, server, and object manager in accordance with the embodiment of Fig. 1. In the example of Fig. 3, the user has already begun execution of browser software 130. In step 302, the user enters the URL of server 140 by way of browser 130. The browser sends a request to the server and, in step 304, the server responds with the HTML to generate a home page. The home page allows the user to enter a URL (or to chose a URL from those known provided within the HTML of the home page). The user can then chose to set/browse objects in the system,

5 as described below. The user can also request information and statistics about once or more objects in the system.

10 In step 306, the user enters a URL of an object by way of browser 130. Server 140 converts the URL to a request to an object manager. For example, in the described embodiment, server 140 converts the URL to an ORM request, as described below. The ORM request is sent to the object manager, which returns object data in steps 308 and 310. Server 140 converts the object data into HTML, which is sent to browser 130 in step 312. The HTML may be based on a predetermined page template known to the server. Alternately, the format of a page may be determined "on the fly" based on the information obtained from the object manager. Server 140 converts all pathnames, such as object-links in the object data (see Fig. 4) to URLs in HTML and vice versa. Thus, if a user clicks on an area in a page displayed by the browser that corresponds to an object-link, browser 130 has the URL corresponding to that object-link. This new URL is sent to the server, which obtains the page information and sends HTML to display information for the object connected to the object-link.

15 Steps 314 through 320 represent a "set" mode, in which the user enters new values for an object by way of browser 130. In step 314, the user indicates that he wishes to enter "set" mode. This indication is usually accomplished by clicking on a button in the current page (thus, the HTML generated by server 140 should include HTML for this button). In step 316, server 140 sends a "form" for set mode. In step 318, the user enters new values into the form and clicks on "submit" (or "apply", (see Fig. 5), as is known to persons of ordinary skill in the art. Server 140 converts the submitted form to, for example, an ORM request, as described below. The ORM request is sent to the object manager, which returns object data in steps 317 and 319. Server 140 converts the object data of step 319 into HTML, which is sent to browser 130 in step 320.

20 Steps 322 through 332 represent a "browse" mode, in which the user views values associated with an object by way of browser 130. In step 322,

5 the user indicates that he wishes to enter "browse" mode. This indication is usually accomplished by clicking on a button in the current page (thus, the HTML generated by server 140 should include HTML for this button). In step 324, server 140 sends a "form" for browse mode. In step 326, the user enters new values into the form and clicks on "submit" (or "apply", see Fig. 5),  
10 as is known to persons of ordinary skill in the art. Server 140 converts the submitted form to, for example, an ORM request, as described below. The ORM request is sent to the object manager, which returns data corresponding to the object in steps 328 and 330. Server 140 converts the response of step 330 into HTML, which is sent to browser 130 in step 332.

## 15 II. Hypermedia Object Management

### A. Object Organization

Fig. 4 is a diagram of a format in which objects are organized in a preferred embodiment. This organization is transparent to server 140 and browser 130. It will be understood that the present invention can be used with  
20 a number of object organizations and with a number of object management protocols. The embodiment described herein uses the ORM protocol, as described below.

The model of Fig. 4 assumes the following:

1) Management operations can be mapped to two basic operations:  
25 a) Get an attribute (or a set of attributes) of an object and b) set an attribute (or set of attributes) of an object.

2) All entities to be managed can be organized as a directed tree with nodes and leaves where the nodes are either (callable) objects or components (sub-parts of objects) with attributes as the leaves (with combined name/pair values), and  
30

3) All knowledge about management operations and attributes is built into and controlled by the managed object.

Fig. 4 shows the following types of entities:

1) Objects

35 Objects encapsulate and control management aspects and respective



5 management operations. In the described embodiment, an object is identified by a "pathname," which is the destination for object calls. Each manageable object has its own virtual tree of components, attributes, and object-links.

#### 2) Components

10 Components are the primary structuring mechanism within an object. Component sub-trees may be of arbitrary depth and component nodes may contain any number of object-links, other sub-components, or attributes.

#### 3) Attributes

15 Attributes describe specific aspects of a component within an object (for example, "status=running" describes the state of a resource). Attribute nodes have additional properties beyond name and value, such as access mode and data type. Attribute nodes are leaves and do not have children.

#### 4) Object-links

20 Object-links contain an object reference to a related object. As every object is responsible for its own virtual tree of resources, one object can provide a reference (hyperlink) to another object. Thus, in the described embodiment, a first object can have links to a second object, so that objects can be "walked" by way of browser 130.

#### 5) Relations

25 Objects and components are the primary means for structuring and navigation in the described embodiment. Attributes have values that characterize the state of the resource. All operations (browsing and attribute retrieval/setting) are performed with respect to a single level of the tree (e.g., relative to a specific parent).

30 Server 140 preferably issues the following requests to object manager 142:

- 1) Get a list of linked objects,
- 2) Get a list of components and/or sub-components,
- 3) Get a list of attributes,
- 35 4) Set a list of attributes (Along with name/value pairs for each attribute), and

5           5) Get an extended list of attributes, which returns meta-information about the attribute, such as data type, allowed access mode (ro, rw) or valid ranges of new attribute values. Within the ORM model, all management operations are mapped to these five operations. Thus, every managed object preferably supports these five operations.

10           It should be understood that the attributes and object types shown in the examples herein are included only for the purposes of example. The present invention can be practiced using any appropriate object organization and type.

#### **B. Server Interface**

15           In the described embodiment, all messages passing in and out of server 140 are ASCII messages.

          A example URL for object 402 of Fig. 4 would look like:

Http://ham/get/objectRoot/Component1/Component2/

          A example URL for attribute 404 of Fig. 4 would look like:

20       Http://ham/get/objectRoot/Component1/Component2/Attr1/

          In both of these URLs, "ham" stands for "HyperMedia Adapter to Management" and represents the address of server 140; "get" (this could also be "set") represents an operation to be performed on an object or attribute; and the remainder of the URL represents the tree of the object or attribute known to the object manager. Other URLs may also include additional information use, for example, by the object manager.

25           Fig. 5 shows a page displayed by browser 130 in "set" mode. Fig. 5 shows the values of attributes for a "Configuration" object component. These attributes include:

- 30           1) Status 520,  
          2) Maximum Concurrency 523,  
          3) Trace Level 524,  
          4) OSL Traces Enabled 526,  
          5) Script directory/Vol. 528,  
35           6) Script File 530,

- 5           7) Cache Tcl Scripts 532,  
          8) Tcl Trace Enabled 534, and  
          9) Maximum Size of Synthesized Page 536.

10           Fig. 5 also shows an entry 522 for changing the status attribute. It should be understood that the attributes of Fig. 5 are presented for the sake of example only and are not to be taken in a limiting sense. Fig. 5 also shows a reset button 540 and an apply button 550. When the user clicks reset button, original attribute values are returned. When apply button 550 is clicked, browser 130 posts a form, as is known to persons of ordinary skill in the art.

15           Figs. 6(a) and 6(b) show an example of HTML generated by server 140. When browser 130 interprets the HTML of Fig. 6, it generates the portion containing attribute values 520-536 and buttons 540, 550 of Fig. 5. Figs. 7(a) through 7(c) show an example of HTML generated by server 140. When browser 130 interprets the HTML 702, 704, and 706 of Figs. 7(a) through 7(c), it generates portions 502, 506, and 504, respectively, of Fig. 5.

20           Fig. 9 shows another page displayed by browser 130 in accordance with HTML generated by server 140. The page of Fig. 9 is used to browse objects, but cannot change the attributes of objects.

25           The previous paragraphs discuss the browser GUI presented to the user and how server 140 translates between HTML and a protocol understood by the object manager. The following paragraphs describe the protocol used to communicate with object manager 142 about objects and to change objects in accordance with the HTML received by the server.

30           Figs. 8(a) and 8(b) show several examples of ORM requests made by the server 140 to object manager 142 and the resulting responses from object manager 142. Pages of the description shows formats of such requests and responses. Request 802 is an example of an OrmGet request sent from server 140 to object manager 142. The format of an OrmGet request is:

OrmGet: pathname  
      entity types

35           where pathname is a name of an object or an attribute. Possible entity types

5 are: "Object" (all known objects at this level), "Component" (a list of all components below the level of the path specified in the OrmGet), "Attribute" (a list of attributes for the current node; for every attribute, its name and "stringified value is returned; if the pathname already navigates to an attribute, the object manager returns the empty string), "Info" (returns "meta-attributes" such as mode, range and unit), and <none> (i.e., an empty string).

10 In request 802 of Fig. 8, the server "knows" about an object "HyperMedia Adapter NSK", possibly from receiving a URL from browser 130. Line 820 represents a version of the server (e.g., version 1.0). Line 822 is an "OrmGet" request for object "HyperMedia Adapter NSK". Server 140 requests information from object manager 142 about entity types (Info), Component, and Object (lines 824).

15 Response 804 is generated by object manager 142 and sent to server 140. The object has four components, no info, and no objects at the same level. As seen in step 312 of Fig. 3, server 140 generates HTML 604 of Fig. 7(c) in accordance with response 804 and sends the generated HTML to browser 130.

20 Assuming that the user wants to browse information about the Configuration component of object "HyperMedia Adapter NSK", browser 130 sends a request to server 140 to this effect. Server 140 then sends request 806 to the object manager, which responds with response 808. Request 806 is similar to request 802, but the pathname in line 830 is "HyperMedia Adapter NSK/Configuration".

25 Response 808 includes attributes for the "Info" entity. Thus, the response includes an attribute value, mode, field, and range for each of ten attributes of the component "Configuration". As seen in step 332 of Fig. 3, server 140 generates the HTML of Figs. 6(a) and 6(b) in accordance with response 808 and sends the generated HTML to browser 130 (see Fig. 5).

30 Assuming that the user wants to change one or more attributes of the Configuration component of object "HyperMedia Adapter NSK", browser 130 sends a request to server 140 to this effect (assuming that the browser is in

5 "set" mode). Server 140 then sends request 810 to the object manager, which responds by sending a status value (not shown).

A format of the OrmSet request is:

OrmSet: pathname

Attribute: name

10 Value: val

where "name" and "val" respectively, represent an attribute name and an attribute value. This command is shown in line 840. The command can include more than one Attribute/Value pairs.

15 In the example, request 810 specifies new values for eight attributes of component "Configuration." Assuming that no error occurs when the object manager changes the attribute values, server 140 generates HTML reflecting the new attribute values in accordance with the response and sends the generated HTML to browser 130 (not shown).

20 A preferred embodiment of the present invention has a server that interfaces with "self describing" (or "introspective") objects. The server sends requests to and receives responses from an ORM (Object Resource Manager). The system may include more than one ORM and more than one server. Each server may "know" about zero or more ORMs. Thus, the system is not centralized and does not necessarily depend on a central point to interface with  
25 the objects.

### C. The Object Manager

#### 1. Self Describing Objects

30 Fig. 4 shows an example of object organization in a preferred embodiment of the present invention. Pages of the description, shows examples of an ORM Server Support Library API (Application Program Interface) supported by the object manager to access objects in a preferred embodiment of the present invention. The routines in the API of pages are used by object manager (e.g., ORM 142 of Fig. 1) to receive requests from server 140 and to prepare responses to the requests. It will be understood be  
35 persons of ordinary skill in the art that any object manager can be used in

5 conjunction with the present invention, as long as the object manager is capable of communicating with server 140 and of fulfilling GetOrm and SetOrm requests from server 140.

Fig. 10 shows layers of functions available to the object manager. A Protocol layer 1002 handles the ORM protocol, e.g., decodes the request from server 140, initiates the corresponding functions, and assembles an ORM response. Protocol layer 1002 is the lowest layer and drives all calls to the upper layers by calling "registered" functions. A Node layer 1004 handles navigation between nodes, i.e., parsing the pathname to locate the virtual node, which represents some management entity.

15 A Handle layer 1006 maps "virtual nodes" to real objects/data. Such a mapping results in a "handle." Handles are explicitly requested and released. An Aspects layer 1008 handles instances that are made up from more than one ORM tree. For example, the "statistics" Component is not a single Component in the tree, but is generated by the object manager. As another example, some attributes depend on others and cannot be modified independently, but have to be treated as a single, atomic operation. These groups of attributes within an instance are called "aspects" and the corresponding Aspect layer is provided to extract and modify groups of attributes within an instance.

20 An Attribute layer 1010 retrieves or updates a single attribute (of an aspect) and provides the meta information corresponding to this attribute. A Conversion layer handles the actual conversion of attributes between the external (ORM) and the internal (native) presentation. This layer also converts states and bitmaps to "friendly strings."

## 30 2. Web Agents

In another preferred embodiment of the present invention, the objects are not self-describing. In such an embodiment, one or more servers 140 in the system performs a "worm" function, i.e., one or more servers 140 follow object-links between objects and save all the information available concerning those objects. When a request is received from browser 130, server 140

5 sends its collected data to browser 130 (assuming that the collected data is not older than a threshold age value).

10 In summary, the present invention allows a user to manage objects by way of hypermedia, such as the world wide web. In a preferred embodiment, the objects are self-describing and respond to questions about themselves from one or more object managers. A server communicates with the object manager(s) and generates HTML from responses received from the object manager. Conventional browser software allows a user to indicate which objects he wishes to browse or change. Using a conventional hypermedia request/response protocol, the browser and server communicate to obtain  
15 information about objects and their attributes. The server also translates HTML/URLs received from the browser to requests to the object manager. Such a system allows a non-centralized object management system.

20 Other embodiments will be apparent to those skilled in the art from consideration of the specification and practice of the invention disclosed herein. It is intended that the specification and examples be considered as exemplary only, with a true scope of the invention being indicated by the following claims and equivalents.

5. Browse (readonly) and Change (read write) modes are differentiated by different URL's. For the Change mode an HTML input form is created with user interface elements dependant on the meta information provided with attributes. Dependant from this meta information, simple text input or numeric input fields, popup boxes or radio buttons are generated.
6. A submit of this HTML form results in an HTTP POST request, which appears at the managements adapter with a special URL and the list of name/value pairs from the input form. These values can be checked against the information now retrieved from the object (see above) and a ORM SetAttribute request is send to the respective object. This object initiates the intended state changes or returns an error to the adapter, which then creates a new page, which reflects the outcome of the operation.
7. Access control can be either applied by the generic HTTP adapter, filtering POST methods for example or by the called object itself using principal identifiers in the ORM request.

### 5.3 Events and Alerts

Compared to SNMP or CMIP, ORM has no event or trap mechanism. With this respect it is much closer to the NSK management protocol called SPI. Instead event and alert support is provided by another mechanism (and object) within MSF, called *Alert Facility*, which is built on top of the *Common Execution Environment (CEE)*.



## 6.0 ORM Protocol

### 6.1 General Characteristics

The ORM protocol is a simple request response protocol constructed out of lines of ASCII text and terminated by newline, to browse through the managed object and change its state. In this way, it is directly comparable to the HTTP protocol.

It is bytestream oriented, as it contains no length fields or has any fixed structure, but the individual items are separated by special characters, i.e. colons and newlines.

The logical end of a protocol unit is determined by an empty line i.e. a line with a newline character as the first character.

**Remark:** The decision for a pure ASCII protocol may be surprising, as MSF has a well defined presentation layer/protocol by IDL/PCU/GLU, but ORM is designed to support self contained object management, e.g. all manageable aspects of an object should be described by the managed object (or its manager) itself and just known by it. This implies, that a lot of human readable information has to be shipped with this protocol and using a pure ASCII protocol seemed quite natural. In addition the browsing nature of the protocol would have resulted in a quite unhandy IDL structure of unbounded sequences of any's or unbounded strings, probably with another layer of unbounded types above. Using ASCII strings simplifies the creation of protocol units. Last not least, ORM must also support the entities provided to build the MSF infrastructure itself, e.g. must also be available with and without these execution environments.

Internally the ORM protocol (ORM-P) is a tagged line protocol, as every line starts with a tag word followed by the tag value separated by a colon and terminated by an end of line character (newline). The protocol itself supports boxcarring (e.g. consolidated requests). Errors are reported inline with the data i.e., where they occur, by special error tags.

Request and response units are constructed the same way and the response is merely a "filled out" or "completed" request and as such, the information in it is self describing (i.e. need not necessarily correlated with the request). This is not true for authentication information, which is not mirrored in the response.

The ORM-P basically supports two kinds of operations:

1. get entities (OrmGet)
2. set attributes (OrmSet).

The two keywords OrmGet and OrmSet at the beginning of every sub-request describe the intended operation (i.e. are the operation tags).

In addition to these two basic operations there is a preamble required for every (consolidated) ORM packet, identifying the originators protocol version (tag OrmVersion) and in the case of a request, there is an optional tag, to pass authentication information from the client to the server (OrmWho).

## 6.2 Pathnames:

Pathnames provide the necessary navigation and identification information to locate a specific aspect of an object (or its internal entities). The different parts of the pathname usually are also the names of the entities in the generated user interface and as such, should be "friendly", descriptive names.

A pathname consists of a "/" separated list of components, which in turn may contain any printable character including whitespace.

For the current version, characters in the pathname components are restricted to the printable set of ASCII characters but may be extended to cover all printable ISO-8859-1 characters in the future (ORM-P is "8-bit clean", but this restriction is just to be more flexible in the choice of user interface construction).

There are two special component names, which follow the POSIX convention for filesystem tree navigation: "." and "..". These are only allowed in conjunction with an OrmPath tag and refer to the current node and to the parent of the current node respectively, where the "current node" must have been defined by a previous ORM sub-request in the same protocol unit (i.e. OrmGet or OrmSet). An attempt to traverse the parent of the root of the tree is treated as an erroneous pathname (compare with "cd" in a POSIX system).

## 6.3 Error Reporting (OrmError):

Errors are reported, where they are detected, i.e. the error tag (OrmError) usually appears directly behind the error causing tag in the response. This allows some kind of identification of the failed subrequest or protocol item even for large consolidated requests.

An error tag in the response also indicates, where the request was aborted. Previous sub-requests in the same ORM packet may have been processed<sup>1</sup>. An error tag (OrmError) has a constant and a variable part, where the constant part identifies the kind of error, the variable part is used for additional hints, what caused the error. As the error is reported with a context, this context provides valuable information for error explanations.

The Format of an error protocol item:

```
OrmError:<decimal errorcode> <stringcode>:<variable part*\n*
```

## 6.4 Version Support: OrmVersion:

The first line in every protocol unit must identify the highest protocol version number, this protocol unit complies to. The actual version is 1.0!

---

1. When the ORM requests are issued via the IDL interface, the error also appears as an exception with the exception detail showing the actual ORM error code.

e.g.

OrmVersion: <major> . <minor>

A server/application may respond to this request with an error VersionMismatch followed by its desired version id.

#### 6.4.1 Protocol Conformance & Protocol Errors:

Unknown tags should be ignored unless they appear at a position, where another tag is required (an attribute tag must be followed by a value tag for example). If the latter occurs, the unidentified or unexpected tag is placed in the response message followed by an error tag with "ProtocolFailure"

### 6.5 Principal Identification Information: OrmWho

For the purpose of propagating the identification of a principal causing the request or to be used with the request, ORM-P supports a protocol item to ship any kind of (encoded) principal identifier with a request in the following way:

OrmWho: <principal identifier>

Up to now, neither the <id-scheme> nor the encoding or interpretation of the "encoded-principal-identifier" are specified in any detail by ORM-P but are up to the application and require an agreement between the ORM-P client and server. This may be subject of change!

### 6.6 Browse Operation: OrmGet

The operation tag "OrmGet" has to be followed by a colon (":" - the tag separator) and usually is followed by a "pathname". As every ORM protocol item, the operation tag followed by the optional pathname has to be terminated by a newline character. This first line is followed by a list of entity type specifyers requested.

#### 6.6.1 Entity Types

The OrmGet request is followed by a list of type specifyers to describe the kinds of entities requested for browsing. Allowed types (and entity specifyers) are:

- *Object*  
requests a list of all known object links at this level. A friendly name and the link-address (NOR) is returned per object.
- *Component*  
A list of all components below the level of the path specified in the GET line is requested. A list of names is returned.
- *Attribute*

A list of attributes for the current node is requested. For every attribute its name (*Name:<name>*) and its stringified value (*Value:<value>*) is returned. If the path in the *OrmGet* line already navigates to a single attribute, the returned name is the empty string.

- *Info* (implies *Attributes*)

This type addresses the same type of entities as "*Attributes*", but here meta information in addition to the name/value pair is requested. as there is *Field* for identifying the type of input expected, *Mode* to describe the access mode (read, read-write or write) and the so called hints (*Range* and *Unit*), which can be used by the user-interface to generate a more sophisticated presentation of the attribute. These *info* fields are described below in detail in the response section.

- *<Empty>*

An empty type specifier indicates that the validity of the pathname should be checked, but no information is requested.

As all ORM protocol items, every single type specifier is terminated by a newline.

**Examples:**

```
1)OrmGet:/telnet/windows\n
   Component\n
   \n

2)OrmGet:/telnet/windows/#pty1/status\n
   Attribute\n
   OrmPath:../pty2/status\n
   Attribute\n
   \n

3)OrmGet:/telnet\n
   Object\n
   Component\n
   Info\n
   \n
```

Note: Leading blanks in front of the tag or the tag value are ignored as well as lines, whose first character is a "#".

The *OrmGet* request may be called without a pathname specification (e.g. "*OrmGet:\n*") in which case the root object itself is referenced and the only valid type specifier is "*Object*", which will return the "friendly" name of this object manger and its link address (Note: this link address may be another one, as the request was sent to, i.e. this allows redirecting the management requests to another object reference).

### 6.6.2 *OrmGet* Response

The response to a *OrmGet* request merely mirrors the request, but here the type specifiers are used as tags (to type the following entity) followed by the name of the entity (separated by ":") followed by a newline character. The name item is followed by a type dependent list of additional tagged items, describing further properties of this entity.

Although the partial responses below are listed per entity type, they appear in the same response unit in the same sequence as in the corresponding request unit, i.e. if the latter

requested entity types by the sequence "Object\nComponent\nAttribute\n", then the response will first list all available objects at this level followed by all available components followed by all available attribute/value pairs. If a requested type is not available at the specified level, an empty tag of that type (with a colon) is returned.

The response unit for an *OrmGet* request starts with the common response header (i.e. *OrmVersion:<version>*) followed by the "OrmGet: <pathname>" line followed by any number of the following constructs.

**6.6.3 Object Entities:**

Syntax:  
 <object-item> ::= "Object" [":" <string> "\n"  
   <object-link>] "\n"  
 <object-link> ::= "Link" ":" <string>

Example:

```
Object:Network Service Layer\n
Link:<obj-reference>\n
Object:Media Access Layer\n
Link:<obj-reference>\n
\n (if this is the end of the response!)
```

**6.6.4 Component Entities: (have no additional properties)**

Syntax:  
 <component-item> ::= "Component" [":" <string>] "\n"

Example:

```
Component:Configuration\n
Component:Statistics\n
\n (if this is the end of the response!)
```

**6.6.5 Attribute Entities (simple request)**

Syntax:  
 <attribute-item> ::= "Attribute" [":" <string> "\n"  
   <attribute-value>] "\n"  
 <attribute-value> ::= "Value" ":" <string>

Example:

```
Attribute:Packets sent\n
Value:1234\n
Attribute:Packets received\n
Value:4321\n
```

### 6.6.6 Attribute Entities (Info request):

Syntax:

```

<attribute-item> ::= "Attribute" [ ":" <string> "\n"
                        <attribute-info> | "\n" ]
<attribute-info> ::= <attribute-value> "\n" (cnt)
                        <attribute-mode> "\n" (cnt)
                        <attribute-field> "\n" (cnt)
                        [ <attribute-range> "\n" ]
                        [ <attribute-unit> "\n" ]
<attribute-mode> ::= "Mode:" [ "RO" | "WO" | "RW" ] [ "P" ]
<attribute-field> ::= "Field:" <ORM-fieldtype>
<attribute-range> ::= "Range:" <ORM-range-definition>
<attribute-unit> ::= "Unit:" <string>

```

Example:

```

Attribute:Status\n
Value:Running\n
Mode:RO\n
Field:String\n
Attribute:New Status\n
Value:Running\n
Mode:WO\n
Type:Enum\n
Range:Stopped,Aborted\n
....

```

Here the sequence of the different *info* tags is not relevant, but "Attribute" always starts a new property set for the next attribute. The "Range" and "Unit" tags are optional.

For "ORM-field type" and "ORM-range-definition" see below.

Errors & Exceptions specific to this request:

- NoSuchNode:

The path specified does not point to a legal virtual node. This error is only reported immediately after a "Orm[Get/Set/Path]:<pathname>" command.

- InvalidOperation:

The path specifies a node, which can not support the requested entity type, an "Attribute" request for an "Object" node or vice versa for example. This error is reported after a type specifier, listing the type specifier (without a colon) followed by a newline followed by the error tag.

### 6.7 Modification Requests: OrmSet

The set request starts with the set-tag "OrmSet:" followed by a pathname followed by a newline. The pathname at minimum must contain the name of the root e.g. "/<root-name>" (i.e. an empty pathname is not allowed!).

This line is followed by a sequence of line pairs containing the name of the attribute and its value, e.g.

Syntax:

```
<attribute-item> ::= "Attribute" [ ":" <string> "\n"
                        <attribute-value> ] "\n"
<attribute-value> ::= "Value:" <string>
```

Example:

```
Attribute:New Status\n
Value:Suspended\n
Attribute:Reset Statistics\n
Value:Yes\n
\n (if this is the end of the protocol unit)
```

### 6.7.1 OrmSet Responses:

If no error occurred, the response to the *OrmSet* request is a copy of the request itself. Otherwise an error-tag may appear somewhere in the response, and if the underlying request/response protocol permits, the response is flagged with an error indicator.

The effect of an erroneous *OrmSet* sub-request is application dependant, but it is recommended, that a *OrmSet* sub-request either succeeds completely or has no effects at all (atomicity).

Any *OrmSet* requests preceding a failed one are not affected, any subsequent requests are ignored.

### 6.7.2 Error>Returns:

- **NoSuchNode:**  
The path specified does not point to a legal virtual node. This error is only reported immediately after a "SET:<pathname>" command.
- **NoSuchAttribute:**  
The string following an "Attribute:" tag does not identify a legal attribute. The error-tag follows the "Attribute:" tag (including the string).
- **ValueOutOfRange**  
The value specified is not within the range of legal new values for this attribute. The error tag follows the value-tag line.
- **ValueInconsistent**  
The set of attribute values in the request were no consistent or contradictory
- **InvalidOperation:**  
The designated Attribute is not writeable.
- **NoPermission:**  
The access rights of the requester do not allow to set the designated attribute.(This error and the previous one may have some overlap)

## 6.8 Request Type Independent Errors:

- **ProtocolError:**  
If pairs of tagged lines are expected and the sequence of pairs is not completed or an unknown or context illegal tag is detected, this error is generated, following the erroneous tagged line.
- **InternalError:**  
An internal error in the application/server was the cause, that this request could not be completed. (allocation failure, mangled structures). This indicates a severe error at the server side.
- **BufferTooSmall:**  
The response buffer specified is too small to return the full response.
- **NoSpace:**  
Some internal buffer could not be allocated or was too small for the requested operation.

## 6.9 ORM Attribute Info Descriptions:

Within the ORM protocol there are two ways to retrieve an attribute: the short form returns just the name of the attribute and its value and the long form, returning additional meta information for every attribute, which can be used to create reasonable user interface elements by the ORM client.

The following fields appear in the extended description:

- **Field:** identifies the kind of field, this attribute should be presented in
- **Mode:** identifies applicable operations (readonly, read/write, writeonly)
- **Range:** Provides hints for input checking and for user interface generation
- **Unit:** a free form field often describing the metric of the value or scale.

### 6.9.1 ORM Field Types:

Although in principle, the ORM field type item (*Field*) allows any principal character string, the ORM support library and the user interface generator (HTML synthesizer) will only support a limited set of predefined field types, to ease the presentation of attributes. If a field type is not recognized, the default "String" is assumed.

#### 6.9.1.1 Field: Integer

The ORM protocol does not distinguish between unsigned and signed integers, e.g. every ascii string representing an integer may be prefixed by a "-" or a "+". There is also no size information in the field type. Any range restrictions have to be specified in the *Range* section.

Syntax:

Field:Integer



### 6.9.1.2 Field: Real

The field type *Real* identifies decimal floating point values. The allowed input formats are those of the POSIX 1003.2 *scanf* function for float and double values.

Syntax:

Field:Real

### 6.9.1.3 Field:HexOctet

This field type is used to display and enter binary data as pairs of hexadecimal character

Syntax:

Field:HexOctet (a sequence of hexadecimal digits)

### 6.9.1.4 Compound-Field Types:

The last set of field types allow much finer control of the input, an end user may provide to the ORM-P client side (or the client of the client...). These types are named *Enum* and *Set*, where *Enum* specifies a "one out of *m*" field and *Set* specifies a "n out of *m*" field.

Both types are only valid with an appropriate *Range* field in the hints section, where the possible alternatives must appear in a comma separated list.

These two types often transformed into "Pop-Up" menus (*Enum*) or option lists (*Set*) or similar by the user interface generator.

Syntax:

Field:Enum (single selection from "Range:")  
Field:Set (multiple choices from "Range:")

### 6.9.2 ORM Attribute Modes:

To generate reasonable user interfaces (as far as possible without object/component specific knowledge), the generator must know, whether an attribute is "read-only", "read-write" or "write-only". The latter is used to signal to the user interface, that this attribute should be only shown in "change-attribute" frames, if those are distinguished from pure browsing frames. An extension to these basic modes is provided for writable attributes to indicate, that an attribute value is mandatory, by appending the letter "M"

The different modes are simply encoded as two-letter strings followed by an optional "P", e.g.

Syntax

- RO	Read-Only
- WO[M]	Write-Only(non-null value mandatory)
- RW[M]	Read-Write(non-null value mandatory)

### 6.9.3 Range Identifiers:

The range identifier, tagged with "Range:" is used as a kind of hint (and therefore it is optional except for the compound fields) to the user interface generator, what kind of input/output field it should generate. In addition the information can be used to check any optional input and give the end-user appropriate responses or hints, if these input checks fail.

The range hints are type specific and as such different conventions are defined to specify valid ranges for an input field. The type independent convention is to separate alternatives by a comma "," and sequences by three subsequent dots "...":

#### 6.9.3.1 Range Specifications for Integer Fields:

Valid range specifications for the integer types are:

Range:1,4,8,16	valid: 1 of the values listed
Range:20...60	valid: all numbers between 20 and 60 including
Range:0...	valid: every integer including 0 (up to typemax)
Range:-20...+20	valid: every integer between -20 and +20 incl.
Range:	valid: every pos/neg integer within type

#### 6.9.3.2 Range Specifications fro Floating Point Fields:

Valid range specifications for the real types are:

Range:0.1,0.5,0.8	valid: one of the values listed
Range:0.1e-3...0.1	valid: reals between 0.0001 and 0.1

#### 6.9.3.3 Range Specifications for String Fields

If the first range value starts with a digit, the range indicates either the maximum or the range of valid string lengths. If the first character is non numeric, the range is interpreted similar to the compound *Enum* field below, i.e. one of these strings may be selected, but a different user interface element may be used (a list box). If the first character of the range string is a comma ",", this provided strings in the comma separated list are treated as examples, where the possible input is not restricted to the given alternatives. A major use of this kind of string selection is in file selection boxes.

Syntax:

Range:1...20	valid: strings with minimum length of 1 and max length of 20 characters
Range:10	valid: a string with at max 20 characters.
Range: ,file1.c,file2.c	file1.c or file2.c are valid options, but other input is also valid

#### 6.9.3.4 Range Specification for Compound Fields:

For the *Enum* and the *Set* type fields lists of alternatives are required in the range section. The comma separated list identifies the different options a user is allowed to select.

Syntax:

Range:<comma separated list of alternatives>

Example:

For Enum (choose one of)

Range:STOP, SUSPEND, ABORT

For field type Set (choose n or none of)

Range:Trace IP, Trace UDP, Trace TCP

#### 6.9.4 The Unit Specification:

The "Unit" specification is a free form string and currently not interpreted by the user interface generator. If present, it will append this string behind the value field as one would do with a unit description like "1.4 inches". Another important purpose of this field is for the use with customized object specific management pages (if used within an HTML environment). Here the unit could be used to identify an application specific type for example.

#### 6.10 Navigation Request: Path

This request extends the previous operation (*OrmGet* or *OrmSet*) to a new subtree and follows these tags in its syntax. It may appear everywhere, where a *OrmGet* or *OrmSet* tag may appear, except that it must be preceded by one of these items in the same protocol unit. It usually is only found in sequences of ORM statements resulting from a "Dump" request!

Syntax:

OrmPath:<pathname>

Semantics: Extends the previous *OrmGet* or a *OrmSet* request into another subtree within the same object.

#### 6.11 Summary of ORM Error Codes:

NoPermission: 1

The current authentication can not be used to perform the requested operation

NoSuchNode: 2

The pathname specified in a *Orm[Get/Set/Path]* request does not point to a known

node.

**NoSuchAttribute: 3**

The attribute specified in a OrmSet request could not be found

**NoSuchObject: 4**

The Object specified could not be found.

**InvalidOperation: 5**

The operation requested is not valid for this type of entity. (Example: attribute is not writable, request components of an attribute)

**ProtocolFailure: 6**

A sequence of tags was encountered, which could not be parsed and decoded.

**VersionMismatch: 7**

The object could not deal with the version of the request packet.

**CommunicationError: 8**

This is a client side error to map lower level communication errors too, if necessary.

**ValueOutOfRange: 9**

The value passed in with a set request for an attribute is not within the allowed range and could not be accepted.

**ValueInconsistent: 10**

The combination of values passed with a set request is not acceptable.

**NoSpace: 12**

The request could not be completed because of internal space restrictions in the object.

**BufferTooSmall: 13**

The response to the request exceeds the size of the response buffer provided by the underlying protocol.

**InternalError: 14**

**ApplicationError: 15**

These two errors are used to report back implementation problems like corrupt data structures, where the `InternalError` usually is generated by the ORM support library, the `ApplicationError` instead is issued by the higher "application" layers.

### 3.1 ORM Protocol Layer And Upcall Interfaces

This section was generated from <stdin> by CDOC on Sun Jan 29 17:00:50 1995.

#### ORM Application Context

Application Server Capsules may serve different kind of requests and therefor may have multiple domains of objects to be managed listening on multiple ports. Following the ORM model, this may result in multiple parallel independant trees.

The ORM parser supports this by maintaining an application context, which has to be passed to the protocol layer to handle a request (there is also an opaque *call-context*, which may be passed to the protocol layer, but this isn't interpreted by the ssl).

The application context contains beside (an opaque pointer) to the (virtual) root of the virtual tree, mainly a list of tree/application specific function pointers. Before the first request can be passed on to the ORM protocol layer, this context has to be established with the ORM SSL via a call to `ORM_ContextInitialize`.

Accordingly there exists a function to inform ORM that this application context is not needed anymore (release).

The following lists the function prototype definitions for actual functions to be provided, when establishing a context.

*Note:* Some functions are defined to return pointers to character strings (`ORM_String`). If the ORM protocol handler is used it is guaranteed, that the same function will not be called, before the string is copied or otherwise not needed anymore. This allows the use of a single private string buffers per function, if necessary.

#### 3.1.1 Authentication

The following list of functions are included to enable an application to maintain its own authenticated context. The ORM protocol just allows to forward some authentication related information from the client to the server (WHO...). This is passed on to the application layer as is, if encountered by the parser. The actual meaning of this data is application and user interface dependant.

#### 3.1.2 Function Type `ORM_AuthenticateFunc`

Performs any necessary authentication or preparation of authentication structures. Usually, the authentication information is used to setup some context in the call-context, which is passed to the node/handle layer upcalls. It is up to the application layer to free/clear such context after return from the protocol layer.

**Declaration:**

```
typedef CRM_Status (*ORM_AuthenticateFunc) (
    CRM_AppCallContextDef  callcontext, /* in */
    CRM_String              authstring  /* in */
)
```

**Fields:**

<i>callcontext</i>	An opaque pointer to any kind of context, the caller has established. This passed to the node and handle layer.
<i>authstring</i>	The string, the client passed in his request, if any. Usually uid:passwd
<i>status</i>	CRM_ENoError: if successfull, CRM_EPermissionDenied, if authentication unknown.

**3.1.3 ORM\_AuthFuncDef**

This structure is used to pass the Authentication function to ORM\_ContextInitialize

**Declaration:**

```
typedef struct CRM_AuthFuncTag {
    CRM_AuthenticateFunc  auth;
} CRM_AuthFuncDef;
```

**3.1.4 Virtual Node & Tree Function Types**

The following list of functions (function types) are used to access the virtual tree of components, attributes and linked objects. They usually don't deal with application specific data.

**3.1.5 Function Type ORM\_NodeLookUpFunc**

This is the central function for the traversal of *the* tree .

Returns an opaque pointer to a virtual node, which may subsequently be called to retrieve properties or children of specific types.

**Declaration:**

```
typedef CRM_Status (*ORM_NodeLookUpFunc) (
    CRM_AppCallContextDef  callcontext, /* in */
    CRM_AppNodeDef         root,        /* in */
    CRM_String              pathname,    /* in */
    CRM_AppNodeDef         *node,       /* out */
    CRM_NodeTypeDef         *nd_type     /* out */
);
```

**Fields:**

*callcontext* is an opaque pointer to the application specific call context provided with the Do\_Request function.

*root* Opaque Pointer to root of virtual tree. This may be NULL, and is taken from the application context.

*pathname* is a / separated list of component names optionally preceded by the name of the object (e.g. if the first component matches the roots object name, strip it, else take the first component to be a child under the applications root). Support for *unix* style directory navigation *.* and *..* is highly recommended/required. A pathname of *..* applied to the root with request type *Object* should return the root name and the actual servers link address (NOR)

*node* The opaque node pointer, if found

*nd\_type* The ORM\_NodeType of the node found

*return* ORM\_ENoError in case of success, or any other ORM error in case of failure.

### 3.1.6 Function Type ORM\_NodeChildNextFunc

Used to subsequently scan the children of a single parent. Returns the next child of type *type* of parent *parent*, which logically follows the child returned by the previous call to *NodeChildNext()*, now passed in as *lastchild*. E.g. If *lastchild* is set to NULL the logically first child of this parent is requested. If there are no children (of the requested type), then NULL must be returned with ORM\_Status set to ORM\_NoError.

Declaration:

```
typedef ORM_Status (*ORM_NodeChildNextFunc) (
    OPM_AppCallContextDef callcontext, /* in */
    ORM_AppNodeDef parent, /* in */
    ORM_AppNodeDef lastchild, /* in */
    ORM_NodeTypeDef type, /* in */
    OPM_AppNodeDef *child, /* out */
    ORM_String *name /* out */
);
```

Fields:

*callcontext* is an opaque pointer to the application specific call context provided with the Do\_Request function.

*parent* Opaque pointer to the virtual parent node.

*lastchild* Opaque pointer to the last child returned by a call to this function (in this request), or NULL to request the first child.

*type* The type of entity, which is requested (ORM\_ObjectType, ORM\_ComponentType, ORM\_AttributeType or ORM\_AnyType).



*node* Pointer where to store the reference to the node found

*name* Pointer to name of node found.

*returns* status value. Possible status values, see below!

### 3.1.7 Function Type ORM\_NodeChildByNameFunc

The little sister of ORM\_NodeLookUp. Looks for a child with name *childname* directly under the given parent *parent*. This function is primarily used within the processing of Set- Attribute requests. If there is no child with this name, return NULL and an error status (see below)

Declaration:

```
typedef CRM_Status (*CRM_NodeChildByNameFunc) (
    CRM_AppCallContextDef callcontext, /* in */
    CRM_AppNodeDef parent, /* in */
    CRM_String childname, /* in */
    CRM_AppNodeDef *child, /* out */
    CRM_NodeTypeDef *child_type /* out */
);
```

Fields:

*parent* Opaque pointer to the virtual parent node.

*childname* Name of the child (attribute), i.e. every printable char except '/'

*child* Pointer where to store the reference to the node found

*child\_type* Pointer to type of node found.

*returns* ORM\_ENoError if child was found, else ORM\_ENoSuchNode.

### 3.1.8 Function Type ORM\_NodeTypeGetFunc

returns the type (enum ORM\_NodeTypeDef) of the given node.

Declaration:

```
typedef CRM_NodeTypeDef (*CRM_NodeTypeGetFunc) (
    CRM_AppNodeDef node /* in */
);
```

Fields:

*node* a pointer to a virtual node

*returns* a valid type or ORM\_NodeTypeUnknown.

### 3.1.9 Function Type ORM\_NodeNameGetFunc

returns the name (ORM\_String) of the given node.

Declaration:

```
typedef ORM_String (*ORM_NodeNameGetFunc) (
    ORM_AppNodeDef node /* in */
);
```

Fields:

*node* a pointer to a virtual node

*returns* a valid null terminated string of characters or NULL

### 3.1.10 Function Type ORM\_NodeNotFoundTrapFunc

This function is kind of special by providing the application layer a chance, if the lookup of a node failed, to create that node.

Normally, referencing a non-existent node in the pathname of an ORM request is treated as an error, except this is an internal ORM restore request. Reloading an ORM tree into an application may encounter subtrees, which were dynamically created by the application during a previous run (usually via a *New* subtree).

This function is totally application dependant and is not covered by the ORM-SSL other than via this hook.

Declaration:

```
typedef ORM_Status (*ORM_NodeNotFoundTrapFunc) (
    ORM_AppNodeDef parent, /* in */
    ORM_String name, /* in */
    ORM_RequestTypeDef request, /* in */
    ORM_AppNodeDef *newnode /* out */
);
```

Fields:

*parent* Reference to parent node

*name* Name of node not found under this parent.

*request* Kind of ORM request (get/set/dump/restore) causing this lookup failure.

*newnode* Where to store the reference to the new node, if one was created.

*returns* ORM\_ENoError, if a node with the given name was created else ORM\_ENoSuchNode.

### 3.1.11 Structure ORM\_NodeFuncDef

This structure bundles the virtual tree related functions for passing to ContextInitialize

Note: The ORM\_NodeNotFoundTrapFunc is not included in this function array, because it is application special anyway and must be passed explicitly, see ContextInitialize()

Declaration:

```
typedef struct ORM_NodeFuncTag {
    ORM_NodeLookupFunc      lookup;
    ORM_NodeChildNextFunc   childnext;
    ORM_NodeChildByNameFunc childbyname;
    ORM_NodeTypeGetFunc     typeget;
    ORM_NodeNameGetFunc     nameget;
};
ORM_NodeFuncDef;
```

### 3.1.12 Application Handles

The following two function types are used to link the virtual nodes in the tree to (parts of) actual application data instances, visible to the ORM support layer as opaque handles. When an application handle is requested from the application layer, *real things* happen to start and it is assumed, that the instances are valid and available, until explicitly released by the ORM layer. The handles together with the aspect (identifying the type of handle to the application) will be passed to the application specific functions, when actual values have to be accessed (either for *get* or *set*). If these functions are not set in the ORM context, NULL will be passed into those calls for both, the handle and the handleclass.

### 3.1.13 Function Type ORM\_HandleGetFunc

Request (and lock) an actual handle (pointer to an application level instance) and a handleclass based on the current virtual node and the current principal.

Declaration:

```
typedef void (*ORM_HandleGetFunc) (
    ORM_AppCallContextDef callcontext, /* in */
    ORM_AppNodeDef       node,         /* in */
    ORM_RequestTypeDef   op,           /* in */
    ORM_AppHandleDef     *handle,      /* out */
    ORM_AppAspectDef     *aspect      /* out */
);
```

Fields:

<i>callcontext</i>	is an opaque pointer to the application specific call context provided with the Do_Request function.
<i>node</i>	Pointer to current Node.
<i>op</i>	Operation Code, e.g. ORM_Request
<i>handle</i>	Pointer, where to store the handle reference

*aspect* Pointer, where to store the aspect reference

*returns* ORM\_ENoError if no error occurred or any of the ORM error codes.

### 3.1.14 Function Type ORM\_HandleReleaseFunc

Returns a given handle back to the application layer. This should be more understood as an *unlock* operation than a free!

Declaration:

```
typedef void (*ORM_HandleReleaseFunc) (
    CRM_AppCallContextDef callcontext, /* in */
    CRM_AppHandleDef handle, /* in */
    CRM_AppAspectDef aspect, /* in */
    CRM_RequestTypeDef op /* in */
);
```

Fields:

*callcontext* is an opaque pointer to the application specific call context provided with the Do\_Request function.

*handle* a handle obtained via a call to HandleGet

*aspect* Aspect as returned from HandleGet

*op* Operation Code, e.g. ORM\_Request

### 3.1.15 Function Type ORM\_ObjectLinkGetFunc

Retrieve the Object Link from a node of type Object given the node, the handle and the aspect. The standard Handle Layer functions just return the link stored in the corresponding field in the node struct.

Declaration:

```
typedef CRM_Status (*ORM_HandleObjectLinkGetFunc) (
    CRM_AppNodeDef node, /* in */
    CRM_AppHandleDef handle, /* in */
    CRM_AppAspectDef aspect, /* in */
    CRM_ObjectLinkDef *link /* out */
);
```

Fields:

*node* Reference to node of Object Type

*handle* Reference to application defined handle as returned from HandleGet

*aspect* Reference to application defined aspect as returned from Han-

*link* Location where to store the reference to the stringified link information

*returns* ORM\_ENoError if successfull, else ORM\_InvalidOperation, if the node is not of type Object

### 3.1.16 Function Type ORM\_AttributeDescrGetFunc

Retrieve the opaque reference unique to a node of type Attribute (usually the attribute descriptor), given the node, the handle and the aspect. The standard Handle Layer functions just return the pointer stored in the corresponding field in the node struct.

Declaration:

```
typedef ORM_Status_t(ORM_HandleAttributeDescrGetFunc) (
    ORM_AttrDescDef node, /* in */
    ORM_AppHandleDef handle, /* in */
    ORM_AppAspectDef aspect, /* in */
    ORM_AttrDescDef *attribdesc /* out */
);
```

Fields:

*node* Reference to node of Object Type

*handle* Reference to application defined handle as returned from HandleGet

*aspect* Reference to application defined aspect as returned from HandleGet

*attribdesc* Location where to store the reference to the attribute information

*returns* ORM\_ENoError if successfull, else ORM\_InvalidOperation, if the node is not of type Object

### 3.1.17 Structure ORM\_HandleFuncDef

This structure bundles the handle related functions for passing to ContextInitialize

Declaration:

```
typedef struct CRM_HandleFuncTag {
    CRM_HandleGetFunc      get;
    CRM_HandleReleaseFunc  release;
    CRM_HandleObjectLinkGetFunc  link;
    CRM_HandleAttributeDescGetFunc  attrib;
} CRM_HandleFuncDef;
```

### 3.1.18 Accessing Application Data: Aspects

the following group of functions (function types) has to be provided to access actual values of the application either for retrieval or for updating. All functions in this group are mandatory, if the ORM protocol layer is used.

### 3.1.19 Function Type CRM\_AspectCallGetFunc

This function retrieves an *aspect* from the application layer, e.g. a reference to a blob of native application data (a pointer to a (part of) an application data structure, or a response buffer ....). The ORM protocol layer calls this function once for every unique handle/aspect combination (and not per Attribute) within a single AttributeGet Request. If the HandleGet Function returns a different pair or there are no more attribute nodes to process, the current aspect is released!

Declaration:

```
typedef CRM_Status (* CRM_AspectCallGetFunc) (
    CRM_AppHandleDef  handle, /* in */
    CRM_AppAspectDef  aspect, /* in */
    CRM_AppDataPtrDef *current /* out */
);
```

Fields:

<i>handle</i>	Handle as retrieved from HandleGet
<i>aspect</i>	Aspect Reference, as retrieved from HandleGet
<i>current</i>	Where to store the reference to the current value (opaque)

### 3.1.20 Function Type CRM\_AspectCallInitFunc

This function requests an *aspect* container from the application layer, e.g. a reference to a blob, where new attribute values can be selectively written to to perform AttributeSet requests. In addition the application layer may return a reference to the current aspects values (cmp CallGet), which is passed unchanged to the CallSet routine. The ORM protocol layer calls this function once for every unique handle/aspect combination (and not per Attribute) within a single AttributeSet Request. If the HandleGet Function returns a different pair for a node or there are no more attribute nodes to process, the CallSet function is called (Note: AspectRelease is only called for aspects retrieved via CallGet!) The

ORM SSL Implementation of these functions copies the current values and returns a reference to this copy in *new* and a reference to the current values in *current*.

Declaration:

```
typedef CRM_Status (* CRM_AspectCallInitFunc) (
    CRM_AppHandleDef    handle,      /* in */
    CRM_AppAspectDef    aspect,      /* in */
    CRM_AppDataPtrDef   *new,        /* out */
    CRM_AppDataPtrDef   *current     /* out */
);
```

Fields:

<i>handle</i>	Handle as retrieved from HandleGet
<i>aspect</i>	Aspect Reference, as retrieved from HandleGet
<i>new</i>	Where to store the reference to the native blob to update with new attribute values (opaque)
<i>current</i>	Where to store the reference to the current aspect (opaque)

### 3.1.21 Function Type CRM\_AspectCallSetFunc

This function is called to actually apply the new attribute values for the current aspect by the application layer. It is up to the aspect/application layer, to check the values in the request structure for validity and consistency and to determine which attributes got new values (by comparison with the *current* values). In addition it is the responsibility of the aspect/application layer to deallocate any structures allocated by AspectCallInit. Only if the Set-Function is not called, the call to AspectRelease is performed.

The ORM protocol layer calls Set-function once for every unique handle/aspect combination (and not per Attribute) within a single AttributeSet Request. If the HandleGet Function returns a different pair for a node or there are no more attribute nodes to process, the CallSet function is called (Note: AspectRelease is only called for aspects retrieved via CallGet!) The ORM SSL Implementation of these functions copies the current values and returns a reference to this copy in *new* and a reference to the current values in *current*.

Declaration:

```
typedef CRM_Status (* CRM_AspectCallSetFunc) (
    CRM_AppHandleDef    handle,      /* in */
    CRM_AppAspectDef    aspect,      /* in */
    CRM_AppDataPtrDef   new,         /* in */
    CRM_AppDataPtrDef   current,     /* in */
    CRM_String           *rsdetail    /* out */
);
```

<i>aspect</i>	Aspect Reference, as retrieved from HandleGet
<i>request</i>	Where to store the reference to the native blob to update with new attribute values (opaque)
<i>current</i>	Where to store the reference to the current aspect (opaque)
<i>rsdetail</i>	Where to store a textual hint, why the call failed, if any.
<i>returns</i>	ORM_ENoError if new values could be applied successfully, else ORM_ERange.

### 3.1.22 Function Type ORM\_AspectReleaseFunc

Used to tell the application layer, that the reference retrieved via an AspectGet or AspectInit call is no longer needed anymore by the ORM layer. This function is called, when GetHandle returns a new handle aspect call within a AttributeGet processing or a conversion in an AttributeSet processing failed.

Declaration:

```
typedef void (* ORM_AspectReleaseFunc) (
    ORM_AttrHandleDef  handle,      /* in */
    ORM_AttrAspectDef aspect,      /* in */
    ORM_AttrAspectDef current,    /* in */
    ORM_RequestTypeDef reqtype     /* in */
);
```

Fields:

<i>handle</i>	Handle as retrieved from HandleGet
<i>aspect</i>	Aspect Reference, as retrieved from HandleGet
<i>current</i>	Reference to data as returned from AspectCallInit or AspectCallGet.
<i>reqtype</i>	ORM_RequestGet or ORM_RequestSet depending whether this dataptr resulted from an AspectGet or AspectInit call.

### 3.1.23 ORM\_AspectFuncDef

This function groups the function pointers of the aspect layer



**Declaration:**

```
typedef struct ORM_AspectFuncTag {
    ORM_AspectCallSetFunc  callget;
    ORM_AspectCallInitFunc callinit;
    ORM_AspectCallSetFunc  callset;
    ORM_AspectReleaseFunc  release;
} ORM_AspectFuncDef;
```

**3.1.24 Attribute Functions**

The following group of functions is called to actually perform the the single attribute Get/Set and the corresponding conversions between the applications native and the ORM (ascii) presentation.

**3.1.25 Data Structure: ORM\_AttributeInfoDef**

This structure is used to return the all the meta information and the actual value of an attribute. It is passed by reference to the application/attribute layer to be filled. Note: The string pointers do not point to valid buffers, when passed to the attribute layer!

```
<mode>  RWIP, WOP, RWIP, RWIP;
<type>  INT:(1,4,8), REAL:(32/64), STRING, HEXOCT,
        DEFINT, NOCHICE
<value> the current value in its ascii presentation
<range> Optional: The range string
<unit>  Optional: The unit string
```

**Declaration:**

```
typedef struct ORM_AttributeInfoTag {
    ORM_String  value;
    ORM_String  name;
    ORM_String  field;
    ORM_String  range;
    ORM_String  unit;
} ORM_AttributeInfoDef;
```

**3.1.26 Function Type ORM\_AttributeNativeToStringFunc**

This function converts the applications native value of an *attribute*, specified by *handle*, *aspect* and the attribute descriptor to a C-string (ORM\_String).

**Declaration:**

```
typedef CRM_Status (*CRM_AttributeNativeToStringFunc) (
    CRM_AppHandleDef      handle,      /* in */
    CRM_AppAspectDef      aspect,      /* in */
    CRM_AppAttribDescrDef attribdescr, /* in */
    CRM_AppDataPtrDef     dataptr,     /* in */
    CRM_String            *strvalue    /* out */
);
```

**Fields:**

- handle*                    Handle as obtained from the last call to HandleGet or NULL.
- aspect*                    Aspect as returned from the last call to HandleGet or NULL
- attribdescr*                Attribute Descriptor as returned form *AttribDescrGet* call.
- dataptr*                    Opaque Pointer as returned from *AspectGetCall*.
- strvalue*                    Where to store the reference to the converted value.
- returns*                    ORM\_ENoError (Null) if conversion was successful, else a valid  
ORM Error return code.

**3.1.27 Function Type ORM\_AttributeNativeToInfo**

This function performs the same as the previous function *ORM\_AttributeNativeToString*, except that it also provides the additional meta information to this attribute, as far as available.

**Declaration:**

```
typedef CRM_Status (*CRM_AttributeNativeToInfoFunc) (
    CRM_AppHandleDef      handle,      /* in */
    CRM_AppAspectDef      aspect,      /* in */
    CRM_AppAttribDescrDef attribdescr, /* in */
    CRM_AppDataPtrDef     dataptr,     /* in */
    CRM_AttributeInfoDef  info        /* in, indirect out */
);
```

**Fields:**

- handle*                    Handle as obtained from the last call to HandleGet or NULL
- aspect*                    Aspect as returned from the last call to HandleGet or NULL
- attribdescr*                Attribute Descriptor as returned form *AttribDescrGet* call.
- dataptr*                    Opaque Pointer as returned from *AspectGetCall*.
- extref*                    Pointer to structure, where to store the string references.

*returns* ORM\_ENoError (Null) if conversion was successful, else a valid ORM Error return code.

### 3.1.28 Function Type ORM\_AttributeStringToNativeFunc

This function converts an ORM\_String value for an attribute into the applications native presentation. The conversion should be done into the structure (dataptr) obtained by a call to AspectCallInit().

Declaration:

```
typedef ORM_Status (*ORM_AttributeStringToNativeFunc) (
    ORM_AppHandleDef handle, /* in */
    ORM_AppAspectDef aspect, /* in */
    ORM_AppAttributeDescrDef attribdescr, /* in */
    ORM_AppDataDef dataptr, /* in, indirect out */
    ORM_String strvalue /* in */
);
```

Fields:

*handle* Handle as obtained from the last call to HandleGet or NULL.

*aspect* Aspect as returned from the last call to HandleGet or NULL

*attribdescr* Attribute Descriptor as returned from AttribDescrGet call.

*dataptr* Opaque Pointer as returned from AspectGetCall.

*strvalue* New value as a C-String (ascii).

*returns* ORM\_ENoError (Null) if conversion was successful, else a valid ORM Error return code.

### 3.1.29 Structure ORM\_AttributeFuncDef

This structure bundles the attribute related functions for passing to ContextInitialize

Declaration:

```
typedef struct ORM_AttributeFuncTag {
    ORM_AttributeStringToNativeFunc stringtonative;
    ORM_AttributeNativeToStringFunc nativetostring;
    ORM_AttributeNativeToInfoFunc infotostring;
}; ORM_AttributeFuncDef;
```

### 3.1.30 Structure ORM\_ContextDef

This is an internal structure to ORM and opaque to the application layer. It stores the function pointers and the information of the root node.

Note: This structure and the related procedure definitions may change

*authfuncs* Pointer to list of authentication related functions or NULL, if no application specific authentication is needed.

*notfound* No description

### 3.1.32 ORM\_ContextRelease

Release an Application Context.

Prototype:

```
void
ORM_ContextRelease( ORM_ContextDef context);
```

Parameters:

*contxt* Pointer to application context as obtained from ORM\_ContextInitialize

### 3.1.33 ORM\_DoRequest

This function calls the protocol layer to parse an ORM request received and act on it accordingly via upcalls to functions in the application context, i.e. this is the function to be dispatched, when ORM requests are received on a server port.

Prototype:

```
ORM_Status
ORM_DoRequest(
    ORM_ContextDef appctx,
    ORM_ContextDef callctx,
    ORM_RequestDef request,
    Long reqlen,
    ORM_ResponseDef response,
    Long maxresp
);
```

Parameters:

*appctx* The application context reference as returned from ORM\_ContextInitialize.

*callctx* An arbitrary call context (reference) maintained by the application layer and passed to the authentication, node and handle upcalls.

*request* Pointer to received ORM request

*reqlen* Length of request buffer in bytes

*response* Pointer to allocated response buffer

*maxresp*

Reference to maximum response buffer length in bytes, on return, points to number of bytes used in response buffer

## 3.2 ORM Node Layer

This section was generated from `<stdin>` by CDOC on Fri Jan 27 19:59:34 1995.

The ORM Node layer adds another level of ORM application/server support, as it actually maintains a tree structure to access the application level datastructures.

This level is accessed from the application/server level via the *ORM\_Node...* functions to actually build/destroy the tree of objects, components and attributes.

On the other side it is called from the protocol level and frees up the application to provide the appropriate functions for navigation and name space/entity management itself.

### 3.2.1 Application Handles

The nodes of the node layer provide a tree structured view to application/server level data, but they (usually) do not contain the actual data. A link to the actual instances of application level data is maintained by *handles* and *aspects*. Both are opaque to the ORM-Node level but are interpreted at the layer on top of ORM-Node. Typically the handle is a pointer to some application level instance, and the aspect is a pointer, index or type identifier, which identifies the type of the instance

### 3.2.2 The ORM\_Node Structure

Instances of this structure maintain the tree of virtual components, objects and attributes

Every node has a name and a type, identifying the three different entity types: Object, Component or Attribute. Object and Attribute nodes are leaf nodes, e.g. they can't have children.

In addition, every node has a parent and a next pointer, to link the actual tree structure. Only component nodes have a pointer to the list of children.

Object Nodes have an additional attribute, called the Link (or Link-Info which usually is a stringified NOR).

Attribute Nodes reference a single attribute by, which is characterize by additional information like

- a value type, which describes the kind of value e.g. integer (different sizes), real (sizes!), string, *single-selection* or *multiple choice*
- a value mode, specifying this attribute as read-only read-write, write-only or persistent.
- *hints* section, which contains additional information for use by the user-interface creator, e.g. valid ranges for this value and a unit string. Both values are optional.

The nodes provide a tree structured view to application/ server level data, but they (usually) do not contain the actual data.

### 3.2.3 Struct NodeDef

**Declaration:**

```

typedef struct ORM_NodeTag {
    ORM_NodeTypeDef    type;
    short              flag;
    char               *name;
    struct ORM_NodeTag *parent;
    struct ORM_NodeTag *next;
    ORM_AppHandleDef   handle;
    ORM_AppAspectDef   aspect;
    union {
        struct {
            struct ORM_NodeTag *first;
            struct ORM_NodeTag *last;
        };
        struct {
            void *descr;
            char *attrib;
        };
        struct {
            char *link;
            char *direct;
        };
    };
};
#define ORM_NodeDef
    
```

**Fields:**

<i>type</i>	identifies the type of entity, this node describes, i.e. ORM_NodeType[Object, Component Attribute, Unknown]
<i>flag</i>	Internal use
<i>name</i>	The name of the node (object, component or attribute name)
<i>parent</i>	pointer to the parent in the tree, NUL for the root of the tree.
<i>next</i>	pointer to next sibling in chain. This defines the order in which nodes of a given type appear in the response
<i>handle</i>	an opaque pointer for use by the upper layers
<i>aspect</i>	another opaque identifier for use by the upper layers
<i>u.comp</i>	union variant for component nodes
<i>u.comp.first</i>	pointer to first child of this component node
<i>u.comp.last</i>	pointer to last child of this component node
<i>u.attrib</i>	union variant for attribute nodes
<i>u.attrib.descr</i>	opaque pointer for use by upper layers

*u.object.link* pointer to stringified link-address of this object (NOR), e.g. the *hyperlink*

### 3.2.4 ORM\_NodeCreate

Creates a new unlinked node. Usually only used by convenience functions and to create the root node.

Prototype:

```
ORM_NodeDef
ORM_NodeCreate( ORM_String      name,      /* in */
                ORM_NodeTypeDef type      /* in */
                );
```

Parameters:

*name* The name of this node (for navigation)

*type* The type of this node. This type also determines which functions - can be applied to this node later on.

### 3.2.5 ORM\_NodeDelete

Deletes the given node and all its children e.g. returns the space allocated Note: if the nodes parent pointer is not NULL, the node will not be deleted.

Prototype:

```
int
ORM_NodeDelete( ORM_NodeDef node /* in */
               );
```

Parameters:

*node* The node (and the subtree) to delete

### 3.2.6 ORM\_NodeAttach

Attaches a node (and its subtree) into an existing tree as a new subtree. Every node (subtree) is in at most 1 tree!

Prototype:

```
int
ORM_NodeAttach( ORM_RelationDef relation, /* in */
               ORM_NodeDef      relative, /* in */
               ORM_NodeDef      subtree  /* in */
               );
```

Parameters:



*relation* : Flag either ORM\_NodeSibling or ORM\_NodeChild, specifying the role of the *relative* node, e.g. its a sibling or its the parent of the subtree to attach. If its a parent, the new node will be attached at the end of all children, if its a sibling, it will be placed right before this child.

*relative* : an existing node, either parent of sibling

*subtree* : No description

### 3.2.7 ORM\_NodeDetach

Detaches a subtree from the current root tree. This always has to be called, before a subtree is actually deallocated. The subtree may also be reattached in the same tree again after this call

Prototype:

```
void
ORM_NodeDetach(ORM_NodeDef subtree /* in */)
{ }
```

No parameter descriptions are available.

### 3.2.8 ORM\_NodeHandleSet

Sets the handle in the given node (see also ORM\_Node<convenience functions>)

Prototype:

```
void
ORM_NodeHandleSet(ORM_NodeDef node, /* in */
                 ORM_AppHandleDef handle /* in */)
{ }
```

Parameters:

*node* : Reference to node structure of any type.

*handle* : Reference to opaque handle.

### 3.2.9 ORM\_NodeHandleGet

Retrieves the handle from a given node

Prototype:

```
int
ORM_NodeHandleGet(ORM_NodeDef node, /* in */
                 ORM_AppHandleDef *handle /* out */)
{ }
```

No parameter descriptions are available.

### 3.2.10 ORM\_NodeAspectSet

Sets the aspect in the given node (see also ORM\_Node<convenience functions>).

Prototype:

```
void
ORM_NodeAspectSet(ORM_NodeDef node, /* in */
                  ORM_AppAspectDef aspect /* in */)
;
```

Parameters:

- node* Reference to node structure of any valid node type.
- aspect* Reference to opaque aspect description.

### 3.2.11 ORM\_NodeAspectGet

Retrieves the aspect from a given node

Prototype:

```
/*
 * ORM_NodeAspectGet(ORM_NodeDef node, /* in */
 *                  ORM_AppAspectDef *aspect /* out */)
 */
;
```

No parameter descriptions are available.

### 3.2.12 ORM\_NodeAttributeDescrSet

Sets the attribute description of an attribute node

Prototype:

```
/*
 * ORM_NodeAttributeDescrSet(ORM_NodeDef node, /* in */
 *                            ORM_AppAttribDescrDef attrib /* in */)
 */
;
```

Parameters:

- node* Reference to node structure of type Attribute.
- attrib* Reference to opaque attribute description

### 3.2.13 ORM\_NodeAttributeDescrGet

Cets the attribute description of an attribute node

Prototype:

```

int
CRM_NodeAttributeDescrGet ( CRM_NodeDef node, /* in */
                           CRM_AppAttribDescrDef *attrib /* out */
);
    
```

No parameter descriptions are available.

### 3.2.14 ORM\_NodeObjectLinkSet

Sets the link of an object node

Prototype:

```

int
CRM_NodeObjectLinkSet ( CRM_NodeDef node, /* in */
                        CRM_String link /* in */
);
    
```

Parameters:

- node* Reference to node structure of type Object.
- link* Stringified version of the address/nor to call this object.

### 3.2.15 ORM\_NodeObjectLinkGet

Gets the linkaddress of an object node

Prototype:

```

int
CRM_NodeObjectLinkGet ( CRM_NodeDef node, /* in */
                        CRM_String *link /* out */
);
    
```

No parameter descriptions are available.

### 3.2.16 ORM\_NodeObjectAdd

for an explanations of paramters, see above. Return created node if operation succeeded else NULL.

Prototype:

```

CRM_NodeDef
CRM_NodeObjectAdd( CRM_RelationDef relation, /* in */
CRM_NodeDef       relative, /* in */
CRM_String        name, /* in */
CRM_AppHandleDef  handle, /* in */
CRM_AppAspectDef  aspect, /* in */
CRM_String        linkaddr /* in */
);
    
```

No parameter descriptions are available.

### 3.2.17 ORM\_NodeComponentAdd

Prototype:

```

CRM_NodeDef
CRM_NodeComponentAdd( CRM_RelationDef relation, /* in */
CRM_NodeDef         relative, /* in */
CRM_String          name, /* in */
CRM_AppHandleDef    handle, /* in */
CRM_AppAspectDef    aspect /* in */
);
    
```

No parameter descriptions are available.

### 3.2.18 ORM\_NodeAttributeAdd

Prototype:

```

CRM_NodeDef
CRM_NodeAttributeAdd( CRM_RelationDef relation, /* in */
CRM_NodeDef         relative, /* in */
CRM_String          name, /* in */
CRM_AppHandleDef    handle, /* in */
CRM_AppAspectDef    aspect, /* in */
CRM_AppAttribDescrDef attribdescr /* in */
);
    
```

No parameter descriptions are available.

### 3.3 ORM Aspect Layer

This section was generated from `<stdin>` by CDOC on Sun Jan 29 17:00:51 1995.

The ORM aspect layer adds another level of ORM application/server support on top of the ORM Node/Handle layer, and supports the retrieval and modification of aspects, i.e. groups of attributes from or into application data structures, once those have been registered with this layer.

This level has no additional (down-call) functions but defines data structures to be provided by the application layer. These are then accessed/used by the aspect upcall functions, if those have been registered with the ORM protocol layer.

The Aspect layer implementation of the ORM-SSL works as follows:

On AspectCallGet requests, just a pointer is returned which points at offset bytes (as set in the aspect descriptor) from the beginning of the handle. On AspectCallInit calls, a copy of the aspect, e.g. size bytes from the area pointed to by handle, starting from offset, is taken into a private memory area. This copy is then passed to the Attribute conversion routines to write the new values into. On AspectCallSet calls, the application level set function as denoted by the aspect descriptor is called and the private copy (request structure) is released afterward.

#### 3.3.1 Function Type ORM\_AspectSetFunc

This function is called from the aspect layer to actually apply the new attribute values to the application layer and/or initiate the requested state changes. This function usually should not block, e.g. should not wait until the initiated state change is completed. Any kind of intermediate state should instead be visible to a client on request (i.e. not STOPPED -> STARTED, but STOPPED -> STARTING -> STARTED, if starting implies a heavier operation).

Declaration:

```

CRM_Status
typedef (*ORM_AspectSetFunc) (
    CRM_AppHandleDef      handle,          /* in */
    CRM_AspectDescriptorDef aspect,        /* in */
    CRM_AppDataPtrDef     request,         /* in */
    CRM_AppDataPtrDef     current,         /* in */
    CRM_Status            *errortext      /* out */
);

```

Fields:

<i>handle</i>	the handle as returned from HandleGet
<i>aspect</i>	Reference to the aspectdescr.
<i>request</i>	Copy of the aspect as described by the aspectdescr updated with new values.

*current* Reference to aspect within handle

*errortext* Where to store a pointer to a short textual description if the requested values could NOT be applied.

*returns* ORM\_ENoError if all new values could be applied, or  
 ORM\_EParameterList if parameter set is inconsistent or  
 ORM\_EMissingAttribute if a mandatory attribute is NULL.

### 3.3.2 The ORM\_AspectDescrDef

This descriptor maintains information about the application data structure (usually references by the ORM\_AppHandle) or parts of it. It describes the binary size, the offset within the handle, and contains pointers to functions to actually retrieve or modify this aspect of the application instance.

*Note:* It is currently open, whether there should be a procedural interface to set up the aspect descriptor instead of providing a structure type definition to be passed initialized by the application code.

Declaration:

```

type def struct ORM_AspectDescrDef {
    char *name;
    int offset;
    int size;
    long flag;
    ORM_AspectSetFunc setf;
    long appid;
    void *appext;
} ORM_AspectDescrDef;
    
```

Fields:

*name* Pointer to name string, for identification mainly.

*offset* The offset in bytes within the instance, where this aspect starts. This usually is the offset of a sub structure in the instance.

*size* The size in bytes of the instance, the application handle pointer points to. For set-requests, the container for the new value is created by copying the handle, and inserting the new values in it.

*flag* If set to ORM\_AspectGetIndirect, the offset indicates the offset to a pointer, pointing to another structure of the above size.

*setf* Pointer to function, which is called to apply (a set of) new values to an application instance.

*appext* any value of pointer size the application wants to store with the aspect. This may be used to store a create\_aspect function pointer.

*appid* Opaque identifier, which may be used by the applications layer

### 3.4 ORM Attribute Layer.

This section was generated from <stdin> by CDOC on Fri Jan 27 19:59:34 1995.

The ORM Attribute layer adds another level of ORM application/server support on top of the ORM node layer, by providing (list of) attribute descriptors, which simply initialized by the application code, allowe automatic conversion and generation of the attribute meta information, requested by the ORM protocol layer.

The implementation of the attribute layer in the ORM SSL assumes, that it is converting to and from a binary blob of data, identified by the (lower level) aspect descriptor. The goal of this layer is to reduce the coding effort needed by the application writer at this layer, just to provide some initialized descriptors and pass them to the ORM SSL via single calls per every instance created.

#### 3.4.1 The ORM\_AttributeDescriptorDef

This data structure describes a single attribute, e.g. its native type and mode, its size, pointers to conversion functions. In addition it maintains hooks for preset meta-info like *Unit* and *Range*.

Declaration:

```
typedef struct ORM_AttributeDescrTag {
    ORM_String          name;
    ORM_AttribTypeDef   datatype;
    ORM_AttribModeDef   accessmode;
    ORM_String          range;
    ORM_String          unit;
    size_t              offset;
    size_t              size;
    ORM_ConverterNativeToStringFunc native tostring;
    ORM_ConverterStringToNativeFunc stringtonative;
    ORM_AppConverterArgDef   convarg;
} ORM_AttributeDescrDef;
```

Fields:

<i>name</i>	The name of the attribute.
<i>datatype</i>	The type of data of this attribute (ORM_AttributeTypeDef). This is a superset of the data types, the ORM protocol defines and used to determine implicit conversion routines.
<i>mode</i>	The allowed access modes of this attribute out of ORM_AttribMode values, e.g. read-only, write- only, read-write.
<i>range</i>	A string describing the allowed ranges for new values for read-write or write-only attributes only. This is a <i>ORM hint</i> , and as such optional
<i>unit</i>	A unit string (usually <i>ms</i> , <i>Mb</i> , etc.) which may be used by object specific user interface generators in any way, and by default if

present is placed behind the attribute value. This is also an *ORM hint* and as such optional.

*conversion function* A function pointer to an application specific conversion function, to convert between native and ORM presentations. *Note:* This is not to be confused with the similar functions of the *ORM\_Context* structure. For the <conversion-function> to be called, the *ORM\_Node* conversions functions have to be setup in the *ORM\_Context*.

*conversion-arg* An opaque pointer to any argument, the conversion function may need to convert this attribute.

### 3.4.2 ORM\_AttributeCreate

This function combines several actions required to register an attribute of a (new) instance with the ORM SSL, i.e. it creates an attribute node under the given parent (which must be of *ORM\_NodeTypeComponent*) and attaches the attribute description and the handle information to it.

Prototype:

```
int
ORM_AttributeCreate (
    ORM_NodeDef          relative, /* in */
    ORM_RelationDef      relation, /* in */
    OPM_AttributeDescrListDef attribdescr, /* in */
    OPM_AspectDescrListDef aspectdescr, /* in */
    ORM_AppHandleDef     handle, /* in */
    ORM_NodeDef          *new      /* out */
);
```

Parameters:

<i>relative</i>	pointer to relative node. If <i>relation</i> is set <i>ORM_NodelsParent</i> , then this has to be a node of <i>ORM_NodeTypeComponent</i> . If <i>relation</i> is set to <i>ORM_NodelsSibling</i> , then this node can be of any valid node type.
<i>relation</i>	Either <i>ORM_NodelsParent</i> , if the node <i>relative</i> should be the parent of the new attribute node, or <i>ORM_NodelsSibling</i> , if the new attribute node should be inserted after the <i>relative</i> node as a sibling.
<i>attribdescr</i>	No description
<i>aspectdescr</i>	No description
<i>handle</i>	Pointer to the application instance this attribute belongs to or <i>ORM_HandleInherit</i> (-1), if the handle should be taken from the parent (or its parent and so on).
<i>new</i>	Pointer to new attribute node or NULL on failure.



### 3.4.3 ORM\_AttributeDestroy

This function detaches the attribute node from the tree of nodes if any, deletes the node structure and deletes any depending structures, i.e. the attribute descriptor.

In the current implementation this function maps directly to `ORM_NodeDestroy`, but nevertheless this function should be called for attribute nodes created with functions of this layer to be able to deallocate any dynamic memory.

Prototype:

```
int
ORM_AttributeDestroy(   ORM_NodeDef attrnode);
```

Parameters:

*attrnode*            Pointer to attribute node.

### 3.4.4 ORM\_AttributeListCreate

This is another convenience functions to add a list of attributes to a component. The given node must be a of component type and is used as the parent for the new list of attributes ( which is appended to the end of the list of child-nodes). The pointer to the attribute descriptor now points to an array of those descriptors, where the end of the array is marked by a descriptor whose name pointer is NULL.

Prototype:

```
int
ORM_AttributeListCreate(
    ORM_NodeDef          parent,          /* in */
    ORM_AppHandleDef     handle,         /* in */
    ORM_AspectDescrDef   aspectdescr,    /* in */
    ORM_AttributeDescrListDef attrdescrlist, /* in */
    long                 attrcount      /* in */
);
```

Parameters:

*parent*            Pointer to an existing component node, who is the parent node of all newly created attribute nodes.

*handle*            Pointer to the application instance, all attribute belongs to or `ORM_HandleInherit (-1)`, which indicates, that the actual handle is determined by the parent (which again may have its handle set to `ORM_HandleInherit!`)

*aspectdescr*       No description

*attrdescrlist*    Pointer to an array of `ORM_AttributeDescr`, with `name=NULL` in the last element if *attrcount* is < 0.

*attrcount*

The number of attribute descriptors in the list or the number of initial attributes from this list to attach to this node or -1, if the end of the list (array) should be determined by a NULL nodeinfo pointer.

### 3.5 ORM Attribute Conversion Support

This section was generated from <stdin> by CDOC on Sun Jan 29 18:13:38 1995.

This part of the ORM Server Support Layer provides functions for converting generic ORM data types between their native (binary) and the ORM (ASCII) presentation. The interface between the attribute and the conversion layer is defined by two function types, one for converting application native data into an ORM representation, one to convert ORM attribute value strings into the applications native presentation. Beside the conversion functions provided by the ORM-SSL, every application may provide its own special converters as long as their interfaces conform these function types.

#### 3.5.1 Function Type ORM\_ConverterNativeToString

This function is called to convert a single native value into its string representation. In addition to the value string it may generate the range and unit strings, if the pointer values passed are non-null. If the converter function returns NULL in these pointers, the lower (attribute) layer may provide default strings if any.

*Memory Allocation:* The memory to hold the converted string value(s) has to be provided by the converter function. It is reasonable to use static memory for this purpose, because before the converter function is called again, the ORM protocol layer will copy the strings returned.

Declaration:

```
typedef CRM_Status (* CRM_ConverterNativeToStringFunc) (
    CRM_AppDataForDef      ptr,           /* in */
    size_t                 size,         /* in */
    CRM_AttributeDescrListDef datatype,  /* in */
    CRM_AppConverterArgDef convarg,     /* in */
    CRM_String             *strvalue,    /* out */
    CRM_String             *strrange,    /* out */
    CRM_String             *strunit     /* out */
);
```

Fields:

<i>ptr</i>	Address of native data element (e.g. attribute value)
<i>size</i>	Byte-size of data element
<i>datatype</i>	One of the CRM_AttributeTypes identifying the type of the native data element and its mapping to an ORM Protocol data type (??is this overloaded ??)
<i>convarg</i>	Any kind of argument (pointer) for this converter (as provided with the attribute descriptor for ex.)
<i>strvalue</i>	Where to store the pointer to the converted value string.
<i>strrange</i>	Where to store the reference to the optional range string.

*strunit*                      Where to store the reference to the optional unit string.

### 3.5.2 Function Type ORM\_ConverterStringToNative

This function is called to convert a single ORM string value into its native presentation. The pointer for the result usually points into a set of different attributes, e.g. an aspect, which usually is a (partial) copy of some application data instance.

**Memory Allocation:** The destination pointer provided references some valid memory (e.g. an aspect), but for references (the native value is a C-string for ex.), there is usually not enough space for the referenced value. This space must be allocated/provided by the converter itself. It is legal, to reference the original string as passed in to the converter function, but then the AspectCallSet function should make a copy, if the string is needed beyond this call.

Declaration:

```
typedef OPM_Status (* ORM_ConverterStringToNativeFunc) (
    ORM_AppDataPtrDef      dest,          /* in */
    size_t                 size,          /* in */
    ORM_AttributeTypeDef   datatype,     /* in */
    ORM_AppConverterArgDef convarg,      /* in */
    ORM_String             strvalue      /* in */
);
```

Fields:

<i>dest</i>	Address/destination of native data element (e.g. attribute value)
<i>maxsize</i>	Maximum byte-size of data element
<i>datatype</i>	One of the ORM_AttributeTypes identifying the type of the native data element
<i>convarg</i>	Any kind of data (pointer) for this converter as provided with the attribute descriptor
<i>strvalue</i>	The new attribute value in its ascii presentation.
<i>returns</i>	ORM_ENoError if conversion was successfull and the resulting attribute value is valid or ORM_ERangeError.

### 3.5.3 ORM Built In Conversion Functions

The following functions are provided to convert generic C datatypes between their ORM and their native presentation. In addition sub functions are provided to support the special ORM SELECT and MCHOICE types, which are called by the generic converters. Along with these sets two new data structure (types) are introduced.

### 3.5.4 Function ORM\_GenericNativeToString

This function converts standard C-data types into their ASCII presentation. It returns only the converted value, but does not support the range and unit parts (e.g. returns NULL for those, if requested). In case of SELECT or MCHOICE functions, this function calls the related ORM\_Select.. or ORM\_MChoice functions.

*Note:* It is currently open, whether the conversion argument *convarg* may be used to specify a format string a la *printf*. Furthermore it is currently open, whether a NULL conversion function in the attribute descriptor should be directed to this (default) function.

Arguments as for ORM\_ConverterNativeToString!

Prototype:

```
CRM_Status ORM_GenericNativeToString(
    CRM_AppDataPtrDef    ptr,           /* in */
    size_t               maxsize,      /* in */
    CRM_AttribTypeDef    type,         /* in */
    CRM_AppConverterArgDef convarg,    /* in */
    CRM_String           *strvalue,    /* out */
    CRM_String           *rangevalue,  /* out */
    CRM_String           *strunit     /* out */
);
```

No parameter descriptions are available.

### 3.5.5 Function ORM\_GenericStringToNative

This function converts ASCII C-strings into standard C-datatype. In case of SELECT or MCHOICE functions, this function calls the related ORM\_Select.. or ORM\_MChoice functions.

*Note:* It is currently open, whether the conversion argument *convarg* may be used to specify a format string a la *scanf*. Furthermore it is currently open, whether a NULL conversion function in the attribute descriptor should be directed to this (default) function.

Arguments as for ORM\_ConverterStringToNative!

Prototype:

```
CRM_Status ORM_GenericStringToNative(
    CRM_AppDataPtrDef    ptr,           /* in */
    size_t               maxsize,      /* in */
    CRM_AttribTypeDef    type,         /* in */
    CRM_AppConverterArgDef convarg,    /* in */
    CRM_String           strvalue      /* in */
);
```

No parameter descriptions are available.

### 3.5.6 Structure ORM\_StringMapDef

This type of structure is used to map strings to binary values and vice versa. It may be used to convert internal flags and states to *friendly* names. StringMaps must be terminated by an entry with *name* set to NULL.

Declaration:

```
typedef struct ORM_StringMapTag {
    CRM_String  name;
    CRM_Key    key;
    ... CRM_StringMapDef;
```

Fields:

<i>name</i>	Friendly name for this key.
<i>key</i>	The binary native value of the key

### 3.5.7 ORM\_StringMapToString

This function maps a value key to a string using the given StringMap. It returns the string of that map entry, whose key is equal to the given key, else it returns the string passed in *notfound*.

Prototype:

```
CRM_String
CRM_StringMapToString( CRM_StringMapDef  map,
                      CRM_Key          key,
                      CRM_String       notfound);
```

Parameters:

<i>map</i>	Pointer to a sequence of map entries
<i>key</i>	Binary key value.
<i>notfound</i>	string to give back, if none of the keys in the map matched.

### 3.5.8 ORM\_StringMapToKey

This function maps a string value to a binary key using the given StringMap. It returns the key of that map entry, whose string is equal to the given key, else it returns the key passed in *invalidkey*.

Prototype:

```
CRM_Key
CRM_StringMapToKey( CRM_StringMapDef  map,
                   CRM_String       name,
                   CRM_Key          invalidkey);
```

Parameters:

<i>map</i>	Pointer to a sequence of map entries
<i>name</i>	No description
<i>invalidkey</i>	No description

### 3.5.9 Structure ORM\_StateMapDef

This structure is used to map *states* into strings, where a *state* is assumed to have a distinct set of possible next states, depending on the current value. E.g. this structure can be used to derive the set of possible new values i.e. it can provide the *range* value for a state attribute.

Otherwise it is used similar to the simpler StringMap structure. StateMaps must be terminated by an entry with *name* set to NULL.

Declaration:

```
typedef struct ORM_StateMapTag { /* not of the U.S.A.!! */
    CRM_String name;
    CRM_Key state;
    CRM_String validnexts;
} *ORM_StateMapDef;
```

Fields:

<i>name</i>	Friendly name for this key.
<i>state</i>	The binary native value of this state
<i>validnexts</i>	String of comma separated names of next valid states which may follow this state.

### 3.5.10 ORM\_StateMapToString

Convert an encoding of a state into a *friendly* name using the given statemap. If the state could not be found, the string passed in *notfound* is returned.

Prototype:

```
CRM_String CRM_StateMapToString ( CRM_StateMapDef map,
    CRM_Key state,
    CRM_String notfound );
```

Parameters:

<i>map</i>	Pointer to a (name=NULL) terminated state map.
<i>state</i>	the binary state
<i>notfound</i>	string to return, if none of the entries in the map had exactly the given state key.

### 3.5.11 ORM\_StateMapToKey

Convert a string representation of a state into a native encoding of a *state* using the given *statemap*. If the string could not be found, the state passed in *invalidstate* is returned.

Prototype:

```

CRM_Key
CRM_StateMapToKey ( CRM_StateMapDef map,
                   CRM_String name,
                   CRM_Key   invalidstate);

```

Parameters:

<i>map</i>	Pointer to a (name=NULL) terminated state map.
<i>name</i>	No description
<i>invalidstate</i>	No description

### 3.5.12 ORM\_StateMapNextByKey

Return the comma separated list of valid next states given the current state.

Prototype:

```

CRM_String
CRM_StateMapNextByKey ( CRM_StateMapDef map,
                       CRM_String state);

```

Parameters:

<i>map</i>	Pointer to a (name=NULL) terminated state map.
<i>state</i>	the binary state



### 3.6 ORM Dump & Restore Support

This section was generated from `<stdin>` by CDOC on Fri Jan 27 19:59:34 1995.

This module of the ORM Server Support Library supports the dump and restore of complete subtrees, and therefore can be used to save the current configuration to a persistent storage media (i.e. the MSF Warehouse) and reload it from there. The actual IO functions are currently not supported by this layer or the support library at all!

Dump and Restore are functions of the ORM SSL and not of the ORM protocol (i.e. there is no *DUMP* or *RESTORE* request defined in the protocol).

This implies, that these functions have to be dispatched out of the application layer explicitly. One (intended) way to dispatch those functions interactively is to provide pseudo components in every subtree, which should be independent storable/reloadable. These contain the required parameters like Warehouse location or version name as attributes. An Attribute-Set request to this subtree then results in the execution of the corresponding function.

Under the layered view of the ORM SSL, these two functions belong to the protocol layer, as they use (nearly) the same functionality of the higher layers via upcalls.

#### 3.6.1 General Model:

Starting from a given node, which is used as the root of the relevant subtree to dump, all components, object links and writable attributes with their meta information are recursively extracted relative to the current subtree root. The extended/meta information on the persistent media can be used to interpret the stored attributes and apply changes to the stored version without the ORM server/application alive but through special clients (by an ORM/Warehouse gateway for example).

The dumped ORM tree can be used to reload the whole subtree at any time, by providing the node and call the *restore* function of the ORM SSL (which is a special kind of Set-Request).

This special kind of SET request creates a new situation, as components (or any new subtree) may have been created dynamically by the ORM server application on request. On the next cold start of the application, these subtrees do not exist.

This results in failed lookup requests by the ORM protocol layer, which usually is treated as an error (remember: ORM-P has no direct support for object/component creation, but this is emulated by sets of writeonly attributes in separate subtrees, i.e. *New..*). To handle this case, the application can provide a special function during application context setup to create new instances including the ORM subtree (`ORM_NodeNotFoundTrapFunc()`).

A parameter is passed to this creation function, which indicates, whether this situation was caused by a regular ORM protocol request or by an internally generated *restore* request, so the application code can still decide to refuse the creation.

### 3.6.2 ORM\_Dump

This function extracts the ORM entities in the subtree pointed to by *subtree* into the character buffer, so it can be used by a later *ORM\_Restore* function (or can be used as a subrequest in a regular ORM protocol request).

It is the responsibility of the caller to provide a sufficient buffer, which can hold the subtree information of the given depth!

Prototype:

```

ORM_Status
ORM_Dump(   ORM_AppNodeDef subtree,
            long          depth,
            long          what,
            ORM_String    buffer,
            long          *maxlen
            );
    
```

Parameters:

- subtree*            The root of the subtree to dump. All navigation information is saved relative to this node.
- depth*             The depth, up to which entities in this subtree should be extracted. A depth of 0 means, direct childs of the given sub-root only, i.e. if the subtree points to a component node with an attribute node as one of its direct children, the name of the attribute would be extracted, but not the value or other extended attribute information, if depth=0. A depth of -1 extracts the whole subtree, independent of its depth.
- what*              Is a bitmask, defining what kind of entities should be extracted:  
 ORM\_DumpSetObjects ORM\_DumpSetComponents  
 ORM\_DumpSetAttributes ORM\_DumpSetWritable  
 ORM\_DumpSetDefault = Objects | Components | Writable  
 ORM\_DumpSetEverything = Objects | Components | Attributes
- buffer*            The address of a character buffer, where to store the extracted entity information
- maxlen*            Pointer to the maximum length of this buffer. On return, maxlen will contain the number of bytes used in this buffer including the C-String '\0' terminator.

### 3.6.3 ORM\_Restore

Function to reload the saved ORM information into an existing subtree, where at least the root of the given subtree has to exist. *Note:* Because the restore request may fail with some attribute modifications already performed, an application may want to call *ORM\_Dump*

(into a temporary buffer) before actually calling ORM\_Restore, to be able to *undo* the partial operations.

Prototype:

```

ORM_Status
ORM_Restore(   ORM_AppNodeDef subtree,
               ORM_SeqDef  buffer,
               long        *length
               );

```

Parameters:

*subtree*            Node of the subtree to load the management information into.

*buffer*            pointer to ORM subrequest sequence.

*length*            pointer to length of the request. On return, this will contain the number of bytes processed from this request.

### 3.7 ORM SSL Generic Datatypes

```

#ifndef _ORM_TYPE_H
#define _ORM_TYPE_H

/*
 * Some generic definitions, may become obsolete
 */

typedef enum {
    False,
    True
} boolean;

#ifndef NULL
#define NULL (void *)0
#endif

typedef unsigned long size_t;

#define CRM_Ptr( base, offset) (void *)((size_t)(base)+(size_t)(offset))

#define CRM_Malloc(x) (void *)malloc(x)
#define CRM_Free(x) free(x)

/*
 * more CRM specific stuff
 */

/*
 * How requests and responses are passed to the ORM protocol layer
 */

typedef char *CRM_RequestDef;
typedef char *CRM_ResponseDef;

/*
 * The principal type of every ORM protocol entity, e.g. names and values,
 * but also used most C-strings.
 */

typedef char *CRM_String;

/*
 * Used for StateMaps and String Maps as the lookup key
 */

typedef long CRM_Key;

/*
 * The following are various opaque handles. Opaque mainly to the protocol
 * layer but also for the lower of two stacked layers.
 */

typedef void *CRM_AppNodeDef;
typedef void *CRM_AppHandleDef;
typedef void *CRM_AppAspectDef;
typedef void *CRM_AppDataPtrDef;

```

```

typedef void *CRM_AppAttribDescrDef;
typedef void *CRM_AppCallContextDef;
typedef void *CRM_AppConverterArgDef;

/*
 * Valid Access modes for an attribute
 */

typedef enum {
    CRM_AttrModeNone,
    CRM_AttrModeRW,
    CRM_AttrModeRO,
    CRM_AttrModeWO,
    CRM_AttrModeRWP
} CRM_AttrModeDef;

/*
 * Known (native) datatypes, which are supported by the Generic converter
 */

typedef enum {
    CRM_AttrTypeNone,
    CRM_AttrTypeInt1,
    CRM_AttrTypeUInt1,
    CRM_AttrTypeInt2,
    CRM_AttrTypeUInt2,
    CRM_AttrTypeInt4,
    CRM_AttrTypeUInt4,
    CRM_AttrTypeInt8,
    CRM_AttrTypeUInt8,
    CRM_AttrTypeReal32,
    CRM_AttrTypeReal64,
    CRM_AttrTypeString,
    CRM_AttrTypeHexOct,
    CRM_AttrTypeSelect, /* 1 out of many */
    CRM_AttrTypeState, /* 1 out of many, but with dynamic range */
    CRM_AttrTypeOption, /* binary switch ON/OFF YES/NO */
    CRM_AttrTypeMChoice, /* n out of many */
    CRM_AttrTypeUnknown
} CRM_AttrTypeDef;

/*
 * CRM Error Codes, used as well by the protocol as by the ORM SSL
 */

typedef enum {
    CRM_ENoError, /* Operation successful! */
    CRM_EPermission, /* None or wrong auth.information */
    CRM_ENoSuchNode, /* some name in pathname could not be found */
    CRM_ENoSuchAttribute, /* Attribute in Set-Request doesn't exist */
    CRM_ENoSuchObject, /* Object/Manager could not be found */
    CRM_EInvalidOperation, /* Operation not applicable to node type */
    CRM_EProtocol, /* ORM protocol violation */
    CRM_ECommunication, /* lower level comm error */
    CRM_ERange, /* new attribute value out of range */
    CRM_EParameterList, /* set of attributes not applicable */
    CRM_EMissingAttribute, /* mandatory attrib. missing or NULL */
    CRM_ENoSpace, /* internal allocation */
    CRM_ENoBuffer, /* response buffer */

```

```
    ORM_EInternal,          /* ORM Internal error -> bug */
    ORM_EApplication,      /* application level error -> bug */
} CRM_Status;

/*
 * Types of nodes in the virtual tree. Note, that only nodes of type
 * CRM_NodeTypeComponent can have children!
 */

typedef enum {
    CRM_NodeTypeUnknown,
    CRM_NodeTypeObject,
    CRM_NodeTypeComponent,
    CRM_NodeTypeAttribute,
    CRM_NodeTypeAny
} CRM_NodeTypeDef;

/*
 * Types of CRM requests. Note that the Dump and Restore requests are
 * not part of the CRM protocol, but only available within the ORM
 * server support library
 */

typedef enum {
    CRM_RequestObjectGet,
    CRM_RequestComponentGet,
    CRM_RequestAttributeGet,
    CRM_RequestAttributeInfoGet,
    CRM_RequestAttributeSet,
    CRM_RequestSet,
    CRM_RequestGet,
    CRM_RequestDump,
    CRM_RequestRestore
} CRM_RequestTypeDef;

#endif
```

## 5 WHAT IS CLAIMED IS:

1. A system for managing objects, including a first server, comprising:  
a first receiver portion configured to receive a request in a  
hypermedia format;  
a first translator portion configured to convert the hypermedia  
0 request to an object request;  
a sender portion configured to send the object request to an object  
manager;  
a second receiver portion configured to receive a response from  
the object manager; and  
5 a second translator portion configured to convert the object  
manager response to the hypermedia format.

2. The system of claim 1, further comprising a second server, including:  
a third receiver portion configured to receive a request in a  
hypermedia format;  
0 a third translator portion configured to convert the hypermedia  
request to an object request;  
a second sender portion configured to send the object request to  
an object manager;  
a fourth receiver portion configured to receive a response from the  
5 object manager; and  
a fourth translator portion configured to convert the object  
manager response to the hypermedia format.

3. The system of claim 1, further comprising:  
a second sending portion configured to send the hypermedia  
0 format data from the sender portion to a browser to be displayed.

4. The system of claim 1, where the object manager manages a self-  
describing object.

5. The system of claim 1, where the object manager manages a non-self  
describing object.

5           6. The system of claim 5, where the object manager performs a "worm" function.

          7. A method for browsing objects, where a browser communicates with a server, comprising the steps, performed by the browser, of:

                  sending an initial URL to the server;  
0                   receiving first data from the server, where the first data specifies an object corresponding to the URL;

                  sending user-entered data associated with the object to the server;  
and

                  receiving second data from the server, where the second data  
5 specifies a second object corresponding to the user-entered data.

          8. The method of claim 7,  
                  wherein the step of sending an initial URL to the server comprises the step of sending an initial URL known to the browser, where the URL is the URL of the server.

0           9. The method of claim 7,  
                  wherein the step of sending an initial URL to the server comprises the step of sending an initial URL entered by the user, where the URL is the URL of the server.

          10. The method of claim 7,  
5                   wherein the step of sending user-entered data associated with the object to the server includes the step of indicating a "set" operation in the user-entered data.

          11. The method of claim 7,  
                  wherein the step of sending user-entered data associated with the  
0 object to the server includes the step of indicating a "get" operation in the user-entered data.

          12. The method of claim 7, wherein the step of receiving second data from the server includes the step of receiving data corresponding to an attribute value of the object.

5



5           13. The method of claim 7, wherein the step of receiving second data  
from the server includes the step of receiving data corresponding to a second  
object linked to the first object via an object-link.

          14. A computer program product comprising:

0           a computer usable medium having computer readable code  
embodied therein for managing objects, the computer program product  
comprising:

          computer readable program code devices configured to cause a  
computer to effect receiving a request in a hypermedia format;

5           computer readable program code devices configured to cause a  
computer to effect converting the hypermedia request to an object request;

          computer readable program code devices configured to cause a  
computer to effect sending the object request to an object manager;

          computer readable program code devices configured to cause a  
computer to effect receiving a response from the object manager; and

0           computer readable program code devices configured to cause a  
computer to effect converting the object manager response to a second  
hypermedia format.

          15. The computer program product of claim 14, further comprising:

5           computer readable program code devices configured to cause a  
computer to effect sending the second hypermedia format data to a browser to  
be displayed.

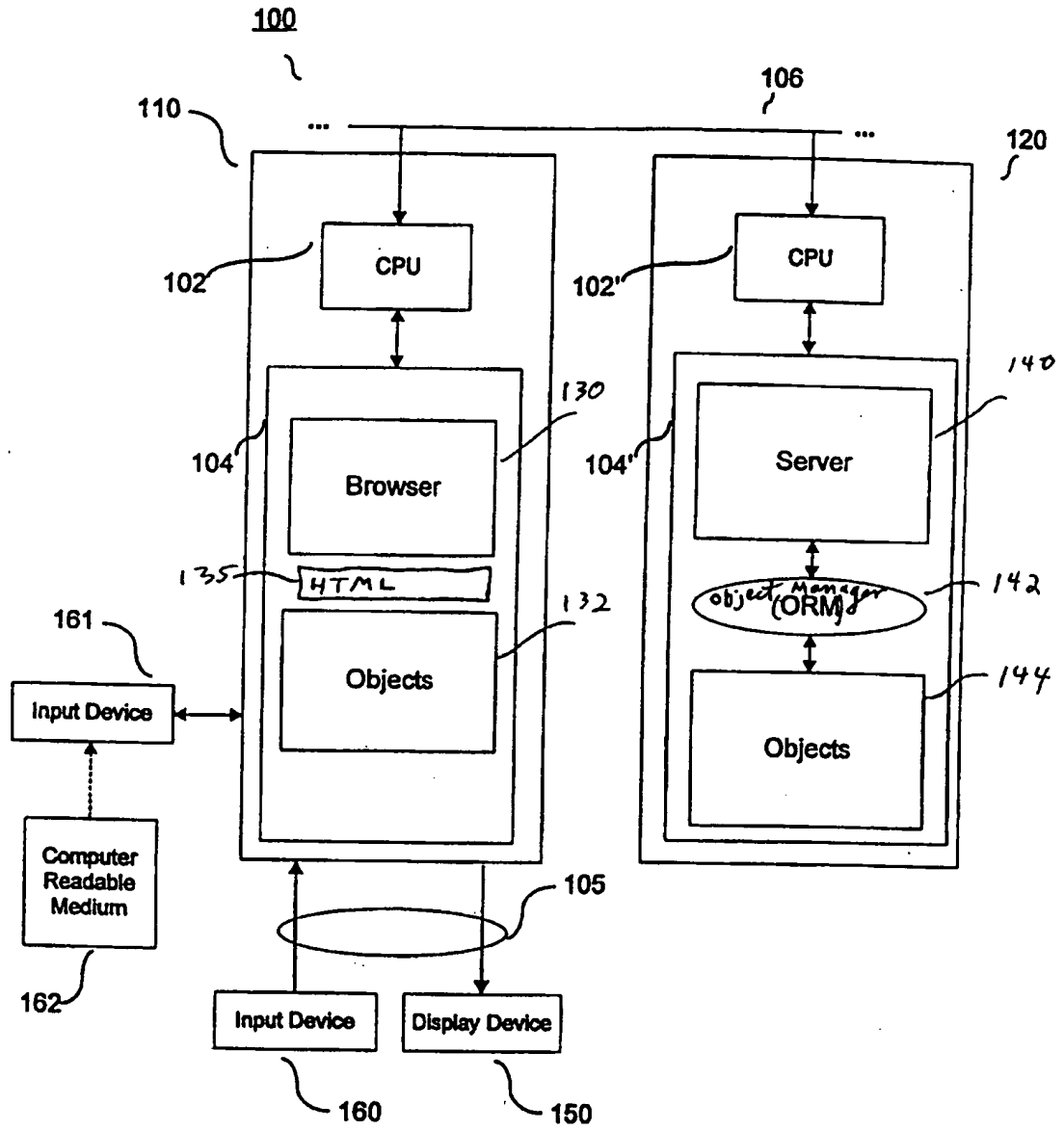


Fig. 1

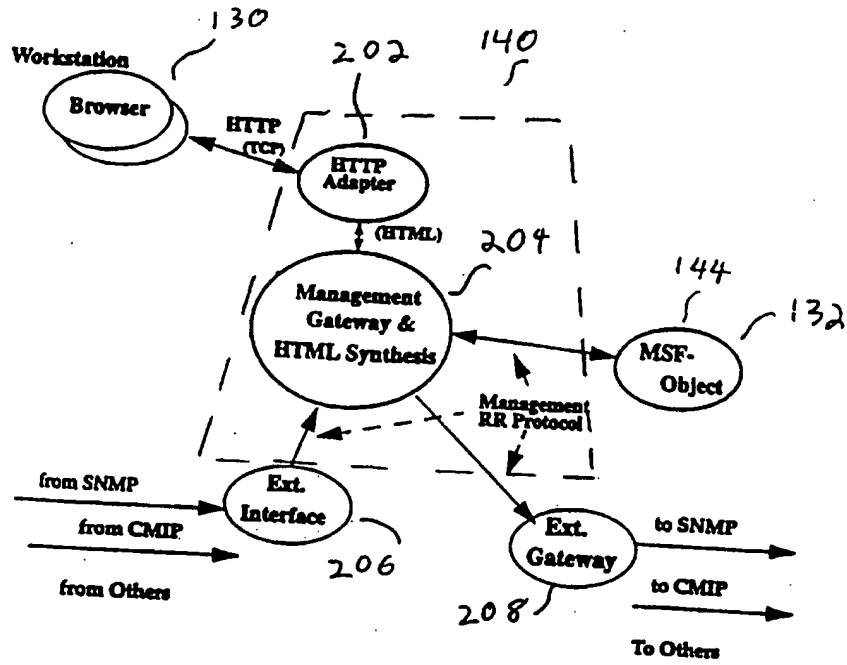


Fig. 2

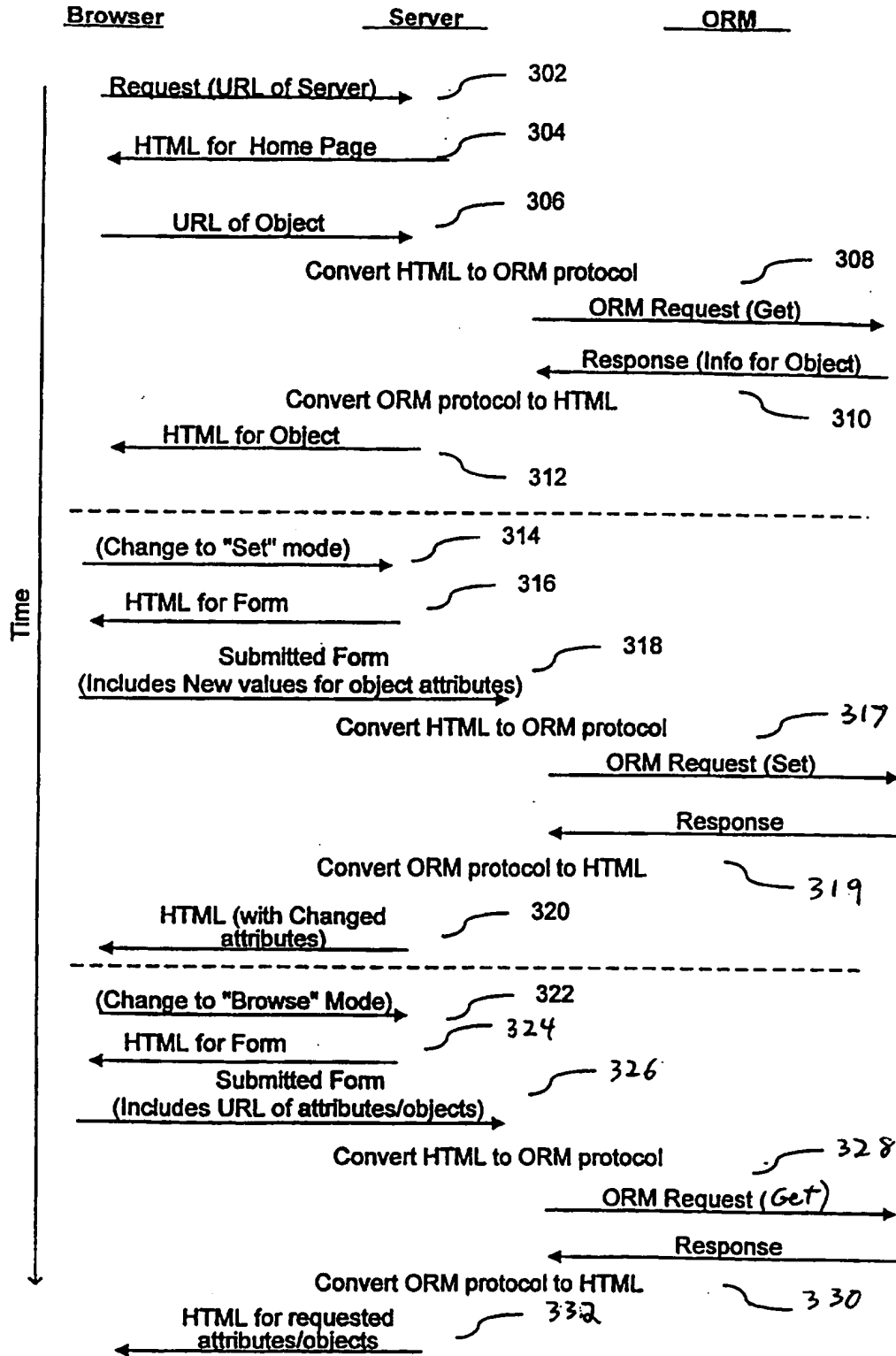


Fig. 3

A System as a Tree of Managed Entities:

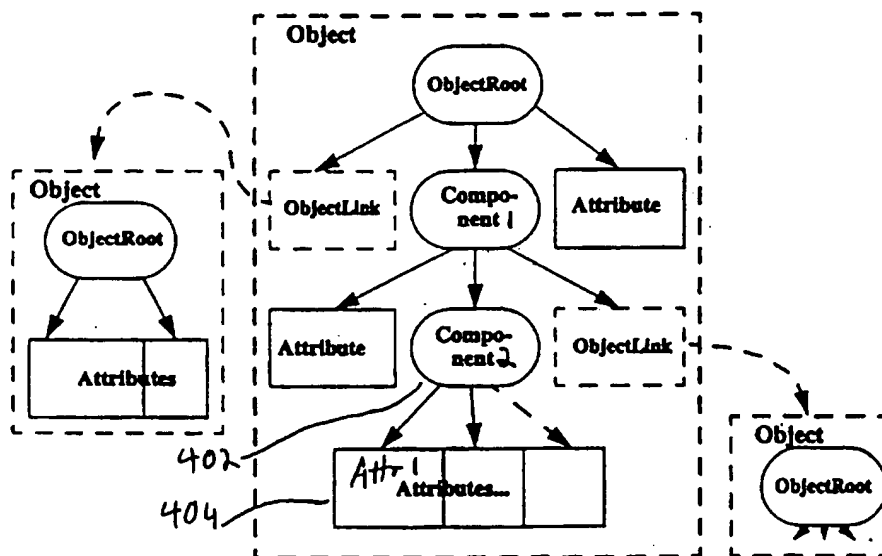


Fig. 4

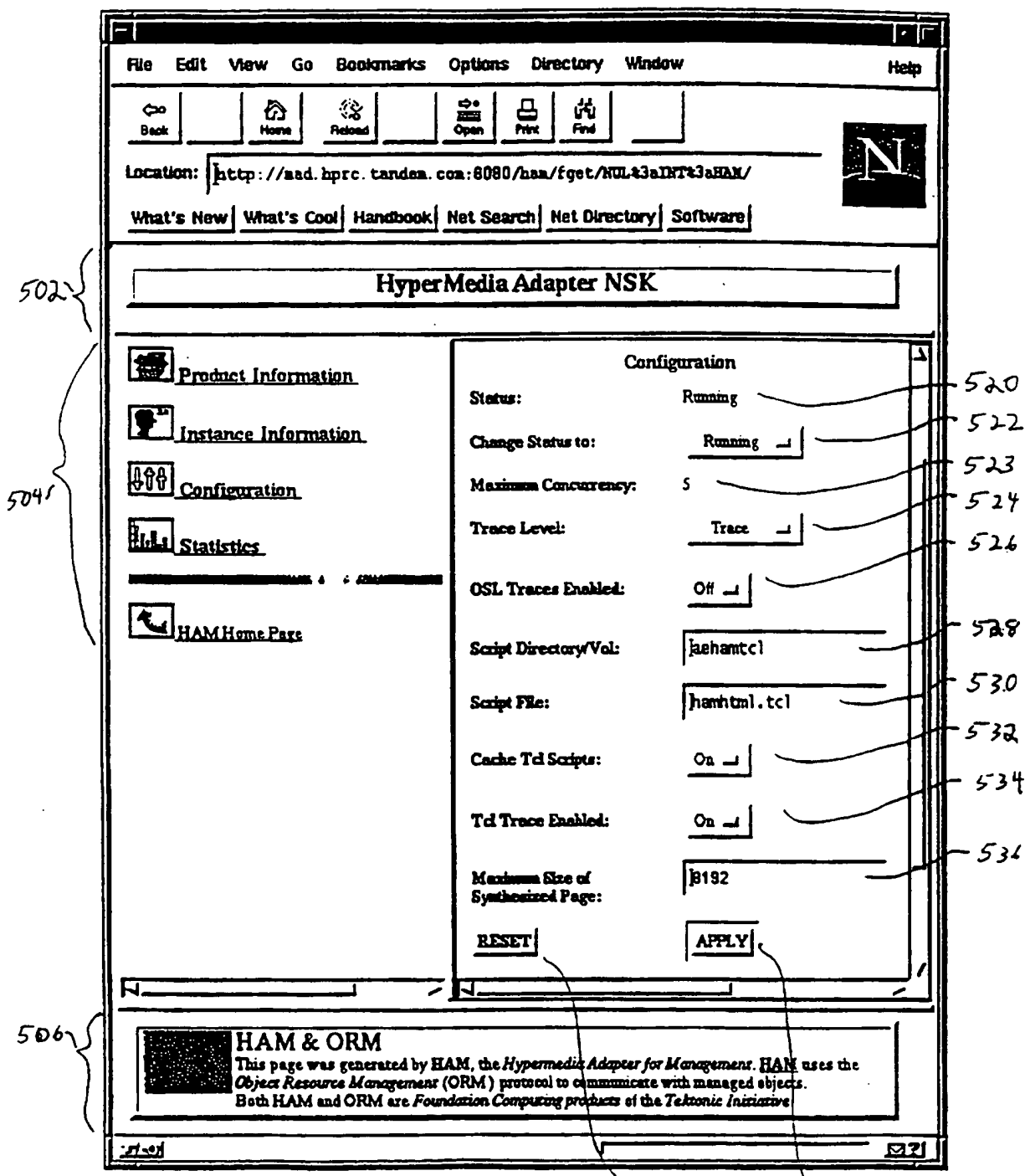


Fig. 5 540 550

```

1 <HTML>
2 <HEAD>
3 <TITLE>Alter Configuration of HyperMedia Adapter NSKITCP:168.87.28.7/8062:HAM </TI
4 <LINK REL="stylesheet" TYPE="text/css" HREF="http://mad.hpcc.tandem.com:8080/ham/set/TCPSJA168.87.28.782806283AHAM
5 </HEAD>
6 <BODY BACKGROUND="/icons/1cstgty.gif">
7 <IMG ALT="" SRC="/icons/pccnfig.gif">
8 </IMG>
9 <H1>Alter Configuration</H1>
10 <H2>HyperMedia Adapter NSK</H2>
11 <FORM METHOD="POST" ACTION="http://mad.hpcc.tandem.com:8080/ham/cset/TCPSJA168.87.28
12 *782806283AHAM22HyperMedia30Adapter20RSK22(Configuration)"/>
13 <TABLE>
14 <TR>
15 <TD><CAPTION><STRONG>FONT SIZE=4>Configuration</FONT></STRONG></CAPTION>
16 <TD><TR><TH ALIGN="LEFT"><P>Status:</TH>
17 <TD><TR><TH ALIGN="LEFT">Remaining</TD>
18 <TR><TH ALIGN="LEFT">Remaining</TD>
19 <TR><TH ALIGN="LEFT"><P>Change Status to:</TH><TD>
20 <TR><TH ALIGN="LEFT"><P>Change Status to:>
21 <OPTION SELECTED> Running
22 <OPTION> Suspended
23 <OPTION> Aborted
24 <OPTION> Stopped
25 </SELECT>
26 </TD>
27 </TR>
28 <TR><TH ALIGN="LEFT"><P>Maximum Concurrency:</TH>
29 <TR><TH ALIGN="LEFT"><P>Maximum Concurrency:>
30 <TR><TH ALIGN="LEFT"><P>Trace Level:</TH><TD>
31 <TR><TH ALIGN="LEFT"><P>Trace Level:>
32 <OPTION SELECTED> Warnings
33 <OPTION> Errors
34 <OPTION> Info
35 <OPTION> Debug
36 <OPTION> Traces
37 <OPTION> Everything
38 </SELECT>
39 </TD></TR>
40 <TR><TH ALIGN="LEFT"><P>OSL Traces Enabled:</TH><TD>
41 <TR><TH ALIGN="LEFT"><P>OSL Traces Enabled:>
42 <OPTION SELECTED> Off
43 <OPTION> On
44 </SELECT>
45 </TD></TR>
46 <TR><TH ALIGN="LEFT"><P>Script Directory/Vol:</TH><TD>
47 <TR><TH ALIGN="LEFT"><P>Script Directory/Vol:>
48 <OPTION SELECTED> /
49 <OPTION> /usr/bin
50 </SELECT>
51 </TD></TR>
52 <TR><TH ALIGN="LEFT"><P>Cache Tcl Scripts:</TH><TD>
53 <TR><TH ALIGN="LEFT"><P>Cache Tcl Scripts:>
54 <OPTION SELECTED> On
55 <OPTION> Off
56 </SELECT>
57 </TD></TR>
58 <TR><TH ALIGN="LEFT"><P>Cache Tcl Scripts:</TH><TD>
59 <TR><TH ALIGN="LEFT"><P>Cache Tcl Scripts:>
60 <OPTION SELECTED> On
61 <OPTION> Off
62 </SELECT>
63 </TD></TR>
64 <TR><TH ALIGN="LEFT"><P>Trace Enabled:</TH><TD>
65 <TR><TH ALIGN="LEFT"><P>Trace Enabled:>
66 <OPTION SELECTED> Off
67 <OPTION> On
68 </SELECT>
69 </TD></TR>
70 </TABLE>

```

Fig 6(a)

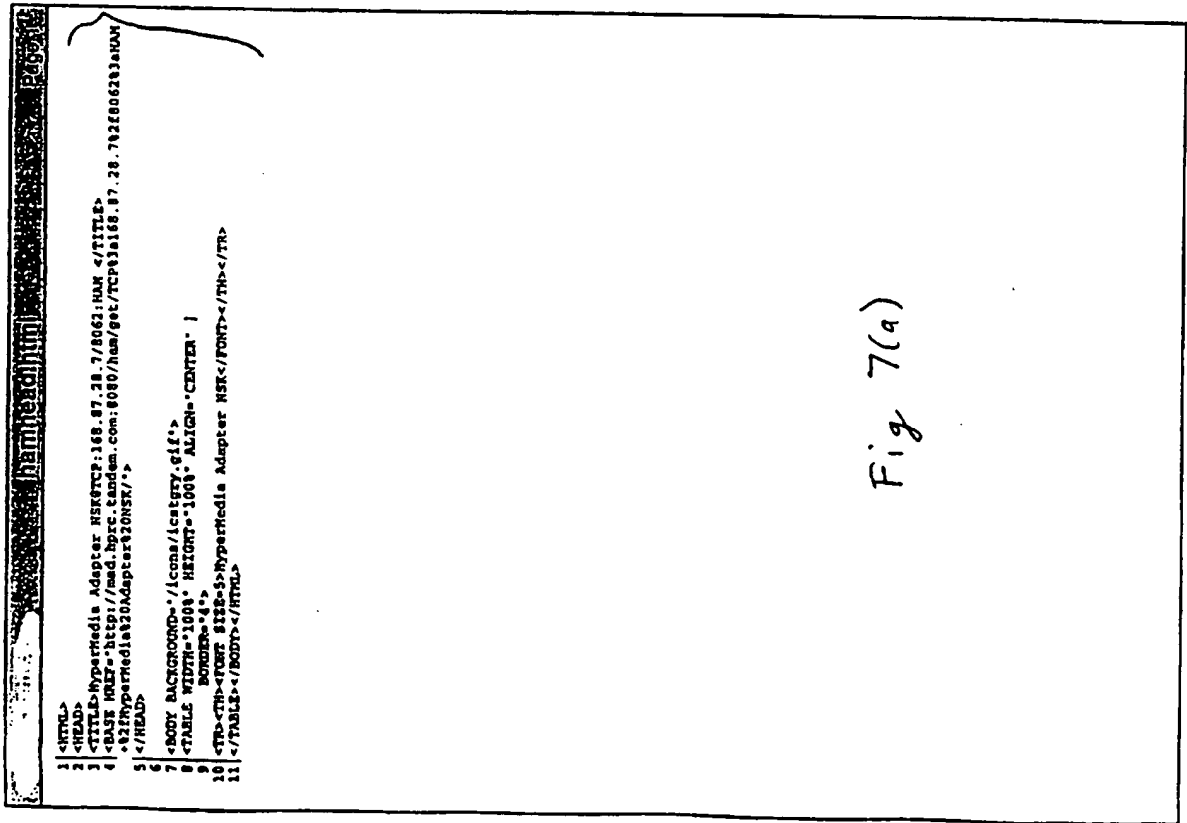
```

71 <TR><TH ALIGN="LEFT"><P>Maximum Size of Synthesized Page:</TH><TD>
72 <TR><TH ALIGN="LEFT"><P>Maximum Size of Synthesized Page:>
73 <OPTION SELECTED> 8192
74 <OPTION> 16384
75 </SELECT>
76 </TD></TR>
77 <TR><TH ALIGN="LEFT"><P>Input Type:reset</TH><TD>
78 <TR><TH ALIGN="LEFT"><P>Input Type:submit</TH><TD>
79 <TR><TH ALIGN="LEFT"><P>Input Type:submit</TH><TD>
80 <TR><TH ALIGN="LEFT"><P>Input Type:submit</TH><TD>
81 <TR><TH ALIGN="LEFT"><P>Input Type:submit</TH><TD>
82 <TR><TH ALIGN="LEFT"><P>Input Type:submit</TH><TD>
83 <TR><TH ALIGN="LEFT"><P>Input Type:submit</TH><TD>
84 <TR><TH ALIGN="LEFT"><P>Input Type:submit</TH><TD>
85 <TR><TH ALIGN="LEFT"><P>Input Type:submit</TH><TD>
86 <TR><TH ALIGN="LEFT"><P>Input Type:submit</TH><TD>
87 <TR><TH ALIGN="LEFT"><P>Input Type:submit</TH><TD>
88 <TR><TH ALIGN="LEFT"><P>Input Type:submit</TH><TD>
89 <TR><TH ALIGN="LEFT"><P>Input Type:submit</TH><TD>
90 <TR><TH ALIGN="LEFT"><P>Input Type:submit</TH><TD>
91 <TR><TH ALIGN="LEFT"><P>Input Type:submit</TH><TD>
92 <TR><TH ALIGN="LEFT"><P>Input Type:submit</TH><TD>
93 <TR><TH ALIGN="LEFT"><P>Input Type:submit</TH><TD>
94 <TR><TH ALIGN="LEFT"><P>Input Type:submit</TH><TD>
95 <TR><TH ALIGN="LEFT"><P>Input Type:submit</TH><TD>
96 <TR><TH ALIGN="LEFT"><P>Input Type:submit</TH><TD>
97 <TR><TH ALIGN="LEFT"><P>Input Type:submit</TH><TD>
98 <TR><TH ALIGN="LEFT"><P>Input Type:submit</TH><TD>
99 <TR><TH ALIGN="LEFT"><P>Input Type:submit</TH><TD>
100 <TR><TH ALIGN="LEFT"><P>Input Type:submit</TH><TD>

```

Fig 6(b)

702





704

```

1 <IMG SRC="" alt="Apple logo" data-bbox="185 500 205 880"/>
2 <TITLE> Home Trailer Frame</TITLE>
3 <BODY BACKGROUND="/icons/ctstory.gif">
4 <TABLE WIDTH="100%" BORDER="0">
5 <TR>
6 <TD style="width: 50%; vertical-align: top; padding: 5px;">
7 <IMG SRC="" alt="Product image" data-bbox="210 620 285 870"/>
8 </TD>
9 <TD style="width: 50%; vertical-align: top; padding: 5px;">
10 This page was generated by <IMG SRC="" alt="HAM & ORM logo" data-bbox="290 680 310 710"/>
11 the <IMG SRC="" alt="Hypermedia Adapter for Management logo" data-bbox="290 715 310 745"/>
12 <A href="http://far.hprc.tandem.com/8888/info/lehman.html" target="_top">
13 <IMG SRC="" alt="HAM & ORM logo" data-bbox="315 680 335 710"/>
14 uses the <IMG SRC="" alt="Object Resource Management logo" data-bbox="315 715 335 745"/>
15 protocol to communicate
16 with managed objects.
17 Both HAM and ORM are <IMG SRC="" alt="Foundation Computing products logo" data-bbox="340 680 360 710"/> of the
18 <IMG SRC="" alt="Tektonic Initiatives logo" data-bbox="340 715 360 745"/>
19 </TD>
20 </TABLE>
21 </BODY>

```

Fig 7(b)

706

```

1 |<HTML>
2 |<HEAD>
3 |<TITLE>Index into Hypermedia Adapter N5K9TCF.168.87.28.7/8062:MAN </TITLE>
4 |<BASE HREF=http://mad.hpcc.tandem.com:8080/ham/get/TCF83168.87.28.7921806283aMAN
5 |>
6 |</HEAD>
7 |<BODY BACKGROUND=/icons/icstgry.gif>
8 |<H3><A HREF=http://mad.hpcc.tandem.com:8080/ham/cget/TCF83168.87.28.7921806283aH
9 |>N5K9TCF.168.87.28.7921806283aProduct</A>
10 |<A HREF=http://mad.hpcc.tandem.com:8080/ham/cget/TCF83168.87.28.7921806283aH
11 |>N5K9TCF.168.87.28.7921806283aInstance</A>
12 |<A HREF=http://mad.hpcc.tandem.com:8080/ham/cget/TCF83168.87.28.7921806283aH
13 |>N5K9TCF.168.87.28.7921806283aConfiguration</A>
14 |<A HREF=http://mad.hpcc.tandem.com:8080/ham/cget/TCF83168.87.28.7921806283aH
15 |>N5K9TCF.168.87.28.7921806283aStatistics</A>
16 |</H3>
17 |<H3><A HREF=http://mad.hpcc.tandem.com:8080/ham/get/TCF83168.87.28.7921806283aM
18 |>AN5K9TCF.168.87.28.7921806283aHome Page</A>
19 |</H3>
20 |</BODY>
21 |</HTML>

```

Fig 7(c)

ORM Protocol examples

```

-85 17:22      ham.trace      Page 1
11A request as issued to create the index frame as seen in
12the left frame of the window dump
13
14>>>>>
150rmVersion:1.0      820
160rmGet:HyperMedia Adapter NSK
17Info      922
18Object      824
19|<<<<<<
20|
21|and its response:
22|
23|>>>>>
24|0rmVersion:1.0
25|0rmGet:HyperMedia Adapter NSK
26|Component:Product Information
27|Component:Instance Information
28|Component:Configuration
29|Component:Statistics
30|<<<<<<
31|
32|11A Browse Request as issued for the frame on the right side
33of the window dump (compare hamstat.html)
34>>>>>
350rmVersion:1.0
360rmGet:HyperMedia Adapter NSK/Configuration      830
37Info
38Component
39|<<<<<<
40|and its response
41>>>>>
420rmVersion:1.0
430rmGet:HyperMedia Adapter NSK/Configuration
44Attribute:Status
45Value:Running
46Mode:RO
47Field:Enum
48Range:Suspended,Aborted,Stopped
49Attribute:Change Status to
50Value:Running
51Mode:RW
52Field:Enum
53Range:Suspended,Aborted,Stopped
54Attribute:Maximum Concurrency
55Value:5
56Field:Int
57Manager:1..70
58Attribute:Trace Level
59Value:Trace
60Mode:RW
61Field:Enum
62Range:Errors,Warnings,Info,Debug,Everything
63Attribute:OSL Traces Enabled
64Value:Off
65Mode:RW
66Field:Enum
67Range:On
68Attribute:Script Directory/Vol
69Value:hamstat
70Mode:RW
71Attribute:Script File
72Value:hamstat.tcl
73Mode:RW

```

Fig. 8(a)

```

-85 17:22      ham.trace      Page 2
74Field:String
75Attribute:Cache Tcl Scripts
76Value:On
77Mode:RW
78Field:Enum
79Range:Off
80Attribute:Tcl Trace Enabled
81Value:On
82Mode:RW
83Field:Enum
84Range:Off
85Attribute:Maximum Size of Synthesized Page
86Value:0192
87Mode:RW
88Field:Int
89Range:1048...16384
90|<<<<<<
91|
92|93IA Set/Change Request as a result from the
93HTML FORM processing.
94|<<<<<<
95|
96|>>>>>
970rmVersion:1.0
980rmGet:HyperMedia Adapter NSK/Configuration      840
99Attribute:Change Status to
100Value:Running
101Attribute:Trace Level
102Value:Trace
103Attribute:OSL Traces Enabled
104Attribute:Script Directory/Vol
105Value:hamstat
106Attribute:Script File
107Value:hamstat.tcl
108Attribute:Cache Tcl Scripts
109Value:On
110Field:Enum
111Range:On
112Attribute:Tcl Trace Enabled
113Attribute:Maximum Size of Synthesized Page
114Value:0192
1150rmGet:HyperMedia Adapter NSK/Configuration
116Info
117|<<<<<<
118|
119|Response similar to Get Request above
120|<<<<<<
121|
122|

```

Fig. 8(b)

File Edit View Go Bookmarks Options Directory Window Help

Back Home Reload Open Print Find

Location: <http://aad.hprc.tandem.com:8080/ham/fget/NUL+3aINT+3aHAM/>

What's New | What's Cool | Handbook | Net Search | Net Directory | Software

---

**HyperMedia Adapter NSK**

[Product Information](#)

[Instance Information](#)

[Configuration](#)

[Statistics](#)

[HAM Home Page](#)

**Configuration**  
of HyperMedia Adapter NSK

---

**Configuration**

Status:	Running
Maximum Concurrency:	5
Trace Level:	Warnings
OSL Traces Enabled:	Off
Script Directory/Vol:	sehamtd
Script File:	hamhtml.tcl
Cache Tcl Scripts:	On
Tcl Trace Enabled:	Off
Maximum Size of Synthesized Page:	8192

[Alter Configuration](#)

---

**HAM & ORM**

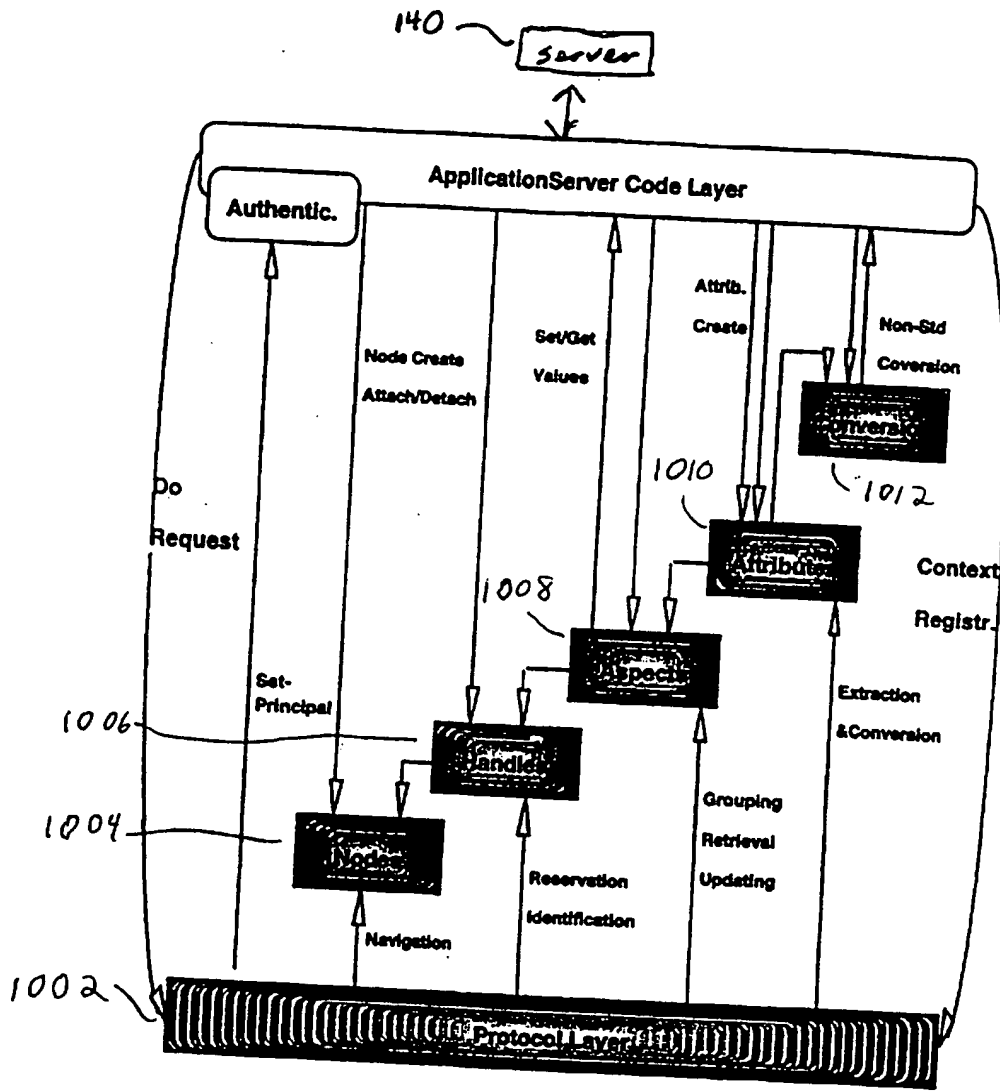
This page was generated by HAM, the *Hypermedia Adapter for Management*. HAM uses the *Object Resource Management (ORM)* protocol to communicate with managed objects. Both HAM and ORM are *Foundation Computing* products of the *Tektonic Initiative*.

9022

9042

9062

Fig. 9



The Layered Structure of the ORM Support Library

Fig. 10

# INTERNATIONAL SEARCH REPORT

Int'l. Application No  
PCT/US 97/11885

<b>A. CLASSIFICATION OF SUBJECT MATTER</b> IPC 6 G06F17/30		
According to International Patent Classification (IPC) or to both national classification and IPC		
<b>B. FIELDS SEARCHED</b>		
Minimum documentation searched (classification system followed by classification symbols) IPC 6 G06F		
Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched		
Electronic data base consulted during the international search (name of data base and, where practical, search terms used)		
<b>C. DOCUMENTS CONSIDERED TO BE RELEVANT</b>		
Category <sup>o</sup>	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	JAGANNATHAN V ET AL: "COLLABORATIVE INFRASTRUCTURES USING THE WWW AND CORBA-BASED ENVIRONMENTS" PROCEEDINGS - THE WORKSHOP ON ENABLING TECHNOLOGIES: INFRASTRUCTURE FOR COLLABORATIVE ENTERPRISES, 19 June 1996, pages 292-297, XP000645510 see page 293, column 1, line 39 - page 294, column 1, line 5 -----	1, 14
<input type="checkbox"/> Further documents are listed in the continuation of box C.		
<input type="checkbox"/> Patent family members are listed in annex.		
<b>* Special categories of cited documents :</b>		
*A* document defining the general state of the art which is not considered to be of particular relevance *E* earlier document but published on or after the international filing date *L* document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified) *O* document referring to an oral disclosure, use, exhibition or other means *P* document published prior to the international filing date but later than the priority date claimed		
*T* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention *X* document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone *Y* document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art. *Z* document member of the same patent family		
Date of the actual completion of the international search  <div style="text-align: center; font-size: 1.2em;">6 November 1997</div>	Date of mailing of the international search report  <div style="text-align: center; font-size: 1.2em;">12. 11. 97</div>	
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Authorized officer  <div style="text-align: center; font-size: 1.2em;">Katerbau, R</div>	



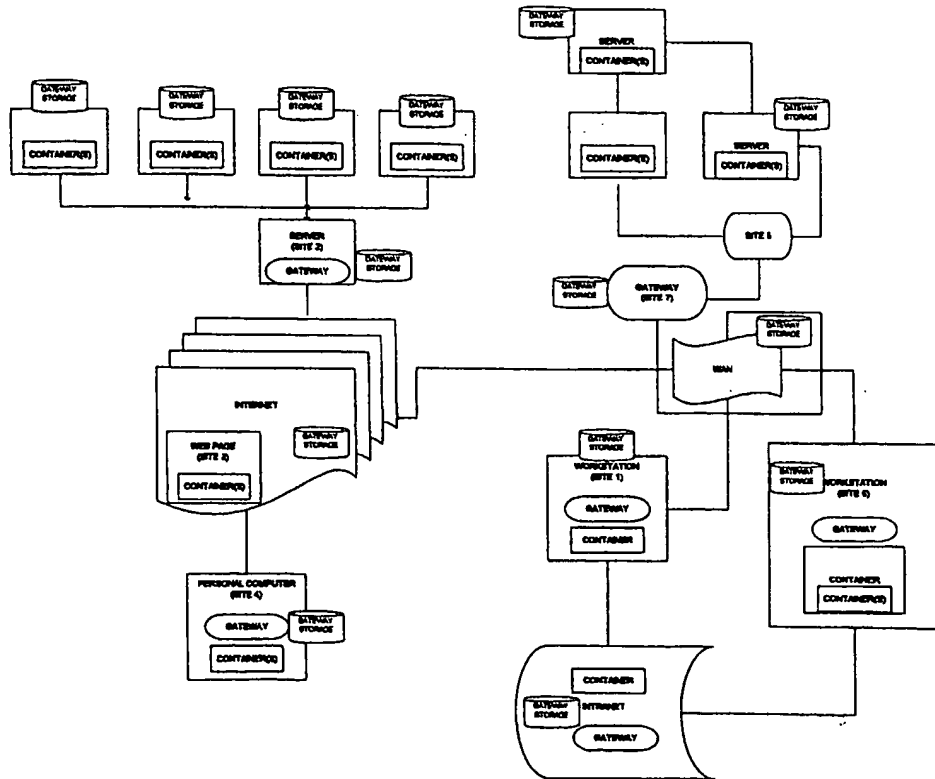
INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

<p>(51) International Patent Classification <sup>6</sup> : G06F 17/30, 3/14</p>	<p>A1</p>	<p>(11) International Publication Number: <b>WO 99/39285</b> (43) International Publication Date: 5 August 1999 (05.08.99)</p>
<p>(21) International Application Number: PCT/US99/01988 (22) International Filing Date: 28 January 1999 (28.01.99) (30) Priority Data: 60/073,209 30 January 1998 (30.01.98) US (71) Applicant (for all designated States except US): EMATRIX CORPORATION [US/US]; 104 West Anapamu, Santa Barbara, CA 93101 (US). (72) Inventor; and (75) Inventor/Applicant (for US only): DE ANGELO, Michael [US/US]; Suite 290, 1324 J State Street, Santa Barbara, CA 93101 (US). (74) Agents: SUEOKA, Greg, T. et al.; Fenwick &amp; West LLP, Two Palo Alto Square, Palo Alto, CA 94306 (US).</p>	<p>(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, 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, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).  Published With international search report. Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.</p>	

(54) Title: SYSTEM AND METHOD FOR CREATING AND MANIPULATING INFORMATION CONTAINERS WITH DYNAMIC REGISTERS

(57) Abstract

A system for creating and manipulating information containers with dynamic registers on a multi-user computer system, or computer network comprises an interactive information container, a container editor, a search interface, a user profile, system-wide hierarchical container gateways (site 7), interactive and evolving container registers, a data collection means, a data reporting means, an analysis engine with editor, an executing engine with editor, and a means of communicating with other computers, computer networks, or digital-based public or published media. The container editor provides an authoring user with the capacity to encapsulate any information component such as a file, set, database, network, event or process, and a set of parameters of multiple container registers to govern the interaction of that container with other containers or processes. The container registers include system-defined, system-alterable, user-defined and user-alterable registers.



**FOR THE PURPOSES OF INFORMATION ONLY**

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakistan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		



## SYSTEM AND METHOD FOR CREATING AND MANIPULATING INFORMATION CONTAINERS WITH DYNAMIC REGISTERS

5

### BACKGROUND OF THE INVENTION

#### 1. Field of the Invention

The present invention relates generally to computer systems in a multi-user mainframe or mini computer system, a client server network, or in local, wide area or public networks, and in particular, to computer networks for creating and manipulating information containers with dynamic interactive registers in a computer, media or publishing network, in order to manufacture information on, upgrade the utility of, and develop intelligence in, a computer network by offering the means to create and manipulate information containers with dynamic registers.

#### 15 2. Description of the Related Art

In the present day, querying and usage of information resources on a computer network is accomplished by individuals directing a search effort by submitting key words or phrases to be compared to those key words or phrases contained in the content or description of that information resource, with indices and contents residing in a fixed location unchanging except by human input. Similarly, the class of storage medium upon which information resides, its class and subclass organizational structures, and its routes of access all remain fundamentally unaltered by ongoing user queries and usage. Only the direct and intended intervention of the owner of the information content or computer hosting site changes these parameters, normally accomplished manually by programmers or systems operators at their own discretion or the discretion of the site owner.

There exists currently in the art a limited means of interfacing a computer user with the information available on computer networks such as the world wide web. Primarily, these means are search engines. Search engines query thousands or tens of thousands of index pages per second to suggest the location of information while the user waits. While factual information can be accessed, the more complex, particular or subtle the inquiry, the more branches and sub-branches need to be explored in a time consuming fashion in order to have any chance of success. Further, there are no such automatic devices that reconstruct the information into more useful groupings or makes it more accessible according to factors

attached to the content by the content creator such as the space or time relevancy of its content, or factors attached to the content by the system's compilation and analysis of the accumulated biography of that specific content's readership.

5 The utility of wide area and public computer networks is thus greatly limited by the static information model and infrastructure upon which those networks operate.

One problem is that on a wide area or public network, specific content such as a document remains inert, except by the direct intervention of users, and is modified neither by patterns or history of usage on the network, or the existence of other content on the network.

10 Another problem is that content does not reside in an information infrastructure conducive to reconstruction by expert rule-based, fuzzy logic, or artificial intelligence based systems. Neither the intelligence of other information users nor the expert intelligence of an observant network computer system can be utilized in constructing, or re-constructing information resources. Where content resides in a fixed location and structure, "information" becomes something defined by the mind of the information provider rather than the mind of the  
15 information user, where the actual construction and utility of information exists. Information remains, like raw ore, in an unrefined state.

Another problem is that the class of storage medium upon which data resides cannot be system or user managed and altered according to the actual recorded and analyzed hierarchically graded usage of any given information resource residing on that storage medium except by  
20 statistical analysis of universal, undefined "hits" or visits to that page or site.

Another problem is that information resource groupings remain fixed on the given storage medium location according to the original installation by the resource author, not altered according to the actual recorded and analyzed hierarchically graded usage of that given information resource. Content itself remains inert, with no possibility of evolution.

25 A further problem with the prior art is that neither the search templates generated by those more knowledgeable in a given field of inquiry, nor the search strategies historically determined to be successful, or system-constructed according to analyses of search strategies historically determined to be successful, are available to inquiring users. A search template is here defined as one or more text phrases, graphics, video or audio bits, alone or in any defined  
30 outline or relational format designed to accomplish an inquiry. Internet or wide area network search may return dozens of briefs to a keyword or key phrase inquiry sometimes requiring the

time-consuming examination of multiple information resources or locations, with no historical relation to the success of any given search strategy.

5 A further problem is that there is limited means to add to, subtract from, or alter the information content of documents, databases, or sites without communicating with the owners or operators of those information resources, e.g., contacting, obtaining permission, negotiating and manually altering, adding or subtracting content. Additionally, once so altered, there is not a means to derive a proportionate value, and thereby a proportionate royalty as the information is used.

10 A final problem is that the physical residence of a body of data or its cyberspace location may not serve its largest body of users in the most expedient manner of access. Neither the expert intelligence of other information users nor the expert intelligence of an observant computer system is presently utilized by inherent network intelligence to analyze, re-design and construct access routes to information medium except by statistical analysis of universal, undefined "hits" or visits to that page or site.

15 Therefore, there is a need for a system and methods for creating and manipulating information containers with dynamic interactive registers defining more comprehensive information about contained content in a computer, media or publishing network, in order to manufacture information on, upgrade the utility of, and develop intelligence in, a computer network by providing a searching user the means to utilize the searches of other users or the  
20 historically determined and compiled searches of the system, a means to containerize information with multiple registers governing the interaction of that container, a means to re-classify the storage medium and location of information resources resident on the network, a means to allow the reconstruction of content into more useful formations, and a means to reconstruct the access routes to that information.

25

### SUMMARY OF THE INVENTION

The present invention is a system and methods for manufacturing information on, upgrading the utility of, and developing intelligence in, a computer or digital network, local, wide area, public, corporate, or digital-based, supported, or enhanced physical media form or  
30 public or published media, or other by offering the means to create and manipulate information containers with dynamic registers.

The system of the present invention comprises an input device, an output device, a processor, a memory unit, a data storage device, and a means of communicating with other computers, network of computers, or digital-based, supported or enhanced physical media forms or public or published media. These components are preferably coupled by a bus and  
5 configured for multi-media presentation, but may also be distributed throughout a network according to the requirements of highest and best use.

The memory unit advantageously includes an information container made interactive with dynamic registers, a container editor, a search interface, a search engine, a search engine editor, system-wide hierarchical container gateways interacting with dynamic container  
10 registers, a gateway editor, a register editor, a data collection means with editor, a data reporting means with editor, an analysis engine with editor, an executing engine with editor, databases, and a means of communicating with other computers as above. These components may reside in a distributed fashion in any configuration on multiple computer systems or networks.

The present invention advantageously provides a container editor for creating  
15 containers, containerizing storing information in containers and defining and altering container registers. A container is an interactive nestable logical domain configurable as both subset and superset, including a minimum set of attributes coded into dynamic interactive evolving registers, containing any information component, digital code, file, search string, set, database, network, event or process, and maintaining a unique network-wide lifelong identity.

The container editor allows the authoring user to create containers and encapsulate any  
20 information component in a container with registers, establishing a unique network lifelong identity, characteristics, and parameters and rules of interaction. The authoring user defines and sets the register with a starting counter and/or mathematical description by utilizing menus and simple graphing tools or other tools appropriate to that particular register. The registers  
25 determine the interaction of that container with other containers, system components, system gateways, events and processes on the computer network.

Containers and registers, upon creation, may be universal or class-specific. The editor provides the means to create system-defined registers as well as the means to create other registers. The editor enables the register values to be set by the user or by the system, in which  
30 case the register value may be fixed or alterable by the user upon creation. Register values are

evolving or non-evolving for the duration of the life of the container on the system. Evolving registers may change through time, space, interaction, system history and other means.

System-defined registers comprise: (1) an historical container register, logging the history of the interaction of that container with other containers, events and processes on the network, (2) an historical system register, logging the history of pertinent critical and processes on the network, (3) a point register accumulating points based upon a hierarchically rated history of usage, (4) an identity register maintaining a unique network wide identification and access location for a given container, (5) a brokerage register maintaining a record of ownership percentage and economic values, and others.

The present invention also includes user-defined registers. User defined registers may be created wholly by the user and assigned a starting value, or simply assigned value by the user when that register is pre-existent in the system or acquired from another user, and then appended to any information container, or detached from any container.

Exemplary user-defined registers comprise (1) a report register, setting trigger levels for report sequences, content determination and delivery target, (2) a triple time register, consisting of a range, map, graph, list, curve or other representation designating time relevance, actively, assigning the time characteristics by which that container will act upon another container or process, passively, assigning the time characteristics by which that container be acted upon by another container or process, and neutrally, assigning the time characteristics by which that container will interact with another container or process, (3) a triple space register, consisting of a range, map, graph, list, curve or other representation designating the domain and determinants of space relevance, actively, assigning the space characteristics by which that content will act upon another container or process, passively, assigning the space, characteristics by which that content will be acted upon by another container or process, and neutrally, assigning the space characteristics by which that container will interact with another container or process, (4) a domain of influence register, determining the set, class and range of containers upon which that container will act, (5) a domain of receptivity register, determining the set, class and range of containers allowed to act upon that container, (6) a domain of neutrality register, determining the set, class and range of containers with which that container will interact, (7) a domain of containment register, determining the set, class and range of containers which that container may logically encompass, (8) a domain of inclusion register, determining the set, class and

range of containers by which that container might be encompassed, (9) an ownership register, recording the original ownership of that containers, (10) a proportionate ownership register, determining the proportionate ownership of that containers, (11) a creator profile register, describing the creator or creators of that container, (12) an ownership address register,  
5 maintaining the address of the creator or creators of that container, (13) a value register, assigning a monetary or credit value to that container, and (14) other registers created by users or the system.

Containers are nestable and configurable as both subset and superset and may be designated hierarchically according to inclusive range, such as image component, image, image  
10 file, image collection, image database, or if text, text fragment, sentence, paragraph, page, document, document collection, document, database, document library, or any arrangement wherein containers are defined as increasingly inclusive sets of sets of digital components.

The present invention also includes, structurally integrated into each container, or strategically placed within a network at container transit points, unique gateways, nestable in a  
15 hierarchical or set and class network scheme. Gateways gather and store container register information according to system-defined, system-generated, or user determined rules as containers exit and enter one another, governing how containers system processes or system components interact within the domain of that container, or after exiting and entering that container, and governing how containers, system components and system processes interact  
20 with that unique gateway, including how data collection and reporting is managed at that gateway. The gateways record the register information of internally nested sub and superset containers, transient containers and search templates, including the grade of access requested, and, acting as an agent of an analysis engine and execution engine, govern the traffic and interaction of those containers and searches with the information resource of which they are the  
25 gateway and other gateways. The gateways' record of internally nested and transient container registers, and its own interaction with those containers, is made available, according to a rules-based determination, to the process of the analysis engine by the data collection and/or data reporting means.

The present invention also includes a means of data storage at any given gateway.

30 The present invention also includes a data collection means, residing anywhere on the network, or located at one or more hierarchical levels of nestable container gateways for

gathering information from other gateways and analysis engines according to system, system-generated or user determined rules. The data collection means manages the gathering of data regarding network-wide user choices, usage and information about information, by collecting it from container and gateway registers as those containers and gateways pass through one another. Such statistics as frequency, pattern, and range of time, space and logical class is collected as directed by the analysis engine, and made that data available to the analysis engine by advancing it directly to the analysis engine, or incrementally, to the next greater hierarchically inclusive collection level. The rules of data collection may be manually set or altered by the system manager, or set by the system and altered by the system in its evolutionary capacity.

The present invention also includes a data reporting means, located at one or more hierarchical levels of nestable container gateways for submitting information to other gateways and analysis engines according to system, system-generated or user determined rules. The data reporting means manages the sending of data from the registers, gateways and search templates in a frequency, pattern, and range of time, space and logical class as directed by the analysis engine, and makes that data available to the analysis engine by advancing it directly to the analysis engine, or incrementally to the next greater hierarchically inclusive reporting level. The rules of data collection may be manually set or altered by the system manager, or set by the system and altered by the system in its evolutionary capacity. The data reporting means may be established to work in concert, in redundancy, or in contiguous or interwoven threads of hierarchically nested containers.

The present invention also includes an analysis engine that receives, reports and collects information regarding the interaction of user searches with gateways and container registers, as well as container registers with other container registers, and container registers with gateways. The analysis engine analyzes the information submitted by the gateways and instructs the execution engine to create new information containers, content assemblages, storage schemes, access routes, search templates, and gateway instructions. The analysis engine includes an editor that provides a system manager with a means of editing the operating principles of that engine, governing data reporting, data collection, search template loading, gateway instructions, and other.

The present invention also includes an execution engine, fulfilling the instructions of the analysis engine, to create new information containers, content sun and superset assemblages, storage schemes, access routes, search templates, and gateway instructions. The execution engine includes an editor that provides a system manager with a means of editing the operating principles of that engine, governing data reporting, data collection, search template loading, gateway instructions, and other.

The present invention also includes a search interface or browser. The search interface provides a means for a searching user to submit, record and access search streams or phrases generated historically by himself, other users, or the system. Search streams or phrases of other users are those that have been historically determined by the system to have the highest probability of utility to the searching user. Search streams or phrases generated by the system are those that have been constructed by the system through the analysis engine based upon the same criteria.

15

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

FIG. 1 is a block diagram of a first and preferred embodiment of a system constructed according to the present invention.

FIG. 2 A is block diagram of a preferred embodiment of the memory unit.

FIG. 2 B is an exemplary embodiment of a computer network showing computer servers, personal computers, workstations, Internet, Wide Area Networks, Intranets in relationship with containers and gateways.

FIG. 2B1 is an exemplary embodiment of a computer network showing computer servers, personal computers, workstations, Internet, Wide Area Networks, Intranets in relationship with containers and gateways and exemplary locations of gateway storage in proximity to one or more of the various sites.

FIGS. 2C through 2H are exemplary embodiments in block diagram form of computer network components showing a possible placement of nested containers, computer servers, gateways, and the software components named in Fig. 2 A on a network.

FIG. 3A is a graphical representation for one embodiment of a container having a plurality of containers nested within that container.



FIG. 3C is a drawing showing elements that might be logically encapsulated by a container. FIG. 4 is a drawing of an information container showing a gateway and registers logically

encapsulating containerized elements.

5 FIG. 5 is a flowchart showing a preferred method for the containerization process and container editor operating on the communication device.

FIG. 6 is a flowchart showing a preferred method for searching for containers within a node.

10 FIG. 7 is a flowchart further showing a preferred method for searching for containers over one or more gateways.

FIG. 8 is a flowchart showing a method for performing the data collection and reporting on containers.

FIG. 9 is a flowchart showing the operation of the analysis engine.

FIG. 10 is a flowchart showing the operation of the execution engine.

15 FIG. 11 is a flowchart showing the operation of the gateway editor.

FIG. 12 is a flowchart showing the operation of the gateway process.

FIG. 13A is a drawing showing an example of nested containers, gateways, registers, analysis engines and an execution engine prior to container reconstruction as depicted in 13 B, 13 C and 13 D.

20 FIG. 13B is a drawing showing the reconstructed nested containers of Figure 13A.

FIG. 13C is a drawing showing further reconstruction of nested containers, with a container relocated to reside within another container.

FIG. 13D is a drawing showing a flowchart of the reconstruction process

FIG. 14 is a drawing showing the screen interface of the container editor.

25 FIG. 15 is a drawing showing the screen interface of the gateway editor.

FIG. 16 is a drawing showing the screen interface of the search interface.

FIG. 17 is a drawing of a generic application program showing a drop-down menu link, and a button link to the containerization process or container editor.

30 **DESCRIPTION OF THE PREFERRED EMBODIMENT**

**THE SYSTEM**

Referring now to FIG. 1, a preferred embodiment of a system 10 for creating and manipulating information containers with dynamic interactive registers in a computer, media, or publishing network 201 in order to manufacture information on, upgrade the utility of, and develop intelligence in that network 201, is shown. The system 10 preferably comprises an input device 24, an output device 16, a processor 18, a memory unit 22, a data storage device 20, and a communication device 26 operating on a network 201. The input device 24, an output device 16, a processor 18, a memory unit 22, a data storage device 20, are preferably coupled together by a bus 12 in a von Neumann architecture. Those skilled in the art will realize that these components 24, 16, 18, 22, 20, and 26 may be coupled together according to various other computer architectures including any physical distribution of components linked together by the communication device 26 without departing from the spirit or scope of the present invention, and may be infinitely nested or chained, both as computer systems within a network 202, and as networks within networks 201.

The output device 16 preferably comprises a computer monitor for displaying high-resolution graphics and speakers for outputting high fidelity audio signals. The output device 16 is used to display various user interfaces 110, 125, 210, 300, 510, 610, 710, as will be described below, for searching for and containerizing information, and editing the container gateways, containers, container registers, the data reporting means and the data collection means, and the search, analysis and execution engines. The author uses the input device 24 to manipulate icons, text, charts or graphs, or to select objects or text, in the process of packaging, searching or editing in a conventional manner such as in the Macintosh or Windows operating systems.

The processor 18 preferably executes programmed instruction steps, generates commands, stores data and analyzes data configurations according to programmed instruction steps that are stored in the memory unit 22 and in the data storage device 20. The processor 22 is preferably a microprocessor such as the Motorola 680(x)0, the Intel 80(x)86 or Pentium, Pentium II, and successors, or processors made by AMD, or Cyrix CPU of the any class.

The memory unit 22 is preferably a predetermined amount of dynamic random access memory, a read-only memory, or both. The memory unit 22 stores data, operating systems, and programmed instructions steps, and manages the operations of all hardware and software components in the system 10 and on the network 201, utilizing the communication device 26

whenever necessary or expeditious to link multiple computer systems 202 within the network 201.

The data storage device 20 is preferably a disk storage device for storing data and programmed instruction steps. In the exemplary embodiment, the data storage device 20 is a  
5 hard disk drive. Historical recordings of network usage are stored on distributed and centralized data storage devices 20.

The preferred embodiment of the input device 24 comprises a keyboard, microphone, and mouse type controller. Data and commands to the system 10 are input through the input device 24.

10 The present invention also includes a communication device 26. The communication device 26 underlies and sustains the operations of, referring now also to Fig 2 the analysis 400 and execution 500 engines, the data reporting 600 and collection 700 means, the container editor 110, the search interface 300, and the search engine 320, providing the means to search, access, move, copy, utilize or otherwise perform operations with and on data. The communication  
15 device 26 utilizes one or more of the following technologies: modem, infrared, microwave, laser, photons, electrons, wave phenomena, cellular carrier, satellite, laser, router hub, direct cabling, physical transport, radio, broadcast or cable TV or other to communicate with other computers, digital-supported television, computer networks, or digital-based or supported public or published media, or physical media forms, on any a local, wide area, public, or any  
20 computer-based computer supported, or computer interfaced network, including but not limited to the Internet. It also allows for the functioning and distribution of any container 100 or container component herein described to reside anywhere on any computer system in any configuration on that local, wide area, public, or corporate computer-based or computer related network, or digital-based or supported media form.

25 Referring now to Figure 2 A, a preferred embodiment of the memory unit 22 is shown. The memory unit includes: an interactive information container 100, a container editor 110, container registers 120, a container register editor 125, system-wide hierarchical container gateways 200, gateway storage 205, gateway editors 210, engine editors 510, a search interface 300, search engine 320, analysis engine 400, execution engine 500, a data reporting module,  
30 600, a data reporting editor 610, a data collection module 700, a data collection editor 710, screen interfaces (GUI's) 936, menu or access buttons from generic computer programs 937,

and databases 900, all residing in memory optimized between a data storage means 20 such as magnetic, optical, laser, or other fixed storage, and a memory means 22 such as RAM. The memory unit 22 functions by operating on communications network 12 with a communication device 26 on multiple computer systems 202 within the network 201. These components will  
5 be described first briefly in the following paragraphs, then in more detail with reference to Figures 3 A through 17.

Those skilled in the art will realize that these components might also be stored in contiguous blocks of memory, and that software components or portions thereof may reside in the memory unit 22 or the data storage means 20.

10 The present invention includes information containers 100 as noted above. The information container 100 is a logically defined data enclosure which encapsulates any element or digital segment (text, graphic, photograph, audio, video, or other), or set of digital segments, or referring now to FIG. 3 C, any system component or process, or other containers or sets of containers. A container 100 at minimum includes in its construction a logically encapsulated  
15 portion of cyberspace, a register and a gateway. A container 100 at minimum encapsulates a single digital bit, a single natural number or the logical description of another container, and at maximum all defined cyberspace, existing, growing and to be discovered, including but not limited to all containers, defined and to be defined in cyberspace. A container 100 contains the code to enable it to interact with the components enumerated in 2 A, and to reconstruct itself  
20 internally and manage itself on the network 201.

The container 100 also includes container registers 120. Container registers 120 are interactive dynamic values appended to the logical enclosure of an information container 100, and serve to govern the interaction of that container 100 with other containers 100, container gateways 200 and the system 10, and to record the historical interaction of that container 100 on  
25 the system 10. Container registers 120 may be values alone or contain code to establish certain parameters in interaction with other containers 100 or gateways 200.

The present invention also includes container gateways 200. Container gateways 200 are logically defined gateways residing both on containers 100 and independently in the system 10. Gateways 200 govern the interactions of containers 100 within their domain, and alter the  
30 registers 120 of transiting containers 100 upon ingress and egress.

The present invention also includes container gateway storage 205 to hold the data collected from registers 120 of transient containers 100 in order to make it available to the data collection means 700 and the data reporting means 600, and to store the rules governing the operations of its particular gateway 200, governing transiting containers upon ingress and egress, and governing the interactive behavior of containers 100 within the container 100 to which that gateway 200 is attached. Gateway storage 205 may be located on gateways 200 themselves, containers 100 or anywhere on the network 202, 201, including but not limited to Internet, Intranet, LAN, WAN, according to best analysis and use.

The memory unit 22 also includes an execution engine 500 to perform the functions on the system 10 as directed by the analysis engine after its analysis of data from the data reporting means 600, the data collection means 700, and the search interface 300.

The memory unit 22 also includes a search interface 300, by which the user enters, selects or edits search phrases or digital strings to be used by the search engine 320 to locate containers 100.

The memory unit 22 also includes an analysis engine 400 which performs rules based or other analysis upon the data collected from the search interface 300 and the data collection 700 and data reporting 600 means.

The memory unit 22 also includes a data reporting means 600, by which means the information collected by gateways 200 from transient containers 100 is sent to the analysis engine 400.

The memory unit 22 also includes a data collection means 700, by which means the analysis engine 400 gathers the information collected by gateways 200 from transient containers 100.

The memory unit 22 also includes a container editor 110 for creating, selecting, acquiring, modifying and appending registers 120 and gateways 200 to containers 100, for creating, selecting, acquiring, and modifying containers, and for selecting content 01 to encapsulate.

The memory unit 22 also includes a register editor 125, for creating, selecting, acquiring and modifying container registers 120 and establishing and adjusting the values therein.

The memory unit 22 also includes a gateway editor 210, by which means the user determines the rules governing the interaction of a given gateway 210 with the registers 120 of

transient containers 100, governing transiting containers upon ingress and egress, and governing the interactive behavior of containers within the container to which that gateway is attached.

The memory unit 22 also includes databases 900, by which means the analysis engine 400, the execution engine 500, the gateways 100, the editors 110, 125, 210, 510, 610, 710, and the search interface 300, store information for later use.

The memory unit 22 present invention also includes a search engine 320 by which means the user is able to locate containers 100 and, referring now to Fig. 4, containerized elements 01.

The memory unit 22 present invention also includes an engine editor 510, by which means the user establishes the rules and operating procedures for the analysis engine 400 and the execution engine 500.

The memory unit 22 present invention also includes a reporting means editor 610, by which means the user establishes the rules and schedule under which the information collected by gateways 200 from transient containers 100 will be sent to the analysis engine 400.

The memory unit 22 present invention also includes a collection means editor 710, by which means the user establishes the rules and schedule under which the analysis engine 400 will gathers the information collected by gateways 200 from transient containers 100.

The memory unit 22 present invention also includes screen interfaces (GUI's) 936, specifically designed to simplify and enhance the operations of the container editor 110, the gateway editor 210, and the search interface 300.

The present invention also includes a menu or button access 937, by which a user utilizing any generic computer program may access the system 10 or the container editor 110 from a menu selection(s) or button(s) within that program.

The present invention also includes a computer, media or publishing network 201, comprising computers, digital devices and digital media 202 and a communication device 26, within which the components enumerated in Fig. 2 A interact, compiling, analyzing, and altering containers 100 and the network 201 according to information gathered from container registers 120.

The memory unit 22 also includes one or more computers 202, by which means the components of Fig 1 sustain the operations described in Fig. 2 A.

The memory unit 22 also includes flat or relational databases 900, used where, and as required. Databases are used to store search phrases, search templates, system history for the

analysis engine and execution engine, container levels and container, sites and digital elements, or any and all storage required to operate the system.

Referring now to FIG. 2 B, a drawing of a computer network 201 as a system 10, showing a possible placement of nested containers 100, computer servers, gateways 200, on the sites described below. (Note: Fig. 2 B utilizes in parts the same numbering scheme as Fig. 13 A, 13 B, 13 C, 13 D and as Fig. 2 A.) In FIG. 2 B various exemplary sites are shown, any or all of which might interact dynamically within the system. Site 1 shows a single workstation with a container and gateway connected to an Intranet. (Individual containers may be a floppy or CD-Rom to be downloaded or inserted.) Site 2 shows a server with a gateway in relationship to various containers.. Site 3 shows an Internet web page with a container residing on it. Site 4 shows a personal computer with containers and a gateway connected to the Internet. Site 5 shows a configuration of multiple servers and containers on a Wide Area Network.. Site 6 shows a workstations with a gateway and containers within a container connected to a Wide Area Network. Site 7 shows an independent gateway, capable of acting as a data collection and data reporting site as it gathers data from the registers of transiting containers, and as an agent of the execution engine as it alters the registers of transient containers. A container 100 contains the code to enable it to interact with the components enumerated in 2A, and to reconstruct itself internally and manage itself on the network 201. The code resides in and with the container in its registers and gateway definitions and controls. Additional system code resides in all sites to manage the individual and collective operation and oversight of the components enumerated in 2A, with the specific components distributed amongst the sites according to the requirements of optimization.

Referring now to Fig. 2 B 1 various exemplary sites are shown as described above in Fig. 2 B, with the addition of possible location of one or more gateway storage 205 locations.

Referring now to Figures 2 C through 2 H, various exemplary sites with one or more of the logical components of the system 10 in relationship are shown. Site 1 comprises an interactive information container 100, a container editor 110, container registers 120, a container register editor 125, system-wide hierarchical container gateways 200, gateway storage 205, gateway editors 210, engine editors 510, a search interface 300, search engine 320, analysis engine 400, execution engine 500, a data reporting means 600, a data reporting means editor 610, a data collection means 700, a data collection means editor 710, and databases 900, all

residing on data storage means 20, utilizing the memory unit to function 22, operating on communications network 12 with a communication device 26.

Site 2 comprises an interactive information container 100, a container editor 110, container registers 120, a container register editor 125, system-wide hierarchical container gateways 200, gateway storage 205, gateway editors 210, engine editors 510, search engine 320, analysis engine 400, execution engine 500, a data reporting means 600, a data reporting means editor 610, a data collection means 700, a data collection means editor 710, and databases 900, all residing on data storage means 20, utilizing the memory unit to function 22, operating on communications network 12 with a communication device 26.

Site 3 comprises an interactive information container 100, a container editor 110, container registers 120, a container register editor 125, hierarchical container gateways 200, gateway storage 205, gateway editors 210, and databases 900, all residing on data storage means 20, utilizing the memory unit to function 22, operating on communications network 12 with a communication device 26.

Site 4 comprises an interactive information container 100, a container editor 110, container registers 120, a container register editor 125, hierarchical container gateways 200, gateway storage 205, gateway editors 210, a search interface 300, and databases 900, all residing on data storage means 20, utilizing the memory unit to function 22, operating on communications network 12 with a communication device 26.

Site 5 comprises an interactive information container 100, container registers 120, a container register editor 125, hierarchical container gateways 200, gateway storage 205, and databases 900, all residing on data storage means 20, accessed and utilized by non-resident memory unit 22, operating on communications network 12 with a communication device 26.

Site 6 includes an independent analysis engine 400, execution engine 500, data collection means 700, and data reporting means 600 gateway editors 210, engine editors 510, a data reporting means editor 610, a data collection means 700, a data collection means editor 710, and databases 900, all residing on data storage means 20, utilizing the memory unit to function 22, operating on communications network 12 with a communication device 26.

Referring now to FIG. 3 A and FIG. 3 B, a block diagram of several nested information containers is shown, including examples of elements, e.g., code 1100, text 1200, audio 1300, video 1400, photograph 1500, graphic images 1600, and examples of possible container level



classifications in increasing size, e.g., element 10900000, document 10800000, database 10700000, warehouse 10600000, domain 10500000, and continuing increasingly larger on Fig 3 (B), subject 10400000, field 10300000, master field 10200000, species 10100000. Containers may be infinitely nested and assigned any class, super class or sub class scheme and description by the creator of the container to govern nesting within that container. In addition to digital elements, containers may also include system process and components, including containerization itself.

Referring now to FIG. 3 C, a block diagram of an information container system is shown, listing, without any relationship indicated, some of the possible system components and processes, or sets thereof, that may be encapsulated as elements 01 in an information container 100. An information container 100 may include one or more of the following: any unique, container 100, gateway 200, output device 16, input device 24, output device process 160, input device process 240, data storage device 20, data storage device process 2000, processor 18, bus 12, content 01, search process 02, interface 04, memory unit 22, communication device 26, search interface 300, search process 98, network 201, class of device, process or content 999, class of process at any unique class of device 990, process at any unique device 99, editor 110, 125, 210, 510, 610, 710, engine 320, 400, 500, containerization process 1098, or process 08.

Any container may include (n) other containers, to infinity. The use of value evolving container registers 120 in conjunction with gateways 200, data reporting modules 600, data collection modules 700, the analysis engine 400, and the execution engine 500 provides the information container 100 with extensive knowledge of the use, operation of its internal contents, prior to, during and after those contents' residence within that container 100, and extensive knowledge of the use, operation and contents of the system 10 external to itself, and allows the container 100 to establish and evolve its own identity and course of interaction on the system 10. Further, containers 100, as logical enclosures, can exist and operate independent of their digital contents, whether encapsulating audio, video, text, graphic, or other.

Referring now to FIG. 4, a block diagram of an information container 100 is shown. The information container 100 is a logically defined data enclosure which encapsulates any element, digital segment (text, graphic, photograph, audio, video, or other), set of digital segments as described above with reference to FIG. 3 (C), any system component or process, or other

containers or sets of containers. The container 100 comprises the containerized elements 01, registers 120 and a gateway 200.

Registers 120 appended to an information container 110 are unique in that they operate independently of the encapsulated contents, providing rules of interaction, history of interaction, identity and interactive life to that container 100 through the duration of its existence on a network 201, without requiring reference to, or interaction with, its specific contents. They enable a container 100 to establish an identity independent of its contents. Additionally, registers 120 are unique in that their internal values evolve through interaction with other containers 100, gateways 200, the analysis engine 400, the execution engine 500, and the choices made by the users in the search interface 300, the container editor 110, the register editor 125, the gateway editor 210, the engine editor 510. Registers 120 are also unique in that they can interact with any register of a similar definition on any container 100 residing on the network 201, independent of that container's contents. Registers 120, once constructed, may be copied and appended to other containers 100 with their internal values reset, to form new containers. Register values, when collected at gateways 200 and made available to the analysis engine 400 through the data collection means 700 and the data reporting means 600, provide an entirely new layer of network observation and analysis and operational control through the execution engine 500. Registers 120 accomplish not only a real time information about information system, but also a real time information about information usage on a network. Further, because the user base of a network determines usage, the system 10, in gathering information about information usage, is observing the choices of the human mind. When these choices are submitted to the analysis of a rules-based or other analysis engine 400, the system 10 becomes capable of becoming progressively more responsive to the need of the user base, in effect, learning to become more useful by utilizing the execution engine 500 to create system-wide changes by altering the rules of gateway 200 interaction and thereby altering the registers 120 of transient containers 100 and establishing a complete evolutionary cycle of enhanced utility.

Further, in establishing the pre-defined registers as described in the following four paragraphs, the following unique aspects of information about information are utilized for the first time: 1) the dynamic governance of information according to its utility through time, in active, passive and neutral aspects, as explained below; 2) the dynamic governance of

information according to its utility through space in active, passive and neutral aspects, as explained below; 3) the dynamic governance of information according to its ownership, as explained below; 4) the dynamic governance of information according to its unique history of interaction as an identity on a network, as explained below; 5) the dynamic governance of information according to the history of the system on which it exists, as explained below; 6) the dynamic governance of information according to established rules of interaction, in active, passive and neutral aspects, as explained below; 7) the dynamic governance of information according to the profile of its creator, as explained below; 8) the dynamic governance of information according to the value established by its ongoing usage, as explained below; 9) the dynamic governance of information according to its distributed ownership, as explained below; 10) the dynamic governance of information according to what class of information it might be incorporated into, and according to what class of information container it might incorporate, as explained below; 11) the dynamic governance of information according to self-reporting, as explained below.

Referring now to Fig 4, registers 120 may be (1) pre-defined, (2) created by the user or acquired by the user, or (3) system-defined or system-created. Pre-defined registers 120 are those immediately available for selection by the user within a given container editor as part of that container editor, in order that the user may append any of those registers 120 to a container 100 and define values for those registers 120 where required. Registers 120 created by the user are those conceived and created by a specific user or user group and made immediately available for selection by the user or user group in conjunction with any of a wide number of container editors, in order that the user may append any of those registers 120 to a container 100 and define values for those registers 120 where required. Registers 120 acquired by the user are those registers existing network-wide 201, created by the user base, that might be located and acquired by the user in order that the user may append any of those registers 120 to a container 100 and define values for those registers 120 where required. System-defined registers are those registers whose values are set and/or controlled by the system 10. System-created registers are those registers created by the system 10.

Registers 120 are user or user-base created or system-created values or ranges made available by the system 10 to attach to a unique container, and hold system-set, user-set, or system-evolved values. Values may be numeric, may describe domains of time or space, or may

provide information about the container 100, the user, or the system 10. Registers 120 may be active, passive or interactive and may evolve with system use. Pre-defined registers include, but are not limited to, system history 110000, container history 101000, active time 102000, passive time 103000, neutral time 104000, active space 111000, passive space 112000, neutral space 113000, containment 105000, inclusion 106000, identity 114000, value 115000, ownership 107000, ownership addresses 116000, proportionate ownership 117000, creator profile 108000, receptivity 118000, influence 119000, points 109000, others 120000, reporting 121000, neutrality 122000, acquire 123000, create 124000, content title 125000, content key phrase(s) 126000, and content description 127000, security 12800, and parent rules 129000.

10 Pre-defined registers comprise an historical container register 101000, logging the history of the interaction of that container 100 with other containers, events and processes on the network 201, an historical system register 110000, logging the history of pertinent critical and processes on the network, a point register 109000 accumulating points based upon a hierarchically rated history of usage, an identity register 114000 maintaining a unique network  
15 wide identification and access location for a given container specifying a unique time and place of origin and original residence, a proportionate ownership register 117000 maintaining a record of ownership percentage and economic values, and others 120000.

User-defined registers include a report register 121000 setting trigger levels for report sequences, content determination and delivery target, three time registers, consisting of a range,  
20 map, graph, list, curve or other designating time relevance, 102000 assigning the time characteristics by which that container will act upon another container or process, 103000 assigning the time characteristics by which that container be acted upon by another container or process, and 104000 assigning the time characteristics by which that container will interact with another container or process, three space registers, consisting of a range, map, graph, list, curve  
25 or other designating the domain and determinants of space relevance, 111000 assigning the space characteristics by which that content will act upon another container or process, 112000 assigning the space, characteristics by which that content will be acted upon by another container or process, and 113000 assigning the space characteristics by which that container will interact with another container or process, a domain of influence register 119000, determining  
30 the set, class and range of containers upon which that container will act, a domain of receptivity register 118000, determining the set, class and range of containers allowed to act upon that

container, a domain of neutrality register 122000, determining the set, class and range of containers with which that container will interact, a domain of containment register 105000, determining the set, class and range of containers which that container may logically encompass, a domain of inclusion 106000 register, determining the set, class and range of containers by which that container might be encapsulated, an ownership register 107000, recording the original ownership of that containers, a creator profile register 108000, describing the creator or creators of that container, an ownership address register 116000, maintaining the address of the creator or creators of that container, a value register 115000, assigning a monetary or credit value to that container, other registers 120000 created by users or the system, a reporting register 121000, determining the content, scheduling and recipients of information about that container, a neutrality register 122000, an acquire register 123000, enabling the user to search and utilize other registers residing on the network, a create register 124000, enabling the user to construct a new register, a content title register 125000, naming the contents of the container, a content key register, 126000, identifying the container contents with a key phrase generated by the user and/or the system based upon successful usage of that phrase in conjunction with the utilization of the information within that container 100, a content description register 127000, identifying the container contents with additional description, a security register 128000, controlling container security, and a parent container register 129000, storing the rules governing container interaction as dictated by the parent (encapsulating) container.

The container also includes a gateway 200 and gateway storage 205.

Gateways 200 are logically defined passageways residing both on containers 100 and independently in the system 10. Gateways 200 govern the interactions of containers 100 encapsulated within their domain by reading and storing register 120 information of containers entering and exiting that container 100.

The present invention also includes container gateway storage 205. Gateway storage 205 stores information regarding the residence, absence, transience, and alteration of encapsulated and encapsulating containers 100, and their attached registers 120, **holding** the data collected from registers 120 of transient containers 100 in order to make it available to the data collection means 700 and the data reporting means 600, and storing the rules governing the operations of its particular gateway 200.

Referring now to FIG. 5, a flow chart of the preferred method for creating a container 100 is shown.

Input is received from the user selecting a container level through use of a drop-down menu 10100. A menu of all possible container classes within the subset and superset scheme of multiple hierarchically nested containers, i.e.; element, document, file, database, warehouse, domain, and more, is displayed on the output device 10200. Input is received from the user selecting a class 10300.

A graphic representation of a container in that class, with registers common to all containers as well as registers unique to its class is displayed 10301.

Input is received from the user choosing to "create" 10400, "edit" 10500, or "locate" 10600.

When the input of "create" 10400 is received from the user, a container template in that class appears 10410. Input from the user is then received adding or selecting a register 10540 to append to that container template. When input is received from the user adding a register, a list of registers that might be added to that class of container is made available to select 10550. Input is received from the user selecting a register 10560 and editing it 10570. The menu returns to "add or select" 10540.

If the input of "locate" 10600 is received from the user, the system prompts the user to enter the identity of the container or class of containers 10605. The system locates the container(s) 10610. Input is received from the user selecting a container 10620. The system prompts the user for a security code for permission to access the container for template use, or to alter its registers, or to alter its content 10630. . Input is received from the user entering a name and password providing access to one of the security levels 10640. Input is received from the user editing the container accordingly by transition to step 10500 and performing the steps for editing.

If the input of "edit" 10500 is received, a list of containers available to edit at that level is shown 10510. Input is received from the user selecting a container 10520. That container appears, available to edit 10530. Input is received from the user selecting "add" or "select" registers 10540 by the user clicking on the graphically depicted register, or from a drop down menu. Input is received from the user selecting the register to edit 10560. Input is received from the user selecting "modify" or "delete" for that register 10565. If input is received from the user

to "delete," that register is severed from the container. If input is received from the user to "modify", the register editor 10570 screen appropriate to that register appears, i.e., an x-y type graph to define a curve of relevant active time, in which the user manipulates the x-y termini, scale and curve, or a global map in which Input is received from the user selecting the locale of active space, whether zip code, city, county, state, country, continent, plant or other. When  
5 input is received from the user saving the definition, the screen returns to the main container screen to make another selection available. . Input is received from the user defining as many registers as he chooses. One of the registers may be named "new register." Input is received from the user selecting the new register, and if chosen by the user, defining a wholly unique and  
10 new kind of register by the user entering input into the register editor 125.

When the input is received from the user choosing to add a register, a list of registers that might be added to that class of container are made available to select 10550. Input is received from the user selecting a register 10560 and editing it 10570. The menu returns to "add or select" 10540, and in turn to Input – Select Container.

15 Input may then be received from the user choosing to add, modify, or delete the container contents 10700. Once the registers are defined, input is received from the user indicating completion and the interface reverts to the container editor. When input is received from the user choosing "select component" (to select the component to containerize) from the main menu bar 10700, a window appears allowing the user to select any file, component, or other  
20 container. If for example, the user were creating a warehouse container, and wishes to incorporate several databases into that container, input would then be received from the user selecting "database." The program would prompt the user for the location (directory) of that database or container. If the requested selection is not containerized, input may then be received from the user choosing to containerize the element at that time, after which the program returns  
25 to "select component." Once input is received from the user defining the database location, the program logically encases the directory or directories in the defined container. The above procedure may be repeated as many times as desired to include multiple databases within a single container. While logical simplicity would dictate that all containers within a container be of the same subset, it would be possible for input to be received from the user choosing  
30 containers of any subset to include in the container. When input is received from the user choosing "finished," the container is created with a unique network identity, preferably through

some combination of exact time and digital device serial number, or centralized numbering system, or other means. The container 100 contains all digital code, including data and program software from the selected items or containers.

Input may then be received from the user to publish the container 11100 at a user-  
5 identified or system suggested location 11200 to be selected 11400.

Input is received from the user to "publish", from the main menu bar 11100. Input is received from the user choosing to leave the container where it was created, move or copy it to another drive, directory, computer, or network the user designates, or select the location from location options offered by the system 11200, or submit, or duplicate and submit, the container  
10 to the analysis engine 400 for intelligent inclusion in other containers, thus allowing the system to publish the container as instructed or choose the residence of the container 11400.

If input is received from the user to choosing to "move," or "copy" a browse function allows the user to name the new location or browse a list of possible locations. If input is received from the user choosing to "submit," a browser function allows the user to name the  
15 analysis search engine 310 or browse a list of possible analyses engines. When input is received from the user choosing the residence of the container 11300, the program restores the search interface screen.

Referring now to FIG. 6, a flow chart of the method for searching for containers 100.

When input is received from the user selecting "search interface" from the main title bar,  
20 the search interface screen appears. The user is given the choice of containerizing selected content or requesting that container levels be displayed 30100. From a drop down menu another menu appears allowing input to be received from the user selecting the container level 30200. Input is received from the user selecting the container level (from the smallest component to the whole system) 30300.

25 Input is received 30310 from the user selecting the phrases, containers or components, which then are re-submitted to the same process, until the input is received from the user selecting a specific site or container.

The search phrase, whether containerized or not, is submitted simultaneously to the search engine 30400 and the analysis engine 30500.

30 The screen then reports in a selection menu, the number of applicable sites found by the search engine 30410, the number of historically proven applicable sites found by the analysis



engine 30410, the number of historically proven applicable containers at the selected container level or any container level found by the analysis engine 30410, and the number of historically proven new search phrases or digital segments found by the analysis engine 30320. . Input is received from the user selecting one of the named sets above 30330. If input is received from the user choosing the search engine, the search interface lists the applicable site titles with a brief description 30410. If input is received from the user choosing the site list of the analysis, the search interface lists the applicable site titles with a brief description 30410. . If input is received from the user choosing the container list of the analysis engine, the search interface lists the applicable container titles with a brief description 30410. . If input is received from the user selecting a container 30420, the system offers the means to view titles and descriptions of sub-containers at any chosen class level. . If input is received from the user choosing the phrase list of the analysis engine, the search interface lists the applicable phrases or digital segments with a brief description 30320. The search and search result cycle repeats until input is received from the user choosing to go to an individual container or site.

Input is received from the user entering text or any digital string describing his search objectives into a text or search box. When input is received from the user submitting the search string, the system provides the option of containerizing the search through the container editor 110. Once the search container 101 is created, the system restores the search interface 300 screen the user.

Input is received from the user selecting “search”, “supported search” or “both” from another drop-down menu and from submitting the search. When input is received from the user selecting “search” 30310, the search phrase is submitted to the search engine 30400, which searches both content and the appropriate container registers, as pre-indexed in the search engine, and returns a list of appropriate locations, components or containers. When input is received from the selecting “supported search”, the search phrase is submitted to the analysis engine search support, which returns a list, in a drop-down menu, of search phrases or individual containers, for any and all container levels, used by other users or created by the system and known to be historically successful for the described effort and the described searching user, as per the results of the analysis search engine. Input is received from the user selecting a new search phrase or specific container from the drop down menu 30330. When input is received from the user choosing a new search phrase, that phrase is also submitted to the

analysis engine 30500 which returns a list of pre-compiled historically proven sites, components or containers associated with that search phrase 30320. Input is received from the user choosing a selection 30420 and the system calls up that specific site, container or component. If input is received from the user selecting a specific site, container or component at any time during the search process, that element is called up by the system 30440.

Input is received from the user choosing to containerize a search or select a container level in which to search 30100. When input is received from the user choosing to containerize the search, the software moves to the container editor as described in Fig. 5, and then returns the user to the search interface screen. Input is received from the user selecting to search a specific container level or the whole network. The system shows the available levels 30200. Input is received from the user selecting a container level 30300, and entering the text or digital component comprising the search string 30310. The system searches the containers 30400 while simultaneously submitting the search string to the analysis engine 30500. While the system is accessing containers, sites or templates 30700, the analysis engine 30500 inquires of the appropriate database 30600 to access historically successful containers, sites or search templates corresponding to the search request 30700, which is then shown on another portion or option of the search interface, either as available containers or sites 30410 or as search template options 30320. On one portion or option of the search interface screen the corresponding containers or sites are listed and/or previewed for selection 30410. Input is received from the user selecting the container to access 30420. The system accesses that container 30430 and shows it on the screen 30440 for user review. Input is received from the user selecting an operation, i.e., preview, read, purchase, move, copy, lease, in any composed schedule with operations assigned specific values 30460, and the system obtains the specified result 30470. The selection of the operation including any interaction with any uniquely defined container 100 is recorded 30800 by the container gateway (Fig. 2 A, 200), stored in the gateway storage 205 and made available to the analysis engine (Fig. 9) by the data collection and reporting means (Fig. 8). Reporting and collection occurs on a regular basis according to user determined times or rules. The analysis engine compiles and analyzes selections according to various rules-based systems applicable to the particular container area of residence in cyberspace.

Input is received from the user selecting the container or site 30410, proceeding as described above, or selecting a search template 30330, and editing it to re-enter the search

30310. All operations on Fig. 6 utilize the communication device 26 whenever necessary or expeditious.

Referring now to FIG. 7, a flow chart of the search process is shown. Steps in FIG. 7 repeated from FIG. 6 are given the same reference number as in FIG. 6 for convenience and ease of understanding. Fig. 7 commences with "SEARCH TRANSITS GATEWAY 32100", continuing from Fig. 6, "SYSTEM SEARCHES CONTAINERS 30400". The submitted search 32100 transits the gateway 200. The gateway 200 interacts with the container registers 32200. The gateways 200 store the information downloaded from the registers 32300, and the container registers are altered 32500. The container registers 120 then interact with the registers 120 of the encapsulated search, which registers, and the values set within, have been constructed and appended to the search through the search interface 32600. Values are exchanged and compared and operations performed under the rules governing both interacting containers 100, and the rules governing the search container 100 and any gateway 200. The search engine 320, operating under the principles and means of search engines presently existing as described elsewhere, then provides to the search interface 32600 a list of containers 100 meeting the requirements of the search and its appended registers, as well as additional search options 32900. The gateway 200 reports and makes available for collection to the analysis engine 400 the information obtained from the interaction 32400. On a periodic basis defined by the user or a rules-based system, the analysis engine 400 (Fig. 9) stores in databases 900, analyzes and instructs the execution engine 500, and the execution engine 500 executes changes in the system components as defined below. (Fig. 10). All operations on Fig. 7 utilize the communication device 26 whenever necessary or expeditious.

On the remaining figures, shapes referring to other figures, to operations external to the scope of the present figures, or to the subject of the present drawing, are indicated with dashed lines, and are shown only to place the described operations in the context of continuous and continual operations external to the drawing.

Referring now to FIG. 8, a flow chart of the preferred process for collecting and reporting information on containers is shown. The data reporting 600 and data collection 700 means utilizes subroutines within the analysis engines 400 and gateways 200 to submit and collect register information and sub level analysis to other analysis engines 400 or other gateways 200 of a higher (larger) logical set in a set pattern and frequency defined by the administrator.

Input is received from the user selecting "data reporting" 70100 from the "edit gateway" drop-down menu. Container levels are displayed 70200. Input is received from the user selecting container level 70300. A menu of all possible gateways 70320 and analysis engines 70330 residing on gateways on the above defined container class appears, depicted graphically as a tree of analysis engines and gateways at that container level. Input is received from the user selecting "source" from "source or destination." Input is received from the user 70400 selecting a container, containers, or class of container by clicking on the graphically depicted container(s) or container level on a display device. Input is received from the user 70410 selecting "destination" from "source or destination" Input is received from the user 70500 selecting an analysis engine, analysis engines, or class of analysis engine by clicking on the graphically depicted analysis engine(s) or analysis engine level on a display device. A time scheduler is displayed. Input is received from the user 70510 selecting the reporting frequency for the selected gateways to report data to the selected engines. The data from the gateways is thenceforth continuously moved or copied to the analysis engines by the system 10 utilizing the execution engine 500 according to the defined schedule, rules and pattern 70420, 70520.

Input is received from the user selecting "choose container level" 70300 from the gateway editor drop-down menu. A menu 70320 appears listing the classes of containers on the system within the defined subset and superset scheme of multiple hierarchically nested containers, i.e.; element, document, file, database, warehouse, domain, appears. Input is received from the user selecting the class of containers. A graphic representation of that container level throughout the system appears. Input 70300 is received from the user selecting individual containers or all the containers in that class.

From the gateway editor drop-down menu input 70100 is received from the user selecting "data collecting" A menu of all possible gateways and analysis engines residing on gateways on the above defined container class appears, depicted graphically as a tree of analysis engines, and gateways at that container level. Input 70510 is received from the user selecting "source" from "source or destination." Input is received from the user selecting a container, containers, or class of container by clicking on the graphically depicted container(s) or container level. Input 70510 is received from the user selecting "destination" from "source or destination." Input 70510 is received from the user selecting an analysis engine, analysis engines, or class of analysis engine by clicking on the graphically depicted analysis engine(s) or analysis engine

level. A time scheduler appears. Input **70510** is received from the user selecting the collecting frequency for the selected engines to collect data from the selected gateways. The data from the gateways is thenceforth continuously moved or copied to the analysis engines by the system **10** utilizing the execution engine **500** according to the defined schedule, rules and pattern.

5           The data collection **700** means, utilizing the communication device **26** and an execution engine **500**, comprises one or more subroutines or agents programmed to travel through the network collecting the accumulated data and analyses from selected analysis engines, gateways or selected subset level of analysis engines or gateways (as above) in a pattern and frequency defined by the gateway administrator at a given container level. Input **70510** is received from  
10 the user or administrator, defining the collection and reporting of data, thus controlling permission within his gateway, and being subject to permission levels defined by others beyond his gateway.

          Input is received from the user or gateway administrator selecting collection or reporting **70100** and the system shows the container levels available **70200**. Input is received from the  
15 user selecting a container level **70300**. Input is received from the user selecting "gateway" **70400** or "engine" **70500**. The system shows gateways **70320** or engines **70330** associated with that level. Input is received from the user editing the reporting parameters associated with a gateway or a class of gateways **70410** or an engine or class of engines **70510**. Input is received from the user selecting the collecting frequency for the chosen engines. When input is received  
20 from the user choosing to user save the definition, the screen returns to the main container screen, step **70100** to make another selection available. Input is received from the user choosing to repeat the cycle, choosing "destination" to describe the destination analysis engines and the data collecting frequency from those destination analysis engines. The data collection means **700** collects the accumulated gateway information in a pattern and frequency defined by the  
25 gateway administrator or user at a given container level.

          The system utilizing the execution engine (see Fig. **10**) distributes the new parameters to the gateways **70420** or engines **70520** by the communication device **26**. Using the new parameters the gateways report to the analysis engines **70430** after, in some cases, conducting sub-analysis **70440**, or using sub-analysis **70440** to submit directly to specified gateways under  
30 certain conditions and parameters, and the analysis engines collect from the gateways **70530**.

The analysis engine uploads, downloads and utilizes information to databases 900 to conduct its analysis.

The invention includes an analysis engine 400. Through the data reporting 600 means and data collection 700 the analysis engine 400 receives data and sub-analysis from the search interface and the gateways. Data includes, for each gateway 200, the frequency and grade of access, the description of the user accessing, the identity of the container 100 accessing, the register parameters, and the historically accumulated register data.

Referring now to FIG. 9, a flow chart of the operation of the analysis engine 400 is shown. Analysis engines 400 may reside at any gateway or anywhere in the system 10. The analysis engine 400, operating under its own programmed sequence, utilizing the communication device 26, works, by means of programmed rules of logical, mathematical, statistical or other analysis upon gateway and register information, in continuous interaction with the search process 410 and the data collection and reporting process 420 to analyze, determine and compile instructions 40100 on container construction 40110 to containerize in an automated process 40115, on container contents 40120 to move, copy or delete containers 40125, on storage schemes 40130 to move or copy containers to new storage 40135, on access routes 40140 to alter gateway pointers to sought information 40145, on search templates 40150 to add, delete or change search phrases and the referenced objects indicated by those search phrases 40155 and on gateway instructions 40160 to alter gateway registers and pointers 40165.

Thus, analyses might include, but are not limited to, the physical locus of the users accessing, the demographic classification of the users accessing, the access frequency for a given container, the range or curve of time relevance affecting a container, the range or region of space relevance affecting a container 100, the number or number of a specific type of container 100 transiting a gateway 200, the hierarchically graded usage of containers 100 or container contents 01 compared with the demographic of those users accessing the container, the hierarchically graded usage of containers 100 or container contents 01 compared with search phrases entered into the search interface 300, the hierarchically graded usage of containers 100 or container contents 01 compared with search phrases entered into the search interface 300 compared with the demographic of the users accessing, the number of pertinent containers nested within a given container 100. Once an analysis is accomplished, the result is compared to

pre-programmed rules triggering instruction sets (such as moving a container to nest within another container).

Instructions are then sent to the execution engine 40200, which utilizes the communication device 26 to execute the instructions derived from the analyses. These containerized instructions transit the gateways 40300 and are utilized in the gateway process (Fig. 12)

Referring now to FIG. 10, a flow chart of the operation of the execution engine is shown. The execution engine 400, operating under its own programmed sequence in response to the instructions from the analysis engine 50100, utilizing the communication device 26, works in continuous process as its containerized execution instructions transit the gateways 50200 to create containers 50210 in an automated containerization process 50215, alter container contents 50230 by moving or copying containers to new containers 50235, to alter storage 50240 by moving or copying containers to new storage 50245, to alter access routes 50250 by altering gateway pointers 50255, to alter search templates 50260 by adding, changing and deleting search phrases and the referenced objects indicated by those search phrases 50265, to alter gateway instructions 50270 by altering gateway registers and pointers 50275. The execution works in a continuous loop with the gateway process 50300, the data collection and reporting process 50400 and the analysis engine process 50300.

The invention includes gateways 200. Gateways may be placed and reside anywhere on the network where containers transit. Gateways also reside on any or all containers. The gateway reads and stores the chosen register information from transient containers entering or exiting its logical boundaries. The resident analysis search engine, if any, performs the specified level of analysis. Data and analysis is both held for the collection means according to the pattern and timing specified in the data reporting 600 editor and submitted according to the pattern and timing specified in the data collection means editor 700.

The gateways are network-wide, hierarchical, and nestable, and reside with a container encompassing any component, digital code, file, search string, set, database, network, event or process and maintaining a unique lifelong network wide identity and unique in all the universe historical identity, or may be strategically placed at such container transit points to gather and store register information attached to any such container, according to system-defined, system-generated, or user determined rules residing in its registers defining the behavior of those

containers and components as they exit and enter one another, or interact with one another or any system process or system component within the logical domain of that container, or after exiting and entering that container, or defining how they interact with that unique gateway.

Gateway's registers comprise both system-defined and user-defined registers, alterable by author, duration, location, network-wide history, individual container history and/or interaction with other containers, gateways, networks or media, and evolve according to that gateway's history on a computer network, or according to the network history of events and processes, or according to that information component's interaction with other information containers, components, system components, network events or processes.

Referring now to FIG. 11, a flow chart of the gateway editor is shown. From the main title bar input is received from the user selecting "containerize" or "gateway level" 20100. When input is received from the user selecting "containerize" the system enters the container editor process 110. When input is received from the user selecting "gateway," the system shows the gateway levels available 20200. A menu of all possible gateways within the subset and superset scheme of defined multiple hierarchically nested gateways appears. Input is received from the user selecting the gateway level 20300. The system searches the gateways 20500 to locate the available gateway templates 20700 and the available gateways 20600. Input is received from the user selecting the gateway 20610 or gateway level template 20720. The system goes to the gateway 20620 or to the template 20720. A graphic representation of the chosen gateway 20630 or template 20730 appears. Input is received from the user to edit 20640 or create a gateway 20740. Once completed, input may be received from the user selecting "analysis level" from the gateway 200 drop-down menu, to select the level of analysis in a multi-level analysis sequence to be accomplished at the local level by a gateway-resident analysis engine. The user accesses the container editor to containerize (Fig. 5). Input is received from the user selecting the registers by clicking on the graphically depicted register, or from a drop down menu. ). Input is received from the user setting the registers as described elsewhere in ("container registers"). Input is received from the user selecting or defining the rules governing the interaction of that gateway with transient containers. Input is received from the user selecting or defining the rules governing the interaction of containers existing within the logical domain of the container 100 to which that gateway is attached. The user publishes the gateway (Fig. 5). Input is received from the user selecting "residence" from the main menu bar.



). Input is received from the user choosing to leave the gateway where it was created, move it to container on another drive, directory, computer, or network. If the user chooses "move," a browse function allows the user to name the new location or browse a list of possible locations. Once input is received from the user choosing the residence of the gateway, the program  
5 restores the search interface screen.

The invention includes a data reporting means editor 610, and a data collection means editor 710, Fig. 2 A, as a menu option under the gateway editor 210.

The present invention also includes a gateway process.

Referring now to FIG. 12, a flow chart of the gateway process is shown. A system  
10 operation, search process or element container or process container is shown in transit 21100 passing through a gateway 21200. The container, operation or process interacts with the gateway 21300, uploading, downloading and exchanging information with the container, operation or process. The gateway stores container information 21400 and the container registers are altered 21500. The container registers also interact with the search interface 21600.  
15 The gateways report the register information or make it available for collection by the data reporting and collection means (Fig. 8) operating on the communication device 26 to provide the information to the analysis engine 21800, which stores 90100, analyzes and instructs the execution engine 21900, which processes and instructions are also stored 90100 by the execution engine upon receipt.

20 All operations in Fig. 12 utilize the communication device 26 whenever necessary or expeditious.

Referring now to FIG. 13 A, a drawing of nested containers 100 prior to the container modification process on a network 201 is shown. (Note: The same container numbering scheme is used in Fig. 13 A, 13 B, 13 C, 13 D and in 2 B.) Information containers  
25 505 and 909, residing within container 908, operating under the rules governing container interaction within that container 908 downloaded to container 505 and 909 from gateway 9081 upon their entrance to container 908, which rules had been downloaded from execution engine 500 acting under the direction of analysis engine 400, and under the rules programmed into their own registers 404120, 909120, compare the specified (by those rules) set of registers 404120,  
30 909120, i.e., time and space, and determine a container 404 encapsulated within 505 would be more appropriately encapsulated within container 909.

Referring now to FIG. 13 B a drawing of nested containers during a container modification process on a network 201 is shown. Container 404 is moved to reside with container 909. As the container 404 exits container 505, the gateway of container 505, being gateway 5051, operating under the rules governing container interaction with a gateway 5051 upon egress or egress as programmed in the gateway editor 210 and modified by the execution engine 500 executing the instructions of the analysis engine 400, or any greater logical analysis engine 408 providing execution instructions to an execution engine 508 operating in a larger encompassing container 108 entering through that container's gateway 208 or an independent gateway 707, or sub-analysis engine operating at any gateway level, records the register information of container 404. The gateway 5051 reports the transaction to the gateway 9081 of container 908, being the next higher logical container. Gateway 9081 holds in gateway storage 205 the information until collected by one or more data collection processes 700, or reported to one or more data reporting processes 600, serving one or more analysis engines 400 residing independently on the system 10 or an analysis engine at higher logical container 303. The analysis engine 400, comparing reports of user hierarchically graded usage under the operations of the search engine 320 and the search interface 300, on information container 808 after receiving reports from the data reporting means of container 404 being moved to container 909 determines, i.e., that the number of time and space relevant containers residing within container 909 is sufficient to warrant an action, and directs the execution engine 500 to copy container 909, nested within container 908, to a third information container 808. As the copy instruction from execution engine 500 transits the gateway of container 908, the gateway 9081 records the instruction. The copy instruction interacts with the registers 909120 of container 909 regarding the rules governing its copying to another location. Once approved by the governing rules of registers 909120 appended to container 909, container 909 is duplicated. As the duplicate container 909 exits the container 908, the gateway records the register information 909120 of container 909, and the registers 909120 of container 909 are altered by special instructions from gateway 9081 under the rules residing in gateway 9081 regarding ingress and egress and the rules residing in the registers 909120 of container 909 regarding alteration by gateways upon ingress and egress. Passing through independent gateway 707, the register information 909120 is recorded, and awaits data collection or reporting 700, 600. As container 909 enters container 808, the gateway records the register information 909120 of container 909, the registers 909120

of 909 are altered by special instructions from gateway 8081, operating under the rules as described in the paragraph above, and container 909 takes up residence within container 808.

Referring now to FIG. 13 C, a drawing of nested containers after the container modification process on a network 201 process is shown. Container 909, now also logically residing within container 808, commences to interact with other containers 606 in 808 under the rules governing container interaction within container 808 as received from gateway 8081 upon transiting that gateway, and under the rules of registers 606120, 909120 of the interacting containers 606, 909, operating under the rules as described in the paragraph above. Through data collection and reporting 700, 600, analysis engine is appraised of container's 909 new duplicate residence. I.e., operating under the registers of space relevance, a body of law pertaining to Boston Municipal tax law may be housed in a container holding Massachusetts tax law, but it would be more appropriately located in a container holding Boston tax law, with only a pointer to that location residing in the Massachusetts tax law container. In this example, such an analysis could be accomplished by comparison of zip code information in the space registers, or logical rules-based analysis, with "state" being a larger set than "city". Or, i.e., operating under the registers of time relevance, the curve of time relevance for a concert might follow an ascending curve for the months prior, hit a brief plateau, and then reach a precipitous decline, at which time certain pertinent information only might be moved to an archival container of city events or rock concerts of that year. In this example, once the curve is mapped into a register, that map would cause an increasing frequency of pointers to that container in other containers or gateways, or inclusion of that container in other containers, as the analysis engine compares that curve with increasing user inquiry.

Referring now to Fig. 13 D, a flowchart of the reconstruction process is shown.

Information containers 505 and 909, residing within container 908, operating under the rules governing container interaction within that container 908 downloaded 888103 to container 505 and 909 from gateway 9081 upon their entrance to container 908, which rules had been downloaded 888102 from execution engine 500 acting under the direction 888101 of analysis engine 400, and under the rules programmed into their own registers 404120, 909120, compare 888104 the specified (by those rules) set of registers 404120, 909120, i.e., time and space, and determine 888105 a container 404 encapsulated within 505 would be more appropriately encapsulated within container 909.

Container 404 is moved 888106 to reside with container 909. As the container 404 exits container 505, the gateway of container 505, being gateway 5051, operating under the rules governing container interaction with a gateway 5051 upon egress or egress as programmed in the gateway editor 210 and modified 888108 by the execution engine 500 executing the instructions of the analysis engine 400, or any greater logical analysis engine 408 providing execution instructions 888107 to an execution engine 508 operating in a larger encompassing container 108 entering through that container's gateway 208 or an independent gateway 707, or sub-analysis engine operating at any gateway level, records 888109 the register information of container 404, and alters the register information of container 404. The gateway 5051 reports 888110 the transaction to the gateway 9081 of container 908, being the next higher logical container. Gateway 9081 holds 888111 in gateway storage 205 the information until collected by one or more data collection processes 700, or reported to one or more data reporting processes 600, serving 888112 one or more analysis engines 400 residing independently on the system 10 or an analysis engine at higher logical container 303. The analysis engine 400, comparing 888114 reports of user hierarchically graded usage on information container 808 under the operations of the search engine 320 and the search interface 300, after receiving 888113 reports from the data reporting means of container 404 being moved to container 909, determines 888115, i.e., that the number of time and space relevant containers residing within container 909 is sufficient to warrant an action, and directs 888115 the execution engine 500 to copy container 909, nested within container 908, to a third information container 808. As the copy instruction from execution engine 500 transits the gateway of container 908, the gateway 9081 records 888116 the instruction. The copy instruction interacts 888117 with the registers 909120 of container 909 regarding the rules governing its copying to another location. Once approved 888118 by the governing rules of registers 909120 appended to container 909, container 909 is duplicated 888118. As the duplicate container 909 exits the container 908, the gateway records 888119 the register information 909120 of container 909, and the registers 909120 of container 909 are altered 888120 by special instructions from gateway 9081 under the rules residing in gateway 9081 regarding ingress and egress and the rules residing in the registers 909120 of container 909 regarding alteration by gateways upon ingress and egress. Passing through independent gateway 707, the register information 909120 is recorded 888121, and awaits 888122 data collection or reporting 700, 600. As container 909 enters container 808,

the gateway records 888123 the register information 909120 of container 909, the registers 909120 of 909 are altered 888124 by special instructions from gateway 8081, operating under the rules as described in the paragraph above, and container 909 takes up residence 888125 within container 808.

5 Container 909, now also logically residing (in addition to its original container residence) within container 808, commences to interact 888126 with other containers 606 in 808 under the rules governing container interaction within container 808 as received from gateway 8081 upon transiting that gateway, and under the rules of registers 606120, 909120 of the interacting containers 606, 909, operating under the rules as described in the paragraph above.  
10 Through data collection and reporting 700, 600, analysis engine is appraised 888127 of container's 909 new duplicate residence.

Referring now to Fig. 14, the screen interface of the container editor is shown. This interface is a process wherein input is received by the user using the main menu 78 or drop down menu 1419, or using an input device to "drag and drop" or click, causing the system 10 to  
15 acquire 1409, edit 1410 or create 1411 a file 1407, container 1408 or digital content 01, to search for 1412, acquire 1413, edit 1414 or create 1415, print 1416, or containerize 1417 a container 100, to select 1402, (or by clicking on register), search 1403, acquire 1404, edit 1405, or create a register 1406 to append or detach registers 120 to those containers, to set register values in those registers 120, to utilize the register editor 125 through 1405 to create new  
20 registers, or to 1418 add, detach, acquire a gateway 200 to append or detach to those containers, and utilize the gateway editor 210 through 1418. (See detailed description referring to Fig. 5)

Referring now to Fig. 15, the screen interface of the gateway editor is shown. This interface is a process wherein input is received by the user using the main menu 1501 or drop down menu 1513, or using an input device to "drag and drop" or click, causing the system 10 to  
25 search for 1507, acquire 1508, edit 1509 create 1510, print 1511 or containerize 1512 gateways, and causing the system 10 to establish rules by which an individual gateway governs the transiting 1502, entering 1503, exiting 1504 of containers and the interaction of containers within its domain 1505, and external of its domain.1506. (See detailed description referring to Fig. 11).

30 Referring now to Fig. 16, the screen interface of the search interface. This interface is a process wherein input is received by the user using the main menu 1625 or drop down menu

1624, or using an input device to “drag and drop” or click, or by entering text, causing the system 10 to select 1615, search for 1616, acquire 1617, edit 1618 create 1619, print 1620, containerize 1621 (by accessing the container editor 110) or insert 1622 digital search strings into the search box 1623 in order to submit that string to the search engine 320, or causing the system 10 to select 1602, search for 1603, acquire 1604, edit 1605, create 1612, containerize 1613 (by accessing the container editor 110), or insert 1614 search keys (templates that comprise search scope in geographic range, container level, and specific key words or digital strings), or containerized searches (containers 110), into the search box 1623 in order to submit that string to the search engine 320 , or causing the system 10 to set a search range by geographic range 1607, container level 1608, or acquire 1609, edit 1610 or create 1611 a scope template. (templates that comprise search scope in geographic range and, container level.) (See detailed description referring to Fig. 6).

Referring now to Fig. 17, a drawing showing, on an input device or computer screen 24, in any generic (dashed lines) software application program, a drop-down menu link 1403 on a drop down menu 1402 dropping down from a main menu 1401, and a free-floating button link 1404, is shown. When input is received at 1402 or 1403, the system 10 makes available to the user the containerization process or container editor 110. When input is received at drop-down menu link 1405 or a button link 1406, the system 10 makes available to the user the means to enter and interact with this system 10 or this network 201 in any of their aspects. The interfaces 1403, 1404 show a process wherein input is received causing the system 10 to encapsulate content or access the container editor 110. The link also allows the user to encapsulate the page or file on which he is currently working, without selecting content, and if so desired, without accessing the container editor. The interfaces 1405, 1406 show a process wherein input is received causing the system 10 to access or interact with the system 10 or the network 201.

The present invention also includes a search engine 320. Once the key word(s), phrase or digital segment is entered into the search interface 300, or an offered selection chosen on the menu, it is utilized by the search engine 320 to locate the desired site or data.

The search engine employed may be any industry standard search engine such as Verity “Topic”, or Personal Library Software, as used in Dow Jones News Retrieval, or Internet search engines such as Webcrawler, Yahoo, Excite, Infoseek, Alexa or any Internet search engine, or

any new engines to be developed capable of searching for and locating digital segments, whether text, audio, video or graphic.

The present invention also includes an analysis engine 400. Utilizing rules-based analysis, the analysis engine determines the class of storage medium upon which containers reside, the subsets and supersets by which and in which containers encompass and reside within one another, the routes of access to those containers, the historically successful search parameters by which those containers are accessed based upon the identity of the user accessing the containers, and the grade of access chosen by the user in accessing that container 100.

Utilizing a pre-programmed sequence of compilation, and inductive, deductive and derivative analysis, the analysis engine manufactures instructions based upon the analysis of the information submitted by the gateways and the search interface, and submits those instructions to the appropriate execution engine 500 in order to create new information containers, content assemblages, storage schemes, access routes, search templates, and gateway instructions, and others, and to provide informed search options through the search interface to the inquiring user.

The present invention also includes an engine editor 510, that provides a system administrator with a means of editing the operating principles of that search engine, and search template loading in the search interface 300, a reporting and collection means editor 610, 710, governing data reporting 600 and data collection 700 at the gateways 200 as defined by the gateway editor 210 and the register editor 125, a container editor 110 for creating and modifying containers and appending registers to containers, a register editor 125 for creating and modifying container registers and establishing and adjusting the values therein, container gateways 200 with their own storage 205, information containers 100 for holding information and container registers for holding information about specific containers and their history on the network.

The present invention also includes an execution engine 300. Based upon instructions received from the analysis engine 400 utilizing the communication device 26, the execution engine 500 provides search phrases to the search interface 300 based upon initially received inquiries, relocates containers including their programs, data and registers to other directories, drives, computers, networks on other classes of storage mediums, i.e., tape drive, optical drive, CD-ROM, deletes, copies, moves containers to nest within or encompass other containers on other directories, drives, computers, networks to nest within other containers, alters the class of

storage medium upon which containers reside, the subsets and supersets by which and in which containers encompass and reside within one another, the routes of access to those containers, and the historically successful search parameters by which those containers are accessed based upon the identity of the user accessing the container and the grade of access chosen by the user  
5 in accessing that container.

The execution engine 400 fulfills the instructions of the analysis search engine 500, to create new information containers, content sub and superset assemblages, storage schemes, access routes, search templates, gateway 200 instructions and other system functions. The execution engine includes an editor 510 that provides a system manager with a means of editing  
10 the operating principles of that search engine, governing data reporting, data collection 700, search template loading, gateway instructions, and other functions.

The present invention also includes flat or relational databases 900, used where, and as required.

The present invention also includes a communication device 26 supporting all operations  
15 on a network wide basis.

The present invention also includes a search engine 300 to locate the desired site or data. The present invention also includes databases 900, flat or relational, to serve the other components of the system as needed and where needed.

The present invention also includes editors, by which the user may alter the governing  
20 aspects of the system. Editors include, but are not limited to, a container editor 110, a register editor 125, a gateway editor 210, an engine editor 510, a reporting means editor 610, a search interface 300, and a collection means editor 710.

The present invention also includes specific screen interfaces for the editors, as described in Fig. 14, Fig. 15. and Fig. 16.

25 The present invention also includes a means for this system 10 and network 201 or container editor 110 to be accessed from a menu or button selection within any program, as described in Fig. 17.

While the present invention has been described with reference to certain preferred embodiments, those skilled in the art will recognize that various modifications may be provided.  
30 For example, both analysis engine and execution engine may be duplicated or modified for distribution at various locations and hierarchical positions in the gateway and container system



throughout the network and designed to work in concert. Also, the physical computing infrastructure may be mainframe, mini, client server or other with various network and distributed computing designs, including digitally supported or based physical or public media, and the components of the system 10, as described in Fig. 1 may be physically distributed  
5 through space. Even the contents of a single container may be logically referenced but be physically distributed through the network and reside at multiple storage locations. The whole system may be hierarchically nested within other systems to the nth degree. Whole systems may also be encapsulated within containers. A single container may also encompass a single physical media, such as a CD-ROM disk, programmed with the container, gateway and register design.

10 Gateways may be strategically placed on containers at ingress and/or egress points or may be placed strategically throughout the system for optimal collection and reporting output and gateway system control. Also, the loop of gateway data collection and reporting, analysis engine analysis, instruction, and gateway modification, and execution engine operations may be infinitely nested, from the smallest container of two sub-containers to whole networks holding  
15 millions of containers and thousands of levels, with analysis itself nested within the multiple levels. Gateways may be established at both logical and physical junctures such as a satellite uplink point. Also, the provision to establish a unique network identity might be designed to include as of yet unknown computer networks as they arise. The analysis and execution engines may operate on a rules-based, fuzzy logic, artificial intelligence, neural net, or other system not  
20 yet devised. Other variations upon and modifications to the preferred embodiments are provided for by the present invention, which is limited only by the following claims. Also, the classification scheme of nested containers, while designated by the container creators, may transform, be utilized otherwise, or be wholly discarded according to usage. Also, hardware configurations, such as the use of RAM or hard drives for storage or lasers for communication  
25 may assume myriad forms without altering the essential operation of this invention.

**WHAT IS CLAIMED IS:**

1. An apparatus for transmitting, receiving and manipulating information on a computer system, the apparatus including a plurality of containers, each container being a logically defined data enclosure and comprising:

5 an information element;

a plurality of registers, the plurality of register forming part of the container, a first register of the plurality of registers for storing a unique container identification value, a second register of the plurality of registers that stores information and evolves according to the relationship, use and interaction of the container with other containers, processes and systems;

10 and

a gateway attached to and forming part of the container, the gateway controlling the interaction of the container with other containers, systems and process.

2. The apparatus of claim 1, wherein the information element is one from the group of text, graphic images, video, audio, a digital pattern, a process, a nested container, bit, natural  
15 number and a system.

3. The system of claim 1, wherein the plurality of registers include at least one container history register for storing information regarding past interaction of the container with other container, system or processes, the container history register being modified.

4. The system of claim 1, wherein the plurality of registers include at least one  
20 system history register for storing information regarding past interaction of the container with different operating system and network processes.

5. The system of claim 1, wherein the plurality of registers include at least one predefined register, the predefined register being a register associated with an editor for user selection, the predefined register appendable to any container.

6. The system of claim 1, wherein the plurality of registers include a user-created register, the user-created register generated by the user, one or more of which is appendable to any container.

7. The system of claim 1, wherein the plurality of registers include a system-  
5 defined register, the system-defined register set, controlled and used by the system, one or more of which is appendable to any container.

8. The system of claim 1, wherein the plurality of registers include at least one register for controlling the relationship of the container with other containers, systems and processes using time as a parameter.

9. The system of claim 8, wherein the plurality of registers include:  
10 an active time register for identifying times at which the container will act upon other containers, processes, systems or gateways;  
an passive time register for identifying times at which the container can be acted upon other containers, processes systems, or gateways; and  
15 a neutral time register for identifying times at which the container may interact with other containers.

10. The system of claim 1, wherein the plurality of registers include at least one acquire register for controlling whether the container adds a register or a container from other containers when interacting with them.

11. The system of claim 1, wherein the plurality of registers include at least one  
20 register for controlling the relationship of the container with other containers using space as a parameter.

12. The system of claim 11, wherein space refers to the geographic location of a the container.

13. The system of claim 11, wherein space refers to the logical address space of a network in which a container resides.

14. The system of claim 11, wherein the plurality of registers include:

5 an active space register for identifying space in which the container will act upon other containers, processes, systems or gateways;

an passive space register for identifying from which the container can be acted upon other containers, processes systems, or gateways; and

a neutral time register for identifying space in which the container may interact with other containers.

10 15. The system of claim 1, wherein the gateway includes means for acting upon another container, the means for acting upon another container using the plurality of register to determine whether and how the container acts upon other containers.

15 16. The system of claim 1, wherein the gateway includes means for allowing interaction, the means for allowing interaction using the plurality of registers to determine whether and how another container can act upon the container.

17. The system of claim 1, wherein the gateway includes means for gathering information, the means for gathering information recording register information from other containers, systems and processes that interact with the container.

20 18. The system of claim 1, wherein the gateway includes means for reporting information, the means for reporting information providing register information to other containers, systems and processes that interact with the container.

19. The system of claim 1, wherein the gateway includes an expert system including rules defining the interaction of the container with other containers, systems and processes.

20. A method for creating an interactive information container, the method including the steps of:

forming a container;

selecting an interactive register for the container;

5 identifying an item for inclusion in a container; and

creating a container element that includes the identified item.

21. The method of claim 20, wherein the step of forming a container further comprising the steps of:

displaying a plurality of container levels;

10 receiving input from a user selecting one of the displayed container level; and

displaying a container template corresponding to the container level input.

22. The method of claim 20, wherein the step of selecting an interactive register further comprising the steps of:

displaying a list of available registers;

15 receiving input selecting an available register from the list of available registers;

receiving input values for the selected available register; and

appending the register to the container.

23. The method of claim 20, wherein the step of creating a container element that includes the identified item further comprising the steps of:

20 providing a data structure as part of the container element;

storing the identified element in the data structure; and

associating the container element with the container.

24. The method of claim 20, wherein the step of forming a container includes the step of providing for the container a gateway that uses the interactive register to control the  
25 interaction of the container with other containers, processes, and systems.

25. The method of claim 24, wherein the step of providing a gateway further comprising the steps of:

determining a current gateway for a system upon which the container is being created;  
replicating the current gateway to create a new gateway; and  
5 associating the new gateway with the container.

26. The method of claim 24, wherein the step of providing a gateway further comprising the steps of:

determining a register for a system upon which the container is being created;  
replicating the determined register to create a new register; and  
10 associating the new register with the container.

27. The method of claim 20, wherein the step of selecting an interactive register further comprising the steps of:

retrieving a list of available registers;  
selecting an available register from the list of available registers by the system;  
15 receiving input values for the selected available register from the system; and  
appending the register to the container.

28. The method of claim 20, wherein the step of creating a container element that includes the identified item is performed by a system interacting with the container, and further comprising the steps of:

20 providing a data structure as part of the container element;  
storing the identified element in the data structure; and  
associating the container element with the container.

29. A method for interacting between a first interactive information container and a second interactive information container, the method including the steps of:

25 determining identification information for the first container using a first gateway;  
determining identification information for the second container using a second gateway;

determining whether the first container can act upon the second container using the first gateway and a register of the first container;

determining whether the second container can be acted upon by the first container using the second gateway and a register of the second container; and

5 performing the interaction between the first and second containers prescribed by the first gateway and the register of the first container if both the first container can act upon the second container and the second container can be acted upon by the first container.

30. The method for interacting of claim 29, wherein the steps of determining identification information are performed by reading respective identification registers of the first  
10 and second containers.

31. The method for interacting of claim 29, further comprising the step of altering a register of the first container and a register of the second container to reflect the interaction between the first container and the second container.

32. The method for interacting of claim 29, further comprising the step of adding  
15 registers to the first container based on the registers in the second container and the second gateway.

33. The method for interacting of claim 29, wherein the step of performing also uses the second gateway and the register of the second container to determine the prescribe action to be taken.

20 34. The method for interacting of claim 29, further comprising the steps of:  
determining whether the first container should add an identified register of the second container as a new register of the first container using an acquire register and the first gateway of the first container; and

25 adding the new register to the first container if it is determined that the new register should be added to the first container.

35. The method for interacting of claim 29, further comprising the step of modifying the first gateway of the first container based on the interaction between the first container and the second container.

36. The method for interacting of claim 35, wherein the step of modifying includes  
5 modifying rules of an expert system that forms the first gateway of the first container.



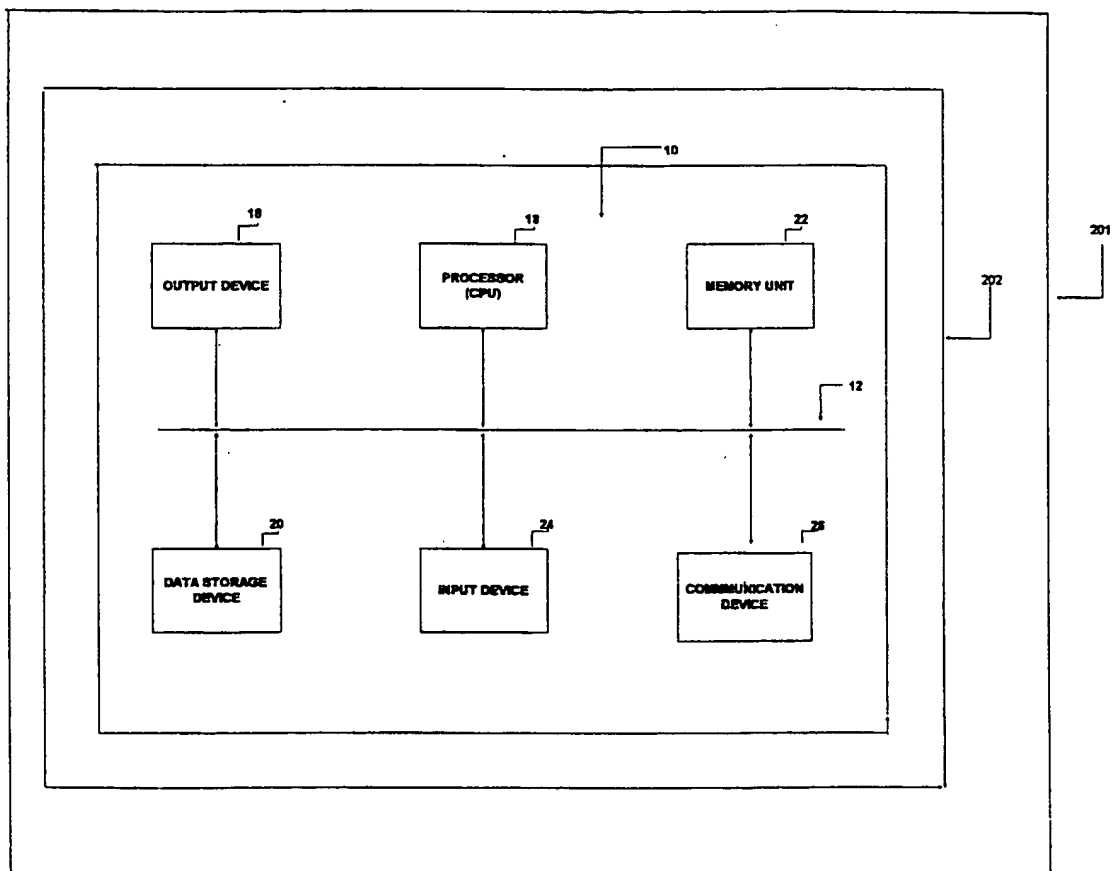


FIG. 1

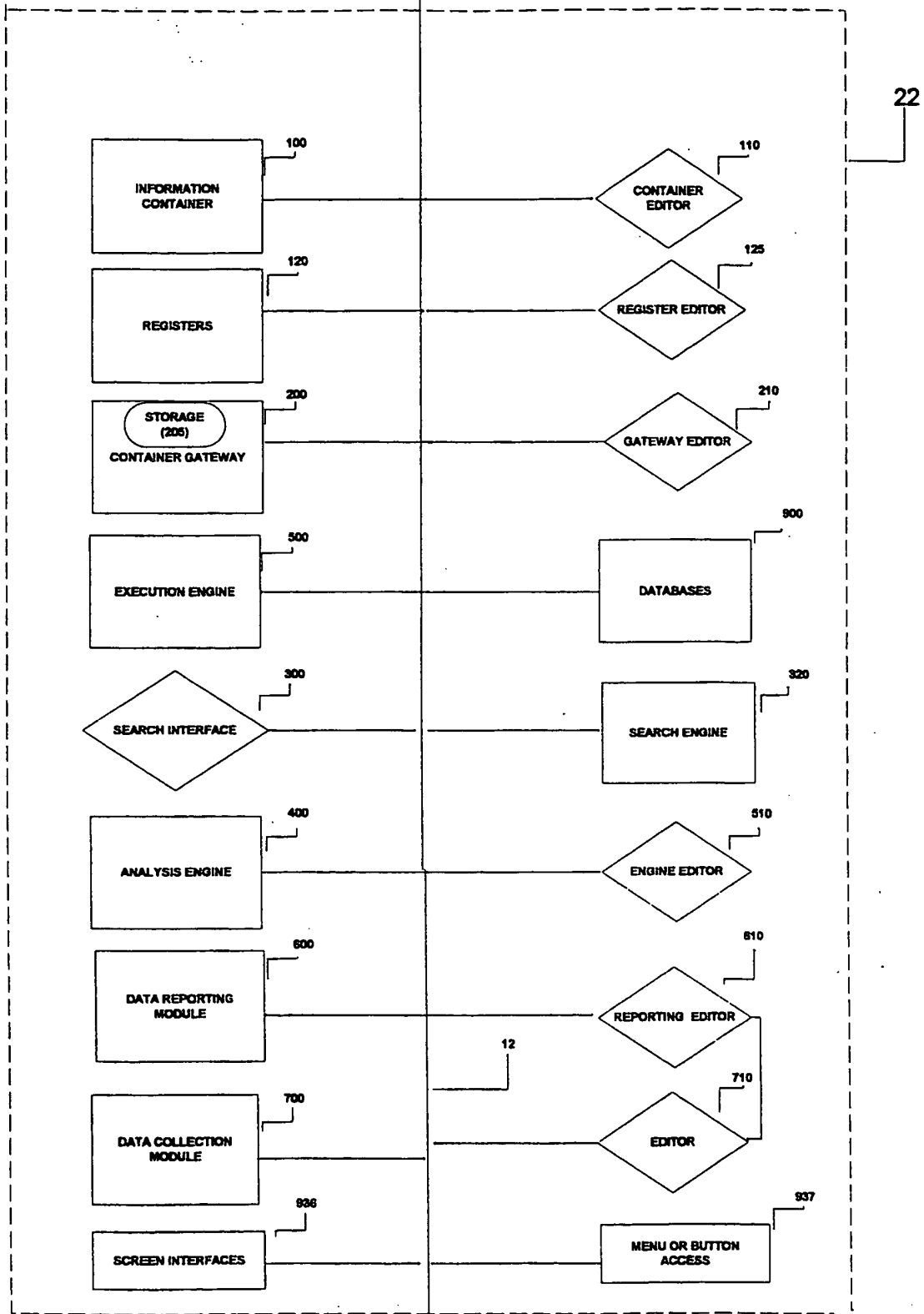


FIG. 2 A

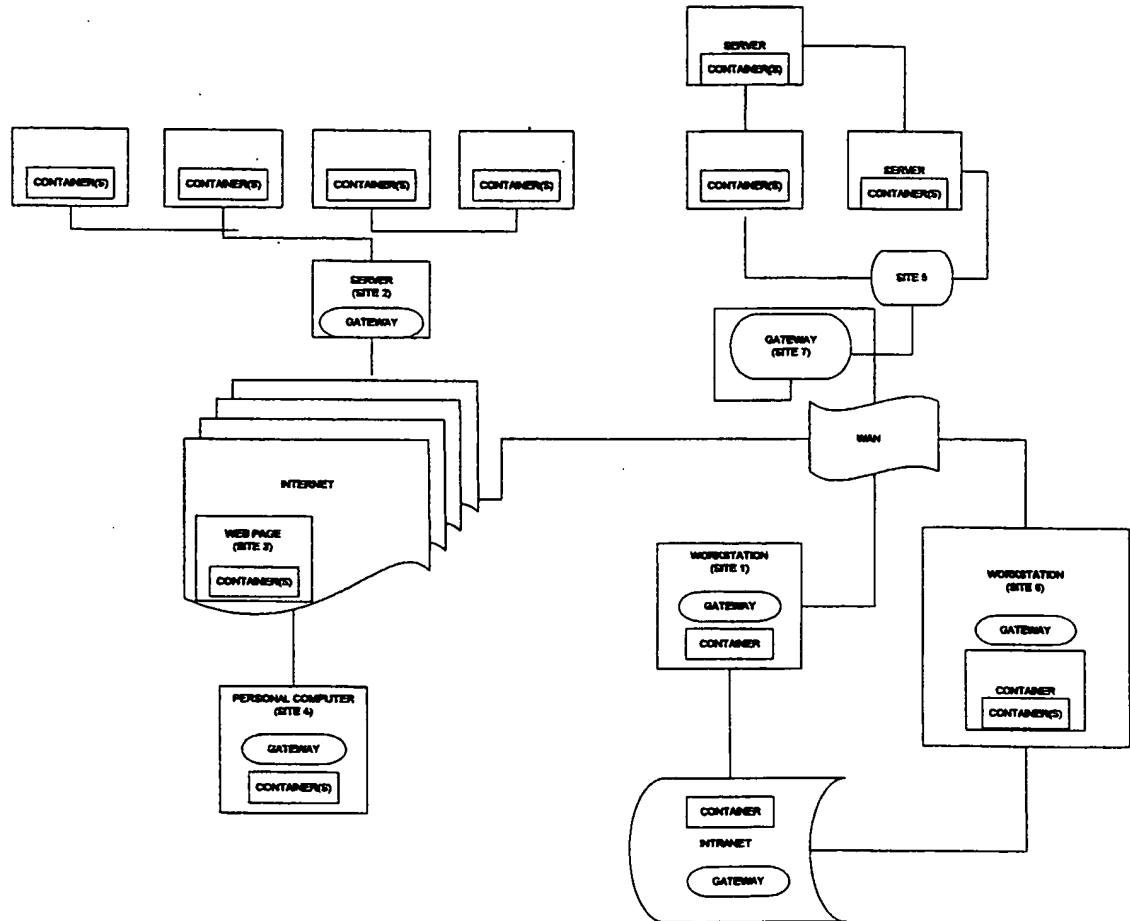


FIG. 2 B

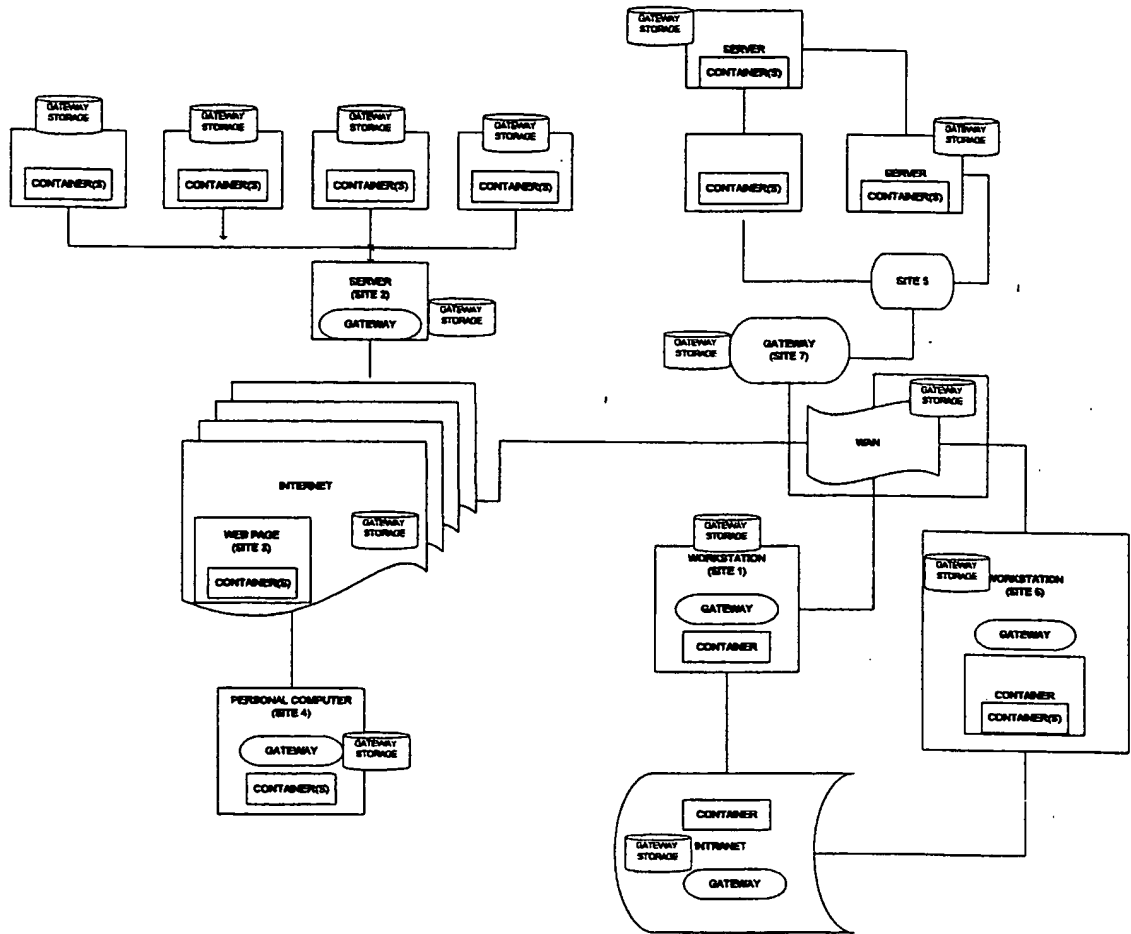


FIG. 2 B 1

5/30

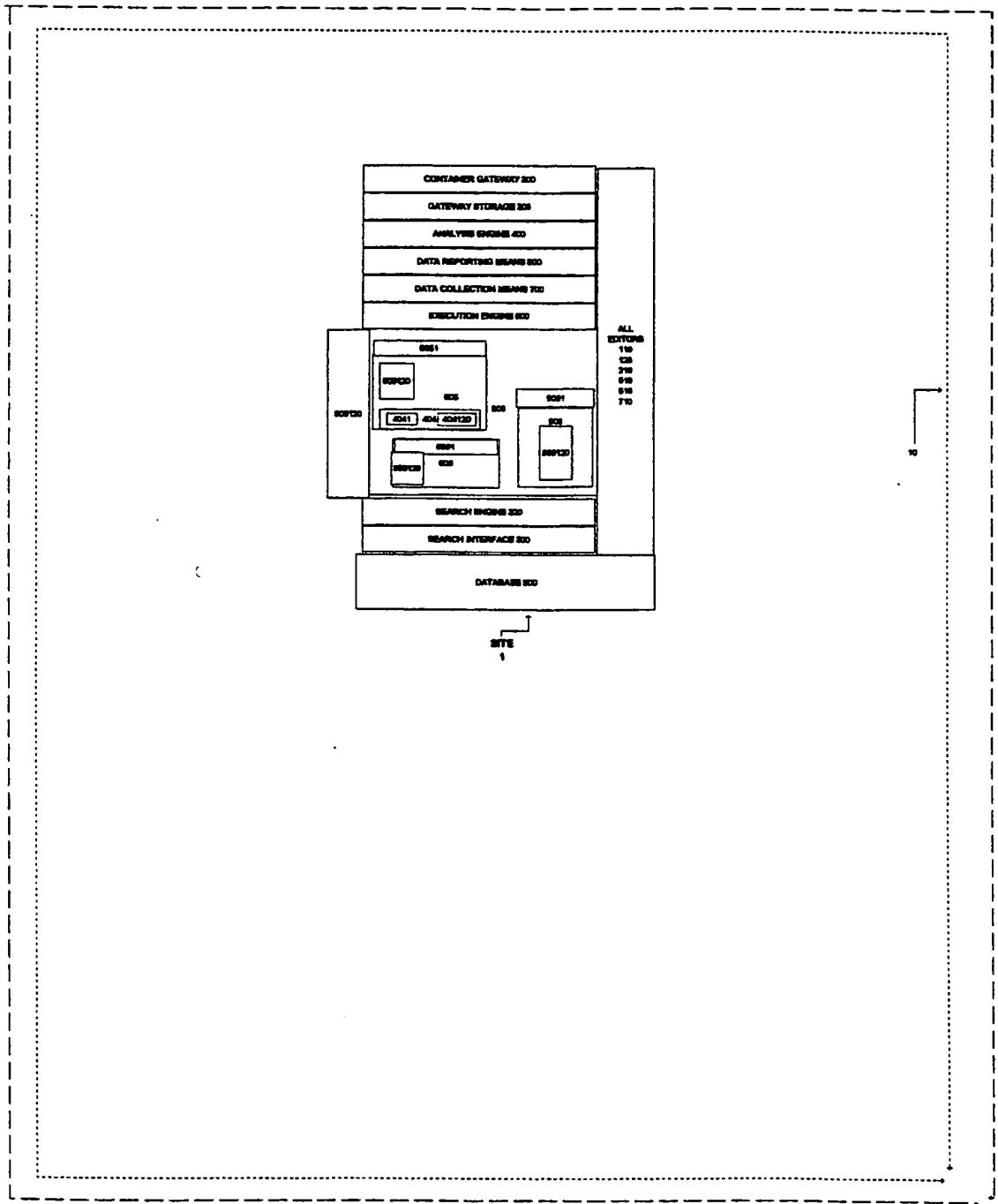


FIG 2 C

6/30

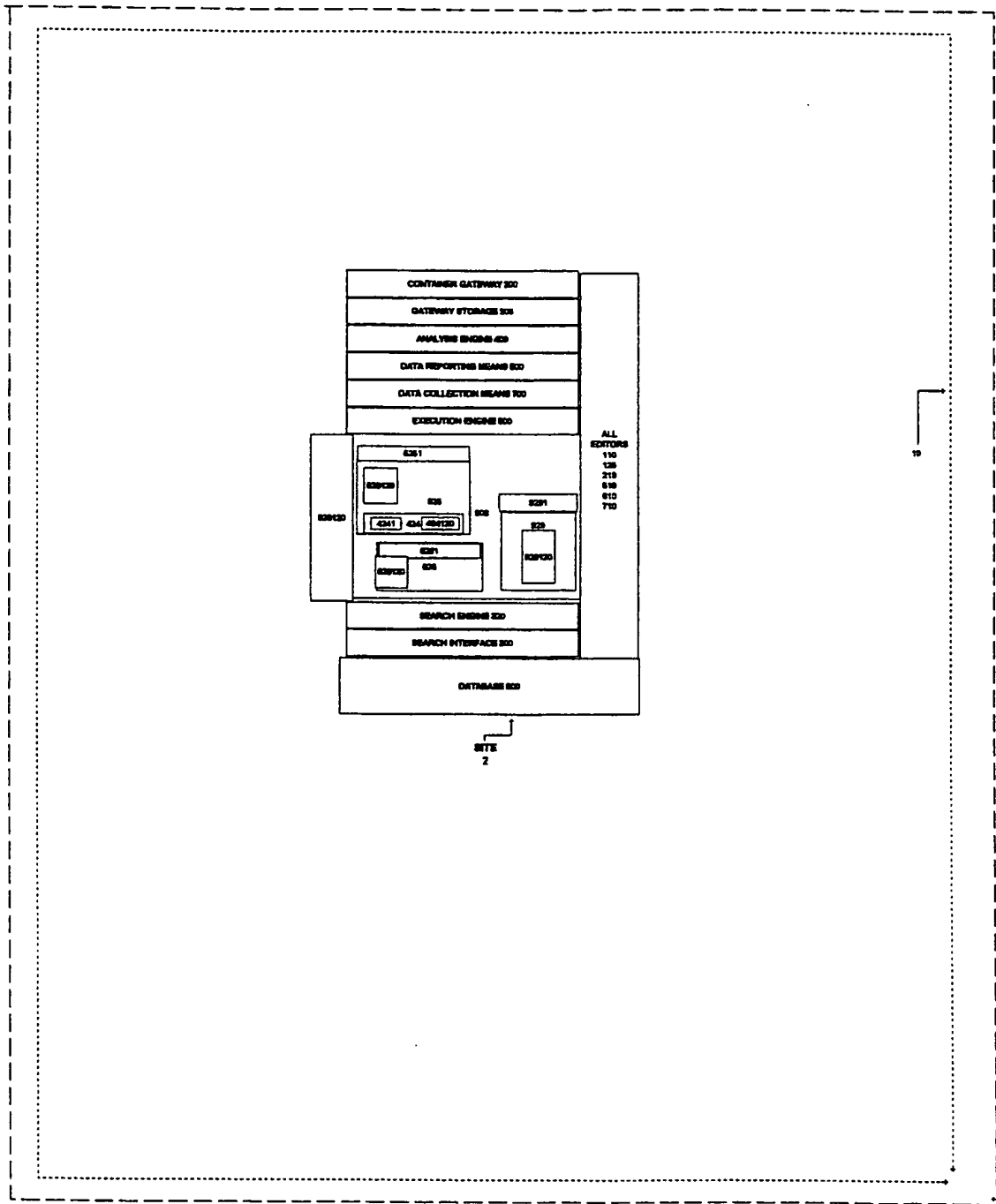
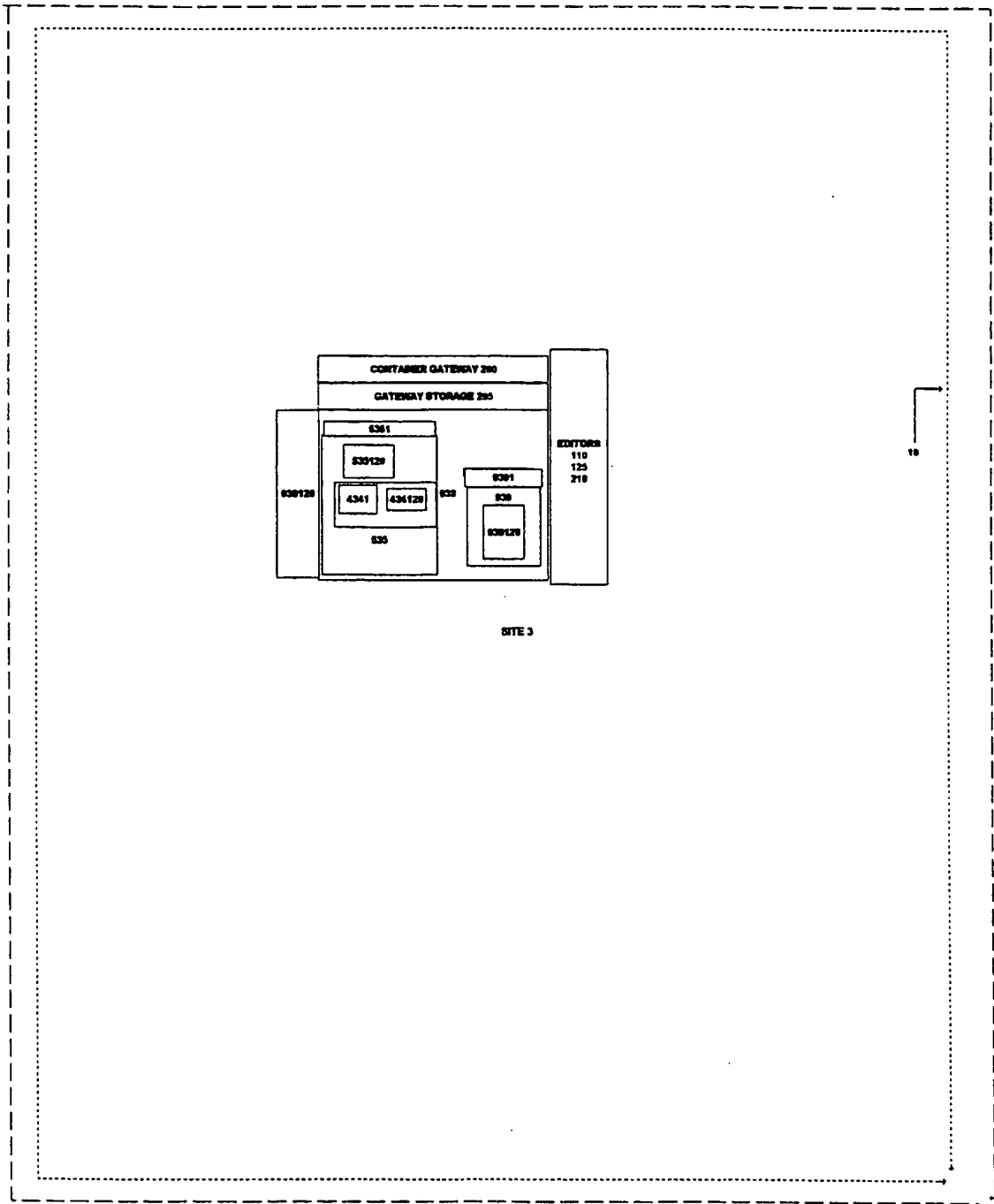


FIG. 2 D

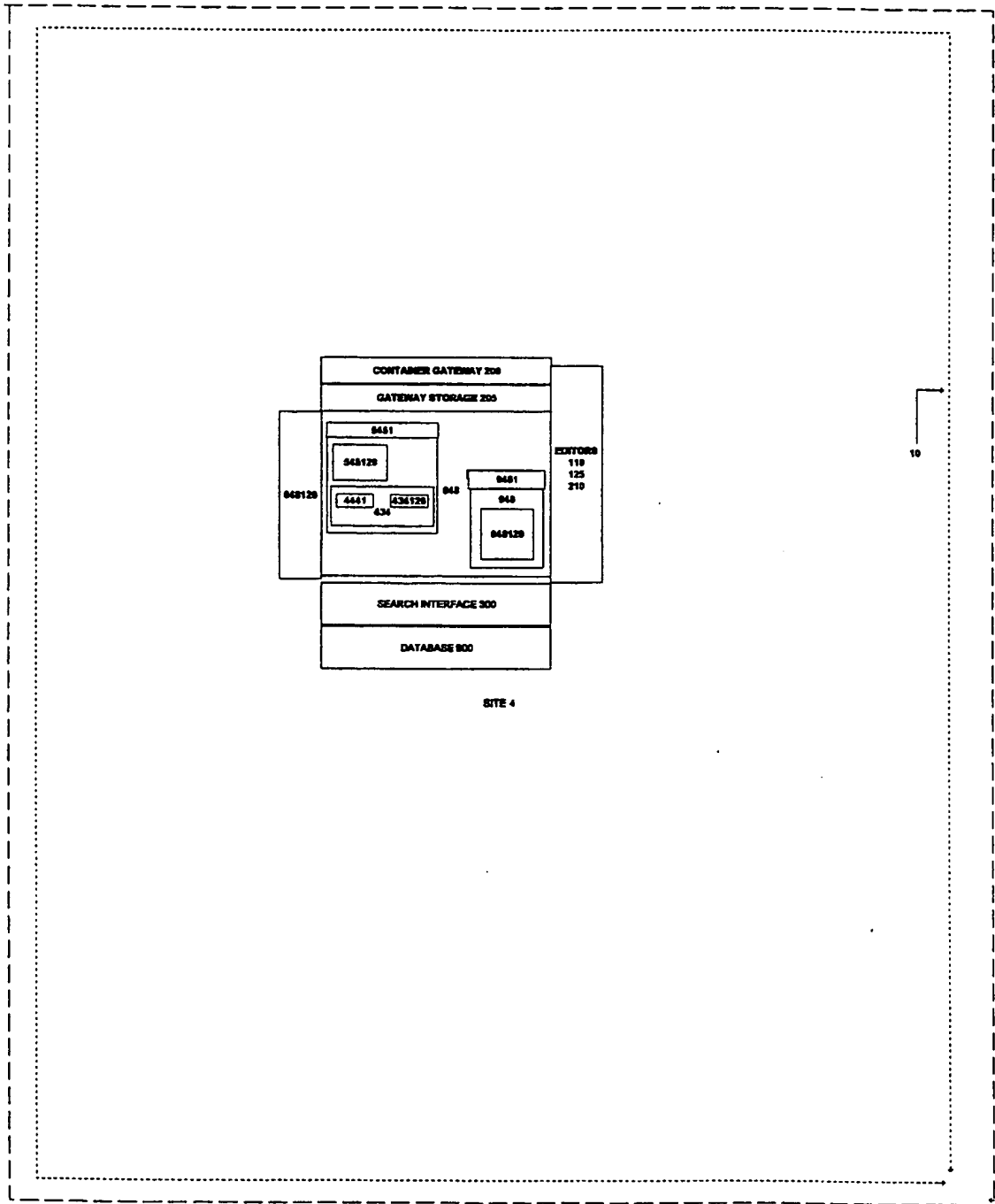
7/30



SITE 3

FIG. 2 E

8/30

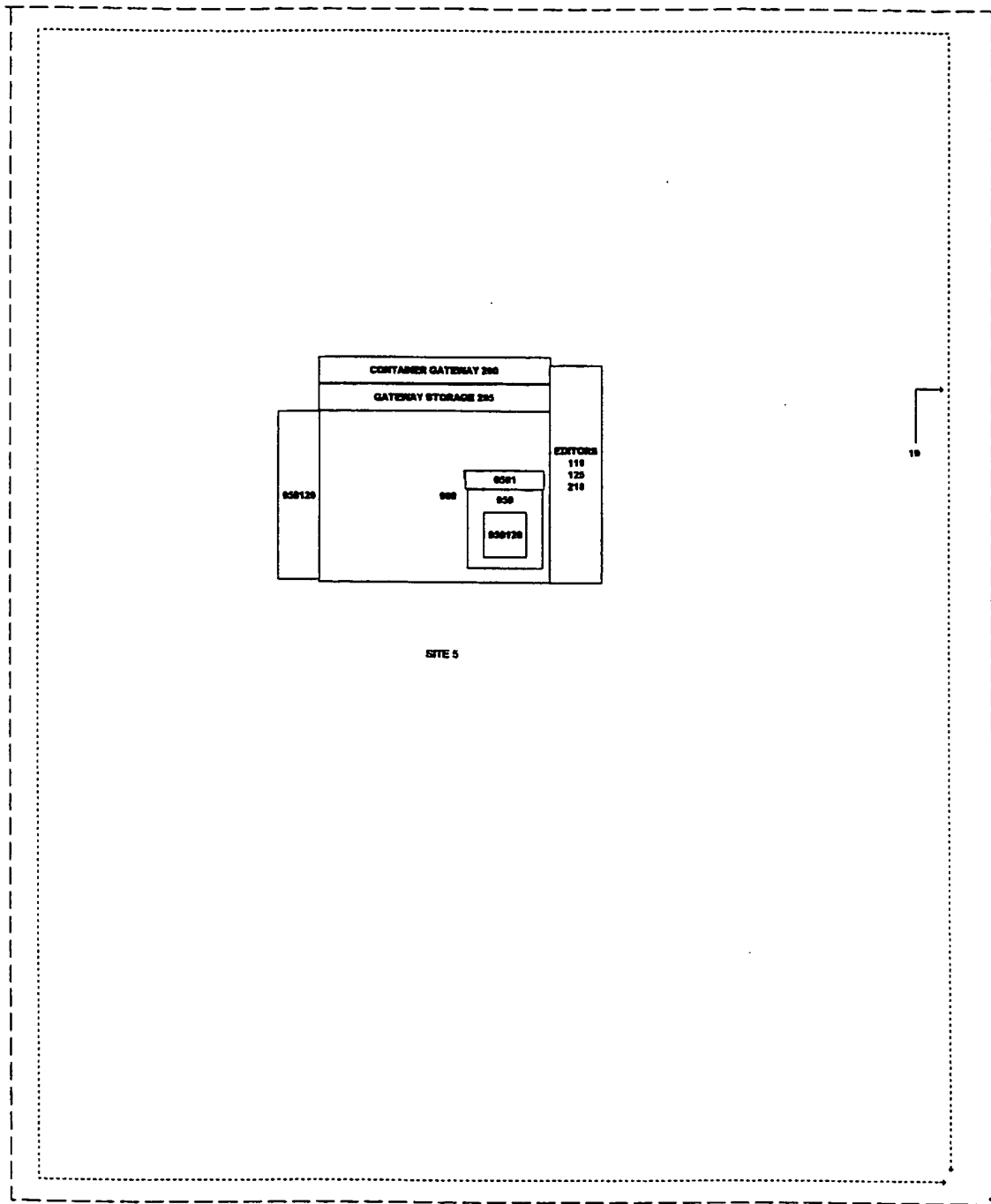


SITE 4

FIG. 2 F



9/30



SITE 5

FIG. 2 G

10/30

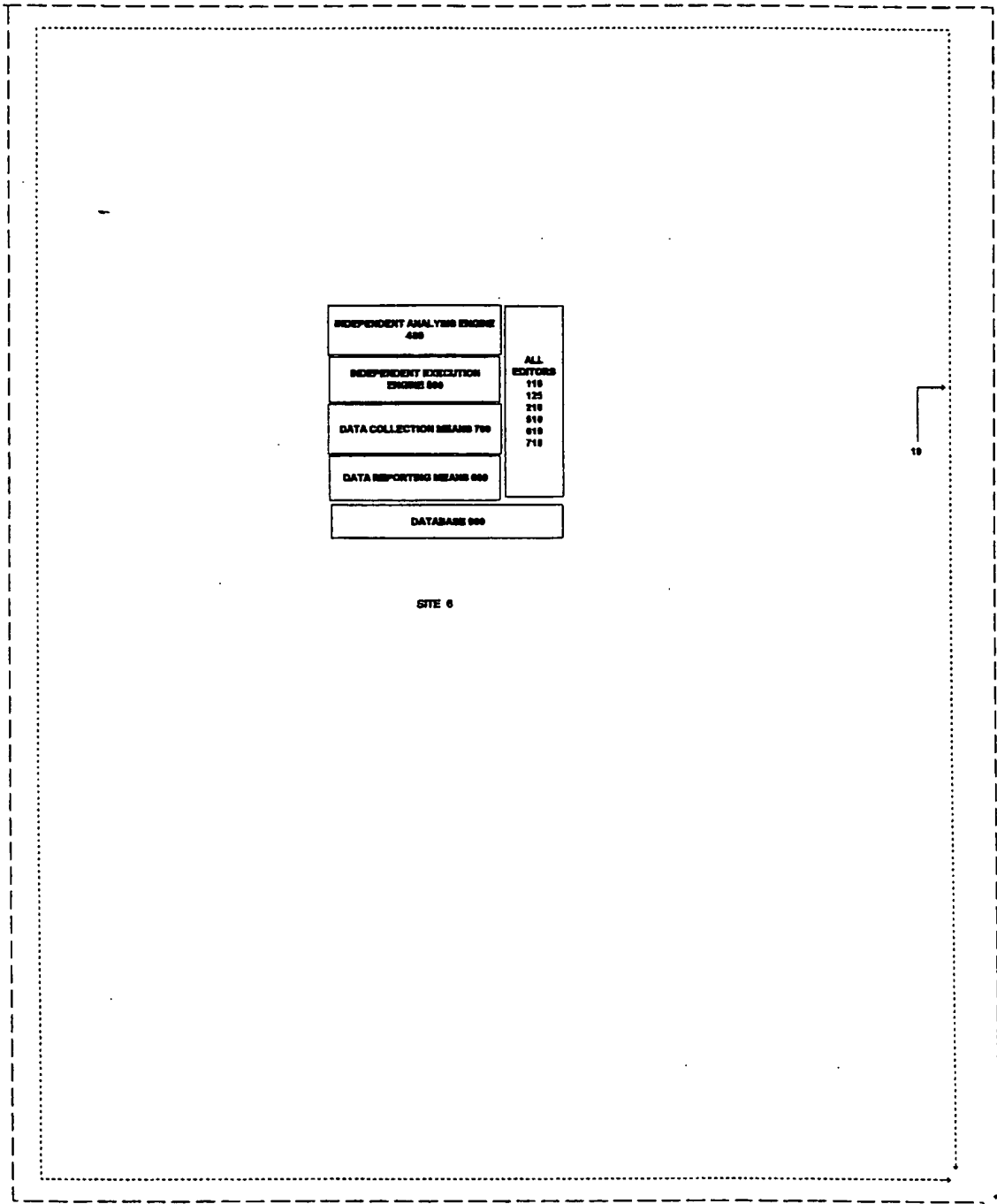


FIG. 2 H

11/30

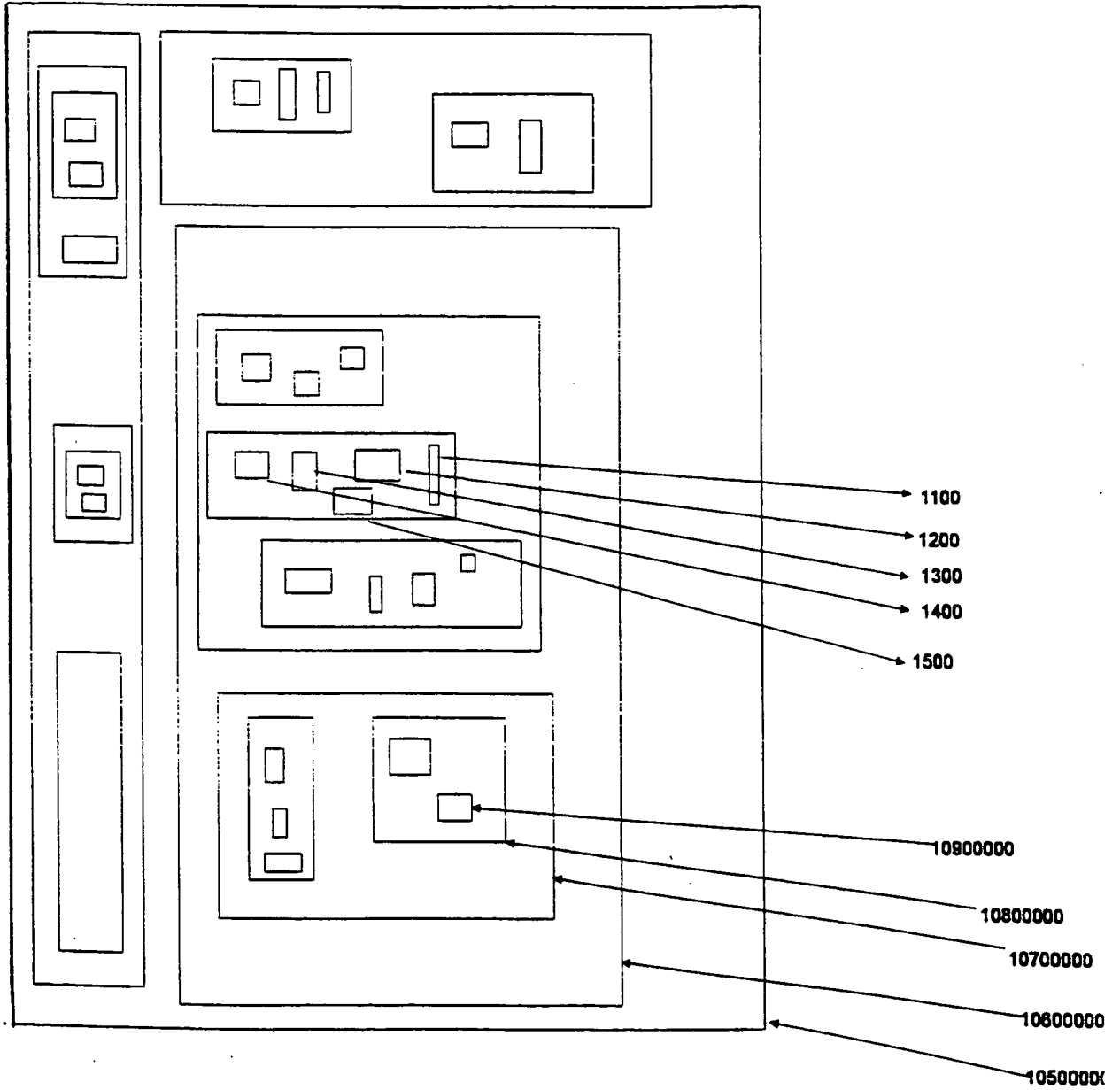


FIG. 3 A

12/30

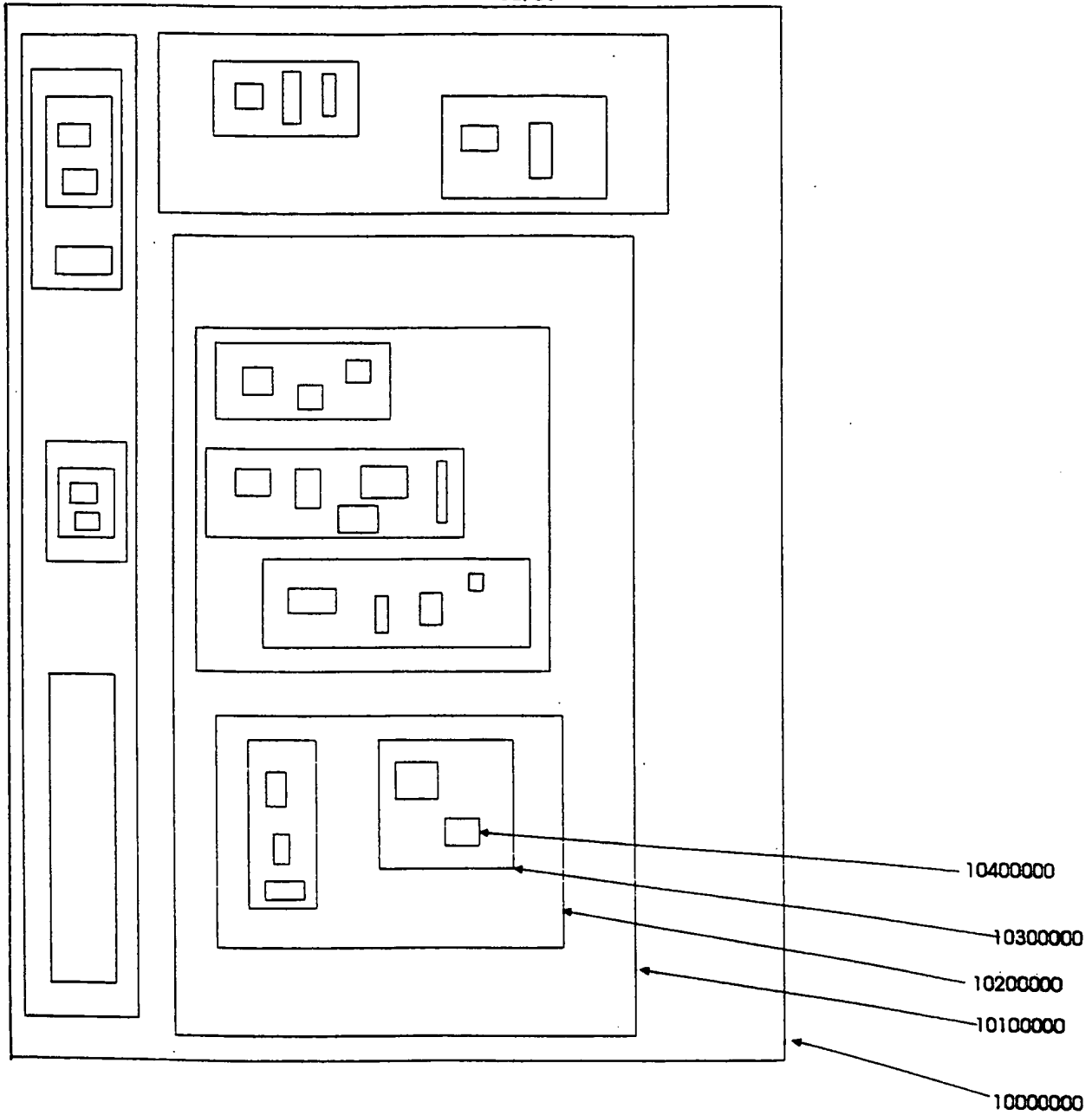


FIG. 3 B

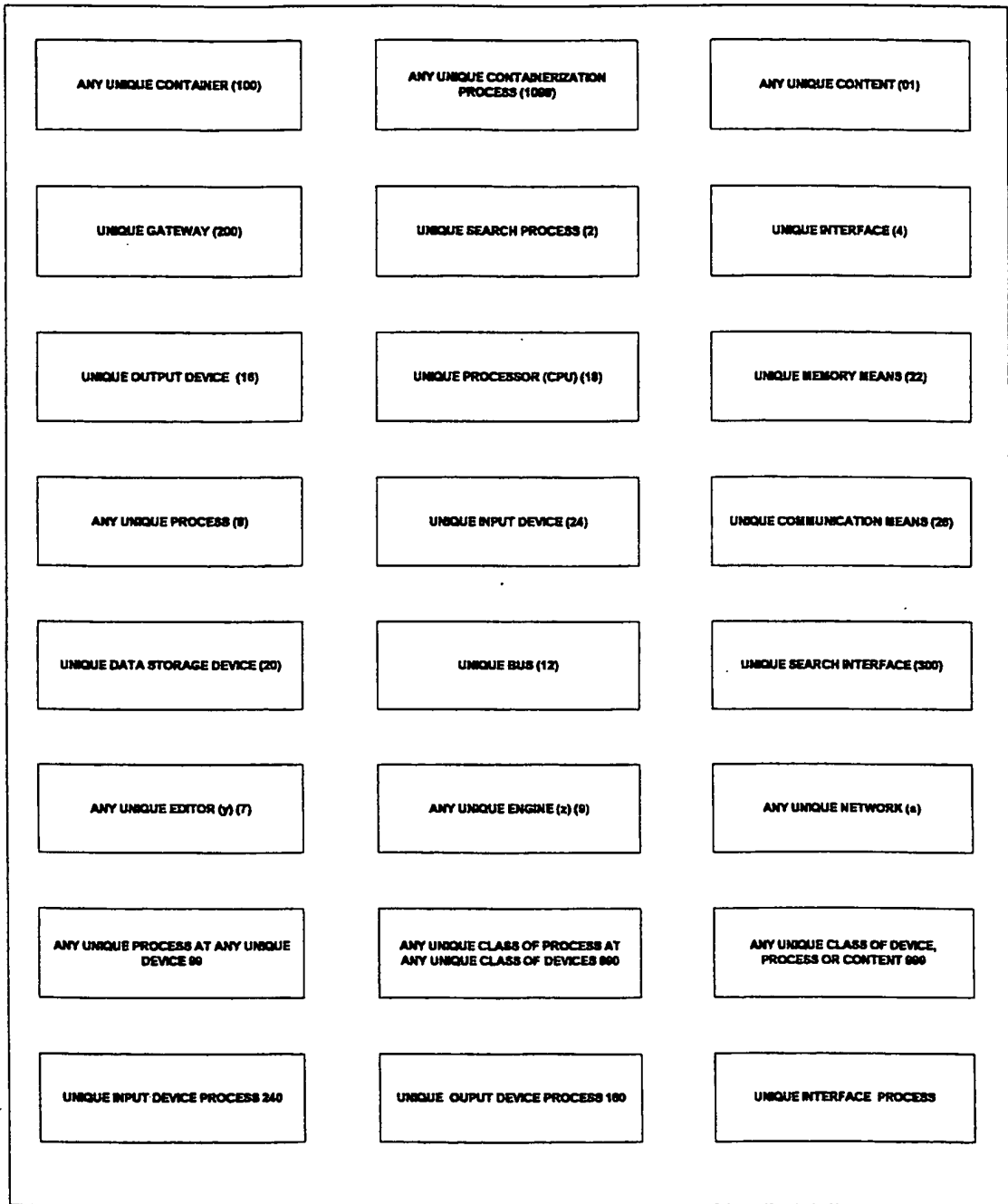
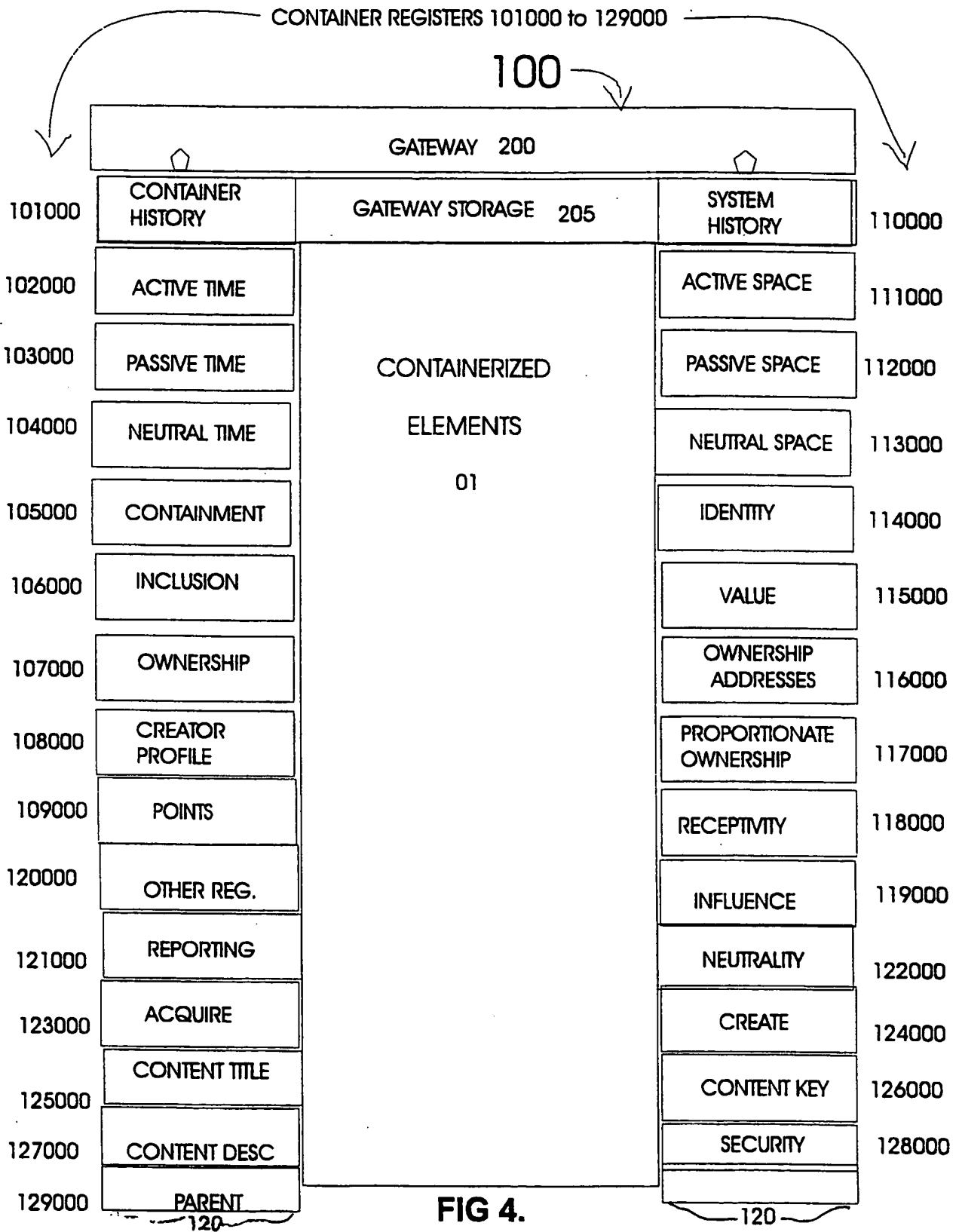


FIG. 3 C

(100)



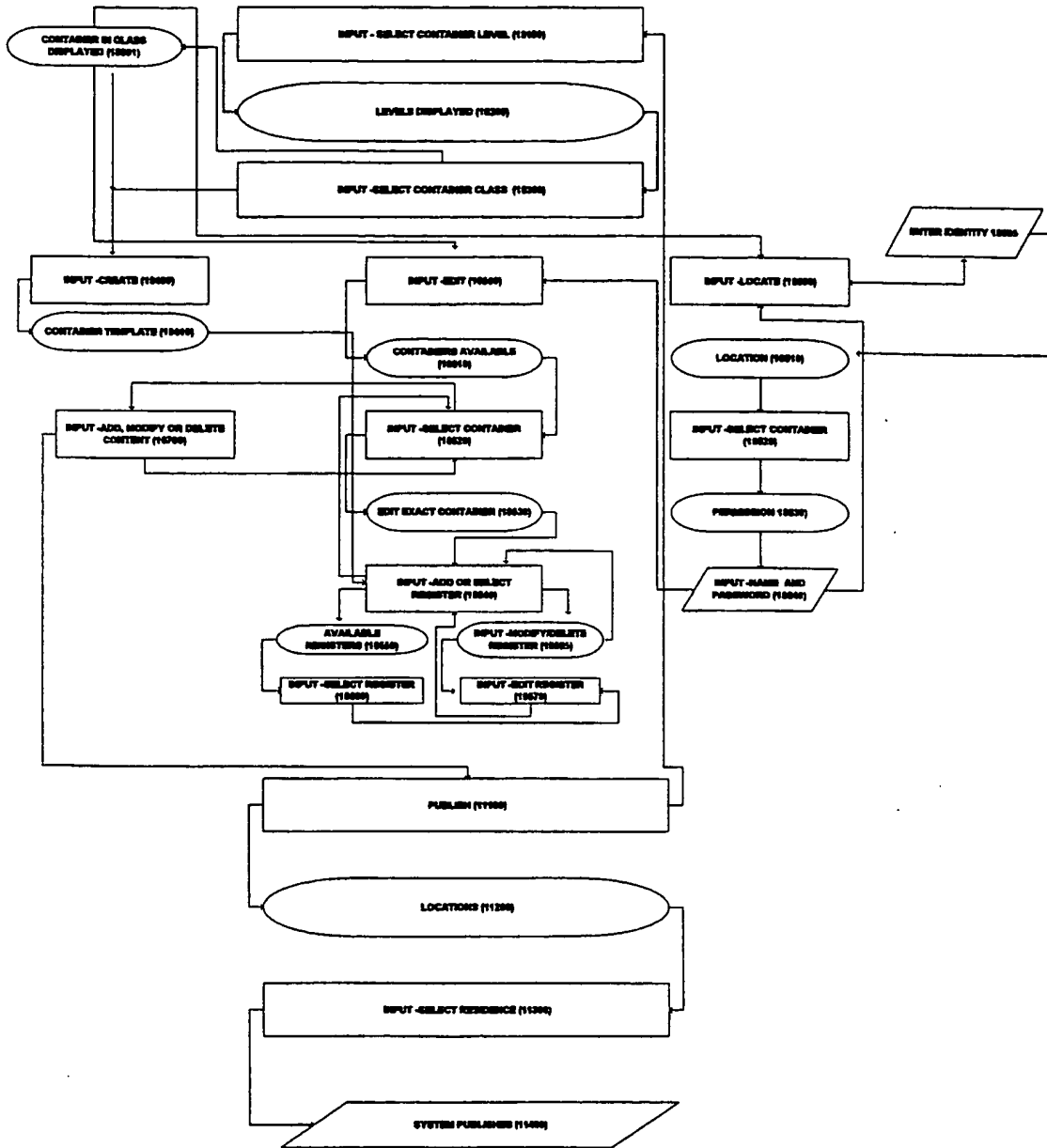


FIG. 5

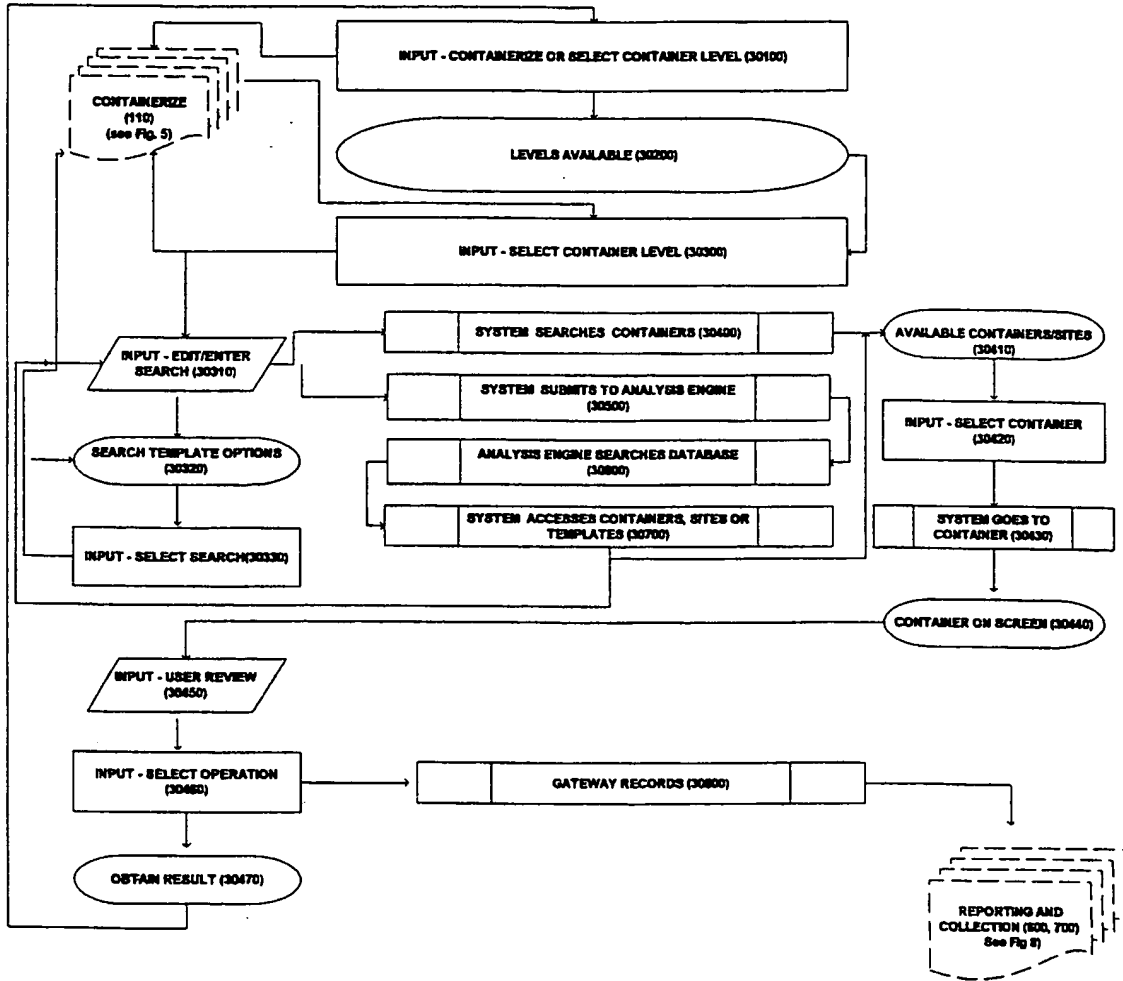


FIG. 6



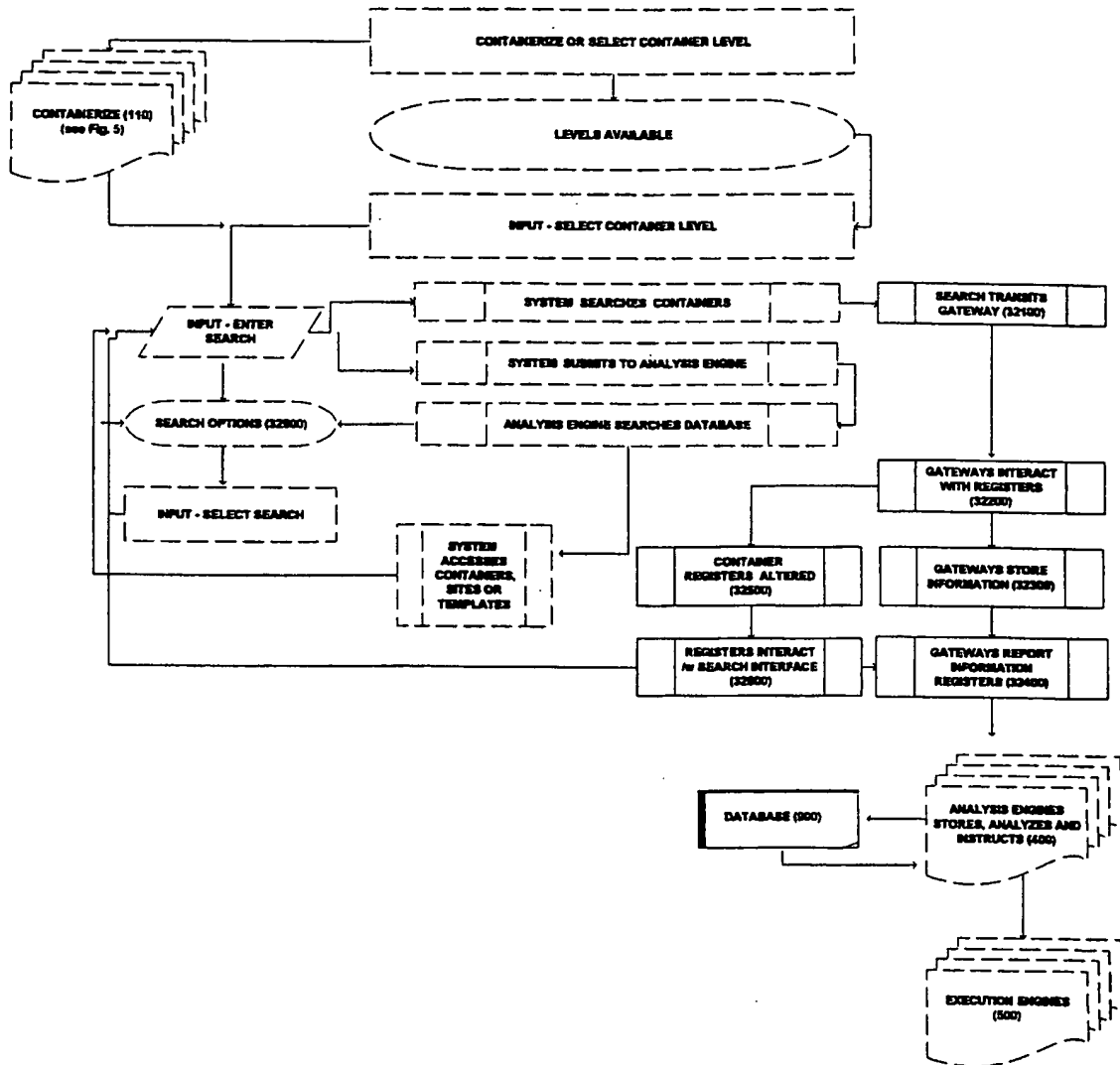


FIG. 7

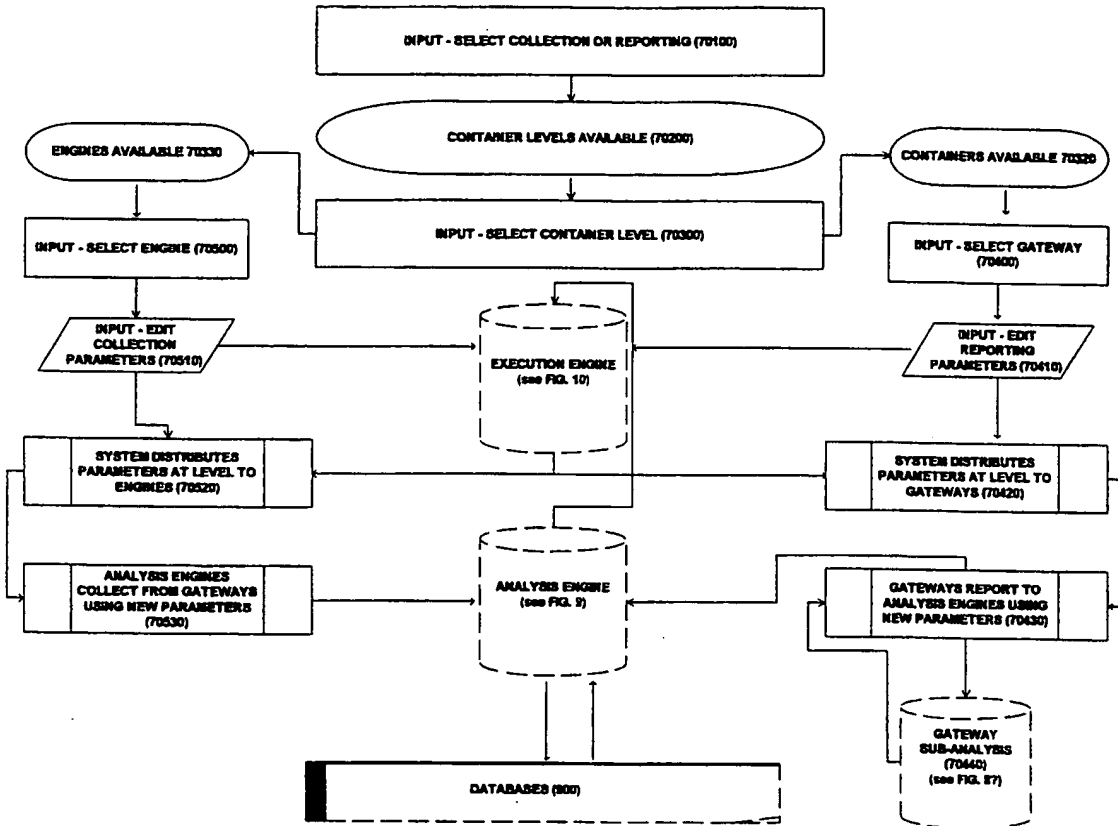


FIG. 8

19/30

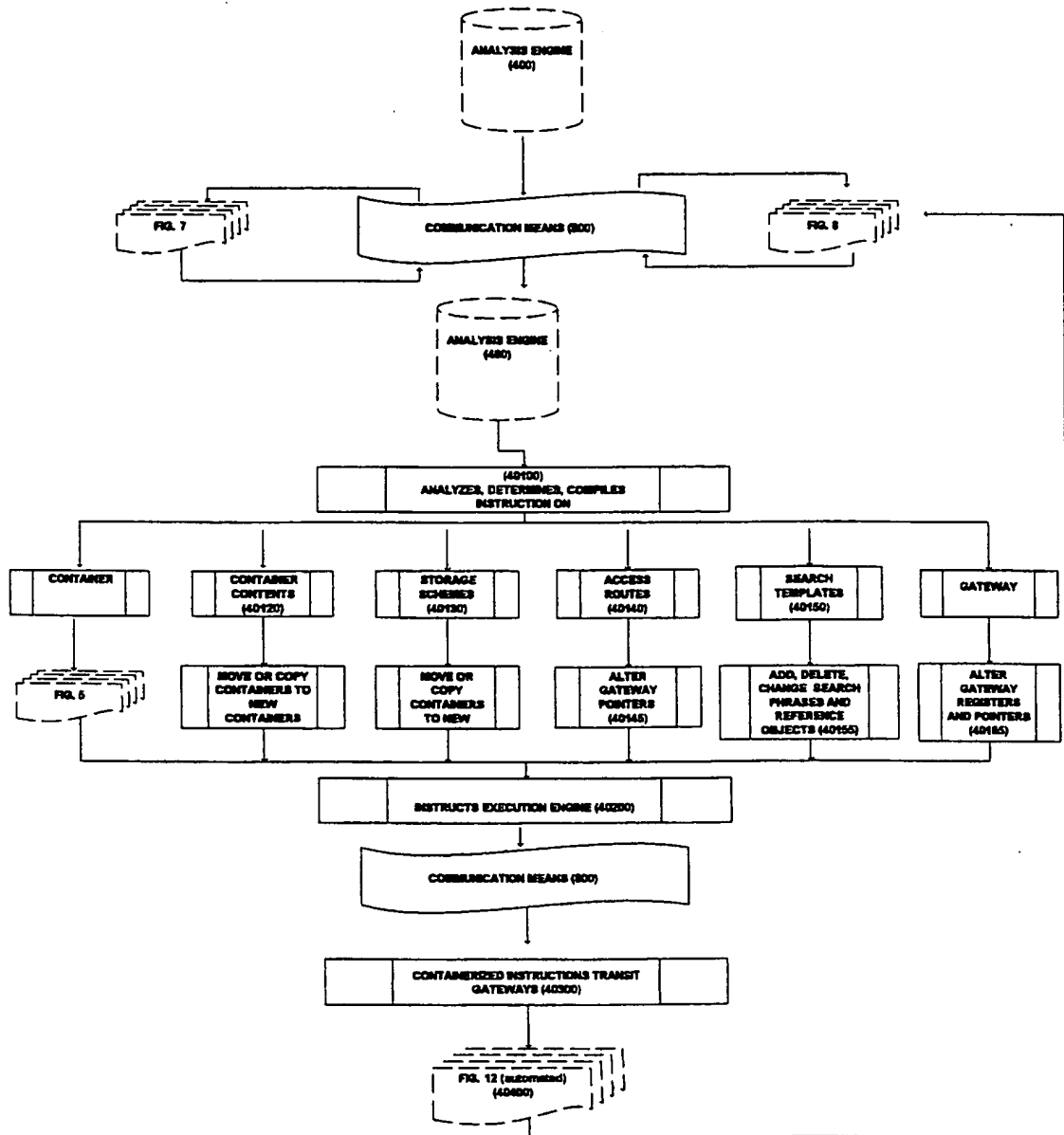


FIG. 9

# EXECUTION ENGINE

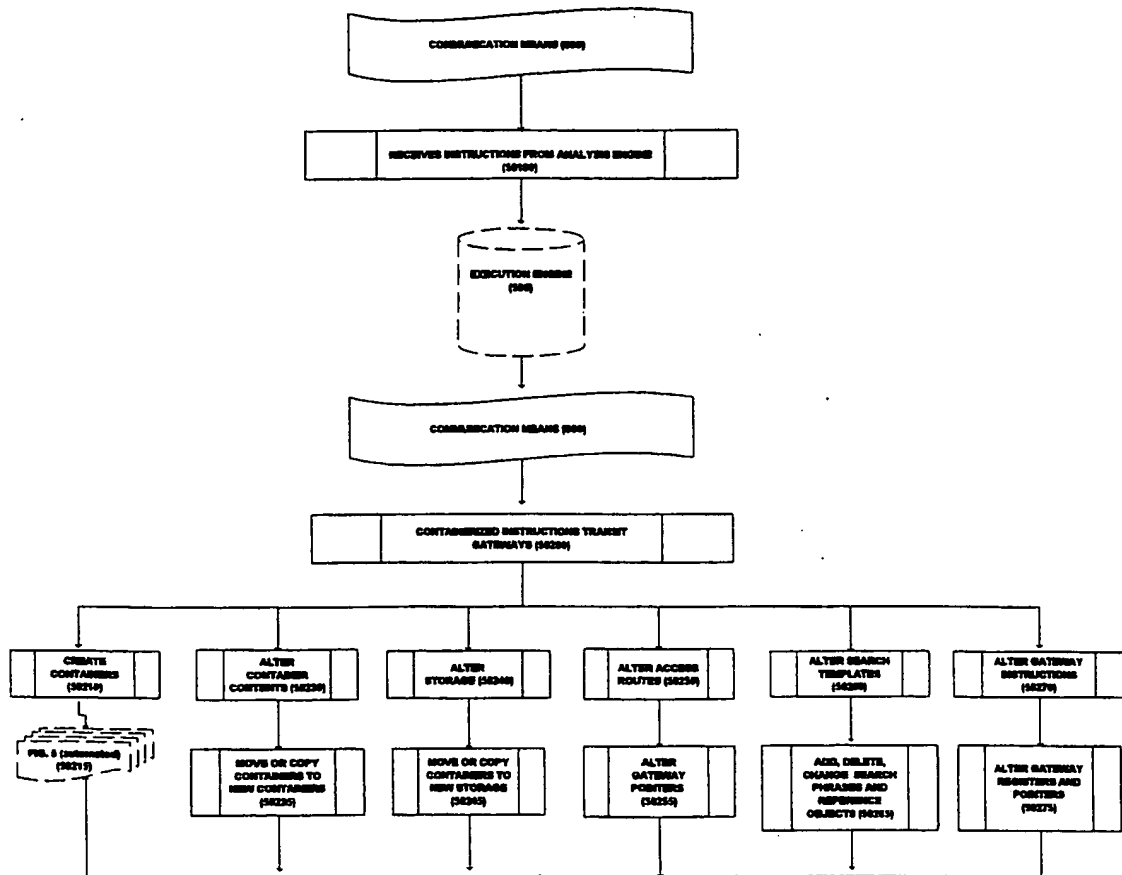


FIG.10

# GATEWAY EDITOR

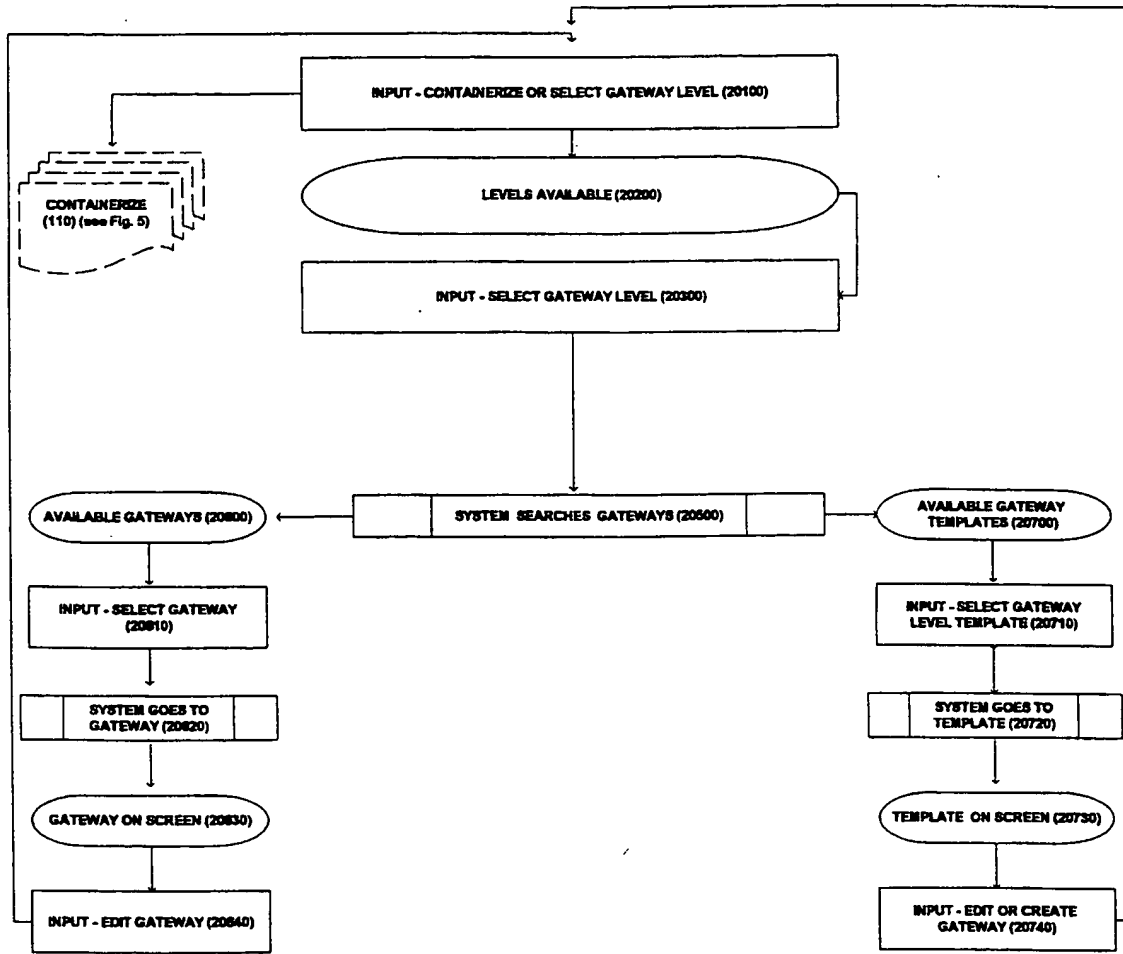


FIG. 11

# GATEWAY PROCESS

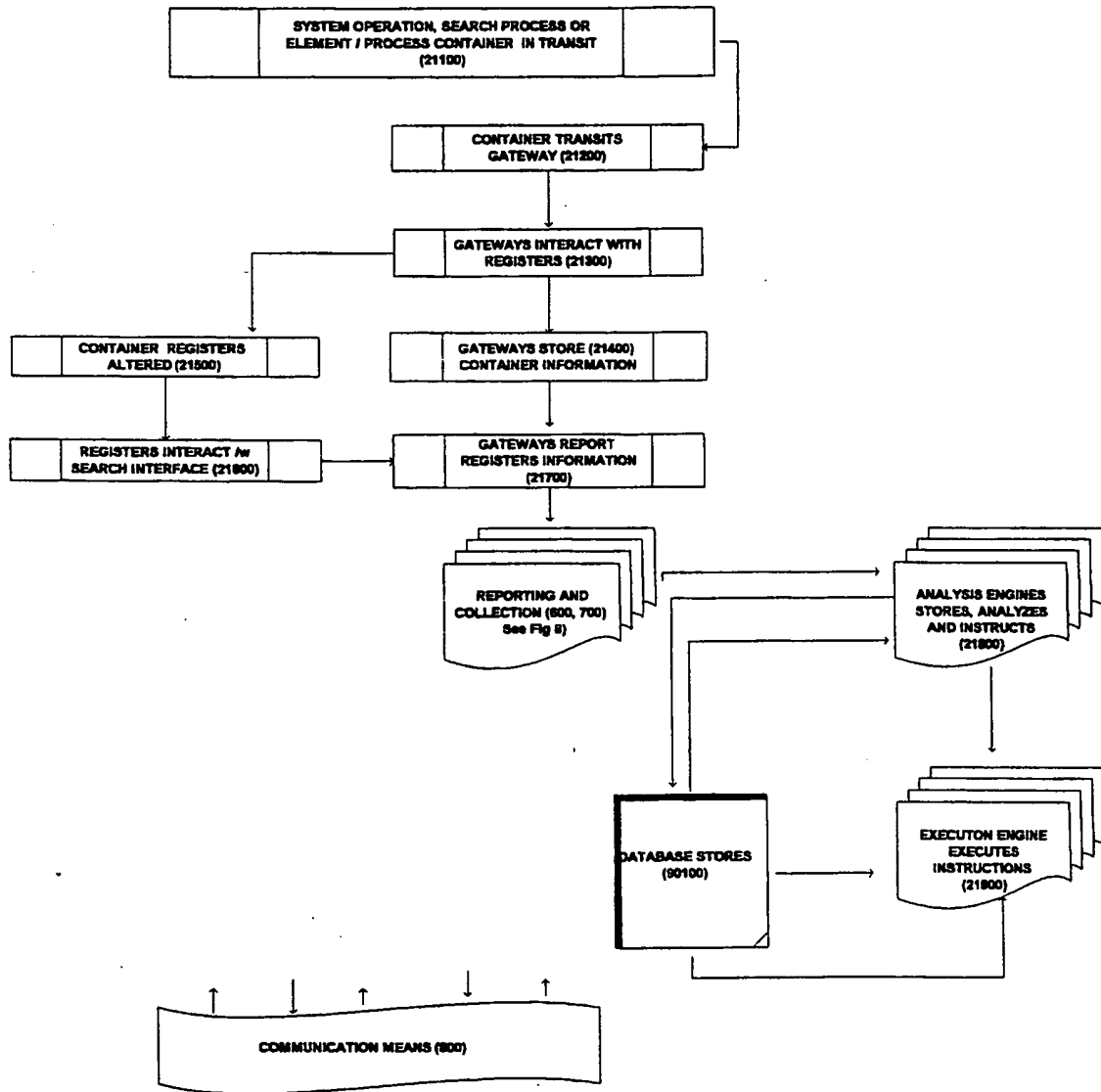


FIG. 12

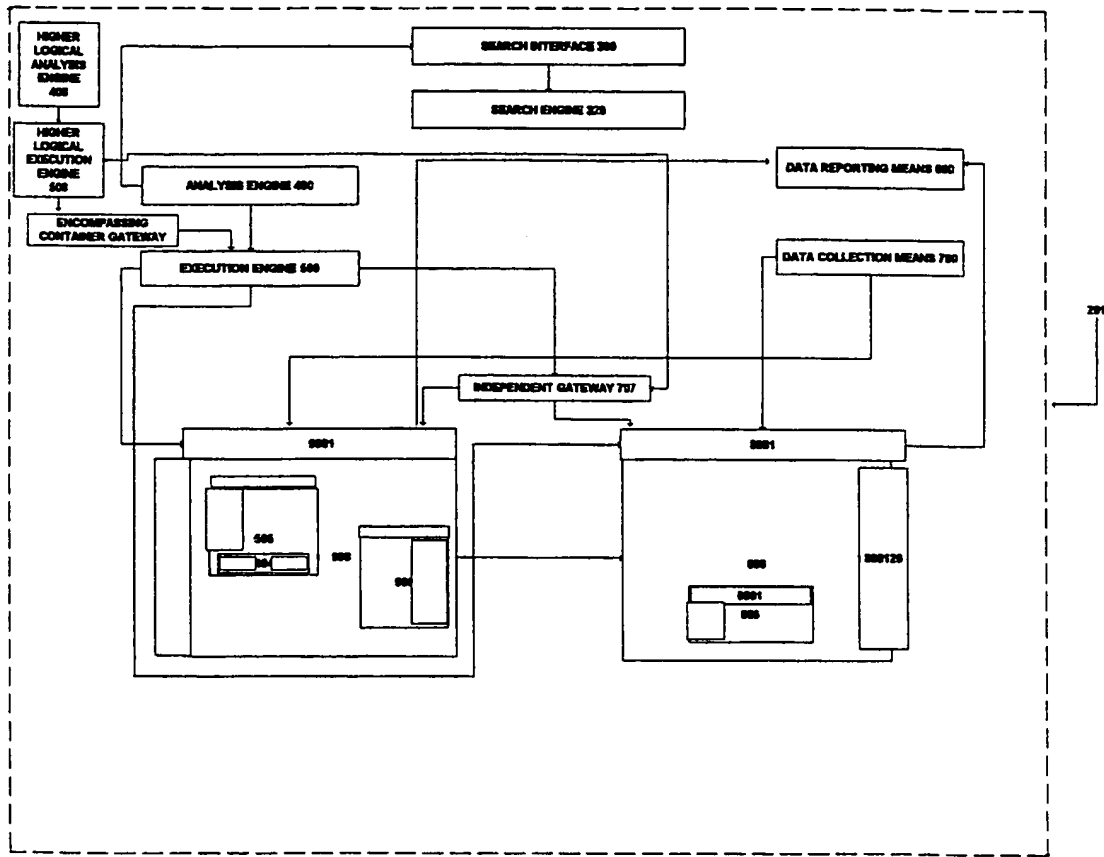


FIG. 13 A

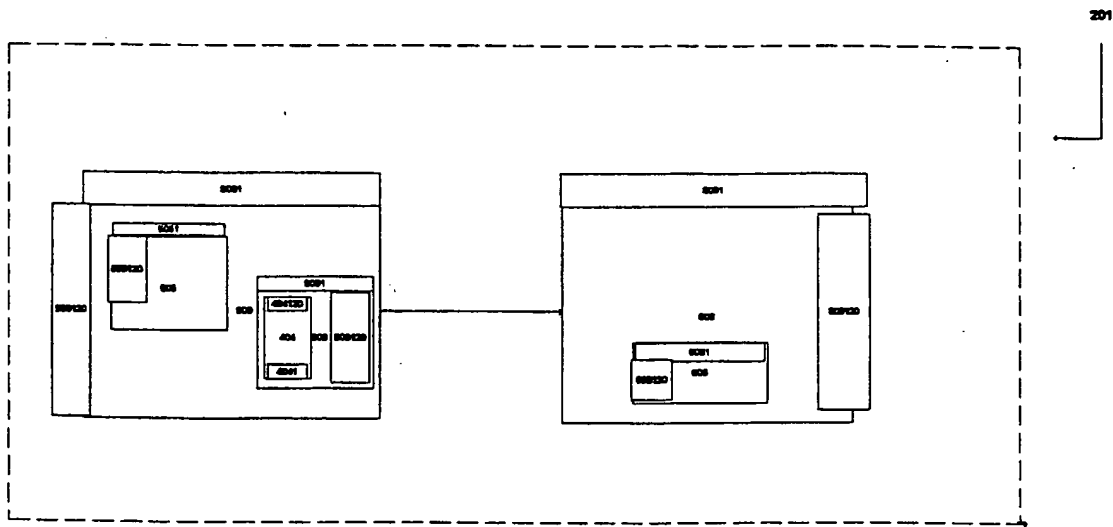


FIG. 13 B



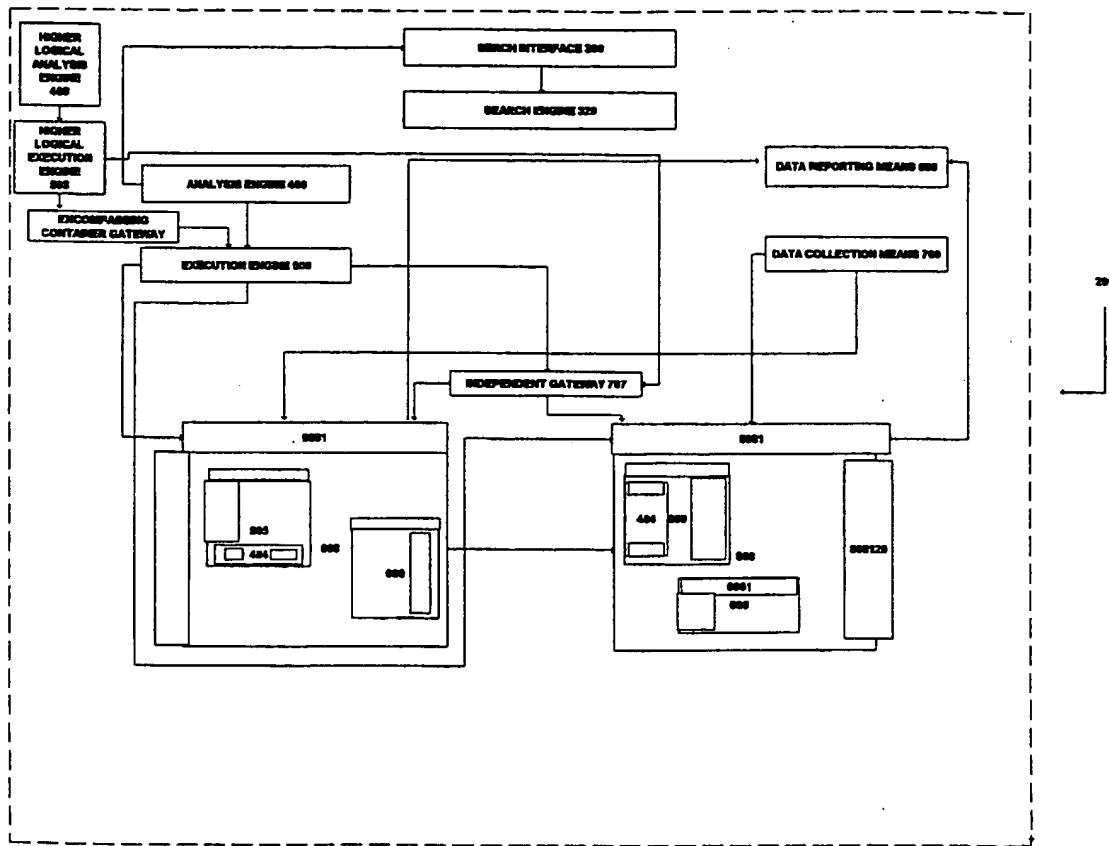


FIG. 13 C

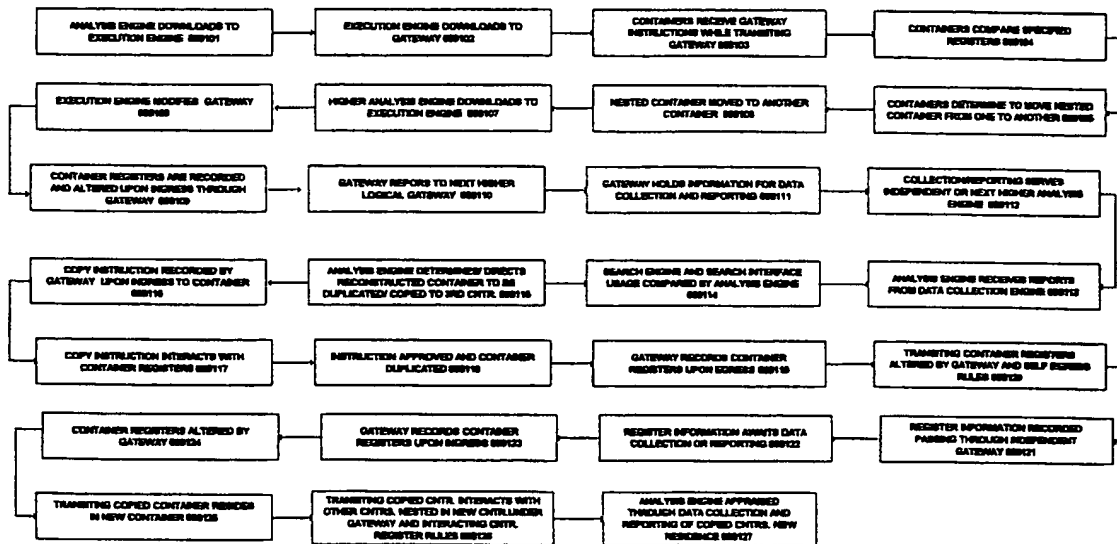


FIG. 13 D

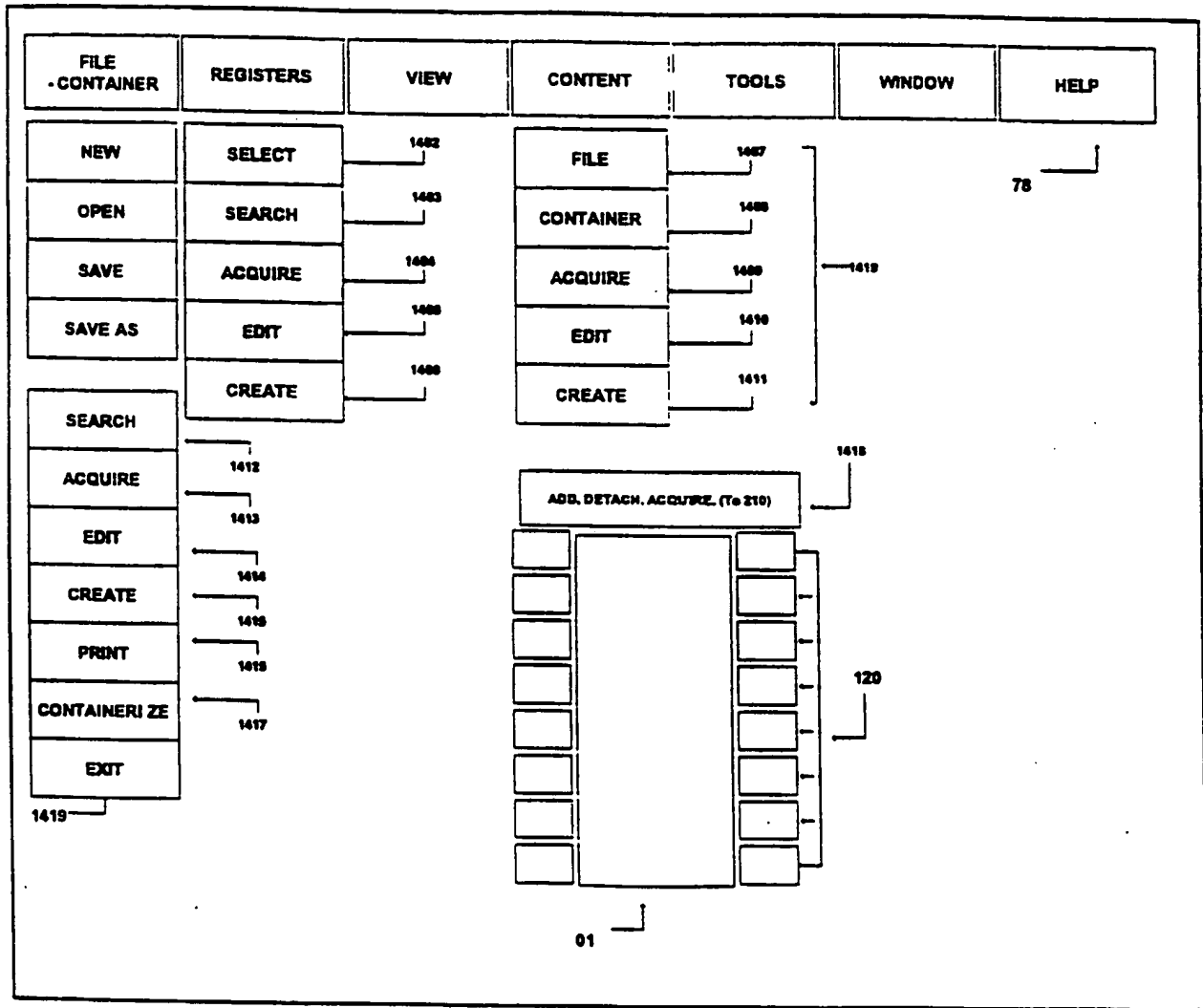


FIG. 14

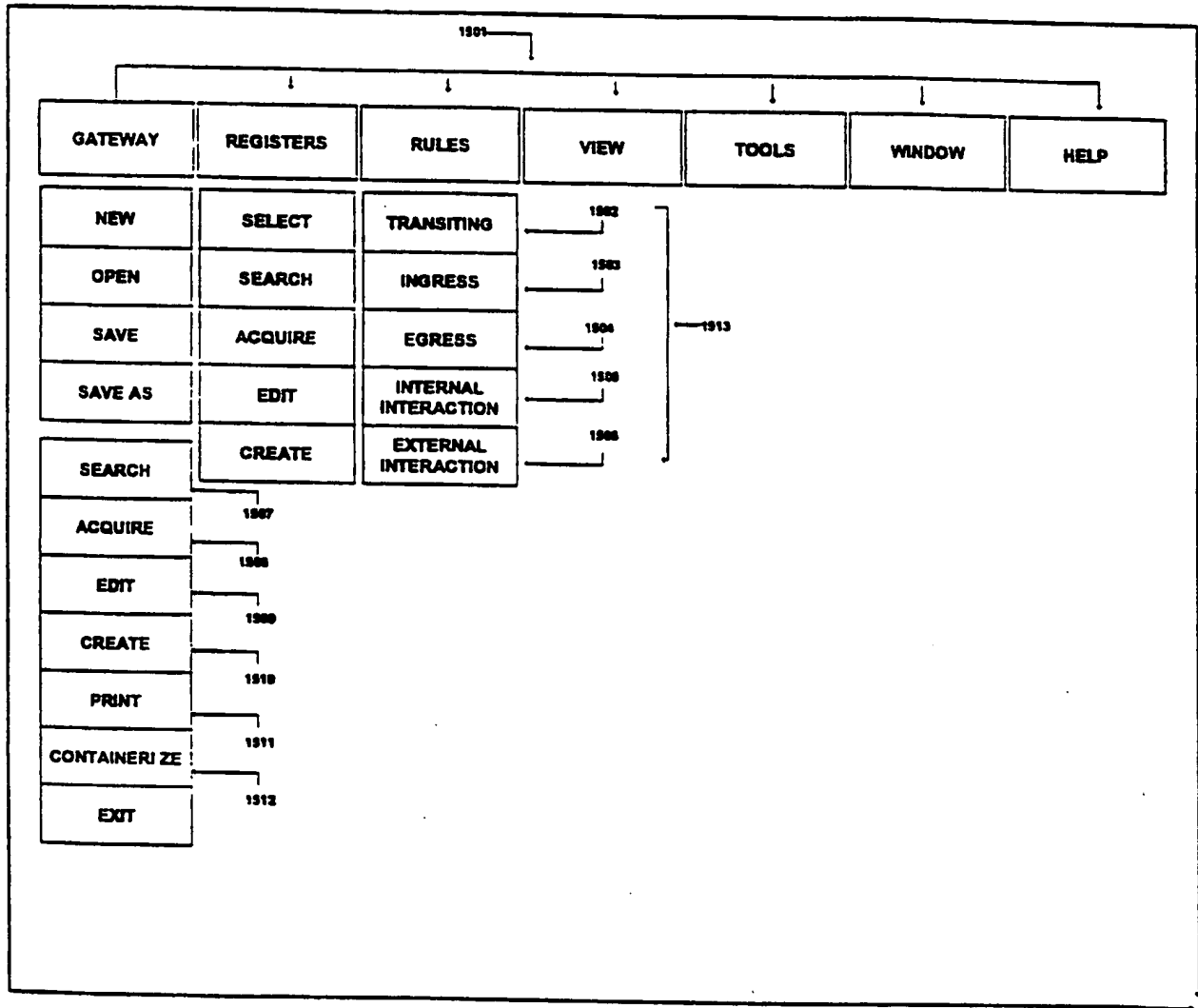


FIG. 15

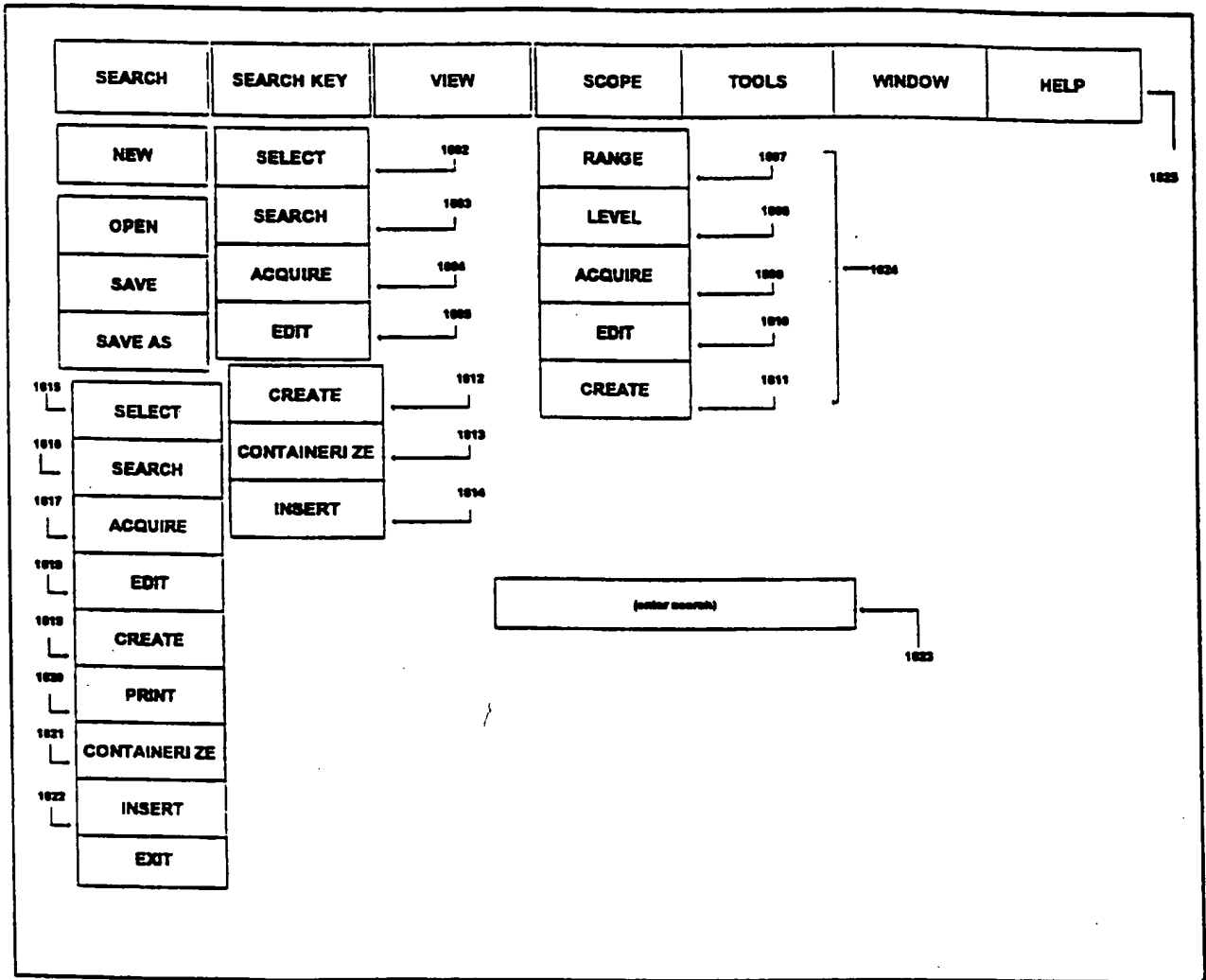


FIG. 16

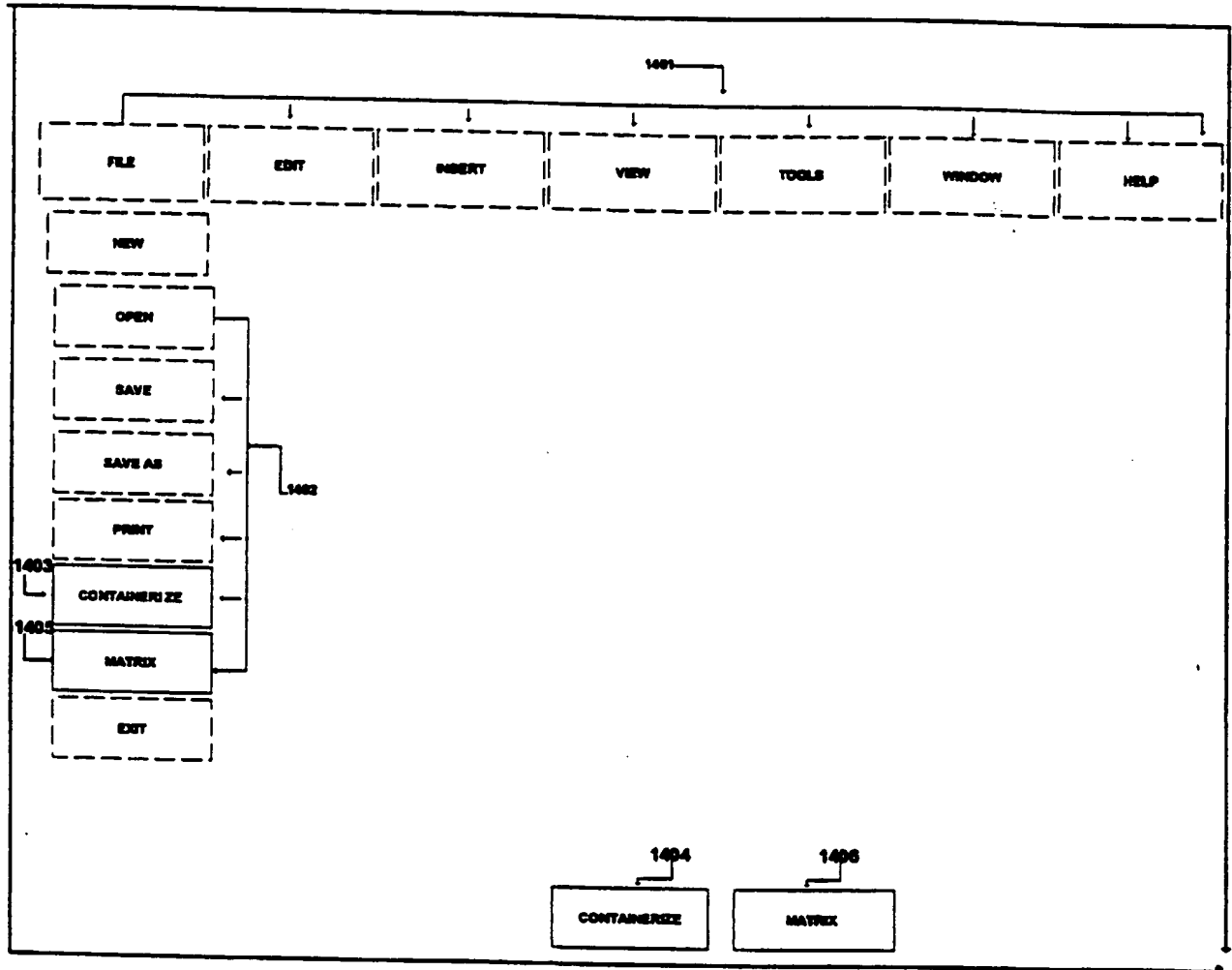


FIG. 17

INTERNATIONAL SEARCH REPORT

International application No. PCT/US99/01988

**A. CLASSIFICATION OF SUBJECT MATTER**  
 IPC(6) : G06F 17/30, 3/14  
 US CL : Please See Extra Sheet.  
 According to International Patent Classification (IPC) or to both national classification and IPC

**B. FIELDS SEARCHED**  
 Minimum documentation searched (classification system followed by classification symbols)  
 U.S. : Please See Extra Sheet.

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched  
 MICROSOFT COMPUTER DICTIONARY

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)  
 APS, PRO-QUEST

**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5,768,510 A (GISH et al) 16 June 1998, column 5.	1-36
A	US 5,848,246 A (GISH et al) 08 December 1998, column 5.	1-36

Further documents are listed in the continuation of Box C.  See patent family annex.

- |   |  |
|---|--|
| * Special categories of cited documents:  | *T* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention  |
| *A* document defining the general state of the art which is not considered to be of particular relevance  | *X* document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone   |
| *E* earlier document published on or after the international filing date  | *Y* document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art |
| *L* document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified) | *A* document member of the same patent family  |
| *O* document referring to an oral disclosure, use, exhibition or other means  |  |
| *P* document published prior to the international filing date but later than the priority date claimed  |  |

Date of the actual completion of the international search 03 JUNE 1999	Date of mailing of the international search report <b>15 JUN 1999</b>
---	--

Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer <i>FOR Ruyian Ho</i> RUAY LIAN HO Telephone No. (703) 305-3834
---	---

INTERNATIONAL SEARCH REPORT

International application No. -- --  
PCT/US99/01988

A. CLASSIFICATION OF SUBJECT MATTER:

US CL :

707/1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 100, 101, 102, 103, 104, 200, 201, 202, 203, 204, 205, 206; 709/202, 203, 218, 228; 713/200, 201

B. FIELDS SEARCHED

Minimum documentation searched

Classification System: U.S.

707/1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 100, 101, 102, 103, 104, 200, 201, 202, 203, 204, 205, 206; 709/202, 203, 218, 228; 713/200, 201





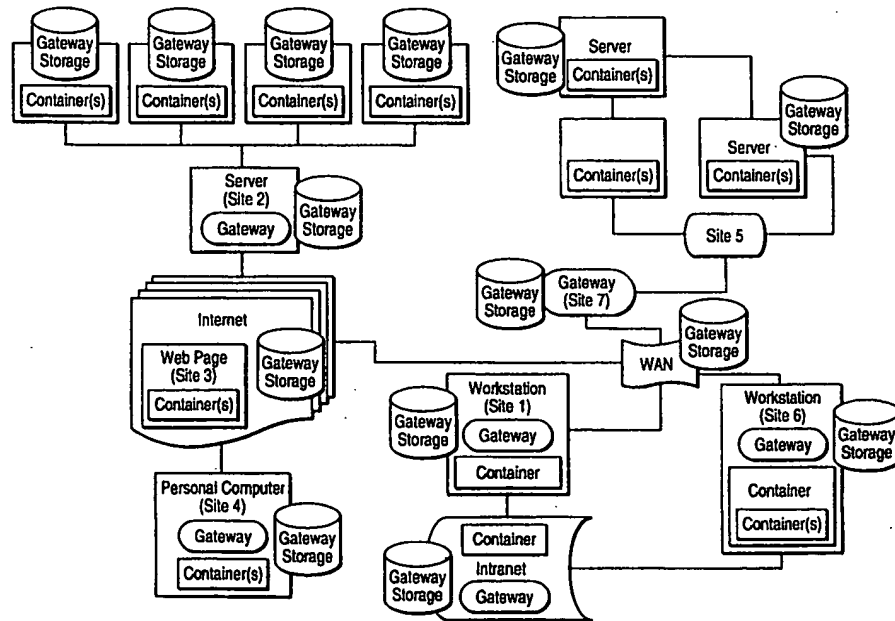
INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

<p>(51) International Patent Classification <sup>6</sup> : <b>G06F 17/30, 3/14</b></p>	<p><b>A1</b></p>	<p>(11) International Publication Number: <b>WO 99/39285</b> (43) International Publication Date: <b>5 August 1999 (05.08.99)</b></p>
<p>(21) International Application Number: <b>PCT/US99/01988</b> (22) International Filing Date: <b>28 January 1999 (28.01.99)</b> (30) Priority Data: <b>60/073,209</b>      <b>30 January 1998 (30.01.98)</b>      <b>US</b> (71) Applicant (for all designated States except US): <b>EMATRIX CORPORATION [US/US]; 104 West Anapamu, Santa Barbara, CA 93101 (US).</b> (72) Inventor; and (75) Inventor/Applicant (for US only): <b>DE ANGELO, Michael [US/US]; Suite 290, 1324 J State Street, Santa Barbara, CA 93101 (US).</b> (74) Agents: <b>SUEOKA, Greg, T. et al.; Fenwick &amp; West LLP, Two Palo Alto Square, Palo Alto, CA 94306 (US).</b></p>		<p>(81) Designated States: <b>AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, 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, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).</b></p> <p><b>Published</b> <i>With international search report. Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.</i></p>

(54) Title: **SYSTEM AND METHOD FOR CREATING AND MANIPULATING INFORMATION CONTAINERS WITH DYNAMIC REGISTERS**

(57) Abstract

A system for creating and manipulating information containers with dynamic registers on a multi-user computer system, or computer network comprises an interactive information container, a container editor, a search interface, a user profile, system-wide hierarchical container gateways (site 7), interactive and evolving container registers, a data collection means, a data reporting means, an analysis engine with editor, an executing engine with editor, and a means of communicating with other computers, computer networks, or digital-based public or published media. The container editor provides an authoring user with the capacity to encapsulate any information component such as a file, set, database, network, event or process, and a set of parameters of multiple container registers to govern the interaction of that container with other containers or processes. The container registers include system-defined, system-alterable, user-defined and user-alterable registers.



**FOR THE PURPOSES OF INFORMATION ONLY**

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

# SYSTEM AND METHOD FOR CREATING AND MANIPULATING INFORMATION CONTAINERS WITH DYNAMIC REGISTERS

## BACKGROUND OF THE INVENTION

### 5       1. Field of the Invention

The present invention relates generally to computer systems in a multi-user mainframe or mini computer system, a client server network, or in local, wide area or public networks, and in particular, to computer networks for creating and manipulating information containers with dynamic interactive registers in a computer, media or publishing network, in order to  
10 manufacture information on, upgrade the utility of, and develop intelligence in, a computer network by offering the means to create and manipulate information containers with dynamic registers.

### 2. Description of the Related Art

In the present day, querying and usage of information resources on a computer network  
15 is accomplished by individuals directing a search effort by submitting key words or phrases to be compared to those key words or phrases contained in the content or description of that information resource, with indices and contents residing in a fixed location unchanging except by human input. Similarly, the class of storage medium upon which information resides, its class and subclass organizational structures, and its routes of access all remain fundamentally  
20 unaltered by ongoing user queries and usage. Only the direct and intended intervention of the owner of the information content or computer hosting site changes these parameters, normally accomplished manually by programmers or systems operators at their own discretion or the discretion of the site owner.

There exists currently in the art a limited means of interfacing a computer user with the  
25 information available on computer networks such as the world wide web. Primarily, these means are search engines. Search engines query thousands or tens of thousands of index pages per second to suggest the location of information while the user waits. While factual information can be accessed, the more complex, particular or subtle the inquiry, the more branches and sub-branches need to be explored in a time consuming fashion in order to have  
30 any chance of success. Further, there are no such automatic devices that reconstruct the information into more useful groupings or makes it more accessible according to factors attached to the content by the content creator such as the space or time relevancy of its content,

or factors attached to the content by the system's compilation and analysis of the accumulated biography of that specific content's readership.

The utility of wide area and public computer networks is thus greatly limited by the static information model and infrastructure upon which those networks operate.

5 One problem is that on a wide area or public network, specific content such as a document remains inert, except by the direct intervention of users, and is modified neither by patterns or history of usage on the network, or the existence of other content on the network.

Another problem is that content does not reside in an information infrastructure conducive to reconstruction by expert rule-based, fuzzy logic, or artificial intelligence based systems. Neither the intelligence of other information users nor the expert intelligence of an  
10 observant network computer system can be utilized in constructing, or re-constructing information resources. Where content resides in a fixed location and structure, "information" becomes something defined by the mind of the information provider rather than the mind of the information user, where the actual construction and utility of information exists.  
15 Information remains, like raw ore, in an unrefined state.

Another problem is that the class of storage medium upon which data resides cannot be system or user managed and altered according to the actual recorded and analyzed hierarchically graded usage of any given information resource residing on that storage medium except by statistical analysis of universal, undefined "hits" or visits to that page or site.

20 Another problem is that information resource groupings remain fixed on the given storage medium location according to the original installation by the resource author, not altered according to the actual recorded and analyzed hierarchically graded usage of that given information resource. Content itself remains inert, with no possibility of evolution.

A further problem with the prior art is that neither the search templates generated by  
25 those more knowledgeable in a given field of inquiry, nor the search strategies historically determined to be successful, or system-constructed according to analyses of search strategies historically determined to be successful, are available to inquiring users. A search template is here defined as one or more text phrases, graphics, video or audio bits, alone or in any defined outline or relational format designed to accomplish an inquiry. Internet or wide area network  
30 search may return dozens of briefs to a keyword or key phrase inquiry sometimes requiring the

time-consuming examination of multiple information resources or locations, with no historical relation to the success of any given search strategy.

5 A further problem is that there is limited means to add to, subtract from, or alter the information content of documents, databases, or sites without communicating with the owners or operators of those information resources, e.g., contacting, obtaining permission, negotiating and manually altering, adding or subtracting content. Additionally, once so altered, there is not a means to derive a proportionate value, and thereby a proportionate royalty as the information is used.

10 A final problem is that the physical residence of a body of data or its cyberspace location may not serve its largest body of users in the most expedient manner of access. Neither the expert intelligence of other information users nor the expert intelligence of an observant computer system is presently utilized by inherent network intelligence to analyze, re-design and construct access routes to information medium except by statistical analysis of universal, undefined "hits" or visits to that page or site.

15 Therefore, there is a need for a system and methods for creating and manipulating information containers with dynamic interactive registers defining more comprehensive information about contained content in a computer, media or publishing network, in order to manufacture information on, upgrade the utility of, and develop intelligence in, a computer network by providing a searching user the means to utilize the searches of other users or the  
20 historically determined and compiled searches of the system, a means to containerize information with multiple registers governing the interaction of that container, a means to re-classify the storage medium and location of information resources resident on the network, a means to allow the reconstruction of content into more useful formations, and a means to reconstruct the access routes to that information.

25

### SUMMARY OF THE INVENTION

The present invention is a system and methods for manufacturing information on, upgrading the utility of, and developing intelligence in, a computer or digital network, local, wide area, public, corporate, or digital-based, supported, or enhanced physical media form or  
30 public or published media, or other by offering the means to create and manipulate information containers with dynamic registers.

The system of the present invention comprises an input device, an output device, a processor, a memory unit, a data storage device, and a means of communicating with other computers, network of computers, or digital-based, supported or enhanced physical media forms or public or published media. These components are preferably coupled by a bus and  
5 configured for multi-media presentation, but may also be distributed throughout a network according to the requirements of highest and best use.

The memory unit advantageously includes an information container made interactive with dynamic registers, a container editor, a search interface, a search engine, a search engine editor, system-wide hierarchical container gateways interacting with dynamic container  
10 registers, a gateway editor, a register editor, a data collection means with editor, a data reporting means with editor, an analysis engine with editor, an executing engine with editor, databases, and a means of communicating with other computers as above. These components may reside in a distributed fashion in any configuration on multiple computer systems or networks.

The present invention advantageously provides a container editor for creating  
15 containers, containerizing storing information in containers and defining and altering container registers. A container is an interactive nestable logical domain configurable as both subset and superset, including a minimum set of attributes coded into dynamic interactive evolving registers, containing any information component, digital code, file, search string, set, database,  
20 network, event or process, and maintaining a unique network-wide lifelong identity.

The container editor allows the authoring user to create containers and encapsulate any information component in a container with registers, establishing a unique network lifelong identity, characteristics, and parameters and rules of interaction. The authoring user defines and sets the register with a starting counter and/or mathematical description by utilizing menus  
25 and simple graphing tools or other tools appropriate to that particular register. The registers determine the interaction of that container with other containers, system components, system gateways, events and processes on the computer network.

Containers and registers, upon creation, may be universal or class-specific. The editor provides the means to create system-defined registers as well as the means to create other  
30 registers. The editor enables the register values to be set by the user or by the system, in which case the register value may be fixed or alterable by the user upon creation. Register values are

evolving or non-evolving for the duration of the life of the container on the system. Evolving registers may change through time, space, interaction, system history and other means.

System-defined registers comprise: (1) an historical container register, logging the history of the interaction of that container with other containers, events and processes on the network, (2) an historical system register, logging the history of pertinent critical and processes on the network, (3) a point register accumulating points based upon a hierarchically rated history of usage, (4) an identity register maintaining a unique network wide identification and access location for a given container, (5) a brokerage register maintaining a record of ownership percentage and economic values, and others.

10 The present invention also includes user-defined registers. User defined registers may be created wholly by the user and assigned a starting value, or simply assigned value by the user when that register is pre-existent in the system or acquired from another user, and then appended to any information container, or detached from any container.

Exemplary user-defined registers comprise (1) a report register, setting trigger levels for report sequences, content determination and delivery target, (2) a triple time register, consisting of a range, map, graph, list, curve or other representation designating time relevance, actively, assigning the time characteristics by which that container will act upon another container or process, passively, assigning the time characteristics by which that container be acted upon by another container or process, and neutrally, assigning the time characteristics by which that container will interact with another container or process, (3) a triple space register, consisting of a range, map, graph, list, curve or other representation designating the domain and determinants of space relevance, actively, assigning the space characteristics by which that content will act upon another container or process, passively, assigning the space, characteristics by which that content will be acted upon by another container or process, and neutrally, assigning the space characteristics by which that container will interact with another container or process, (4) a domain of influence register, determining the set, class and range of containers upon which that container will act, (5) a domain of receptivity register, determining the set, class and range of containers allowed to act upon that container, (6) a domain of neutrality register, determining the set, class and range of containers with which that container will interact, (7) a domain of containment register, determining the set, class and range of containers which that container may logically encompass, (8) a domain

of inclusion register, determining the set, class and range of containers by which that container might be encompassed, (9) an ownership register, recording the original ownership of that containers, (10) a proportionate ownership register, determining the proportionate ownership of that containers, (11) a creator profile register, describing the creator or creators  
5 of that container, (12) an ownership address register, maintaining the address of the creator or creators of that container, (13) a value register, assigning a monetary or credit value to that container, and (14) other registers created by users or the system.

Containers are nestable and configurable as both subset and superset and may be designated hierarchically according to inclusive range, such as image component, image,  
10 image file, image collection, image database, or if text, text fragment, sentence, paragraph, page, document, document collection, document, database, document library, or any arrangement wherein containers are defined as increasingly inclusive sets of sets of digital components.

The present invention also includes, structurally integrated into each container, or  
15 strategically placed within a network at container transit points, unique gateways, nestable in a hierarchical or set and class network scheme. Gateways gather and store container register information according to system-defined, system-generated, or user determined rules as containers exit and enter one another, governing how containers system processes or system components interact within the domain of that container, or after exiting and entering that  
20 container, and governing how containers, system components and system processes interact with that unique gateway; including how data collection and reporting is managed at that gateway. The gateways record the register information of internally nested sub and superset containers, transient containers and search templates, including the grade of access requested, and, acting as an agent of an analysis engine and execution engine, govern the traffic and  
25 interaction of those containers and searches with the information resource of which they are the gateway and other gateways. The gateways' record of internally nested and transient container registers, and its own interaction with those containers, is made available, according to a rules-based determination, to the process of the analysis engine by the data collection and/or data reporting means.

30 The present invention also includes a means of data storage at any given gateway.



The present invention also includes a data collection means, residing anywhere on the network, or located at one or more hierarchical levels of nestable container gateways for gathering information from other gateways and analysis engines according to system, system-generated or user determined rules. The data collection means manages the gathering of data regarding network-wide user choices, usage and information about information, by collecting it from container and gateway registers as those containers and gateways pass through one another. Such statistics as frequency, pattern, and range of time, space and logical class is collected as directed by the analysis engine, and made that data available to the analysis engine by advancing it directly to the analysis engine, or incrementally, to the next greater hierarchically inclusive collection level. The rules of data collection may be manually set or altered by the system manager, or set by the system and altered by the system in its evolutionary capacity.

The present invention also includes a data reporting means, located at one or more hierarchical levels of nestable container gateways for submitting information to other gateways and analysis engines according to system, system-generated or user determined rules. The data reporting means manages the sending of data from the registers, gateways and search templates in a frequency, pattern, and range of time, space and logical class as directed by the analysis engine, and makes that data available to the analysis engine by advancing it directly to the analysis engine, or incrementally to the next greater hierarchically inclusive reporting level. The rules of data collection may be manually set or altered by the system manager, or set by the system and altered by the system in its evolutionary capacity. The data reporting means may be established to work in concert, in redundancy, or in contiguous or interwoven threads of hierarchically nested containers.

The present invention also includes an analysis engine that receives, reports and collects information regarding the interaction of user searches with gateways and container registers, as well as container registers with other container registers, and container registers with gateways. The analysis engine analyzes the information submitted by the gateways and instructs the execution engine to create new information containers, content assemblages, storage schemes, access routes, search templates, and gateway instructions. The analysis engine includes an editor that provides a system manager with a means of editing the operating

principles of that engine, governing data reporting, data collection, search template loading, gateway instructions, and other.

The present invention also includes an execution engine, fulfilling the instructions of the analysis engine, to create new information containers, content sun and superset assemblages, storage schemes, access routes, search templates, and gateway instructions. The execution  
5 engine includes an editor that provides a system manager with a means of editing the operating principles of that engine, governing data reporting, data collection, search template loading, gateway instructions, and other.

The present invention also includes a search interface or browser. The search interface  
10 provides a means for a searching user to submit, record and access search streams or phrases generated historically by himself, other users, or the system. Search streams or phrases of other users are those that have been historically determined by the system to have the highest probability of utility to the searching user. Search streams or phrases generated by the system are those that have been constructed by the system through the analysis engine based upon the  
15 same criteria.

### BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a first and preferred embodiment of a system constructed according to the present invention.

20 FIG. 2 A is block diagram of a preferred embodiment of the memory unit.

FIG. 2 B is an exemplary embodiment of a computer network showing computer servers, personal computers, workstations, Internet, Wide Area Networks, Intranets in relationship with containers and gateways.

FIG. 2B1 is an exemplary embodiment of a computer network showing computer  
25 servers, personal computers, workstations, Internet, Wide Area Networks, Intranets in relationship with containers and gateways and exemplary locations of gateway storage in proximity to one or more of the various sites.

FIGS. 2C through 2H are exemplary embodiments in block diagram form of computer network components showing a possible placement of nested containers, computer servers,  
30 gateways, and the software components named in Fig. 2 A on a network.

FIG. 3A is a graphical representation for one embodiment of a container having a plurality of containers nested within that container.

FIG. 3C is a drawing showing elements that might be logically encapsulated by a container. FIG. 4 is a drawing of an information container showing a gateway and registers  
5 logically  
encapsulating containerized elements.

FIG. 5 is a flowchart showing a preferred method for the containerization process and container editor operating on the communication device.

FIG. 6 is a flowchart showing a preferred method for searching for containers within a  
10 node.

FIG. 7 is a flowchart further showing a preferred method for searching for containers over one or more gateways.

FIG. 8 is a flowchart showing a method for performing the data collection and reporting on containers.

15 FIG. 9 is a flowchart showing the operation of the analysis engine.

FIG. 10 is a flowchart showing the operation of the execution engine.

FIG. 11 is a flowchart showing the operation of the gateway editor.

FIG. 12 is a flowchart showing the operation of the gateway process.

FIG. 13A is a drawing showing an example of nested containers, gateways, registers,  
20 analysis engines and an execution engine prior to container reconstruction as depicted in 13 B,  
13 C and 13 D.

FIG. 13B is a drawing showing the reconstructed nested containers of Figure 13A.

FIG. 13C is a drawing showing further reconstruction of nested containers, with a container relocated to reside within another container.

25 FIG. 13D is a drawing showing a flowchart of the reconstruction process

FIG. 14 is a drawing showing the screen interface of the container editor.

FIG. 15 is a drawing showing the screen interface of the gateway editor.

FIG. 16 is a drawing showing the screen interface of the search interface.

FIG. 17 is a drawing of a generic application program showing a drop-down menu link,  
30 and a button link to the containerization process or container editor.

## DESCRIPTION OF THE PREFERRED EMBODIMENT

### THE SYSTEM

Referring now to FIG. 1, a preferred embodiment of a system 10 for creating and manipulating information containers with dynamic interactive registers in a computer, media, or publishing network 201 in order to manufacture information on, upgrade the utility of, and develop intelligence in that network 201, is shown. The system 10 preferably comprises an input device 24, an output device 16, a processor 18, a memory unit 22, a data storage device 20, and a communication device 26 operating on a network 201. The input device 24, an output device 16, a processor 18, a memory unit 22, a data storage device 20, are preferably coupled together by a bus 12 in a von Neumann architecture. Those skilled in the art will realize that these components 24, 16, 18, 22, 20, and 26 may be coupled together according to various other computer architectures including any physical distribution of components linked together by the communication device 26 without departing from the spirit or scope of the present invention, and may be infinitely nested or chained, both as computer systems within a network 202, and as networks within networks 201.

The output device 16 preferably comprises a computer monitor for displaying high-resolution graphics and speakers for outputting high fidelity audio signals. The output device 16 is used to display various user interfaces 110, 125, 210, 300, 510, 610, 710, as will be described below, for searching for and containerizing information, and editing the container gateways, containers, container registers, the data reporting means and the data collection means, and the search, analysis and execution engines. The author uses the input device 24 to manipulate icons, text, charts or graphs, or to select objects or text, in the process of packaging, searching or editing in a conventional manner such as in the Macintosh or Windows operating systems.

The processor 18 preferably executes programmed instruction steps, generates commands, stores data and analyzes data configurations according to programmed instruction steps that are stored in the memory unit 22 and in the data storage device 20. The processor 22 is preferably a microprocessor such as the Motorola 680(x)0, the Intel 80(x)86 or Pentium, Pentium II, and successors, or processors made by AMD, or Cyrix CPU of the any class.

The memory unit 22 is preferably a predetermined amount of dynamic random access memory, a read-only memory, or both. The memory unit 22 stores data, operating systems,

and programmed instructions steps, and manages the operations of all hardware and software components in the system 10 and on the network 201, utilizing the communication device 26 whenever necessary or expeditious to link multiple computer systems 202 within the network 201.

5           The data storage device 20 is preferably a disk storage device for storing data and programmed instruction steps. In the exemplary embodiment, the data storage device 20 is a hard disk drive. Historical recordings of network usage are stored on distributed and centralized data storage devices 20.

10           The preferred embodiment of the input device 24 comprises a keyboard, microphone, and mouse type controller. Data and commands to the system 10 are input through the input device 24.

15           The present invention also includes a communication device 26. The communication device 26 underlies and sustains the operations of, referring now also to Fig 2 the analysis 400 and execution 500 engines, the data reporting 600 and collection 700 means, the container editor 110, the search interface 300, and the search engine 320, providing the means to search, access, move, copy, utilize or otherwise perform operations with and on data. The communication device 26 utilizes one or more of the following technologies: modem, infrared, microwave, laser, photons, electrons, wave phenomena, cellular carrier, satellite, laser, router hub, direct cabling, physical transport, radio, broadcast or cable TV or other to communicate  
20 with other computers, digital-supported television, computer networks, or digital-based or supported public or published media, or physical media forms, on any a local, wide area, public, or any computer-based computer supported, or computer interfaced network, including but not limited to the Internet. It also allows for the functioning and distribution of any container 100 or container component herein described to reside anywhere on any computer  
25 system in any configuration on that local, wide area, public, or corporate computer-based or computer related network, or digital-based or supported media form.

          Referring now to Figure 2 A, a preferred embodiment of the memory unit 22 is shown. The memory unit includes: an interactive information container 100, a container editor 110, container registers 120, a container register editor 125, system-wide hierarchical container  
30 gateways 200, gateway storage 205, gateway editors 210, engine editors 510, a search interface 300, search engine 320, analysis engine 400, execution engine 500, a data reporting

module, 600, a data reporting editor 610, a data collection module 700, a data collection editor 710, screen interfaces (GUI's) 936, menu or access buttons from generic computer programs 937, and databases 900, all residing in memory optimized between a data storage means 20 such as magnetic, optical, laser, or other fixed storage, and a memory means 22 such as RAM. The memory unit 22 functions by operating on communications network 12 with a communication device 26 on multiple computer systems 202 within the network 201. These components will be described first briefly in the following paragraphs, then in more detail with reference to Figures 3 A through 17.

Those skilled in the art will realize that these components might also be stored in contiguous blocks of memory, and that software components or portions thereof may reside in the memory unit 22 or the data storage means 20.

The present invention includes information containers 100 as noted above. The information container 100 is a logically defined data enclosure which encapsulates any element or digital segment (text, graphic, photograph, audio, video, or other), or set of digital segments, or referring now to FIG. 3 C, any system component or process, or other containers or sets of containers. A container 100 at minimum includes in its construction a logically encapsulated portion of cyberspace, a register and a gateway. A container 100 at minimum encapsulates a single digital bit, a single natural number or the logical description of another container, and at maximum all defined cyberspace, existing, growing and to be discovered, including but not limited to all containers, defined and to be defined in cyberspace. A container 100 contains the code to enable it to interact with the components enumerated in 2 A, and to reconstruct itself internally and manage itself on the network 201.

The container 100 also includes container registers 120. Container registers 120 are interactive dynamic values appended to the logical enclosure of an information container 100, and serve to govern the interaction of that container 100 with other containers 100, container gateways 200 and the system 10, and to record the historical interaction of that container 100 on the system 10. Container registers 120 may be values alone or contain code to establish certain parameters in interaction with other containers 100 or gateways 200.

The present invention also includes container gateways 200. Container gateways 200 are logically defined gateways residing both on containers 100 and independently in the

system 10. Gateways 200 govern the interactions of containers 100 within their domain, and alter the registers 120 of transiting containers 100 upon ingress and egress.

The present invention also includes container gateway storage 205 to hold the data collected from registers 120 of transient containers 100 in order to make it available to the data collection means 700 and the data reporting means 600, and to store the rules governing the operations of its particular gateway 200, governing transiting containers upon ingress and egress, and governing the interactive behavior of containers 100 within the container 100 to which that gateway 200 is attached. Gateway storage 205 may be located on gateways 200 themselves, containers 100 or anywhere on the network 202, 201, including but not limited to Internet, Intranet, LAN, WAN, according to best analysis and use.

The memory unit 22 also includes an execution engine 500 to perform the functions on the system 10 as directed by the analysis engine after its analysis of data from the data reporting means 600, the data collection means 700, and the search interface 300.

The memory unit 22 also includes a search interface 300, by which the user enters, selects or edits search phrases or digital strings to be used by the search engine 320 to locate containers 100.

The memory unit 22 also includes an analysis engine 400 which performs rules based or other analysis upon the data collected from the search interface 300 and the data collection 700 and data reporting 600 means.

The memory unit 22 also includes a data reporting means 600, by which means the information collected by gateways 200 from transient containers 100 is sent to the analysis engine 400.

The memory unit 22 also includes a data collection means 700, by which means the analysis engine 400 gathers the information collected by gateways 200 from transient containers 100.

The memory unit 22 also includes a container editor 110 for creating, selecting, acquiring, modifying and appending registers 120 and gateways 200 to containers 100, for creating, selecting, acquiring, and modifying containers, and for selecting content 01 to encapsulate.

The memory unit 22 also includes a register editor 125, for creating, selecting, acquiring and modifying container registers 120 and establishing and adjusting the values therein.

The memory unit 22 also includes a gateway editor 210, by which means the user determines the rules governing the interaction of a given gateway 210 with the registers 120 of transient containers 100, governing transiting containers upon ingress and egress, and governing the interactive behavior of containers within the container to which that gateway is attached.

The memory unit 22 also includes databases 900, by which means the analysis engine 400, the execution engine 500, the gateways 100, the editors 110, 125, 210, 510, 610, 710, and the search interface 300, store information for later use.

The memory unit 22 present invention also includes a search engine 320 by which means the user is able to locate containers 100 and, referring now to Fig. 4, containerized elements 01.

The memory unit 22 present invention also includes an engine editor 510, by which means the user establishes the rules and operating procedures for the analysis engine 400 and the execution engine 500.

The memory unit 22 present invention also includes a reporting means editor 610, by which means the user establishes the rules and schedule under which the information collected by gateways 200 from transient containers 100 will be sent to the analysis engine 400.

The memory unit 22 present invention also includes a collection means editor 710, by which means the user establishes the rules and schedule under which the analysis engine 400 will gather the information collected by gateways 200 from transient containers 100.

The memory unit 22 present invention also includes screen interfaces (GUI's) 936, specifically designed to simplify and enhance the operations of the container editor 110, the gateway editor 210, and the search interface 300.

The present invention also includes a menu or button access 937, by which a user utilizing any generic computer program may access the system 10 or the container editor 110 from a menu selection(s) or button(s) within that program.

The present invention also includes a computer, media or publishing network 201, comprising computers, digital devices and digital media 202 and a communication device 26, within which the components enumerated in Fig. 2 A interact, compiling, analyzing, and altering containers 100 and the network 201 according to information gathered from container registers 120.



The memory unit 22 also includes one or more computers 202, by which means the components of Fig 1 sustain the operations described in Fig. 2 A.

The memory unit 22 also includes flat or relational databases 900, used where, and as required. Databases are used to store search phrases, search templates, system history for the analysis engine and execution engine, container levels and container, sites and digital elements, or any and all storage required to operate the system.

Referring now to FIG. 2 B, a drawing of a computer network 201 as a system 10, showing a possible placement of nested containers 100, computer servers, gateways 200, on the sites described below. (Note: Fig. 2 B utilizes in parts the same numbering scheme as Fig. 13 A, 13 B, 13 C, 13 D and as Fig. 2 A.) In FIG. 2 B various exemplary sites are shown, any or all of which might interact dynamically within the system. Site 1 shows a single workstation with a container and gateway connected to an Intranet. (Individual containers may be a floppy or CD-Rom to be downloaded or inserted.) Site 2 shows a server with a gateway in relationship to various containers.. Site 3 shows an Internet web page with a container residing on it. Site 4 shows a personal computer with containers and a gateway connected to the Internet. Site 5 shows a configuration of multiple servers and containers on a Wide Area Network.. Site 6 shows a workstations with a gateway and containers within a container connected to a Wide Area Network. Site 7 shows an independent gateway, capable of acting as a data collection and data reporting site as it gathers data from the registers of transiting containers, and as an agent of the execution engine as it alters the registers of transient containers. A container 100 contains the code to enable it to interact with the components enumerated in 2A, and to reconstruct itself internally and manage itself on the network 201. The code resides in and with the container in its registers and gateway definitions and controls. Additional system code resides in all sites to manage the individual and collective operation and oversight of the components enumerated in 2A, with the specific components distributed amongst the sites according to the requirements of optimization.

Referring now to Fig. 2 B 1 various exemplary sites are shown as described above in Fig. 2 B, with the addition of possible location of one or more gateway storage 205 locations.

Referring now to Figures 2 C through 2 H, various exemplary sites with one or more of the logical components of the system 10 in relationship are shown. Site 1 comprises an interactive information container 100, a container editor 110, container registers 120, a

container register editor 125, system-wide hierarchical container gateways 200, gateway storage 205, gateway editors 210, engine editors 510, a search interface 300, search engine 320, analysis engine 400, execution engine 500, a data reporting means 600, a data reporting means editor 610, a data collection means 700, a data collection means editor 710, and  
5 databases 900, all residing on data storage means 20, utilizing the memory unit to function 22, operating on communications network 12 with a communication device 26.

Site 2 comprises an interactive information container 100, a container editor 110, container registers 120, a container register editor 125, system-wide hierarchical container gateways 200, gateway storage 205, gateway editors 210, engine editors 510, search engine  
10 320, analysis engine 400, execution engine 500, a data reporting means 600, a data reporting means editor 610, a data collection means 700, a data collection means editor 710, and databases 900, all residing on data storage means 20, utilizing the memory unit to function 22, operating on communications network 12 with a communication device 26.

Site 3 comprises an interactive information container 100, a container editor 110,  
15 container registers 120, a container register editor 125, hierarchical container gateways 200, gateway storage 205, gateway editors 210, and databases 900, all residing on data storage means 20, utilizing the memory unit to function 22, operating on communications network 12 with a communication device 26.

Site 4 comprises an interactive information container 100, a container editor 110,  
20 container registers 120, a container register editor 125, hierarchical container gateways 200, gateway storage 205, gateway editors 210, a search interface 300, and databases 900, all residing on data storage means 20, utilizing the memory unit to function 22, operating on communications network 12 with a communication device 26.

Site 5 comprises an interactive information container 100, container registers 120, a  
25 container register editor 125, hierarchical container gateways 200, gateway storage 205, and databases 900, all residing on data storage means 20, accessed and utilized by non-resident memory unit 22, operating on communications network 12 with a communication device 26.

Site 6 includes an independent analysis engine 400, execution engine 500, data collection means 700, and data reporting means 600 gateway editors 210, engine editors 510,  
30 a data reporting means editor 610, a data collection means 700, a data collection means editor

710, and databases 900, all residing on data storage means 20, utilizing the memory unit to function 22, operating on communications network 12 with a communication device 26.

Referring now to FIG. 3 A and FIG. 3 B, a block diagram of several nested information containers is shown, including examples of elements, e.g., code 1100, text 1200, audio 1300, video 1400, photograph 1500, graphic images 1600, and examples of possible container level classifications in increasing size, e.g., element 10900000, document 10800000, database 10700000, warehouse 10600000, domain 10500000, and continuing increasingly larger on Fig 3 (B), subject 10400000, field 10300000, master field 10200000, species 10100000. Containers may be infinitely nested and assigned any class, super class or sub class scheme and description by the creator of the container to govern nesting within that container. In addition to digital elements, containers may also include system process and components, including containerization itself.

Referring now to FIG. 3 C, a block diagram of an information container system is shown, listing, without any relationship indicated, some of the possible system components and processes, or sets thereof, that may be encapsulated as elements 01 in an information container 100. An information container 100 may include one or more of the following: any unique, container 100, gateway 200, output device 16, input device 24, output device process 160, input device process 240, data storage device 20, data storage device process 2000, processor 18, bus 12, content 01, search process 02, interface 04, memory unit 22, communication device 26, search interface 300, search process 98, network 201, class of device, process or content 999, class of process at any unique class of device 990, process at any unique device 99, editor 110, 125, 210, 510, 610, 710, engine 320, 400, 500, containerization process 1098, or process 08.

Any container may include (n) other containers, to infinity. The use of value evolving container registers 120 in conjunction with gateways 200, data reporting modules 600, data collection modules 700, the analysis engine 400, and the execution engine 500 provides the information container 100 with extensive knowledge of the use, operation of its internal contents, prior to, during and after those contents' residence within that container 100, and extensive knowledge of the use, operation and contents of the system 10 external to itself, and allows the container 100 to establish and evolve its own identity and course of interaction on the system 10. Further, containers 100, as logical enclosures, can exist and operate

independent of their digital contents, whether encapsulating audio, video, text, graphic, or other.

Referring now to FIG. 4, a block diagram of an information container 100 is shown. The information container 100 is a logically defined data enclosure which encapsulates any element, digital segment (text, graphic, photograph, audio, video, or other), set of digital segments as described above with reference to FIG. 3 (C), any system component or process, or other containers or sets of containers. The container 100 comprises the containerized elements 01, registers 120 and a gateway 200.

Registers 120 appended to an information container 110 are unique in that they operate independently of the encapsulated contents, providing rules of interaction, history of interaction, identity and interactive life to that container 100 through the duration of its existence on a network 201, without requiring reference to, or interaction with, its specific contents. They enable a container 100 to establish an identity independent of its contents. Additionally, registers 120 are unique in that their internal values evolve through interaction with other containers 100, gateways 200, the analysis engine 400, the execution engine 500, and the choices made by the users in the search interface 300, the container editor 110, the register editor 125, the gateway editor 210, the engine editor 510. Registers 120 are also unique in that they can interact with any register of a similar definition on any container 100 residing on the network 201, independent of that container's contents. Registers 120, once constructed, may be copied and appended to other containers 100 with their internal values reset, to form new containers. Register values, when collected at gateways 200 and made available to the analysis engine 400 through the data collection means 700 and the data reporting means 600, provide an entirely new layer of network observation and analysis and operational control through the execution engine 500. Registers 120 accomplish not only a real time information about information system, but also a real time information about information usage on a network. Further, because the user base of a network determines usage, the system 10, in gathering information about information usage, is observing the choices of the human mind. When these choices are submitted to the analysis of a rules-based or other analysis engine 400, the system 10 becomes capable of becoming progressively more responsive to the need of the user base, in effect, learning to become more useful by utilizing the execution engine 500 to create system-wide changes by altering the rules of gateway 200

interaction and thereby altering the registers 120 of transient containers 100 and establishing a complete evolutionary cycle of enhanced utility.

Further, in establishing the pre-defined registers as described in the following four paragraphs, the following unique aspects of information about information are utilized for the first time: 1) the dynamic governance of information according to its utility through time, in active, passive and neutral aspects, as explained below; 2) the dynamic governance of information according to its utility through space in active, passive and neutral aspects, as explained below; 3) the dynamic governance of information according to its ownership, as explained below; 4) the dynamic governance of information according to its unique history of interaction as an identity on a network, as explained below; 5) the dynamic governance of information according to the history of the system on which it exists, as explained below; 6) the dynamic governance of information according to established rules of interaction, in active, passive and neutral aspects, as explained below; 7) the dynamic governance of information according to the profile of its creator, as explained below; 8) the dynamic governance of information according to the value established by its ongoing usage, as explained below; 9) the dynamic governance of information according to its distributed ownership, as explained below; 10) the dynamic governance of information according to what class of information it might be incorporated into, and according to what class of information container it might incorporate, as explained below; 11) the dynamic governance of information according to self-reporting, as explained below.

Referring now to Fig 4, registers 120 may be (1) pre-defined, (2) created by the user or acquired by the user, or (3) system-defined or system-created. Pre-defined registers 120 are those immediately available for selection by the user within a given container editor as part of that container editor, in order that the user may append any of those registers 120 to a container 100 and define values for those registers 120 where required. Registers 120 created by the user are those conceived and created by a specific user or user group and made immediately available for selection by the user or user group in conjunction with any of a wide number of container editors, in order that the user may append any of those registers 120 to a container 100 and define values for those registers 120 where required. Registers 120 acquired by the user are those registers existing network-wide 201, created by the user base, that might be located and acquired by the user in order that the user may append any of those registers

120 to a container 100 and define values for those registers 120 where required. System-defined registers are those registers whose values are set and/or controlled by the system 10. System-created registers are those registers created by the system 10.

Registers 120 are user or user-base created or system-created values or ranges made available by the system 10 to attach to a unique container, and hold system-set, user-set, or system-evolved values. Values may be numeric, may describe domains of time or space, or may provide information about the container 100, the user, or the system 10. Registers 120 may be active, passive or interactive and may evolve with system use. Pre-defined registers include, but are not limited to, system history 110000, container history 101000, active time 102000, passive time 103000, neutral time 104000, active space 111000, passive space 112000, neutral space 113000, containment 105000, inclusion 106000, identity 114000, value 115000, ownership 107000, ownership addresses 116000, proportionate ownership 117000, creator profile 108000, receptivity 118000, influence 119000, points 109000, others 120000, reporting 121000, neutrality 122000, acquire 123000, create 124000, content title 125000, content key phrase(s) 126000, and content description 127000, security 12800, and parent rules 129000.

Pre-defined registers comprise an historical container register 101000, logging the history of the interaction of that container 100 with other containers, events and processes on the network 201, an historical system register 110000, logging the history of pertinent critical and processes on the network, a point register 109000 accumulating points based upon a hierarchically rated history of usage, an identity register 114000 maintaining a unique network wide identification and access location for a given container specifying a unique time and place of origin and original residence, a proportionate ownership register 117000 maintaining a record of ownership percentage and economic values, and others 120000.

User-defined registers include a report register 121000 setting trigger levels for report sequences, content determination and delivery target, three time registers, consisting of a range, map, graph, list, curve or other designating time relevance, 102000 assigning the time characteristics by which that container will act upon another container or process, 103000 assigning the time characteristics by which that container be acted upon by another container or process, and 104000 assigning the time characteristics by which that container will interact with another container or process, three space registers, consisting of a range, map, graph, list,

curve or other designating the domain and determinants of space relevance, **111000** assigning the space characteristics by which that content will act upon another container or process, **112000** assigning the space, characteristics by which that content will be acted upon by another container or process, and **113000** assigning the space characteristics by which that container will interact with another container or process, a domain of influence register **119000**, determining the set, class and range of containers upon which that container will act, a domain of receptivity register **118000**, determining the set, class and range of containers allowed to act upon that container, a domain of neutrality register **122000**, determining the set, class and range of containers with which that container will interact, a domain of containment register **105000**, determining the set, class and range of containers which that container may logically encompass, a domain of inclusion **106000** register, determining the set, class and range of containers by which that container might be encapsulated, an ownership register **107000**, recording the original ownership of that containers, a creator profile register **108000**, describing the creator or creators of that container, an ownership address register **116000**, maintaining the address of the creator or creators of that container, a value register **115000**, assigning a monetary or credit value to that container, other registers **120000** created by users or the system, a reporting register **121000**, determining the content, scheduling and recipients of information about that container, a neutrality register **122000**, an acquire register **123000**, enabling the user to search and utilize other registers residing on the network, a create register **124000**, enabling the user to construct a new register, a content title register **125000**, naming the contents of the container, a content key register, **126000**, identifying the container contents with a key phrase generated by the user and/or the system based upon successful usage of that phrase in conjunction with the utilization of the information within that container **100**, a content description register **127000**, identifying the container contents with additional description, a security register **128000**, controlling container security, and a parent container register **129000**, storing the rules governing container interaction as dictated by the parent (encapsulating) container.

The container also includes a gateway **200** and gateway storage **205**.

Gateways **200** are logically defined passageways residing both on containers **100** and independently in the system **10**. Gateways **200** govern the interactions of containers **100**

encapsulated within their domain by reading and storing register 120 information of containers entering and exiting that container 100.

The present invention also includes container gateway storage 205. Gateway storage 205 stores information regarding the residence, absence, transience, and alteration of 5 encapsulated and encapsulating containers 100, and their attached registers 120, holding the data collected from registers 120 of transient containers 100 in order to make it available to the data collection means 700 and the data reporting means 600, and storing the rules governing the operations of its particular gateway 200.

Referring now to FIG. 5, a flow chart of the preferred method for creating a container 10 100 is shown.

Input is received from the user selecting a container level through use of a drop-down menu 10100. A menu of all possible container classes within the subset and superset scheme of multiple hierarchically nested containers, i.e.; element, document, file, database, warehouse, domain, and more, is displayed on the output device 10200. Input is received from the user 15 selecting a class 10300.

A graphic representation of a container in that class, with registers common to all containers as well as registers unique to its class is displayed 10301.

Input is received from the user choosing to "create" 10400, "edit" 10500, or "locate" 10600.

When the input of "create" 10400 is received from the user, a container template in that 20 class appears 10410. Input from the user is then received adding or selecting a register 10540 to append to that container template. When input is received from the user adding a register, a list of registers that might be added to that class of container is made available to select 10550. Input is received from the user selecting a register 10560 and editing it 10570. The menu 25 returns to "add or select" 10540.

If the input of "locate" 10600 is received from the user, the system prompts the user to enter the identity of the container or class of containers 10605. The system locates the container(s) 10610. Input is received from the user selecting a container 10620. The system prompts the user for a security code for permission to access the container for template use, or 30 to alter its registers, or to alter its content 10630. Input is received from the user entering a name and password providing access to one of the security levels 10640. Input is received



from the user editing the container accordingly by transition to step 10500 and performing the steps for editing.

If the input of "edit" 10500 is received, a list of containers available to edit at that level is shown 10510. Input is received from the user selecting a container 10520. That container appears, available to edit 10530. Input is received from the user selecting "add" or "select" registers 10540 by the user clicking on the graphically depicted register, or from a drop down menu. Input is received from the user selecting the register to edit 10560. Input is received from the user selecting "modify" or "delete" for that register 10565. If input is received from the user to "delete," that register is severed from the container. If input is received from the user to "modify", the register editor 10570 screen appropriate to that register appears, i.e., an x-y type graph to define a curve of relevant active time, in which the user manipulates the x-y termini, scale and curve, or a global map in which Input is received from the user selecting the locale of active space, whether zip code, city, county, state, country, continent, plant or other. When input is received from the user saving the definition, the screen returns to the main container screen to make another selection available. . Input is received from the user defining as many registers as he chooses. One of the registers may be named "new register." Input is received from the user selecting the new register, and if chosen by the user, defining a wholly unique and new kind of register by the user entering input into the register editor 125.

When the input is received from the user choosing to add a register, a list of registers that might be added to that class of container are made available to select 10550. Input is received from the user selecting a register 10560 and editing it 10570. The menu returns to "add or select" 10540, and in turn to Input – Select Container.

Input may then be received from the user choosing to add, modify, or delete the container contents 10700. Once the registers are defined, input is received from the user indicating completion and the interface reverts to the container editor. When input is received from the user choosing "select component" (to select the component to containerize) from the main menu bar 10700, a window appears allowing the user to select any file, component, or other container. If for example, the user were creating a warehouse container, and wishes to incorporate several databases into that container, input would then be received from the user selecting "database." The program would prompt the user for the location (directory) of that database or container. If the requested selection is not containerized, input may then be

received from the user choosing to containerize the element at that time, after which the program returns to "select component." Once input is received from the user defining the database location, the program logically encases the directory or directories in the defined container. The above procedure may be repeated as many times as desired to include multiple  
5 databases within a single container. While logical simplicity would dictate that all containers within a container be of the same subset, it would be possible for input to be received from the user choosing containers of any subset to include in the container. When input is received from the user choosing "finished," the container is created with a unique network identity, preferably through some combination of exact time and digital device serial number, or  
10 centralized numbering system, or other means. The container 100 contains all digital code, including data and program software from the selected items or containers.

Input may then be received from the user to publish the container 11100 at a user-identified or system suggested location 11200 to be selected 11400.

Input is received from the user to "publish", from the main menu bar 11100. Input is  
15 received from the user choosing to leave the container where it was created, move or copy it to another drive, directory, computer, or network the user designates, or select the location from location options offered by the system 11200, or submit, or duplicate and submit, the container to the analysis engine 400 for intelligent inclusion in other containers, thus allowing the system to publish the container as instructed or choose the residence of the container 11400.

20 If input is received from the user to choosing to "move," or "copy" a browse function allows the user to name the new location or browse a list of possible locations. If input is received from the user choosing to "submit," a browser function allows the user to name the analysis search engine 310 or browse a list of possible analyses engines. When input is received from the user choosing the residence of the container 11300, the program restores the  
25 search interface screen.

Referring now to FIG. 6, a flow chart of the method for searching for containers 100.

When input is received from the user selecting "search interface" from the main title bar, the search interface screen appears. The user is given the choice of containerizing selected content or requesting that container levels be displayed 30100. From a drop down menu  
30 another menu appears allowing input to be received from the user selecting the container level

30200. Input is received from the user selecting the container level (from the smallest component to the whole system) 30300.

Input is received 30310 from the user selecting the phrases, containers or components, which then are re-submitted to the same process, until the input is received from the user selecting a specific site or container.

The search phrase, whether containerized or not, is submitted simultaneously to the search engine 30400 and the analysis engine 30500.

The screen then reports in a selection menu, the number of applicable sites found by the search engine 30410, the number of historically proven applicable sites found by the analysis engine 30410, the number of historically proven applicable containers at the selected container level or any container level found by the analysis engine 30410, and the number of historically proven new search phrases or digital segments found by the analysis engine 30320. . Input is received from the user selecting one of the named sets above 30330. If input is received from the user choosing the search engine, the search interface lists the applicable site titles with a brief description 30410. If input is received from the user choosing the site list of the analysis, the search interface lists the applicable site titles with a brief description 30410. . If input is received from the user choosing the container list of the analysis engine, the search interface lists the applicable container titles with a brief description 30410. . If input is received from the user selecting a container 30420, the system offers the means to view titles and descriptions of sub-containers at any chosen class level. . If input is received from the user choosing the phrase list of the analysis engine, the search interface lists the applicable phrases or digital segments with a brief description 30320. The search and search result cycle repeats until input is received from the user choosing to go to an individual container or site.

Input is received from the user entering text or any digital string describing his search objectives into a text or search box. When input is received from the user submitting the search string, the system provides the option of containerizing the search through the container editor 110. Once the search container 101 is created, the system restores the search interface 300 screen the user.

Input is received from the user selecting "search", "supported search" or "both" from another drop-down menu and from submitting the search. When input is received from the user selecting "search" 30310, the search phrase is submitted to the search engine 30400,

which searches both content and the appropriate container registers, as pre-indexed in the search engine, and returns a list of appropriate locations, components or containers. When input is received from the selecting "supported search", the search phrase is submitted to the analysis engine search support, which returns a list, in a drop-down menu, of search phrases or individual containers, for any and all container levels, used by other users or created by the system and known to be historically successful for the described effort and the described searching user, as per the results of the analysis search engine. Input is received from the user selecting a new search phrase or specific container from the drop down menu 30330. When input is received from the user choosing a new search phrase, that phrase is also submitted to the analysis engine 30500 which returns a list of pre-compiled historically proven sites, components or containers associated with that search phrase 30320. Input is received from the user choosing a selection 30420 and the system calls up that specific site, container or component. If input is received from the user selecting a specific site, container or component at any time during the search process, that element is called up by the system 30440.

Input is received from the user choosing to containerize a search or select a container level in which to search 30100. When input is received from the user choosing to containerize the search, the software moves to the container editor as described in Fig. 5, and then returns the user to the search interface screen. Input is received from the user selecting to search a specific container level or the whole network. The system shows the available levels 30200. Input is received from the user selecting a container level 30300, and entering the text or digital component comprising the search string 30310. The system searches the containers 30400 while simultaneously submitting the search string to the analysis engine 30500. While the system is accessing containers, sites or templates 30700, the analysis engine 30500 inquires of the appropriate database 30600 to access historically successful containers, sites or search templates corresponding to the search request 30700, which is then shown on another portion or option of the search interface, either as available containers or sites 30410 or as search template options 30320. On one portion or option of the search interface screen the corresponding containers or sites are listed and/or previewed for selection 30410. Input is received from the user selecting the container to access 30420. The system accesses that container 30430 and shows it on the screen 30440 for user review. Input is received from the user selecting an operation, i.e., preview, read, purchase, move, copy, lease, in any composed

schedule with operations assigned specific values 30460, and the system obtains the specified result 30470. The selection of the operation including any interaction with any uniquely defined container 100 is recorded 30800 by the container gateway (Fig. 2 A, 200), stored in the gateway storage 205 and made available to the analysis engine (Fig. 9) by the data collection and reporting means (Fig. 8). Reporting and collection occurs on a regular basis according to user determined times or rules. The analysis engine compiles and analyzes selections according to various rules-based systems applicable to the particular container area of residence in cyberspace.

Input is received from the user selecting the container or site 30410, proceeding as described above, or selecting a search template 30330, and editing it to re-enter the search 30310. All operations on Fig. 6 utilize the communication device 26 whenever necessary or expeditious.

Referring now to FIG. 7, a flow chart of the search process is shown. Steps in FIG. 7 repeated from FIG. 6 are given the same reference number as in FIG. 6 for convenience and ease of understanding. Fig. 7 commences with "SEARCH TRANSITS GATEWAY 32100", continuing from Fig. 6, "SYSTEM SEARCHES CONTAINERS 30400". The submitted search 32100 transits the gateway 200. The gateway 200 interacts with the container registers 32200. The gateways 200 store the information downloaded from the registers 32300, and the container registers are altered 32500. The container registers 120 then interact with the registers 120 of the encapsulated search, which registers, and the values set within, have been constructed and appended to the search through the search interface 32600. Values are exchanged and compared and operations performed under the rules governing both interacting containers 100, and the rules governing the search container 100 and any gateway 200. The search engine 320, operating under the principles and means of search engines presently existing as described elsewhere, then provides to the search interface 32600 a list of containers 100 meeting the requirements of the search and its appended registers, as well as additional search options 32900. The gateway 200 reports and makes available for collection to the analysis engine 400 the information obtained from the interaction 32400. On a periodic basis defined by the user or a rules-based system, the analysis engine 400 (Fig. 9) stores in databases 900, analyzes and instructs the execution engine 500, and the execution engine 500 executes

changes in the system components as defined below. (Fig. 10). All operations on Fig. 7 utilize the communication device 26 whenever necessary or expeditious.

On the remaining figures, shapes referring to other figures, to operations external to the scope of the present figures, or to the subject of the present drawing, are indicated with dashed lines, and are shown only to place the described operations in the context of continuous and continual operations external to the drawing.

Referring now to FIG. 8, a flow chart of the preferred process for collecting and reporting information on containers is shown. The data reporting 600 and data collection 700 means utilizes subroutines within the analysis engines 400 and gateways 200 to submit and collect register information and sub level analysis to other analysis engines 400 or other gateways 200 of a higher (larger) logical set in a set pattern and frequency defined by the administrator.

Input is received from the user selecting "data reporting" 70100 from the "edit gateway" drop-down menu. Container levels are displayed 70200. Input is received from the user selecting container level 70300. A menu of all possible gateways 70320 and analysis engines 70330 residing on gateways on the above defined container class appears, depicted graphically as a tree of analysis engines and gateways at that container level. Input is received from the user selecting "source" from "source or destination." Input is received from the user selecting a container, containers, or class of container by clicking on the graphically depicted container(s) or container level on a display device. Input is received from the user selecting "destination" from "source or destination" Input is received from the user selecting an analysis engine, analysis engines, or class of analysis engine by clicking on the graphically depicted analysis engine(s) or analysis engine level on a display device. A time scheduler is displayed. Input is received from the user selecting the reporting frequency for the selected gateways to report data to the selected engines. The data from the gateways is thenceforth continuously moved or copied to the analysis engines by the system utilizing the execution engine 500 according to the defined schedule, rules and pattern.

Input is received from the user selecting "choose container level" 70300 from the gateway editor drop-down menu. A menu 70320 appears listing the classes of containers on the system within the defined subset and superset scheme of multiple hierarchically nested containers, i.e.; element, document, file, database, warehouse, domain, appears. Input is

received from the user selecting the class of containers. A graphic representation of that container level throughout the system appears. Input 70300 is received from the user selecting individual containers or all the containers in that class.

From the gateway editor drop-down menu input 70100 is received from the user selecting "data collecting" A menu of all possible gateways and analysis engines residing on gateways on the above defined container class appears, depicted graphically as a tree of analysis engines, and gateways at that container level. Input 70510 is received from the user selecting "source" from "source or destination." Input is received from the user selecting a container, containers, or class of container by clicking on the graphically depicted container(s) or container level. Input 70510 is received from the user selecting "destination" from "source or destination." Input 70510 is received from the user selecting an analysis engine, analysis engines, or class of analysis engine by clicking on the graphically depicted analysis engine(s) or analysis engine level. A time scheduler appears. Input 70510 is received from the user selecting the collecting frequency for the selected engines to collect data from the selected gateways. The data from the gateways is thenceforth continuously moved or copied to the analysis engines by the system 10 utilizing the execution engine 500 according to the defined schedule, rules and pattern.

The data collection 700 means, utilizing the communication device 26 and an execution engine 500, comprises one or more subroutines or agents programmed to travel through the network collecting the accumulated data and analyses from selected analysis engines, gateways or selected subset level of analysis engines or gateways (as above) in a pattern and frequency defined by the gateway administrator at a given container level. Input 70510 is received from the user or administrator, defining the collection and reporting of data, thus controlling permission within his gateway, and being subject to permission levels defined by others beyond his gateway.

Input is received from the user or gateway administrator selecting collection or reporting 70100 and the system shows the container levels available 70200. Input is received from the user selecting a container level 70300. Input is received from the user selecting "gateway" 70400 or "engine" 70500. The system shows gateways 70320 or engines 70330 associated with that level. Input is received from the user editing the reporting parameters associated with a gateway or a class of gateways 70410 or an engine or class of engines 70510.

Input is received from the user selecting the collecting frequency for the chosen engines. When input is received from the user choosing to user save the definition, the screen returns to the main container screen, step 70100 to make another selection available. Input is received from the user choosing to repeat the cycle, choosing "destination" to describe the destination analysis engines and the data collecting frequency from those destination analysis engines. 5 The data collection means 700 collects the accumulated gateway information in a pattern and frequency defined by the gateway administrator or user at a given container level.

The system utilizing the execution engine (see Fig. 10) distributes the new parameters to the gateways 70420 or engines 70520 by the communication device 26. Using the new 10 parameters the gateways report to the analysis engines 70430 after, in some cases, conducting sub-analysis 70440, or using sub-analysis 70440 to submit directly to specified gateways under certain conditions and parameters, and the analysis engines collect from the gateways 70530. The analysis engine uploads, downloads and utilizes information to databases 900 to conducts its analysis.

15 The invention includes an analysis engine 400. Through the data reporting 600 means and data collection 700 the analysis engine 400 receives data and sub-analysis from the search interface and the gateways. Data includes, for each gateway 200, the frequency and grade of access, the description of the user accessing, the identity of the container 100 accessing, the register parameters, and the historically accumulated register data.

20 Referring now to FIG. 9, a flow chart of the operation of the analysis engine 400 is shown. Analysis engines 400 may reside at any gateway or anywhere in the system 10. The analysis engine 400, operating under its own programmed sequence, utilizing the communication device 26, works, by means of programmed rules of logical, mathematical, statistical or other analysis upon gateway and register information, in continuous interaction 25 with the search process 410 and the data collection and reporting process 420 to analyze, determine and compile instructions 40100 on container construction 40110 to containerize in an automated process 40115, on container contents 40120 to move, copy or delete containers 40125, on storage schemes 40130 to move or copy containers to new storage 40135, on access routes 40140 to alter gateway pointers to sought information 40145, on search templates 30 40150 to add, delete or change search phrases and the referenced objects indicated by those



search phrases 40155 and on gateway instructions 40160 to alter gateway registers and pointers 40165.

Thus, analyses might include, but are not limited to, the physical locus of the users accessing, the demographic classification of the users accessing, the access frequency for a given container, the range or curve of time relevance affecting a container, the range or region of space relevance affecting a container 100, the number or number of a specific type of container 100 transiting a gateway 200, the hierarchically graded usage of containers 100 or container contents 01 compared with the demographic of those users accessing the container, the hierarchically graded usage of containers 100 or container contents 01 compared with search phrases entered into the search interface 300, the hierarchically graded usage of containers 100 or container contents 01 compared with search phrases entered into the search interface 300 compared with the demographic of the users accessing, the number of pertinent containers nested within a given container 100. Once an analysis is accomplished, the result is compared to pre-programmed rules triggering instruction sets (such as moving a container to nest within another container).

Instructions are then sent to the execution engine 40200, which utilizes the communication device 26 to execute the instructions derived from the analyses. These containerized instructions transit the gateways 40300 and are utilized in the gateway process (Fig. 12)

Referring now to FIG. 10, a flow chart of the operation of the execution engine is shown. The execution engine 400, operating under its own programmed sequence in response to the instructions from the analysis engine 50100, utilizing the communication device 26, works in continuous process as its containerized execution instructions transit the gateways 50200 to create containers 50210 in an automated containerization process 50215, alter container contents 50230 by moving or copying containers to new containers 50235, to alter storage 50240 by moving or copying containers to new storage 50245, to alter access routes 50250 by altering gateway pointers 50255, to alter search templates 50260 by adding, changing and deleting search phrases and the referenced objects indicated by those search phrases 50265, to alter gateway instructions 50270 by altering gateway registers and pointers 50275. The execution works in a continuous loop with the gateway process 50300, the data collection and reporting process 50400 and the analysis engine process 50300.

The invention includes gateways 200. Gateways may be placed and reside anywhere on the network where containers transit. Gateways also reside on any or all containers. The gateway reads and stores the chosen register information from transient containers entering or exiting its logical boundaries. The resident analysis search engine, if any, performs the specified level of analysis. Data and analysis is both held for the collection means according to the pattern and timing specified in the data reporting 600 editor and submitted according to the pattern and timing specified in the data collection means editor 700.

The gateways are network-wide, hierarchical, and nestable, and reside with a container encompassing any component, digital code, file, search string, set, database, network, event or process and maintaining a unique lifelong network wide identity and unique in all the universe historical identity, or may be strategically placed at such container transit points to gather and store register information attached to any such container, according to system-defined, system-generated, or user determined rules residing in its registers defining the behavior of those containers and components as they exit and enter one another, or interact with one another or any system process or system component within the logical domain of that container, or after exiting and entering that container, or defining how they interact with that unique gateway.

Gateway's registers comprise both system-defined and user-defined registers, alterable by author, duration, location, network-wide history, individual container history and/or interaction with other containers, gateways, networks or media, and evolve according to that gateway's history on a computer network, or according to the network history of events and processes, or according to that information component's interaction with other information containers, components, system components, network events or processes.

Referring now to FIG. 11, a flow chart of the gateway editor is shown. From the main title bar input is received from the user selecting "containerize" or "gateway level" 20100. When input is received from the user selecting "containerize" the system enters the container editor process 110. When input is received from the user selecting "gateway," the system shows the gateway levels available 20200. A menu of all possible gateways within the subset and superset scheme of defined multiple hierarchically nested gateways appears. Input is received from the user selecting the gateway level 20300. The system searches the gateways 20500 to locate the available gateway templates 20700 and the available gateways 20600. Input is received from the user selecting the gateway 20610 or gateway level template 20720.

The system goes to the gateway 20620 or to the template 20720. A graphic representation of the chosen gateway 20630 or template 20730 appears. Input is received from the user to edit 20640 or create a gateway 20740. Once completed, input may be received from the user selecting "analysis level" from the gateway 200 drop-down menu, to select the level of analysis in a multi-level analysis sequence to be accomplished at the local level by a gateway-resident analysis engine. The user accesses the container editor to containerize (Fig. 5). Input is received from the user selecting the registers by clicking on the graphically depicted register, or from a drop down menu. ). Input is received from the user setting the registers as described elsewhere in ("container registers"). Input is received from the user selecting or defining the rules governing the interaction of that gateway with transient containers. Input is received from the user selecting or defining the rules governing the interaction of containers existing within the logical domain of the container 100 to which that gateway is attached. The user publishes the gateway (Fig. 5). Input is received from the user selecting "residence" from the main menu bar. ). Input is received from the user choosing to leave the gateway where it was created, move it to container on another drive, directory, computer, or network. If the user chooses "move," a browse function allows the user to name the new location or browse a list of possible locations. Once input is received from the user choosing the residence of the gateway, the program restores the search interface screen.

The invention includes a data reporting means editor 610, and a data collection means editor 710, Fig. 2 A, as a menu option under the gateway editor 210.

The present invention also includes a gateway process.

Referring now to FIG. 12, a flow chart of the gateway process is shown. A system operation, search process or element container or process container is shown in transit 21100 passing through a gateway 21200. The container, operation or process interacts with the gateway 21300, uploading, downloading and exchanging information with the container, operation or process. The gateway stores container information 21400 and the container registers are altered 21500. The container registers also interact with the search interface 21600. The gateways report the register information or make it available for collection by the data reporting and collection means (Fig. 8) operating on the communication device 26 to provide the information to the analysis engine 21800, which stores 90100, analyzes and

instructs the execution engine 21900, which processes and instructions are also stored 90100 by the execution engine upon receipt.

All operations in Fig. 12 utilize the communication device 26 whenever necessary or expeditious.

5 Referring now to FIG. 13 A, a drawing of nested containers 100 prior to the container modification process on a network 201 is shown. (Note: The same container numbering scheme is used in Fig. 13 A, 13 B, 13 C, 13 D and in 2 B.) Information containers 505 and 909, residing within container 908, operating under the rules governing container interaction within that container 908 downloaded to container 505 and 909 from gateway 9081  
10 upon their entrance to container 908, which rules had been downloaded from execution engine 500 acting under the direction of analysis engine 400, and under the rules programmed into their own registers 404120, 909120, compare the specified (by those rules) set of registers 404120, 909120, i.e., time and space, and determine a container 404 encapsulated within 505 would be more appropriately encapsulated within container 909.

15 Referring now to FIG. 13 B a drawing of nested containers during a container modification process on a network 201 is shown. Container 404 is moved to reside with container 909. As the container 404 exits container 505, the gateway of container 505, being gateway 5051, operating under the rules governing container interaction with a gateway 5051 upon egress or egress as programmed in the gateway editor 210 and modified by the execution  
20 engine 500 executing the instructions of the analysis engine 400, or any greater logical analysis engine 408 providing execution instructions to an execution engine 508 operating in a larger encompassing container 108 entering through that container's gateway 208 or an independent gateway 707, or sub-analysis engine operating at any gateway level, records the register information of container 404. The gateway 5051 reports the transaction to the  
25 gateway 9081 of container 908, being the next higher logical container. Gateway 9081 holds in gateway storage 205 the information until collected by one or more data collection processes 700, or reported to one or more data reporting processes 600, serving one or more analysis engines 400 residing independently on the system 10 or an analysis engine at higher logical container 303. The analysis engine 400, comparing reports of user hierarchically graded usage  
30 under the operations of the search engine 320 and the search interface 300, on information container 808 after receiving reports from the data reporting means of container 404 being

moved to container 909 determines, i.e., that the number of time and space relevant containers residing within container 909 is sufficient to warrant an action, and directs the execution engine 500 to copy container 909, nested within container 908, to a third information container 808. As the copy instruction from execution engine 500 transits the gateway of container 908, the gateway 9081 records the instruction. The copy instruction interacts with the registers 909120 of container 909 regarding the rules governing its copying to another location. Once approved by the governing rules of registers 909120 appended to container 909, container 909 is duplicated. As the duplicate container 909 exits the container 908, the gateway records the register information 909120 of container 909, and the registers 909120 of container 909 are altered by special instructions from gateway 9081 under the rules residing in gateway 9081 regarding ingress and egress and the rules residing in the registers 909120 of container 909 regarding alteration by gateways upon ingress and egress. Passing through independent gateway 707, the register information 909120 is recorded, and awaits data collection or reporting 700, 600. As container 909 enters container 808, the gateway records the register information 909120 of container 909, the registers 909120 of 909 are altered by special instructions from gateway 8081, operating under the rules as described in the paragraph above, and container 909 takes up residence within container 808.

Referring now to FIG. 13 C, a drawing of nested containers after the container modification process on a network 201 process is shown. Container 909, now also logically residing within container 808, commences to interact with other containers 606 in 808 under the rules governing container interaction within container 808 as received from gateway 8081 upon transiting that gateway, and under the rules of registers 606120, 909120 of the interacting containers 606, 909, operating under the rules as described in the paragraph above. Through data collection and reporting 700, 600, analysis engine is appraised of container's 909 new duplicate residence. I.e., operating under the registers of space relevance, a body of law pertaining to Boston Municipal tax law may be housed in a container holding Massachusetts tax law, but it would be more appropriately located in a container holding Boston tax law, with only a pointer to that location residing in the Massachusetts tax law container. In this example, such an analysis could be accomplished by comparison of zip code information in the space registers, or logical rules-based analysis, with "state" being a larger set than "city". Or, i.e., operating under the registers of time relevance, the curve of time relevance for a

concert might follow an ascending curve for the months prior, hit a brief plateau, and then reach a precipitous decline, at which time certain pertinent information only might be moved to an archival container of city events or rock concerts of that year. In this example, once the curve is mapped into a register, that map would cause an increasing frequency of pointers to that container in other containers or gateways, or inclusion of that container in other containers, as the analysis engine compares that curve with increasing user inquiry.

Referring now to Fig. 13 D, a flowchart of the reconstruction process is shown.

Information containers 505 and 909, residing within container 908, operating under the rules governing container interaction within that container 908 downloaded 888103 to container 505 and 909 from gateway 9081 upon their entrance to container 908, which rules had been downloaded 888102 from execution engine 500 acting under the direction 888101 of analysis engine 400, and under the rules programmed into their own registers 404120, 909120, compare 888104 the specified (by those rules) set of registers 404120, 909120, i.e., time and space, and determine 888105 a container 404 encapsulated within 505 would be more appropriately encapsulated within container 909.

Container 404 is moved 888106 to reside with container 909. As the container 404 exits container 505, the gateway of container 505, being gateway 5051, operating under the rules governing container interaction with a gateway 5051 upon egress or egress as programmed in the gateway editor 210 and modified 888108 by the execution engine 500 executing the instructions of the analysis engine 400, or any greater logical analysis engine 408 providing execution instructions 888107 to an execution engine 508 operating in a larger encompassing container 108 entering through that container's gateway 208 or an independent gateway 707, or sub-analysis engine operating at any gateway level, records 888109 the register information of container 404, and alters the register information of container 404. The gateway 5051 reports 888110 the transaction to the gateway 9081 of container 908, being the next higher logical container. Gateway 9081 holds 888111 in gateway storage 205 the information until collected by one or more data collection processes 700, or reported to one or more data reporting processes 600, serving 888112 one or more analysis engines 400 residing independently on the system 10 or an analysis engine at higher logical container 303. The analysis engine 400, comparing 888114 reports of user hierarchically graded usage on information container 808 under the operations of the search engine 320 and the search

interface 300, after receiving 888113 reports from the data reporting means of container 404 being moved to container 909, determines 888115, i.e., that the number of time and space relevant containers residing within container 909 is sufficient to warrant an action, and directs 888115 the execution engine 500 to copy container 909, nested within container 908, to a  
5 third information container 808. As the copy instruction from execution engine 500 transits the gateway of container 908, the gateway 9081 records 888116 the instruction. The copy instruction interacts 888117 with the registers 909120 of container 909 regarding the rules governing its copying to another location. Once approved 888118 by the governing rules of registers 909120 appended to container 909, container 909 is duplicated 888118. As the  
10 duplicate container 909 exits the container 908, the gateway records 888119 the register information 909120 of container 909, and the registers 909120 of container 909 are altered 888120 by special instructions from gateway 9081 under the rules residing in gateway 9081 regarding ingress and egress and the rules residing in the registers 909120 of container 909 regarding alteration by gateways upon ingress and egress. Passing through independent  
15 gateway 707, the register information 909120 is recorded 888121, and awaits 888122 data collection or reporting 700, 600. As container 909 enters container 808, the gateway records 888123 the register information 909120 of container 909, the registers 909120 of 909 are altered 888124 by special instructions from gateway 8081, operating under the rules as described in the paragraph above, and container 909 takes up residence 888125 within  
20 container 808.

Container 909, now also logically residing (in addition to its original container residence) within container 808, commences to interact 888126 with other containers 606 in 808 under the rules governing container interaction within container 808 as received from gateway 8081 upon transiting that gateway, and under the rules of registers 606120, 909120 of  
25 the interacting containers 606, 909, operating under the rules as described in the paragraph above. Through data collection and reporting 700, 600, analysis engine is appraised 888127 of container's 909 new duplicate residence.

Referring now to Fig. 14, the screen interface of the container editor is shown. This interface is a process wherein input is received by the user using the main menu 78 or drop  
30 down menu 1419, or using an input device to "drag and drop" or click, causing the system 10 to acquire 1409, edit 1410 or create 1411 a file 1407, container 1408 or digital content 01, to

search for 1412, acquire 1413, edit 1414 or create 1415, print 1416, or containerize 1417 a container 100, to select 1402, (or by clicking on register), search 1403, acquire 1404, edit 1405, or create a register 1406 to append or detach registers 120 to those containers, to set register values in those registers 120, to utilize the register editor 125 through 1405 to create  
5 new registers, or to 1418 add, detach, acquire a gateway 200 to append or detach to those containers, and utilize the gateway editor 210 through 1418. (See detailed description referring to Fig. 5)

Referring now to Fig. 15, the screen interface of the gateway editor is shown. This interface is a process wherein input is received by the user using the main menu 1501 or drop  
10 down menu 1513, or using an input device to “drag and drop” or click, causing the system 10 to search for 1507, acquire 1508, edit 1509 create 1510, print 1511 or containerize 1512 gateways, and causing the system 10 to establish rules by which an individual gateway governs the transiting 1502, entering 1503, exiting 1504 of containers and the interaction of containers within its domain 1505, and external of its domain.1506. (See detailed description  
15 referring to Fig. 11).

Referring now to Fig. 16, the screen interface of the search interface. This interface is a process wherein input is received by the user using the main menu 1625 or drop down menu  
1624, or using an input device to “drag and drop” or click, or by entering text, causing the system 10 to select 1615, search for 1616, acquire 1617, edit 1618 create 1619, print 1620,  
20 containerize 1621 (by accessing the container editor 110) or insert 1622 digital search strings into the search box 1623 in order to submit that string to the search engine 320, or causing the system 10 to select 1602, search for 1603, acquire 1604, edit 1605, create 1612, containerize 1613 (by accessing the container editor 110), or insert 1614 search keys (templates that comprise search scope in geographic range, container level, and specific key words or digital  
25 strings), or containerized searches (containers 110), into the search box 1623 in order to submit that string to the search engine 320 , or causing the system 10 to set a search range by geographic range 1607, container level 1608, or acquire 1609, edit 1610 or create 1611 a scope template. (templates that comprise search scope in geographic range and, container level.) (See detailed description referring to Fig. 6).

30 Referring now to Fig. 17, a drawing showing, on an input device or computer screen 24, in any generic (dashed lines) software application program, a drop-down menu link 1403



on a drop down menu 1402 dropping down from a main menu 1401, and a free-floating button link 1404, is shown. When input is received at 1402 or 1403, the system 10 makes available to the user the containerization process or container editor 110. When input is received at drop-down menu link 1405 or a button link 1406, the system 10 makes available to the user the means to enter and interact with this system 10 or this network 201 in any of their aspects. The interfaces 1403, 1404 show a process wherein input is received causing the system 10 to encapsulate content or access the container editor 110. The link also allows the user to encapsulate the page or file on which he is currently working, without selecting content, and if so desired, without accessing the container editor. The interfaces 1405, 1406 show a process wherein input is received causing the system 10 to access or interact with the system 10 or the network 201.

The present invention also includes a search engine 320. Once the key word(s), phrase or digital segment is entered into the search interface 300, or an offered selection chosen on the menu, it is utilized by the search engine 320 to locate the desired site or data.

The search engine employed may be any industry standard search engine such as Verity "Topic", or Personal Library Software, as used in Dow Jones News Retrieval, or Internet search engines such as Webcrawler, Yahoo, Excite, Infoseek, Alexa or any Internet search engine, or any new engines to be developed capable of searching for and locating digital segments, whether text, audio, video or graphic.

The present invention also includes an analysis engine 400. Utilizing rules-based analysis, the analysis engine determines the class of storage medium upon which containers reside, the subsets and supersets by which and in which containers encompass and reside within one another, the routes of access to those containers, the historically successful search parameters by which those containers are accessed based upon the identity of the user accessing the containers, and the grade of access chosen by the user in accessing that container 100.

Utilizing a pre-programmed sequence of compilation, and inductive, deductive and derivative analysis, the analysis engine manufactures instructions based upon the analysis of the information submitted by the gateways and the search interface, and submits those instructions to the appropriate execution engine 500 in order to create new information containers, content assemblages, storage schemes, access routes, search templates, and

gateway instructions, and others, and to provide informed search options through the search interface to the inquiring user.

The present invention also includes an engine editor 510, that provides a system administrator with a means of editing the operating principles of that search engine, and search  
5 template loading in the search interface 300, a reporting and collection means editor 610, 710, governing data reporting 600 and data collection 700 at the gateways 200 as defined by the gateway editor 210 and the register editor 125, a container editor 110 for creating and modifying containers and appending registers to containers, a register editor 125 for creating and modifying container registers and establishing and adjusting the values therein, container  
10 gateways 200 with their own storage 205, information containers 100 for holding information and container registers for holding information about specific containers and their history on the network.

The present invention also includes an execution engine 300. Based upon instructions received from the analysis engine 400 utilizing the communication device 26, the execution  
15 engine 500 provides search phrases to the search interface 300 based upon initially received inquiries, relocates containers including their programs, data and registers to other directories, drives, computers, networks on other classes of storage mediums, i.e., tape drive, optical drive, CD-ROM, deletes, copies, moves containers to nest within or encompass other containers on other directories, drives, computers, networks to nest within other containers, alters the class  
20 of storage medium upon which containers reside, the subsets and supersets by which and in which containers encompass and reside within one another, the routes of access to those containers, and the historically successful search parameters by which those containers are accessed based upon the identity of the user accessing the container and the grade of access chosen by the user in accessing that container.

25 The execution engine 400 fulfills the instructions of the analysis search engine 500, to create new information containers, content sub and superset assemblages, storage schemes, access routes, search templates, gateway 200 instructions and other system functions. The execution engine includes an editor 510 that provides a system manager with a means of editing the operating principles of that search engine, governing data reporting, data collection  
30 700, search template loading, gateway instructions, and other functions.

The present invention also includes flat or relational databases 900, used where, and as required.

The present invention also includes a communication device 26 supporting all operations on a network wide basis.

5 The present invention also includes a search engine 300 to locate the desired site or data. The present invention also includes databases 900, flat or relational, to serve the other components of the system as needed and where needed.

The present invention also includes editors, by which the user may alter the governing aspects of the system. Editors include, but are not limited to, a container editor 110, a register  
10 editor 125, a gateway editor 210, an engine editor 510, a reporting means editor 610, a search interface 300, and a collection means editor 710.

The present invention also includes specific screen interfaces for the editors, as described in Fig. 14, Fig. 15. and Fig. 16.

The present invention also includes a means for this system 10 and network 201 or  
15 container editor 110 to be accessed from a menu or button selection within any program, as described in Fig. 17.

While the present invention has been described with reference to certain preferred embodiments, those skilled in the art will recognize that various modifications may be provided. For example, both analysis engine and execution engine may be duplicated or  
20 modified for distribution at various locations and hierarchical positions in the gateway and container system throughout the network and designed to work in concert. Also, the physical computing infrastructure may be mainframe, mini, client server or other with various network and distributed computing designs, including digitally supported or based physical or public media, and the components of the system 10, as described in Fig. 1 may be physically  
25 distributed through space. Even the contents of a single container may be logically referenced but be physically distributed through the network and reside at multiple storage locations. The whole system may be hierarchically nested within other systems to the nth degree. Whole systems may also be encapsulated within containers. A single container may also encompass a single physical media, such as a CD-ROM disk, programmed with the container, gateway and  
30 register design. Gateways may be strategically placed on containers at ingress and/or egress points or may be placed strategically throughout the system for optimal collection and

reporting output and gateway system control. Also, the loop of gateway data collection and reporting, analysis engine analysis, instruction, and gateway modification, and execution engine operations may be infinitely nested, from the smallest container of two sub-containers to whole networks holding millions of containers and thousands of levels, with analysis itself  
5 nested within the multiple levels. Gateways may be established at both logical and physical junctures such as a satellite uplink point. Also, the provision to establish a unique network identity might be designed to include as of yet unknown computer networks as they arise. The analysis and execution engines may operate on a rules-based, fuzzy logic, artificial intelligence, neural net, or other system not yet devised. Other variations upon and  
10 modifications to the preferred embodiments are provided for by the present invention, which is limited only by the following claims. Also, the classification scheme of nested containers, while designated by the container creators, may transform, be utilized otherwise, or be wholly discarded according to usage. Also, hardware configurations, such as the use of RAM or hard drives for storage or lasers for communication may assume myriad forms without altering the  
15 essential operation of this invention.

**WHAT IS CLAIMED IS:**

1. An apparatus for transmitting, receiving and manipulating information on a computer system, the apparatus including a plurality of containers, each container being a logically defined data enclosure and comprising:

5 an information element;

a plurality of registers, the plurality of register forming part of the container, a first register of the plurality of registers for storing a unique container identification value, a second register of the plurality of registers that stores information and evolves according to the relationship, use and interaction of the container with other containers, processes and systems;

10 and

a gateway attached to and forming part of the container, the gateway controlling the interaction of the container with other containers, systems and process.

2. The apparatus of claim 1, wherein the information element is one from the group of text, graphic images, video, audio, a digital pattern, a process, a nested container, bit,  
15 natural number and a system.

3. The system of claim 1, wherein the plurality of registers include at least one container history register for storing information regarding past interaction of the container with other container, system or processes, the container history register being modified.

4. The system of claim 1, wherein the plurality of registers include at least one  
20 system history register for storing information regarding past interaction of the container with different operating system and network processes.

5. The system of claim 1, wherein the plurality of registers include at least one predefined register, the predefined register being a register associated with an editor for user selection, the predefined register appendable to any container.

6. The system of claim 1, wherein the plurality of registers include a user-created register, the user-created register generated by the user, one or more of which is appendable to any container.

7. The system of claim 1, wherein the plurality of registers include a system-  
5 defined register, the system-defined register set, controlled and used by the system, one or more of which is appendable to any container.

8. The system of claim 1, wherein the plurality of registers include at least one register for controlling the relationship of the container with other containers, systems and processes using time as a parameter.

10 9. The system of claim 8, wherein the plurality of registers include:  
an active time register for identifying times at which the container will act upon other containers, processes, systems or gateways;  
an passive time register for identifying times at which the container can be acted upon other containers, processes systems, or gateways; and  
15 a neutral time register for identifying times at which the container may interact with other containers.

10. The system of claim 1, wherein the plurality of registers include at least one acquire register for controlling whether the container adds a register or a container from other containers when interacting with them.

20 11. The system of claim 1, wherein the plurality of registers include at least one register for controlling the relationship of the container with other containers using space as a parameter.

12. The system of claim 11, wherein space refers to the geographic location of a the container.

13. The system of claim 11, wherein space refers to the logical address space of a network in which a container resides.

14. The system of claim 11, wherein the plurality of registers include:  
an active space register for identifying space in which the container will act upon other  
5 containers, processes, systems or gateways;  
an passive space register for identifying from which the container can be acted upon  
other containers, processes systems, or gateways; and  
a neutral time register for identifying space in which the container may interact with  
other containers.

10 15. The system of claim 1, wherein the gateway includes means for acting upon another container, the means for acting upon another container using the plurality of register to determine whether and how the container acts upon other containers.

16. The system of claim 1, wherein the gateway includes means for allowing  
interaction, the means for allowing interaction using the plurality of registers to determine  
15 whether and how another container can act upon the container.

17. The system of claim 1, wherein the gateway includes means for gathering information, the means for gathering information recording register information from other containers, systems and processes that interact with the container.

18. The system of claim 1, wherein the gateway includes means for reporting  
20 information, the means for reporting information providing register information to other containers, systems and processes that interact with the container.

19. The system of claim 1, wherein the gateway includes an expert system including rules defining the interaction of the container with other containers, systems and processes.

20. A method for creating an interactive information container, the method including the steps of:

- forming a container;
- selecting an interactive register for the container;
- 5 identifying an item for inclusion in a container; and
- creating a container element that includes the identified item.

21. The method of claim 20, wherein the step of forming a container further comprising the steps of:

- displaying a plurality of container levels;
- 10 receiving input from a user selecting one of the displayed container level; and
- displaying a container template corresponding to the container level input.

22. The method of claim 20, wherein the step of selecting an interactive register further comprising the steps of:

- displaying a list of available registers;
- 15 receiving input selecting an available register from the list of available registers;
- receiving input values for the selected available register; and
- appending the register to the container.

23. The method of claim 20, wherein the step of creating a container element that includes the identified item further comprising the steps of:

- 20 providing a data structure as part of the container element;
- storing the identified element in the data structure; and
- associating the container element with the container.

24. The method of claim 20, wherein the step of forming a container includes the step of providing for the container a gateway that uses the interactive register to control the  
25 interaction of the container with other containers, processes, and systems.



25. The method of claim 24, wherein the step of providing a gateway further comprising the steps of:

determining a current gateway for a system upon which the container is being created;  
replicating the current gateway to create a new gateway; and  
5 associating the new gateway with the container.

26. The method of claim 24, wherein the step of providing a gateway further comprising the steps of:

determining a register for a system upon which the container is being created;  
replicating the determined register to create a new register; and  
10 associating the new register with the container.

27. The method of claim 20, wherein the step of selecting an interactive register further comprising the steps of:

retrieving a list of available registers;  
selecting an available register from the list of available registers by the system;  
15 receiving input values for the selected available register from the system; and  
appending the register to the container.

28. The method of claim 20, wherein the step of creating a container element that includes the identified item is performed by a system interacting with the container, and further comprising the steps of:

20 providing a data structure as part of the container element;  
storing the identified element in the data structure; and  
associating the container element with the container.

29. A method for interacting between a first interactive information container and a second interactive information container, the method including the steps of:

25 determining identification information for the first container using a first gateway;  
determining identification information for the second container using a second gateway;

determining whether the first container can act upon the second container using the first gateway and a register of the first container;

determining whether the second container can be acted upon by the first container using the second gateway and a register of the second container; and

5 performing the interaction between the first and second containers prescribed by the first gateway and the register of the first container if both the first container can act upon the second container and the second container can be acted upon by the first container.

30. The method for interacting of claim 29, wherein the steps of determining identification information are performed by reading respective identification registers of the  
10 first and second containers.

31. The method for interacting of claim 29, further comprising the step of altering a register of the first container and a register of the second container to reflect the interaction between the first container and the second container.

32. The method for interacting of claim 29, further comprising the step of adding  
15 registers to the first container based on the registers in the second container and the second gateway.

33. The method for interacting of claim 29, wherein the step of performing also uses the second gateway and the register of the second container to determine the prescribe action to be taken.

20 34. The method for interacting of claim 29, further comprising the steps of:  
determining whether the first container should add an identified register of the second container as a new register of the first container using an acquire register and the first gateway of the first container; and

25 adding the new register to the first container if it is determined that the new register should be added to the first container.

35. The method for interacting of claim 29, further comprising the step of modifying the first gateway of the first container based on the interaction between the first container and the second container.

36. The method for interacting of claim 35, wherein the step of modifying includes  
5 modifying rules of an expert system that forms the first gateway of the first container.

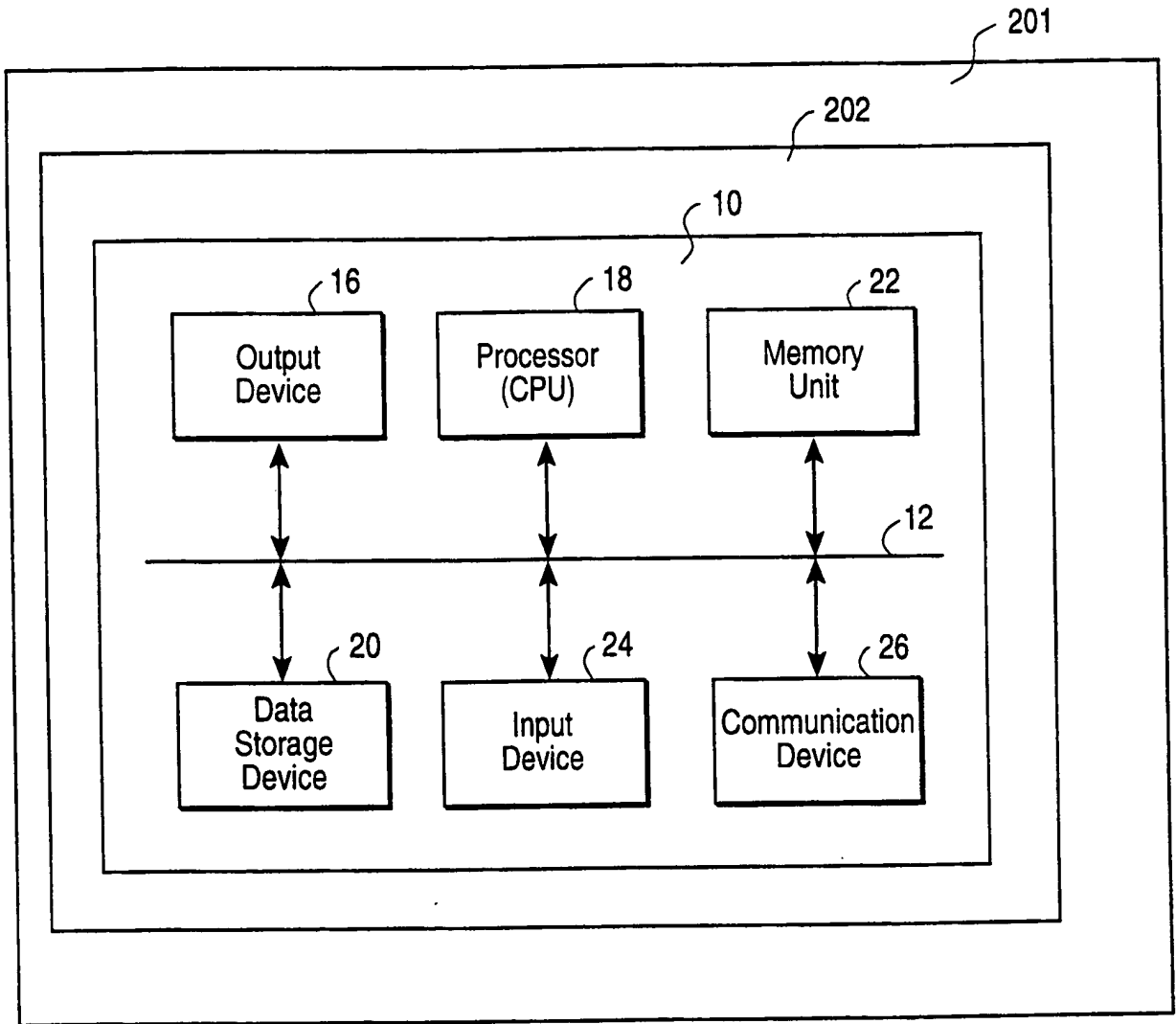


FIG. 1

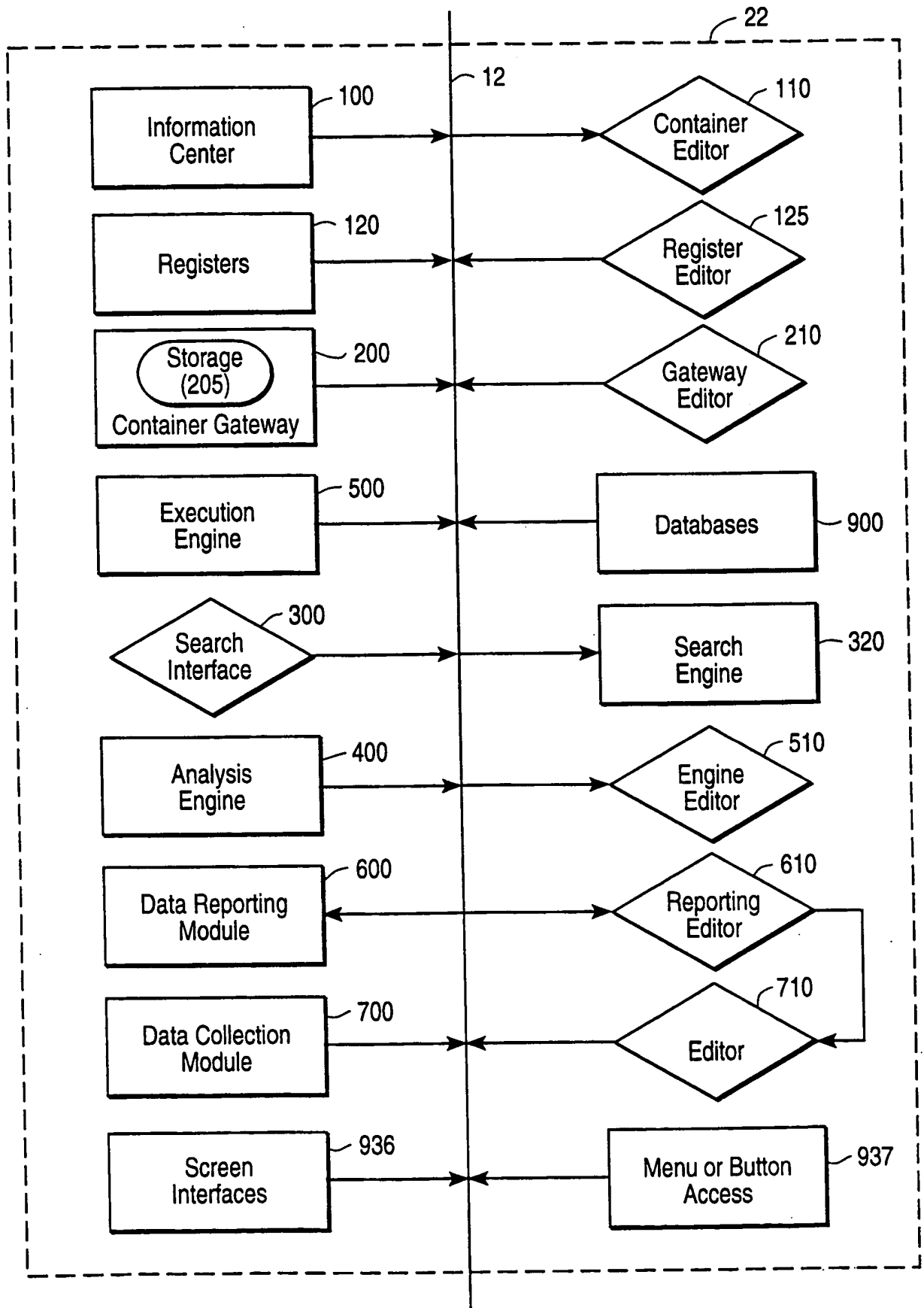


FIG. 2A

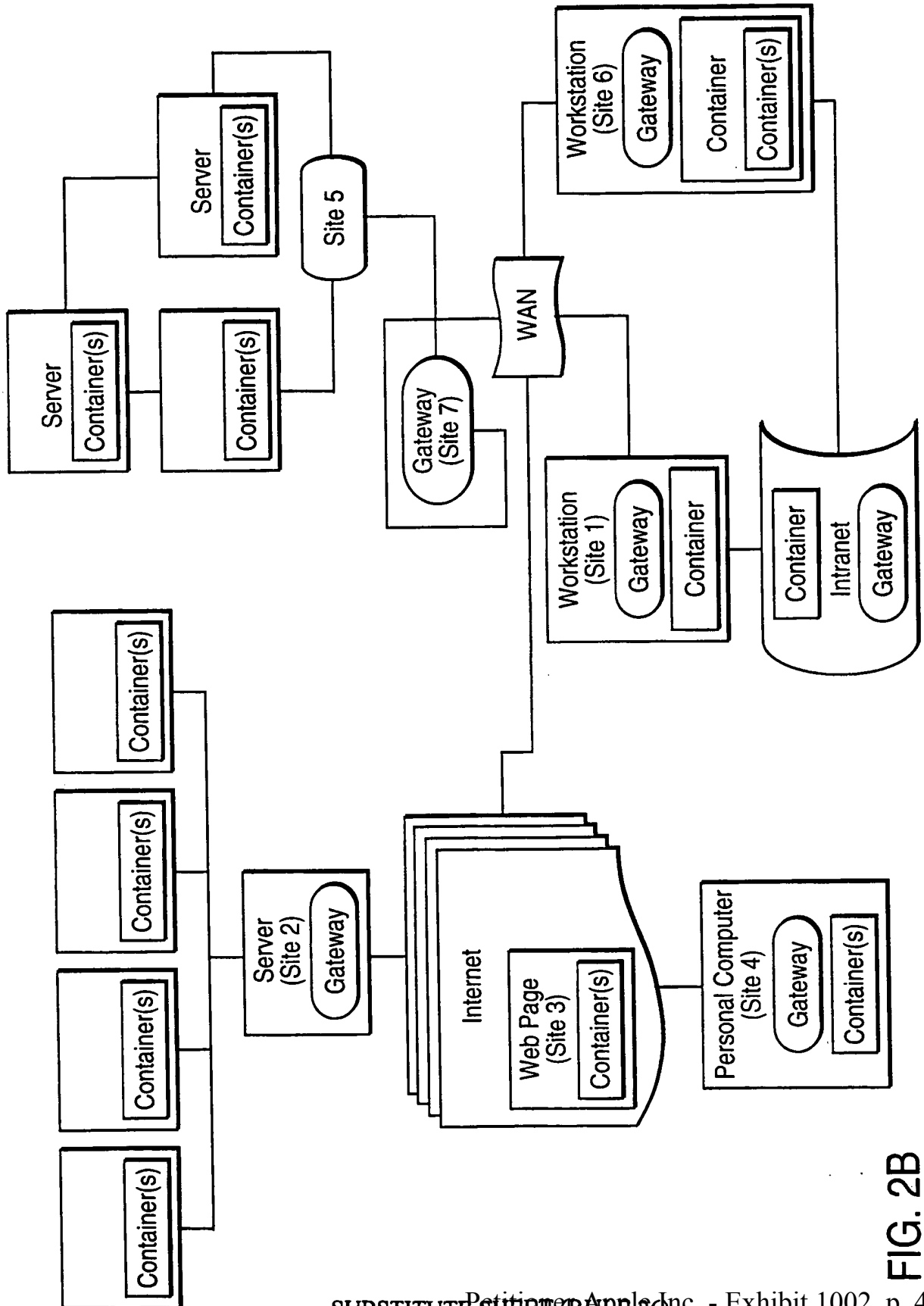


FIG. 2B

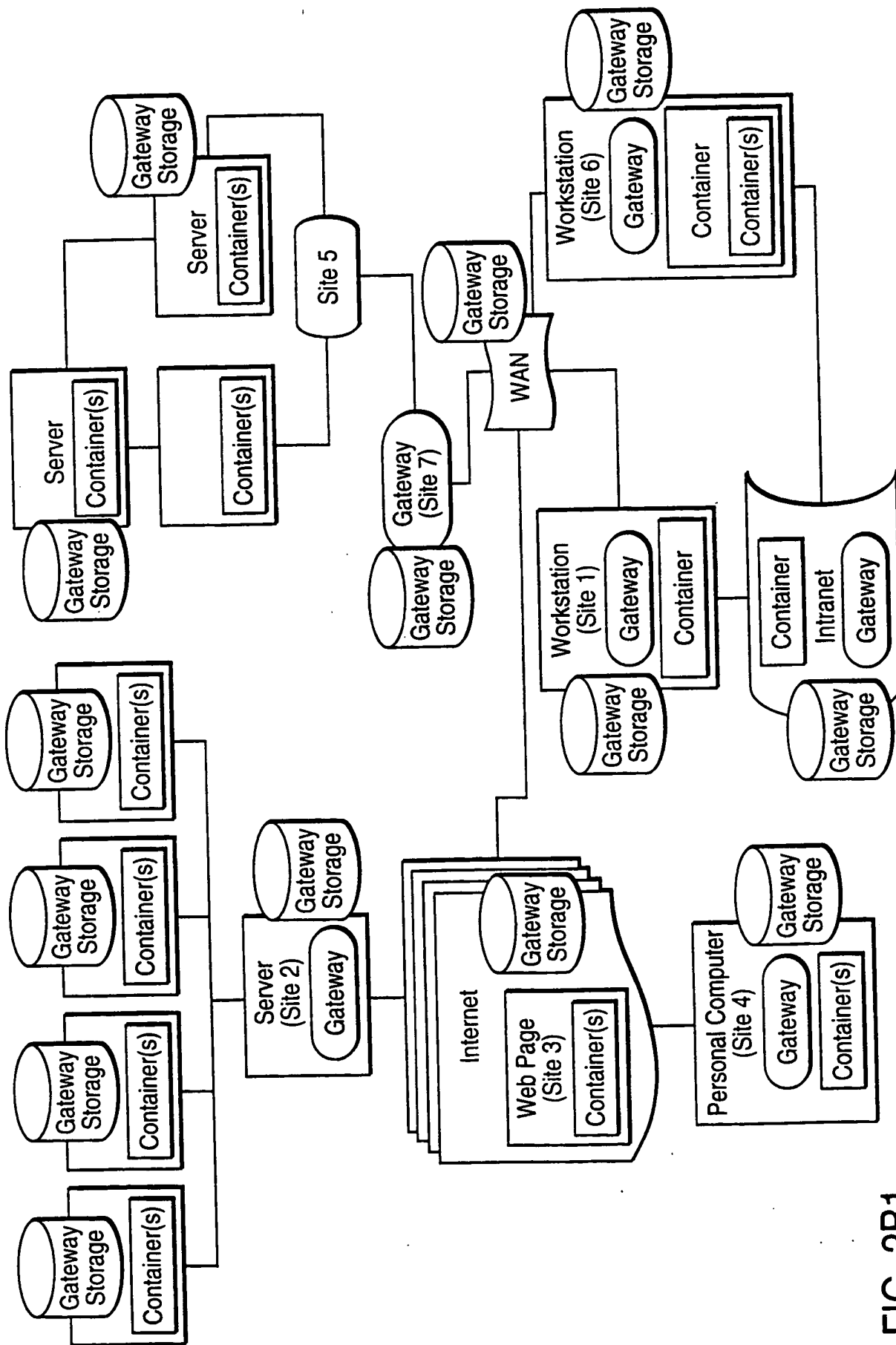


FIG. 2B1

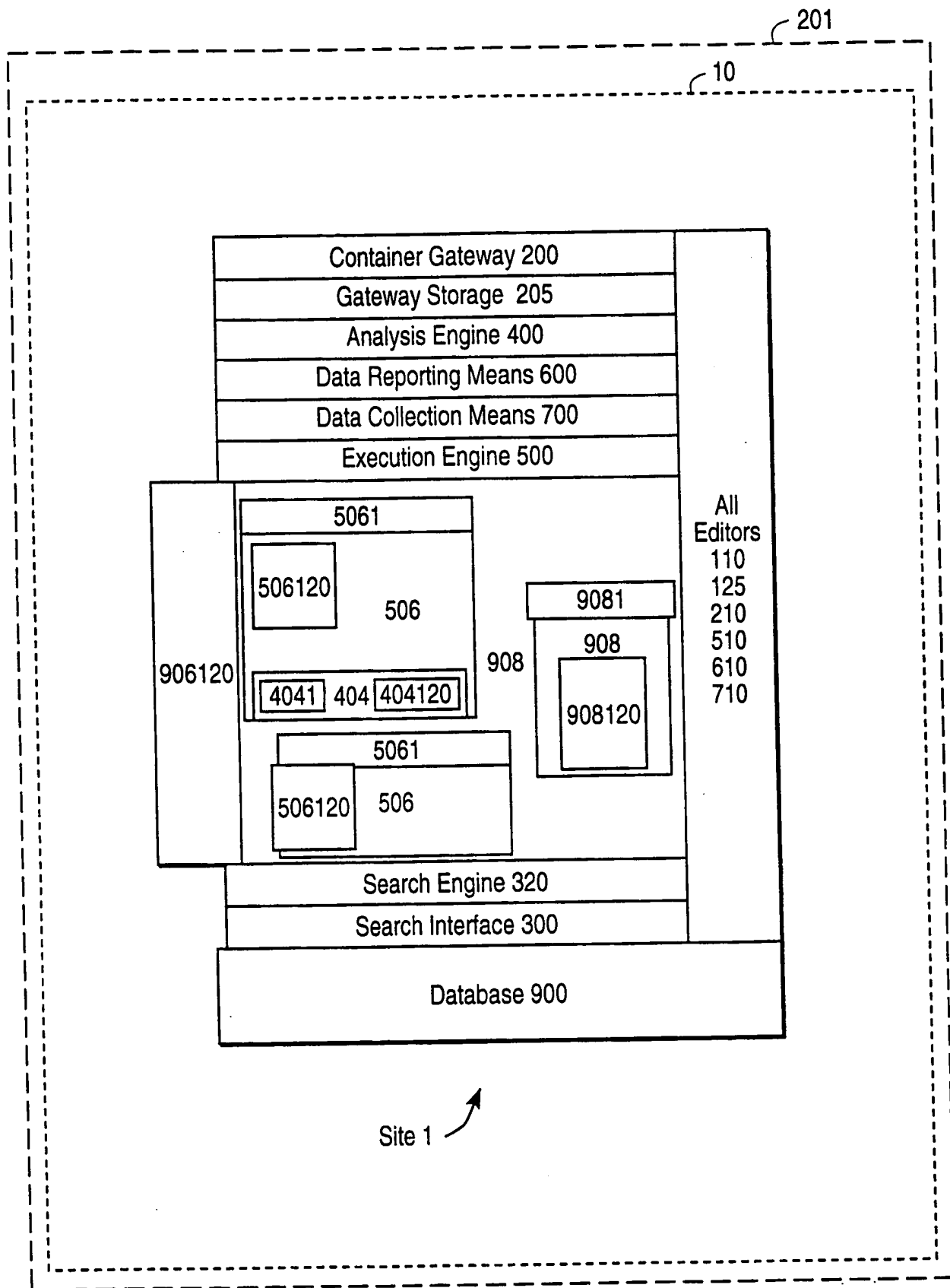


FIG. 2C



6/30

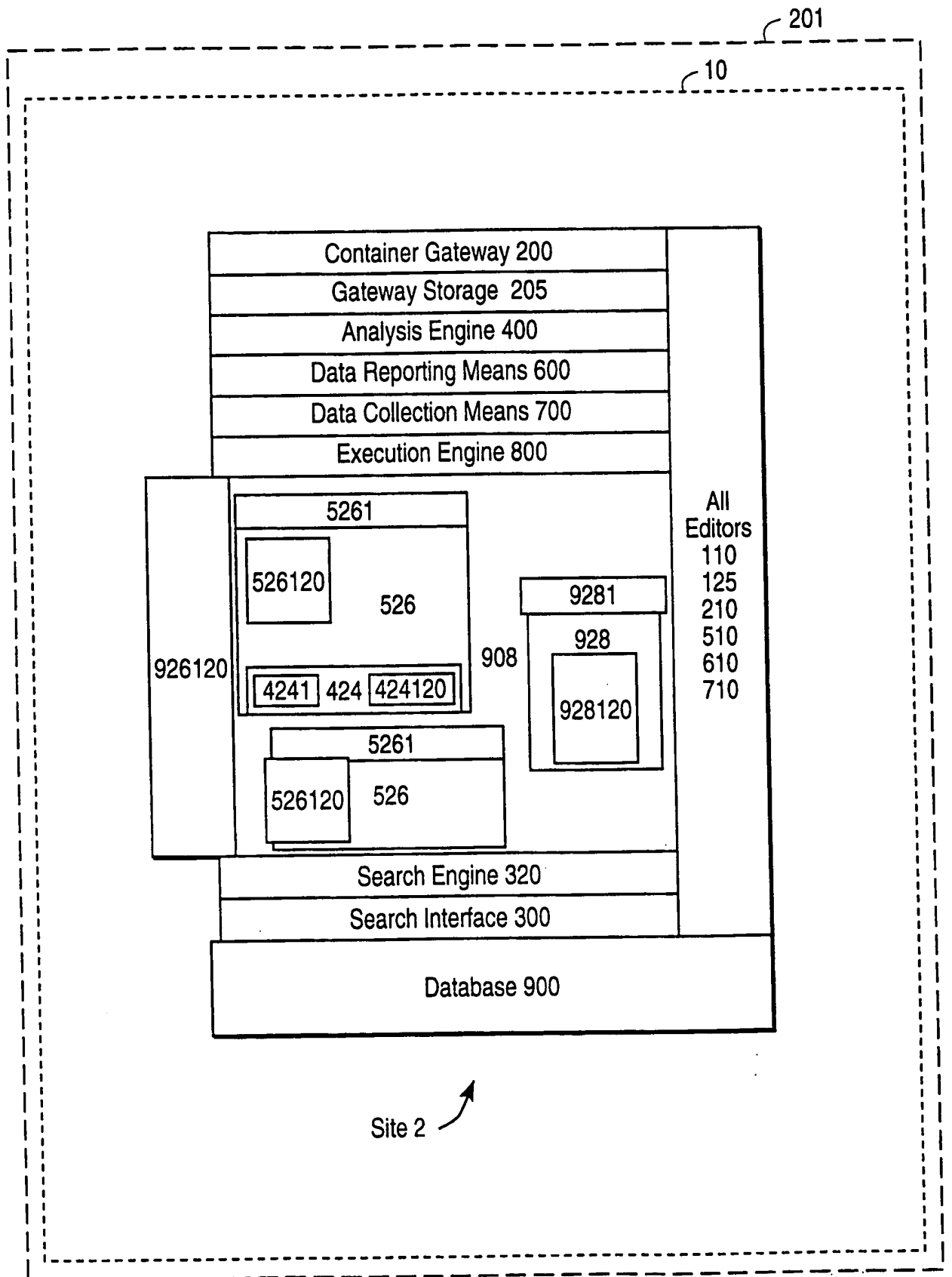


FIG. 2D

7/30

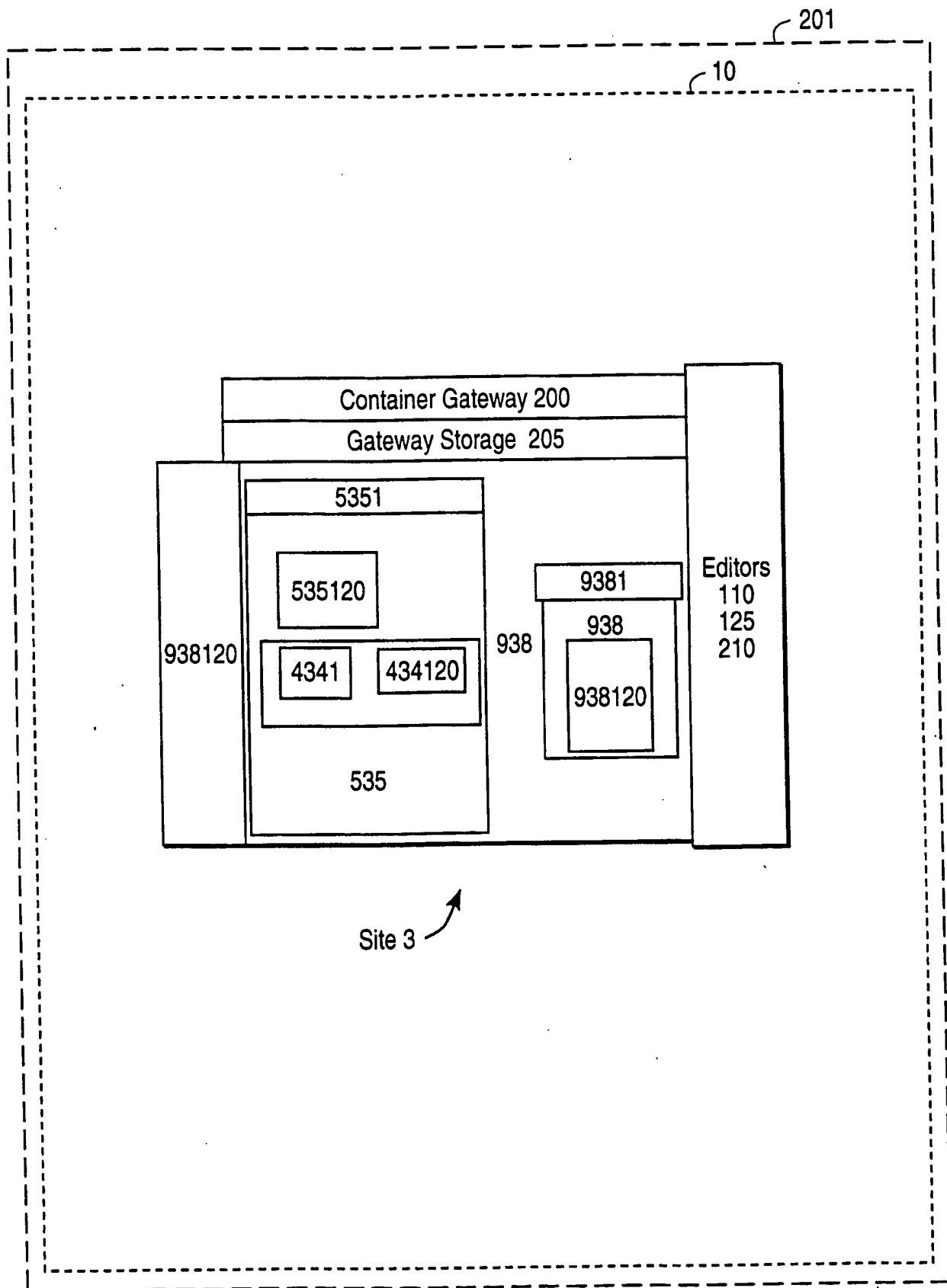


FIG. 2E

8/30

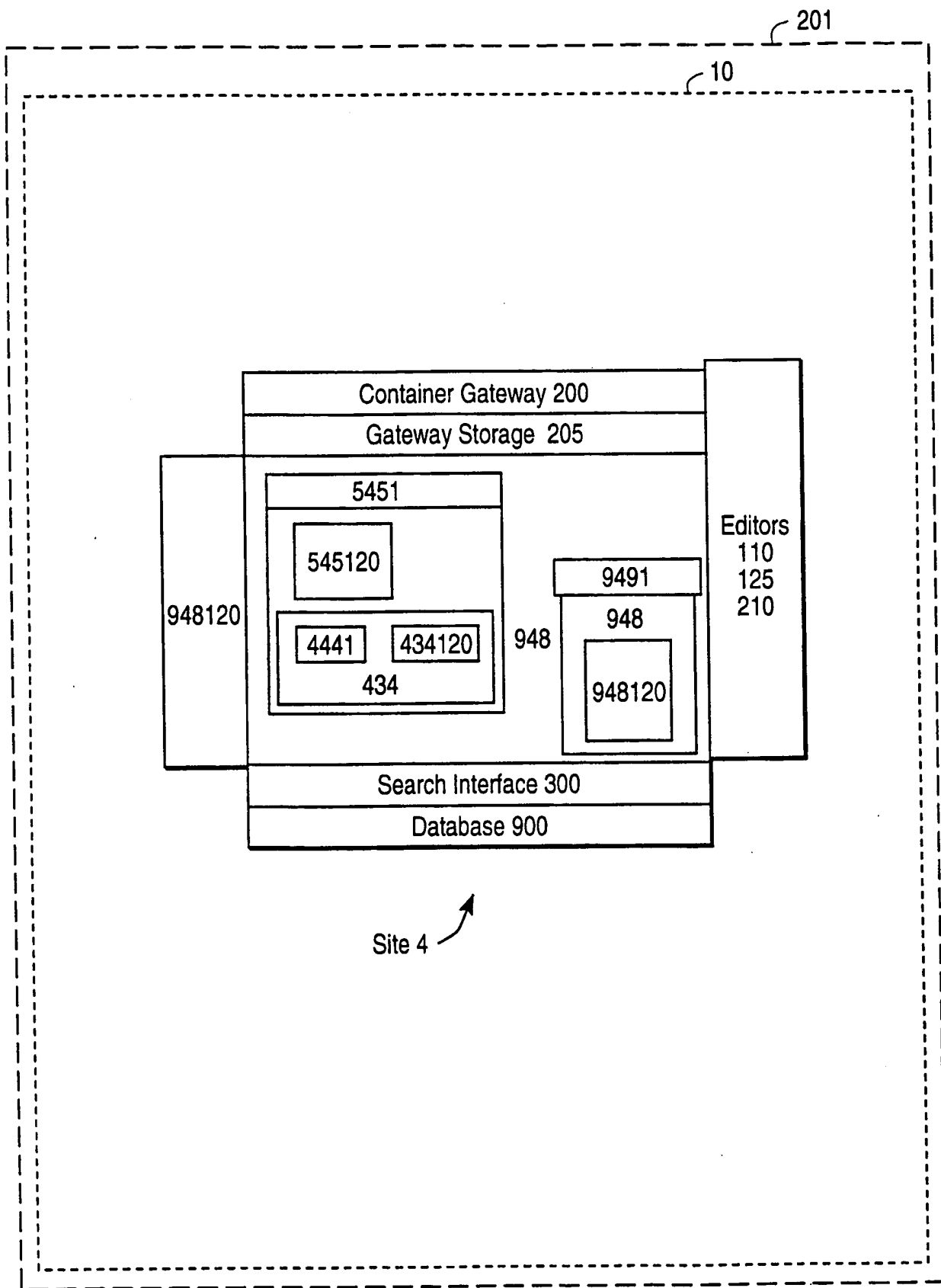


FIG. 2F

9/30

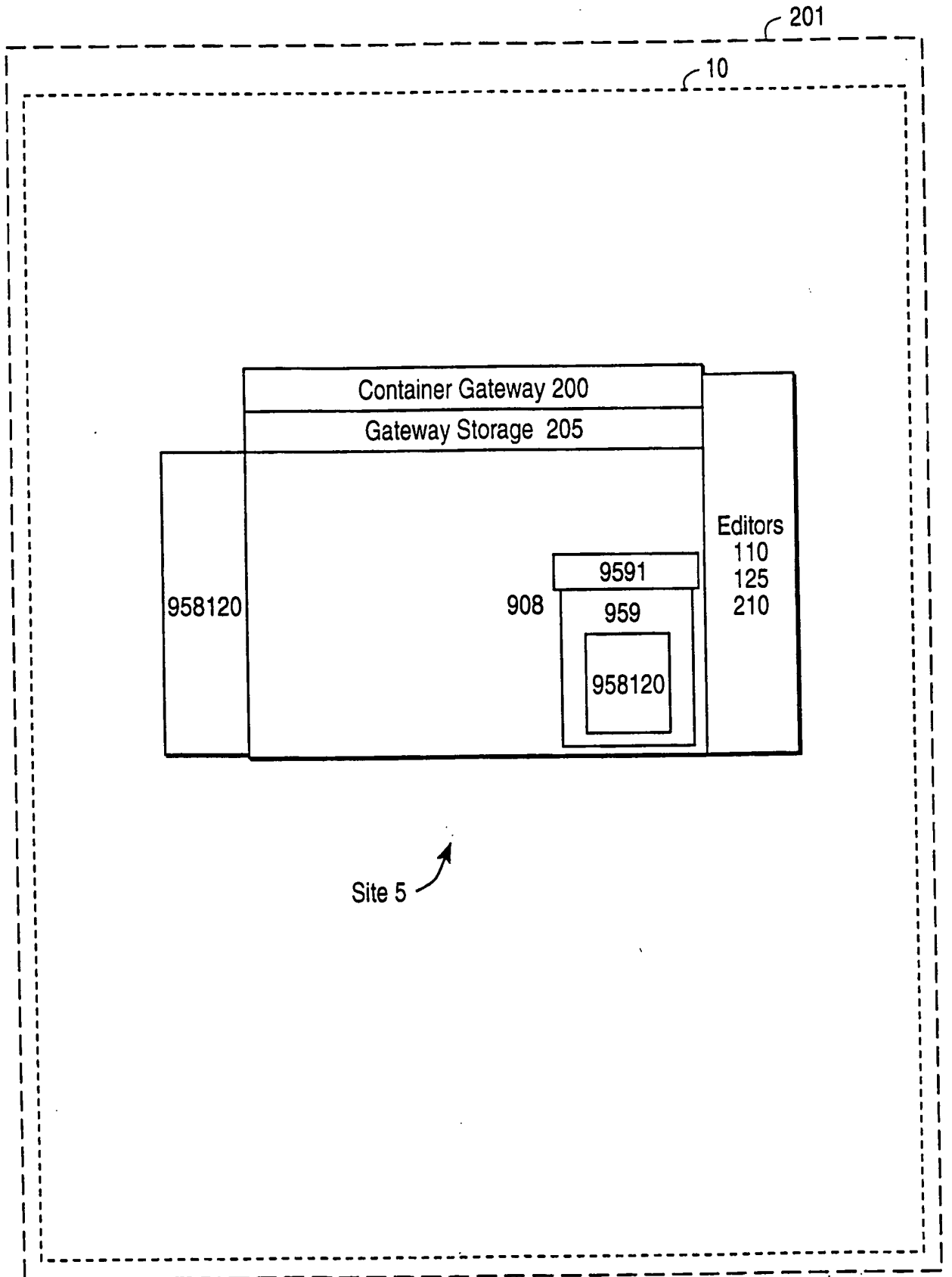


FIG. 2G

10/30

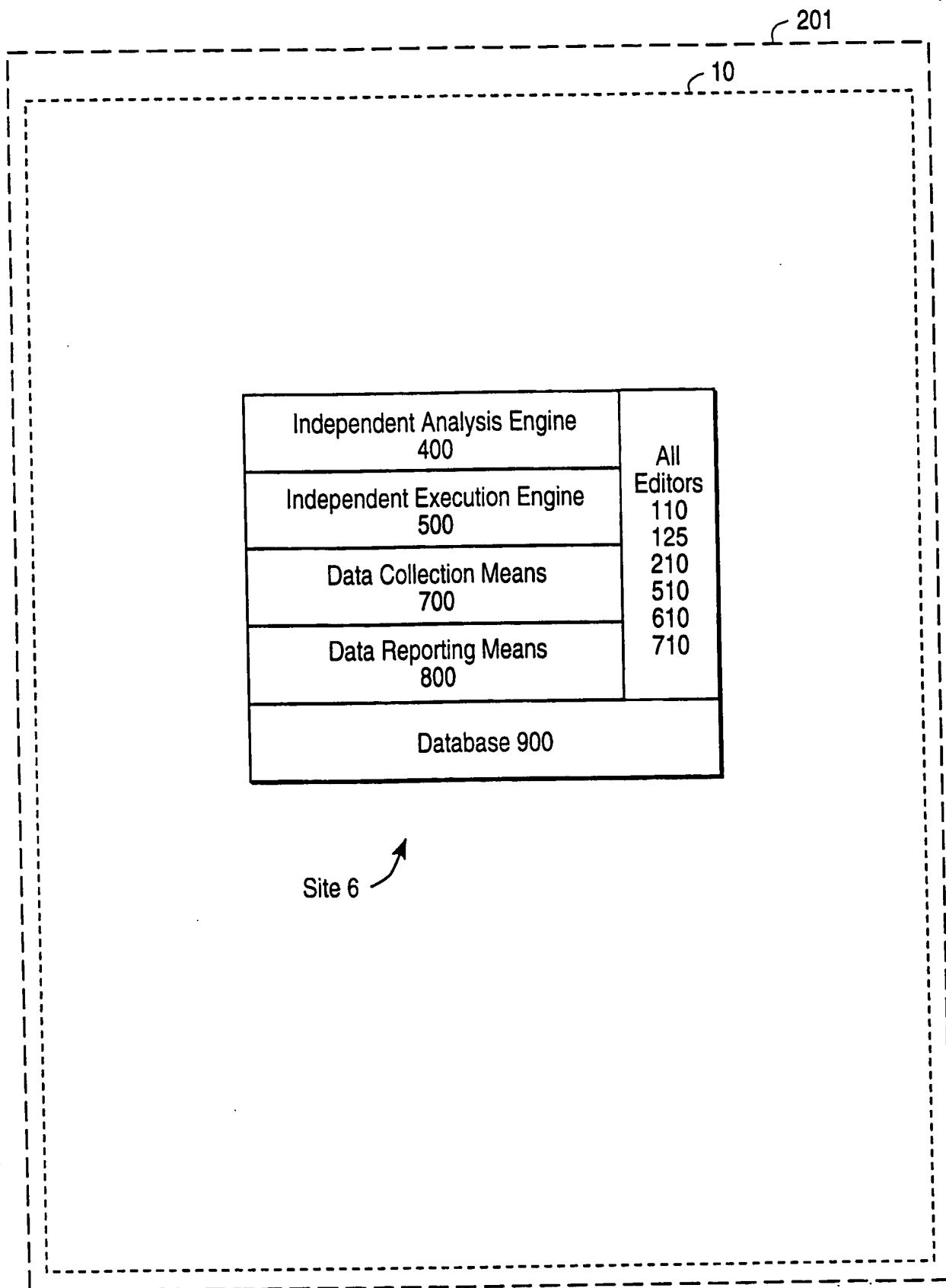


FIG. 2H

11/30

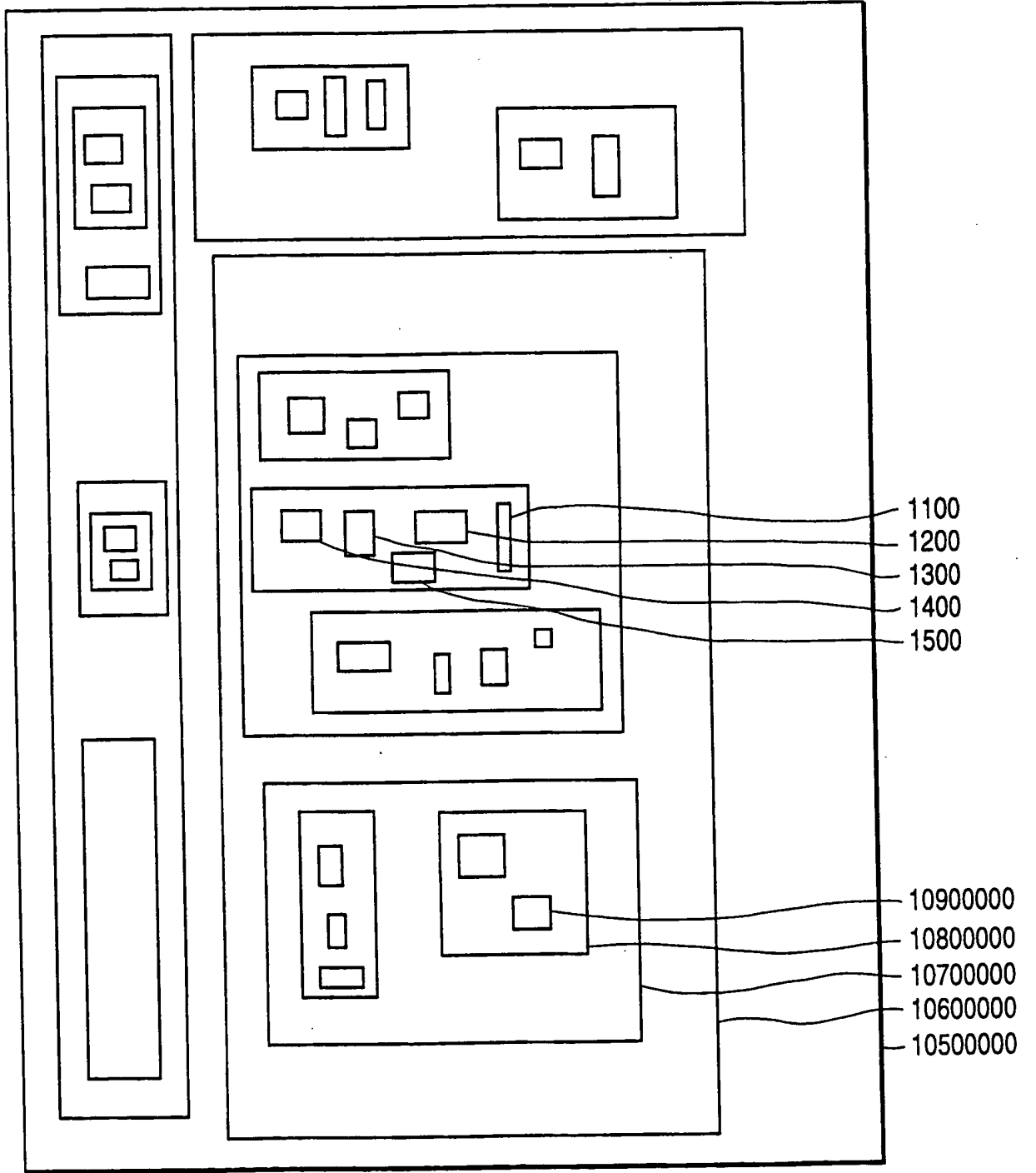


FIG. 3A

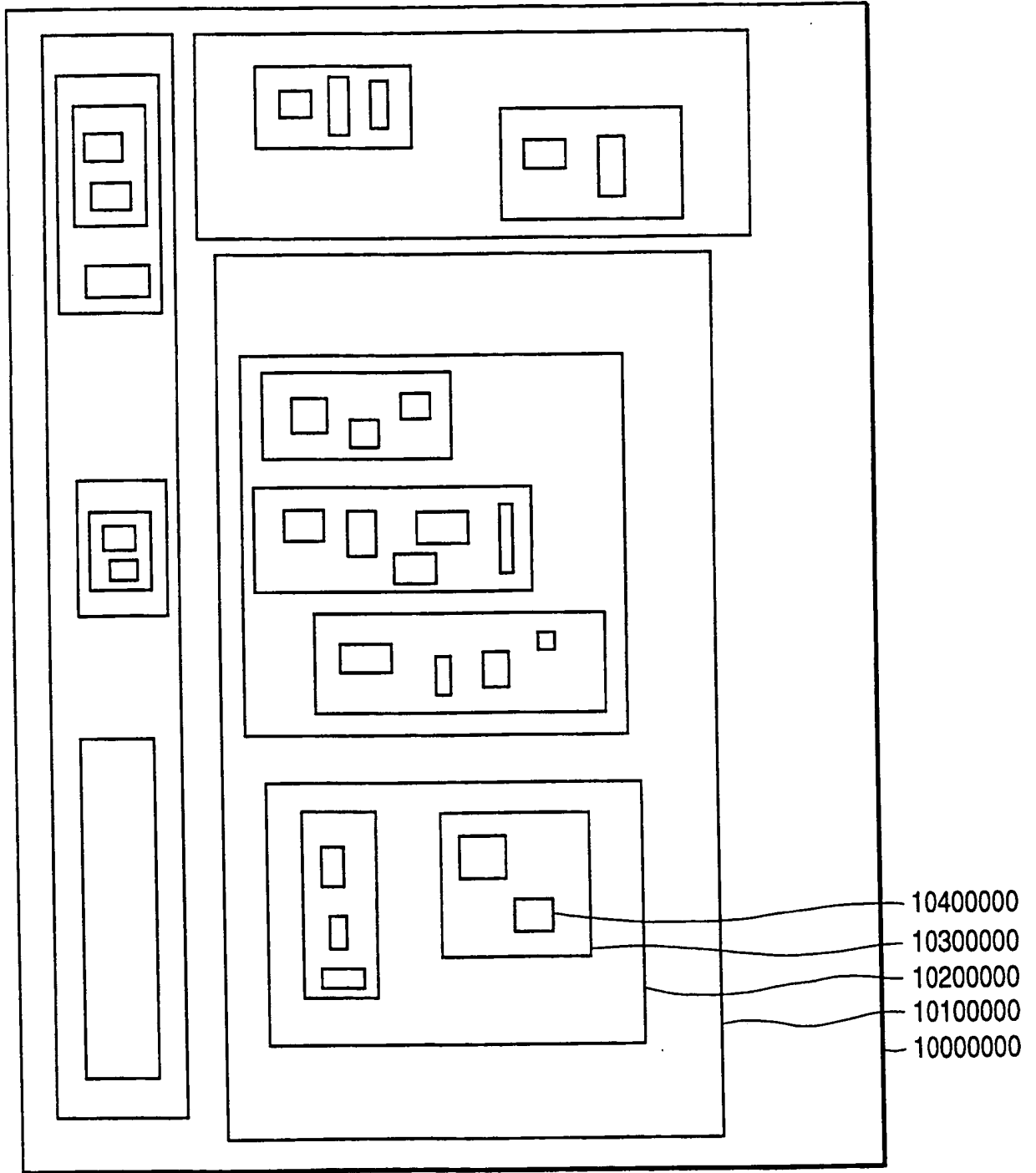


FIG. 3B

(100)

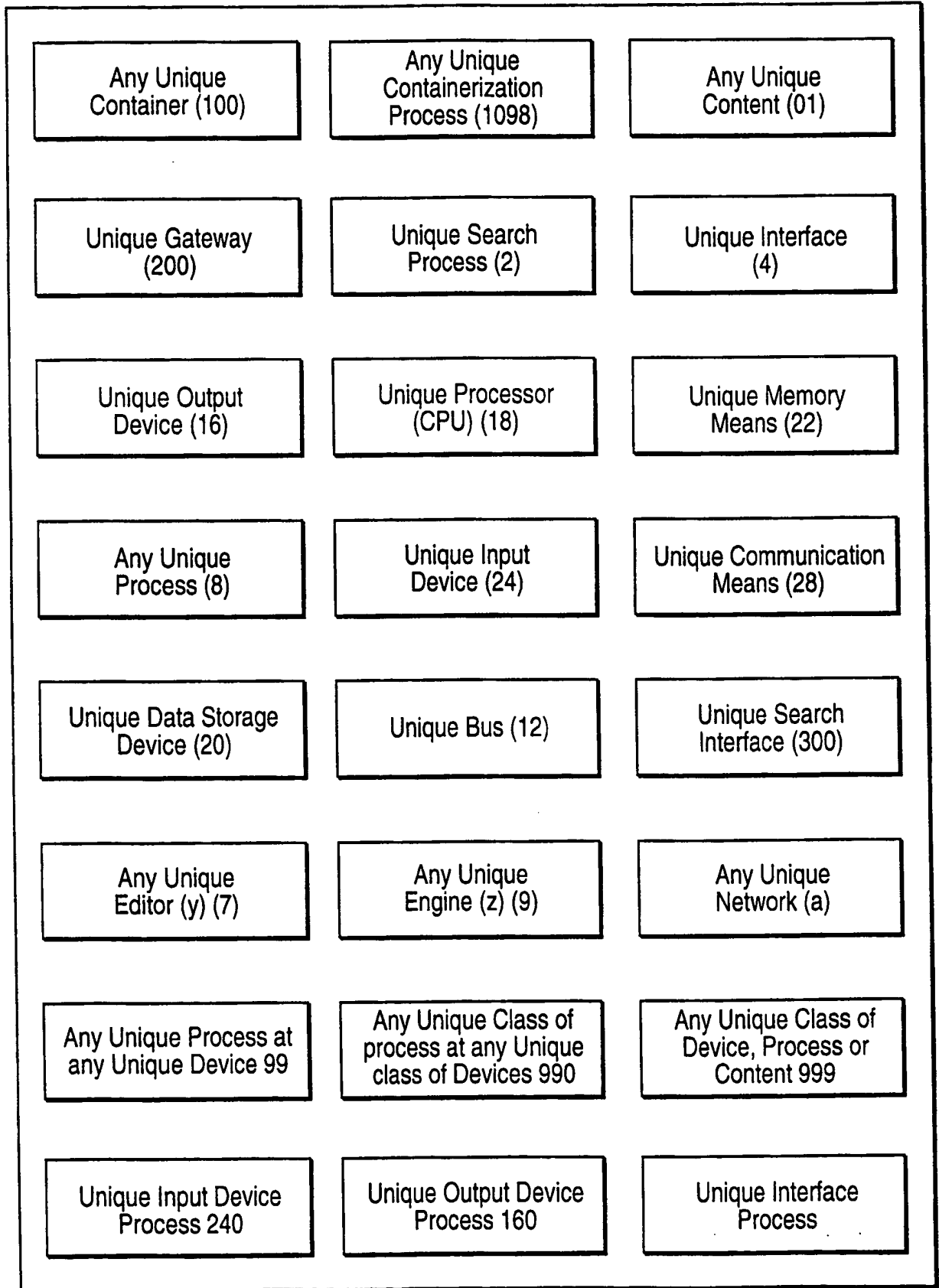


FIG. 3C



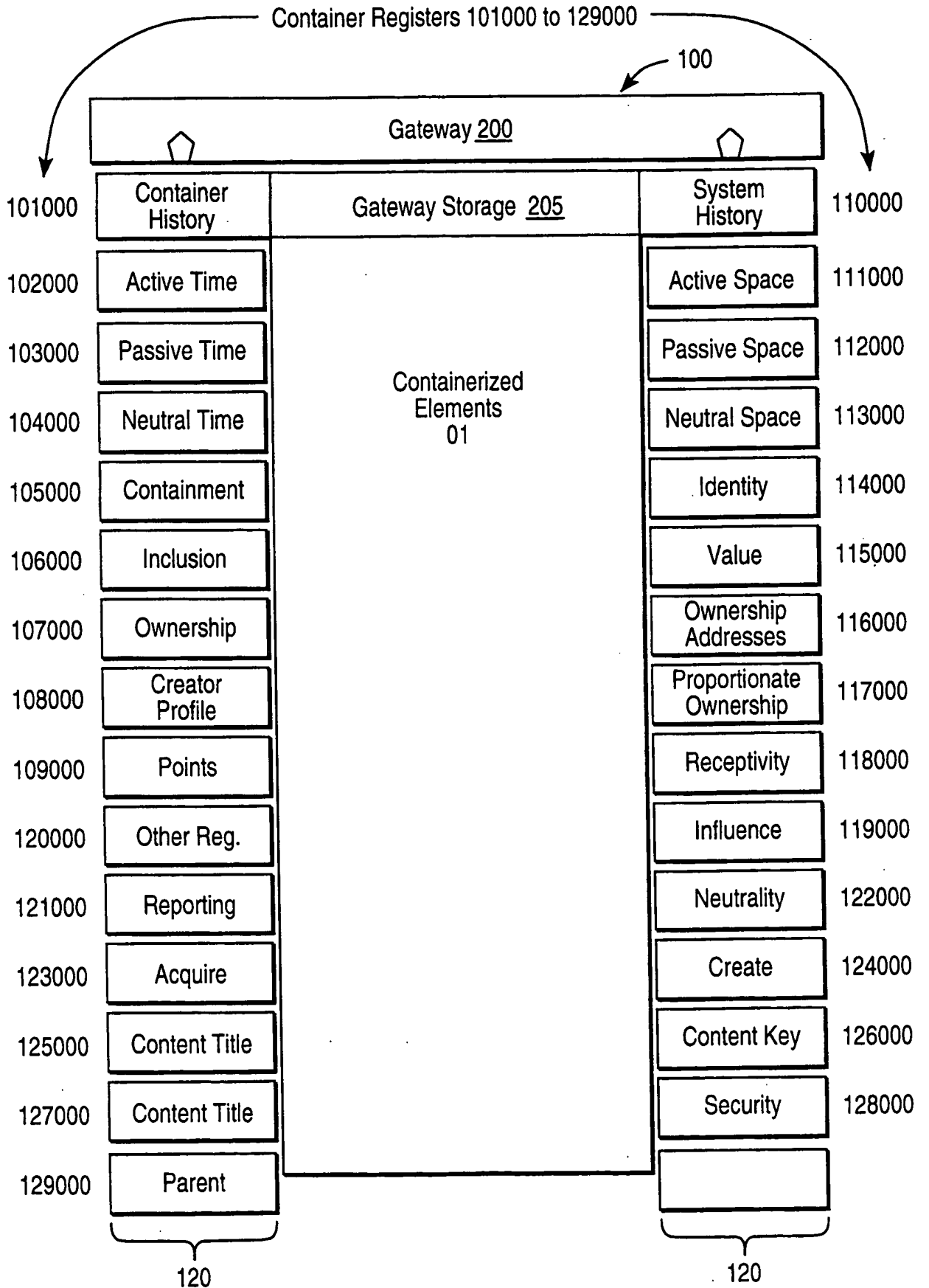


FIG. 4

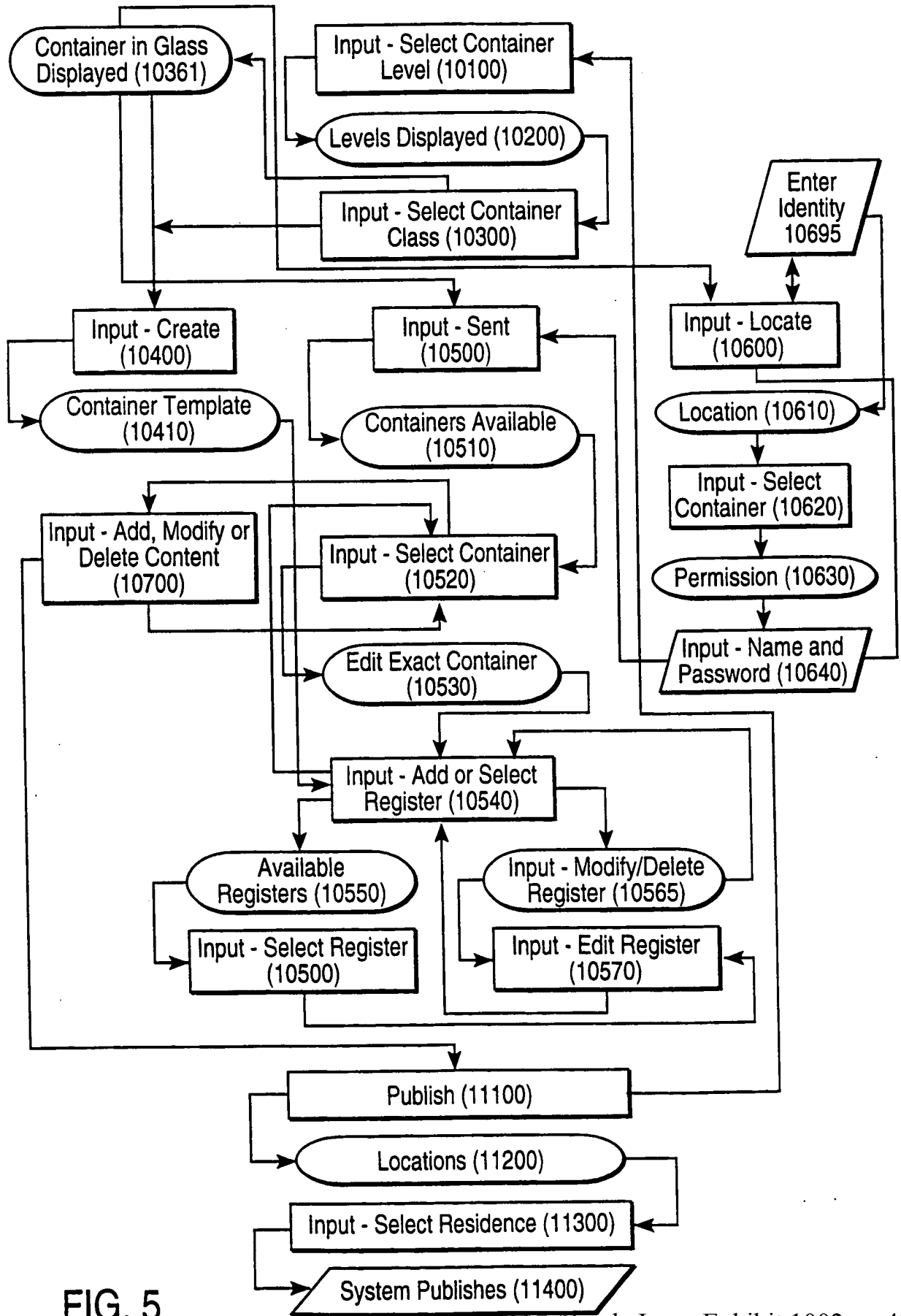


FIG. 5

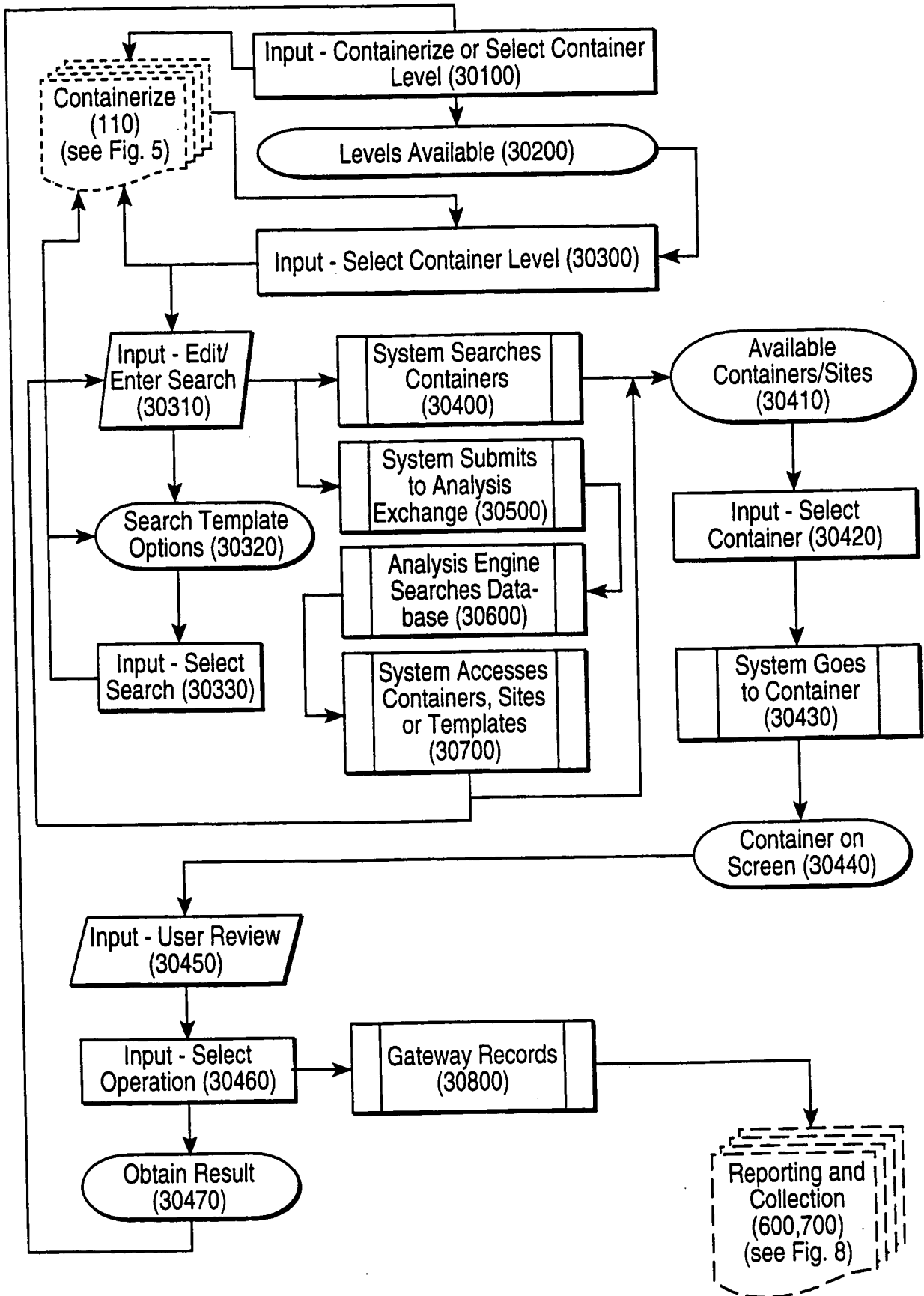


FIG. 6

17/30

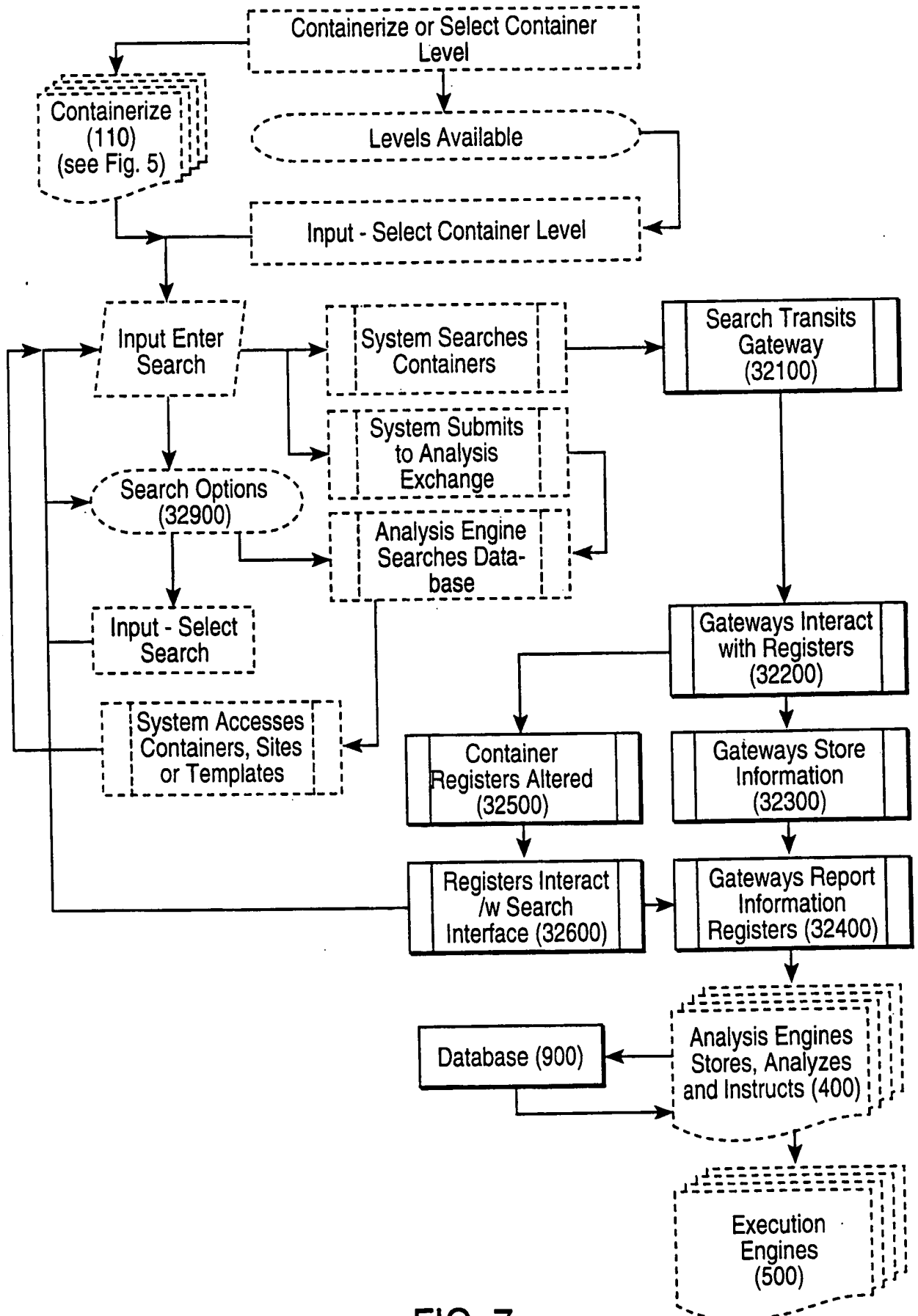


FIG. 7

18/30

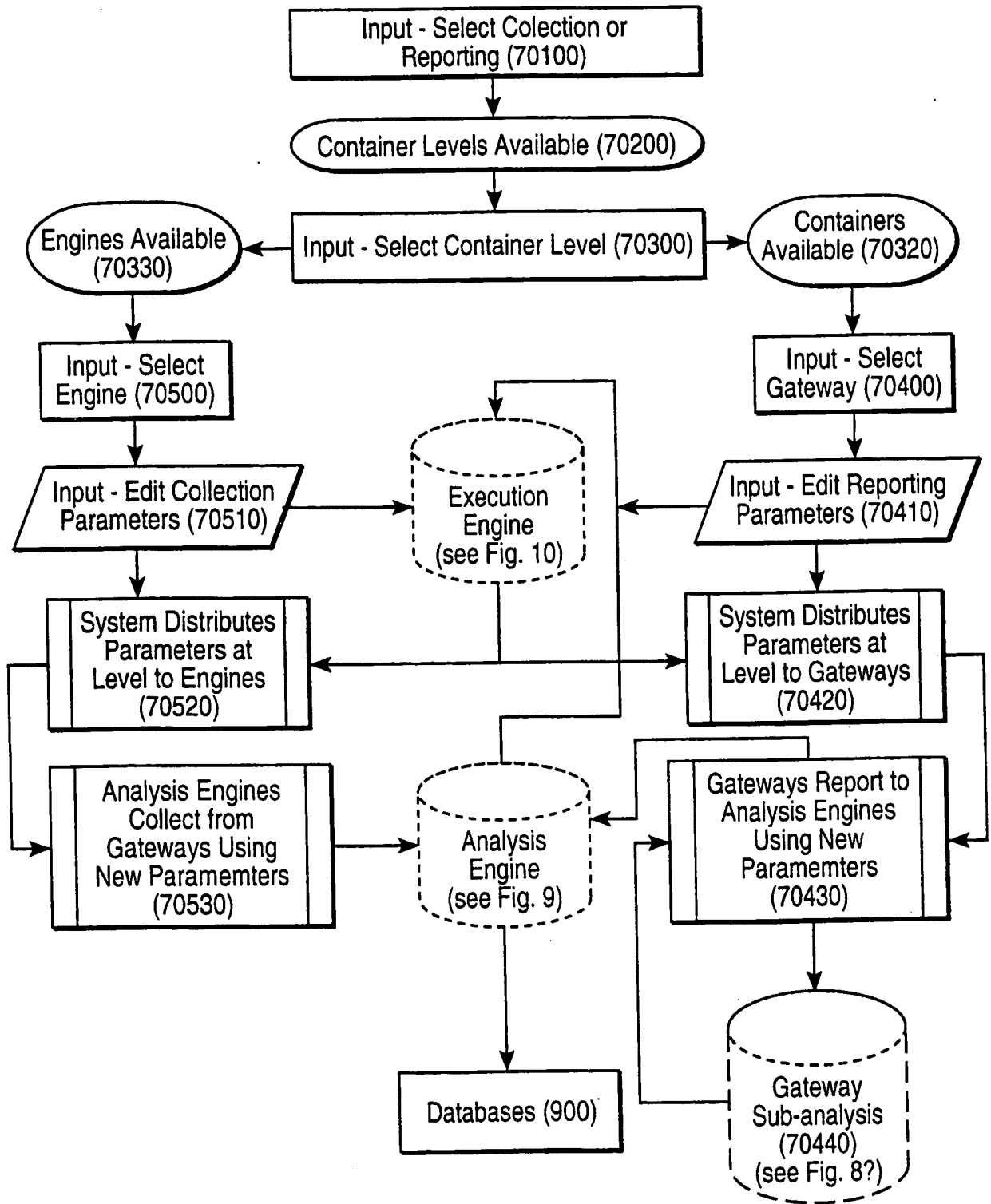


FIG. 8

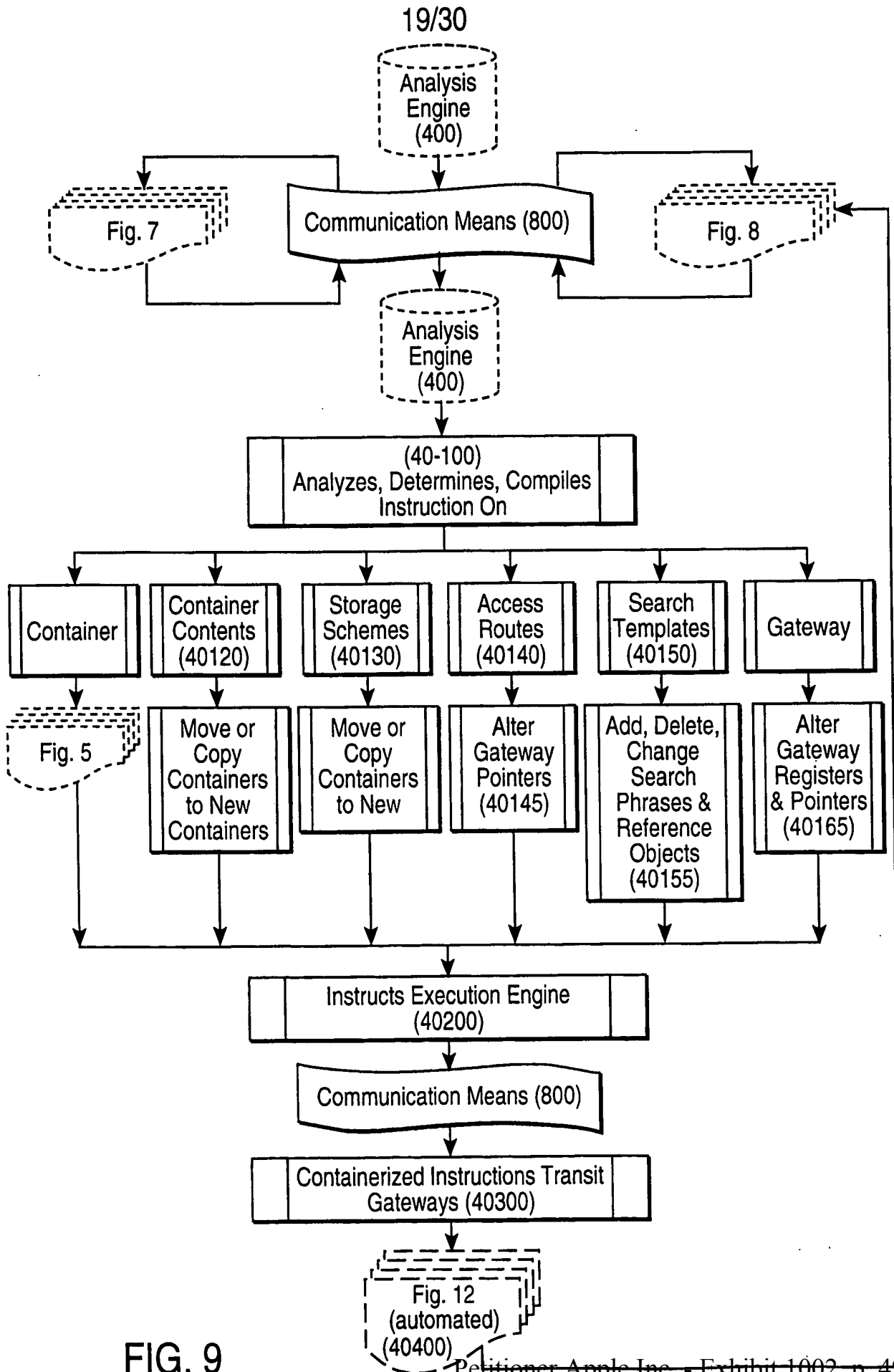


FIG. 9

20/30

EXECUTION ENGINE

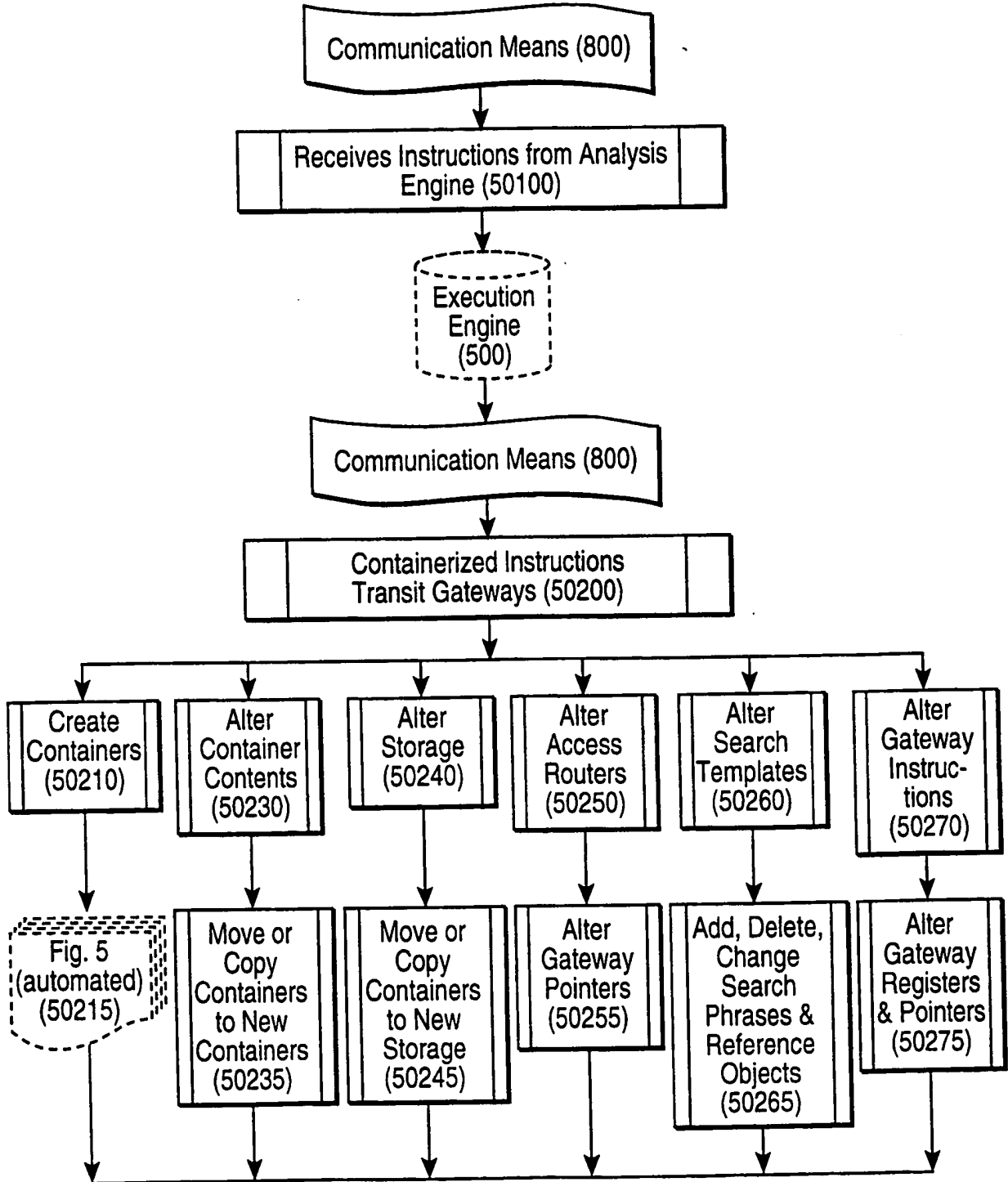


FIG. 10

21/30

GATEWAY EDITOR

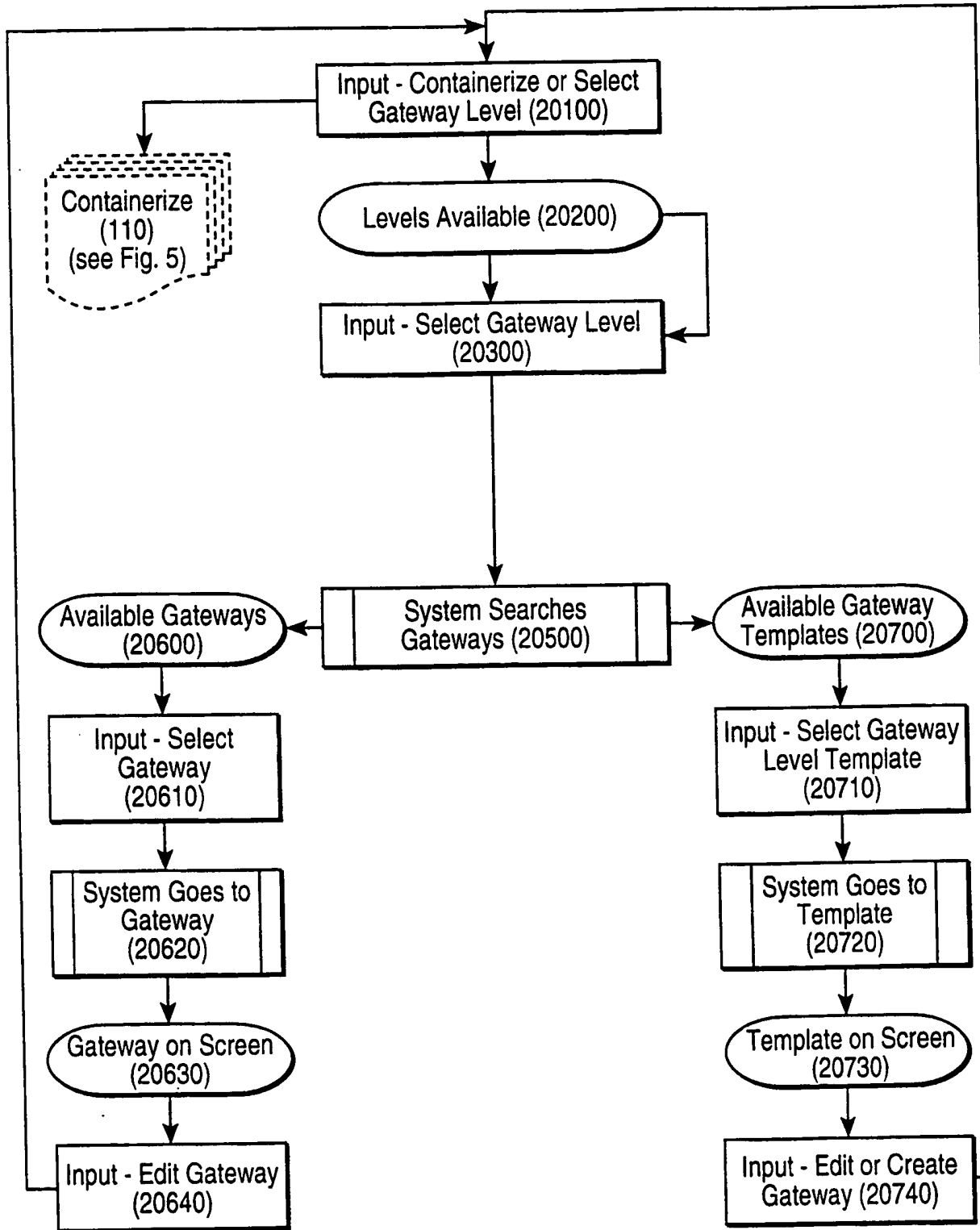


FIG. 11



GATEWAY PROCESS

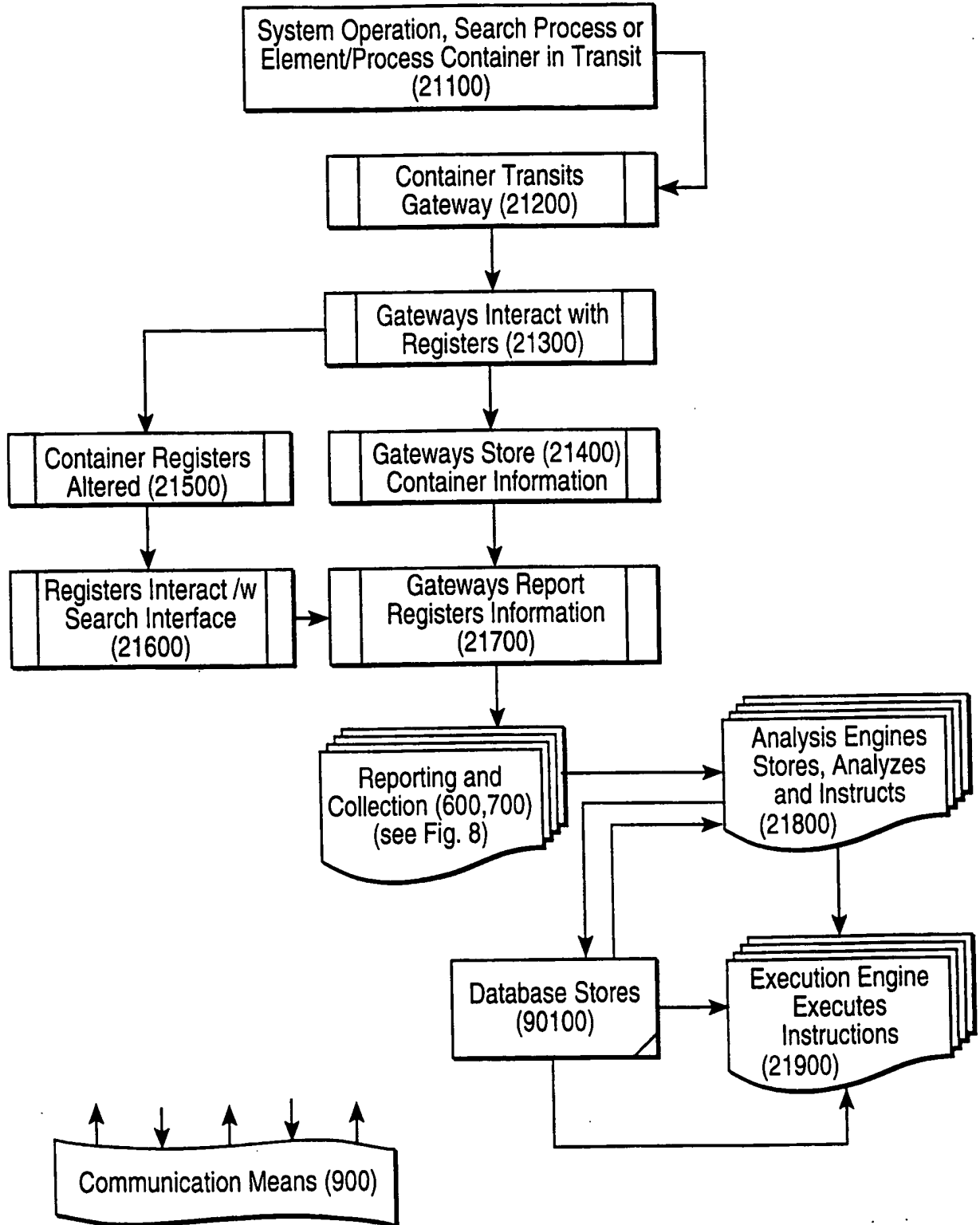


FIG. 12

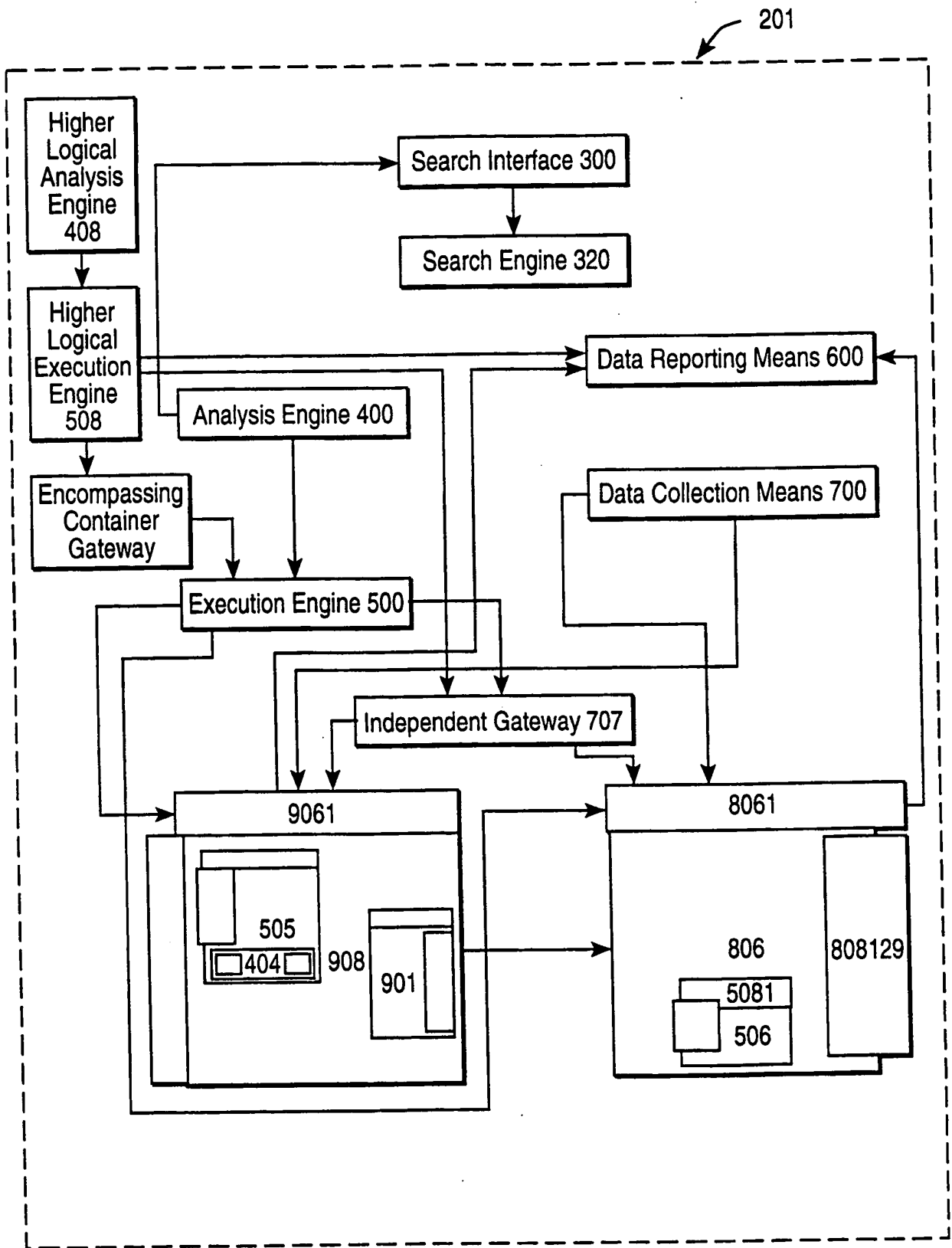


FIG. 13A

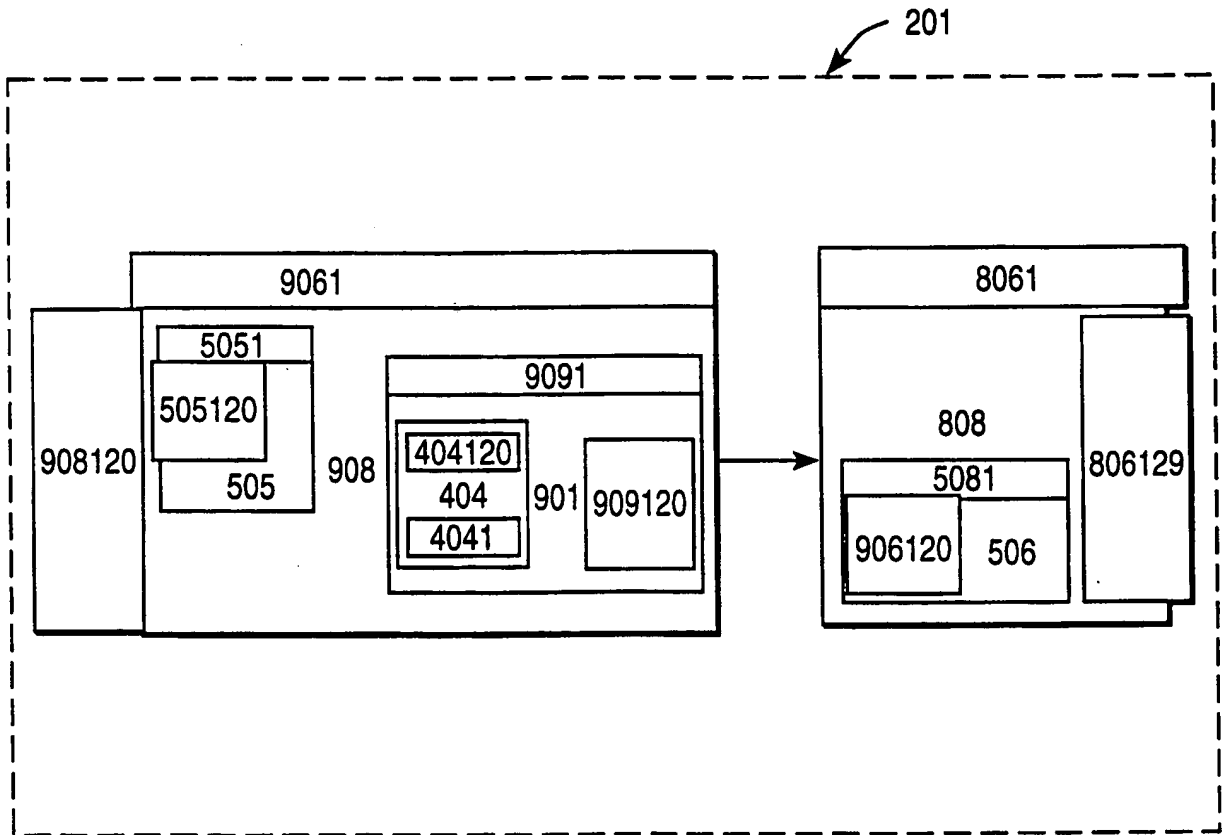


FIG. 13B

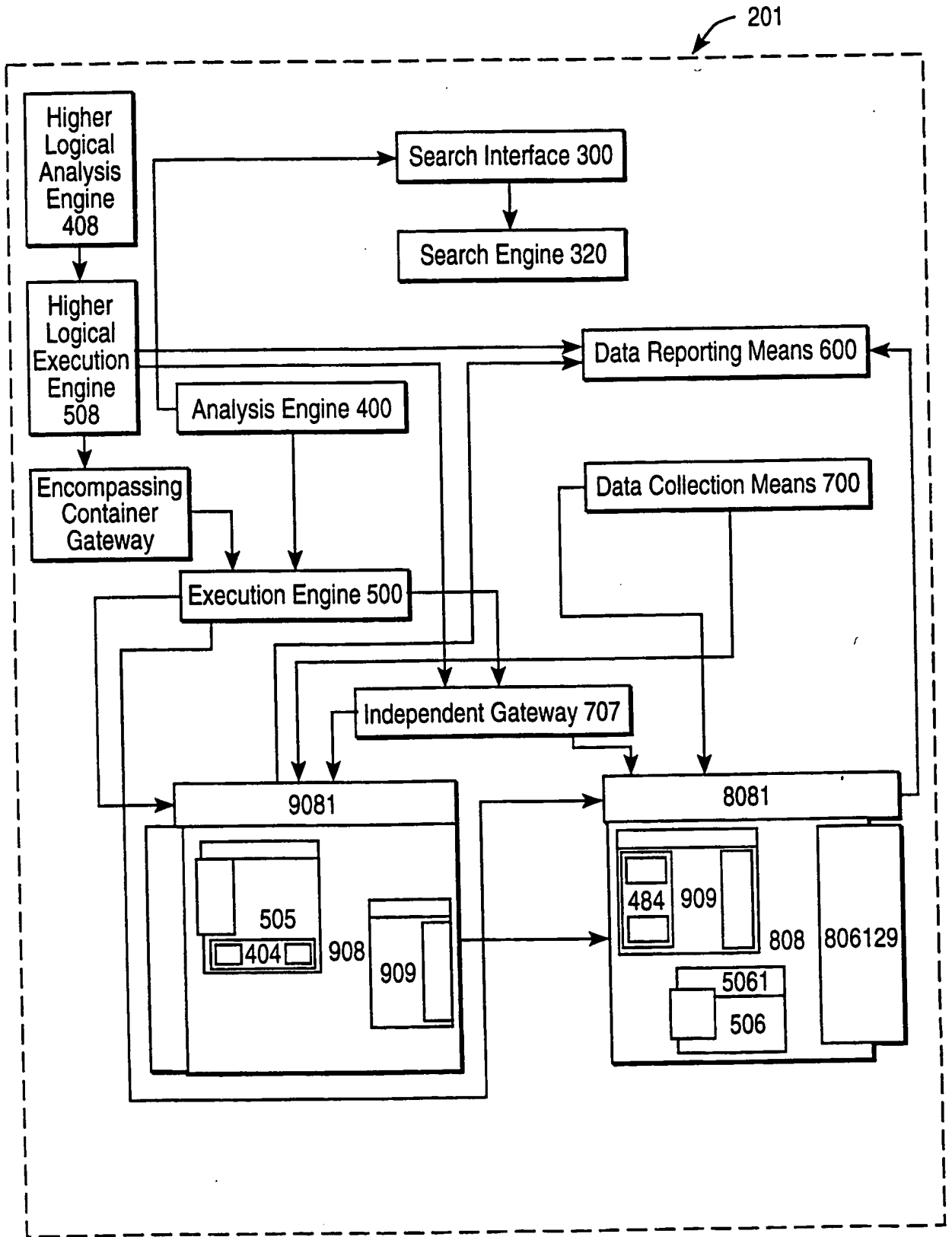


FIG. 13C

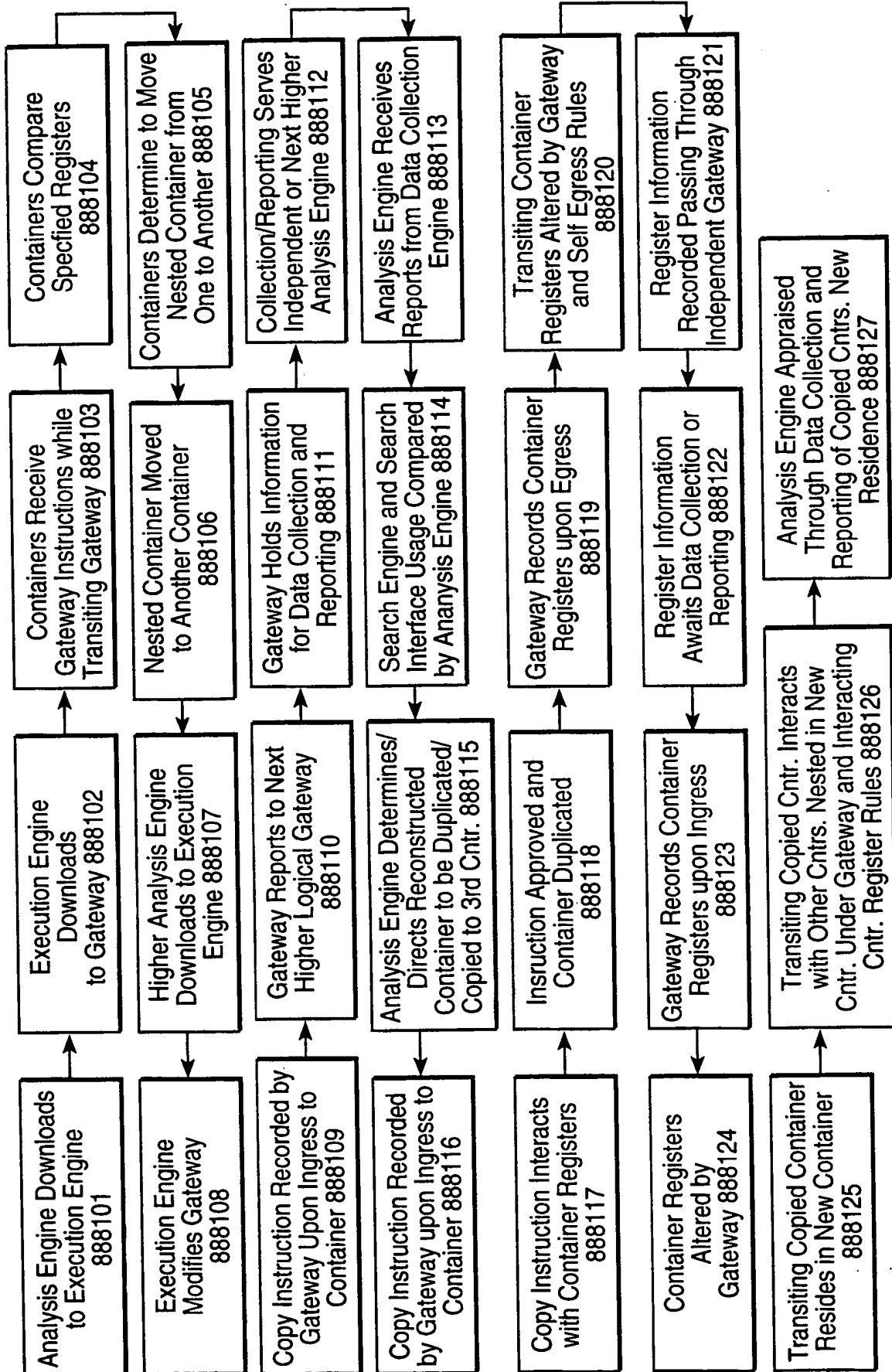


FIG. 13D

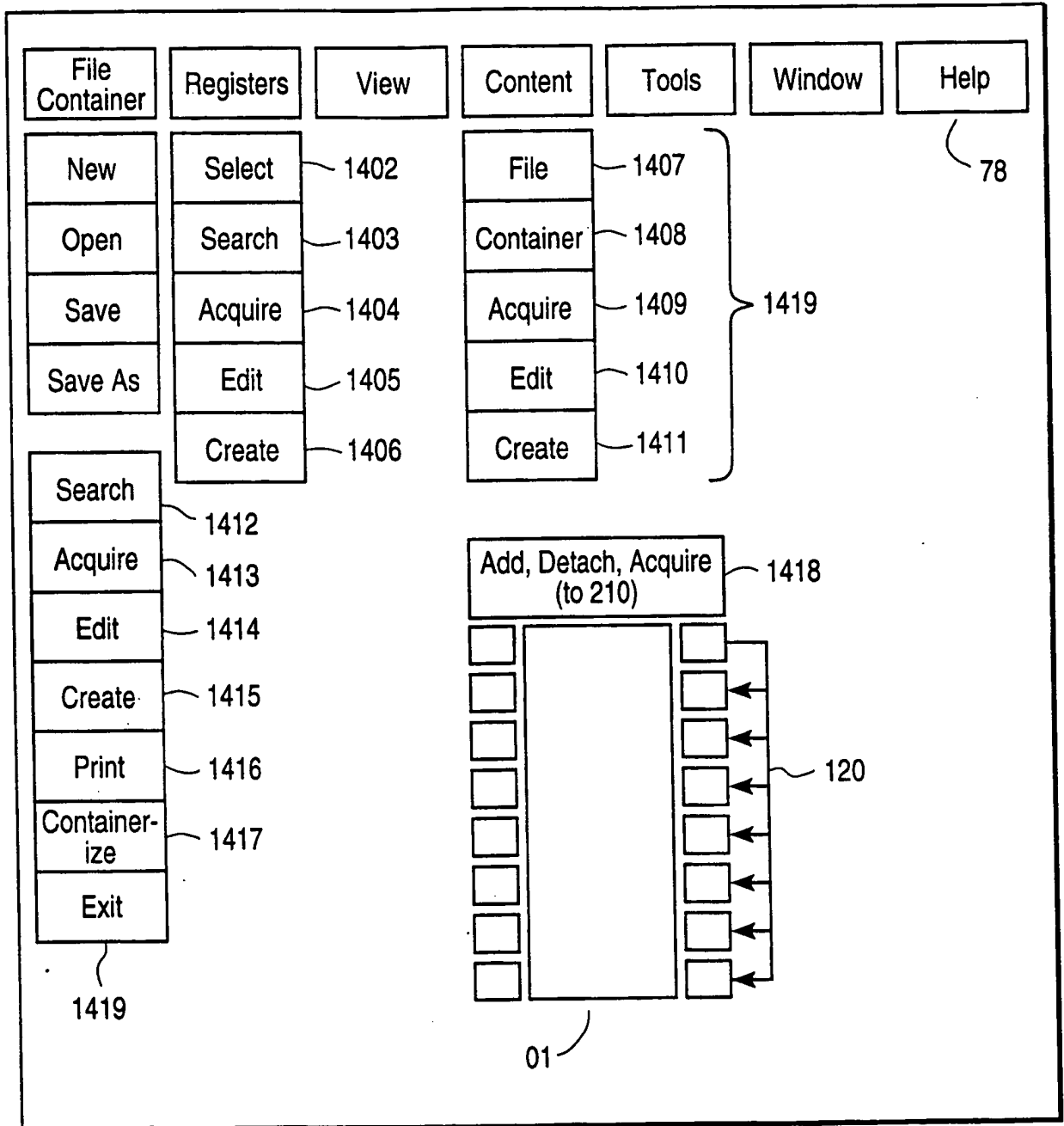


FIG. 14

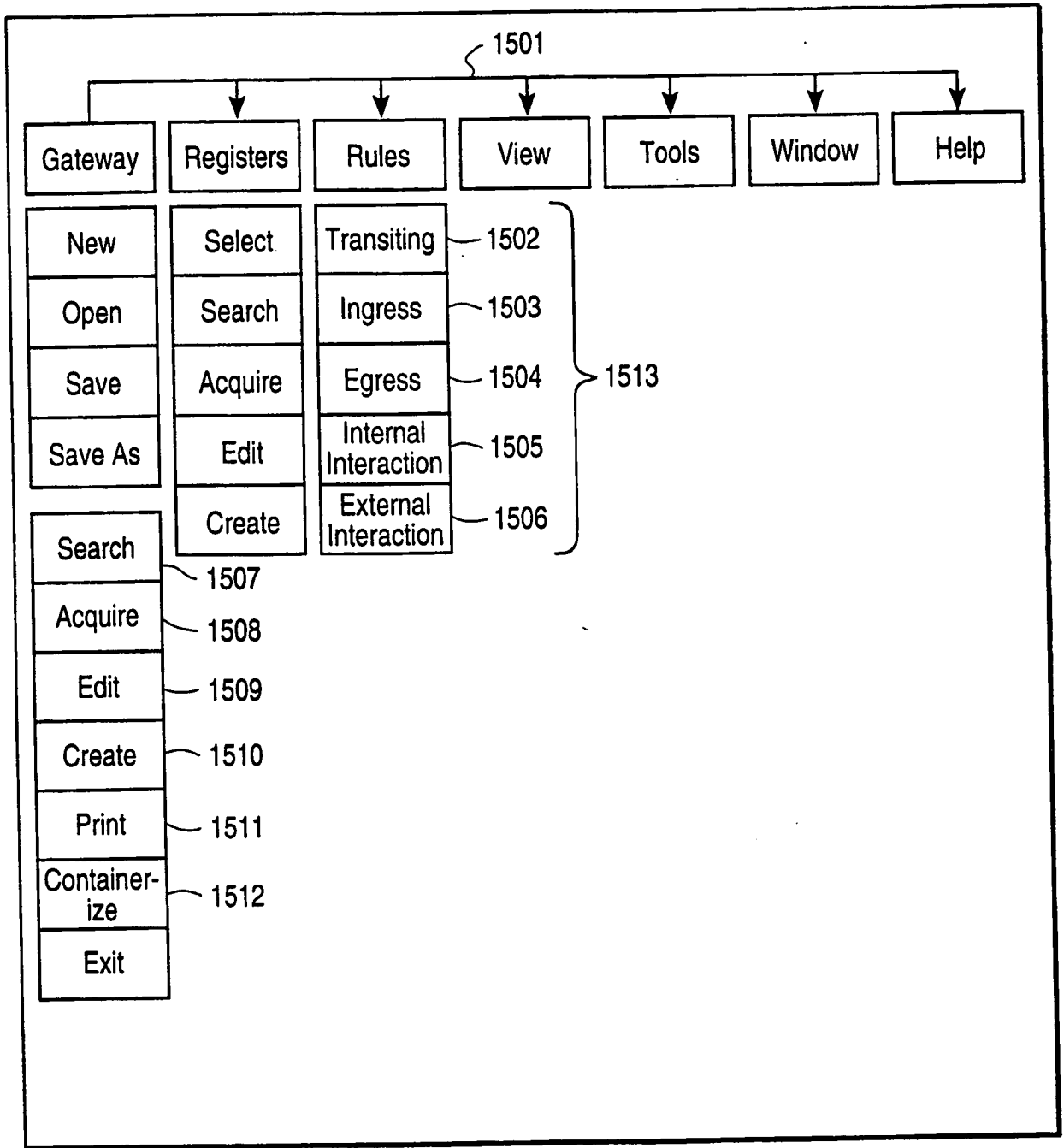


FIG. 15

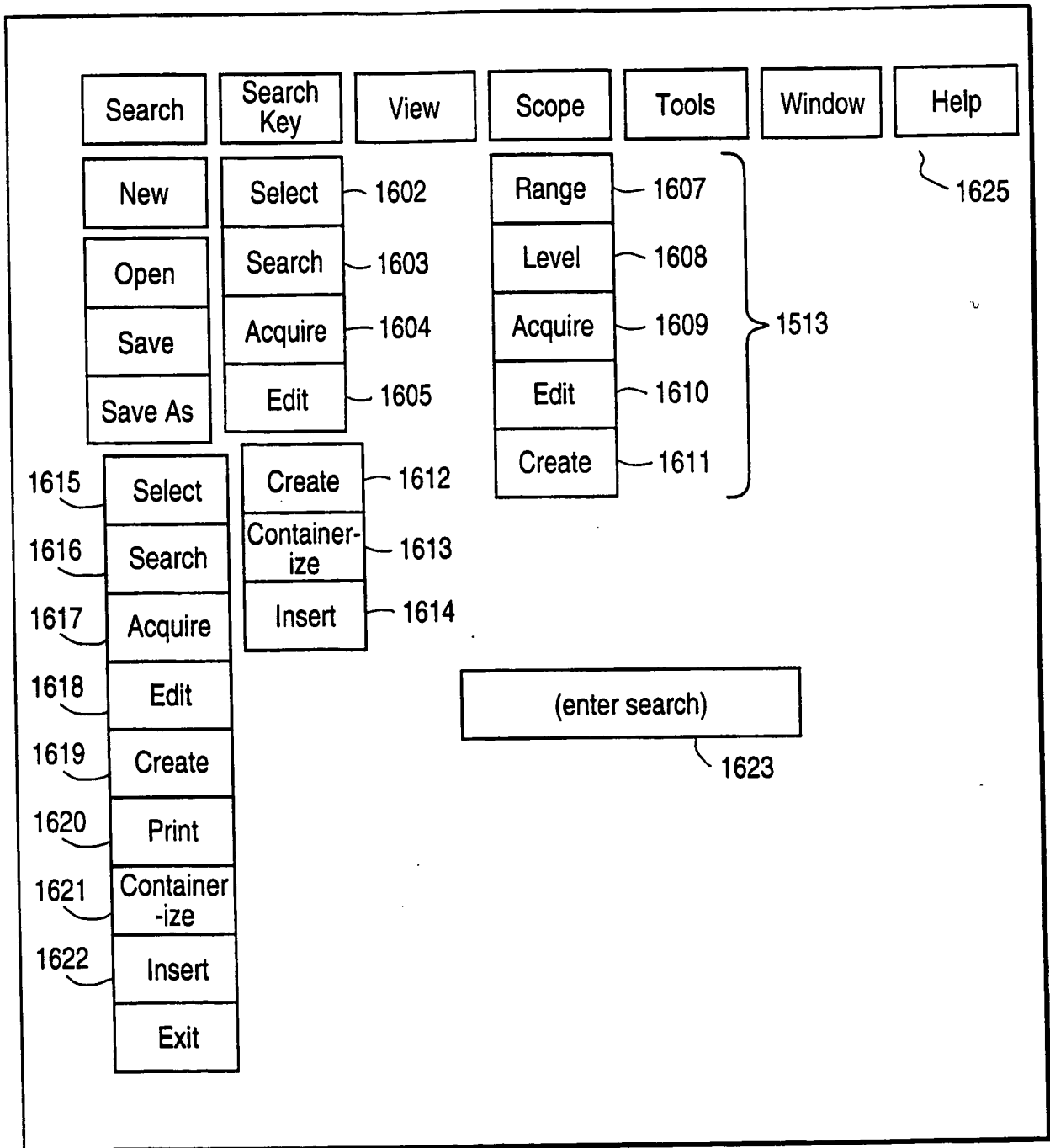


FIG. 16



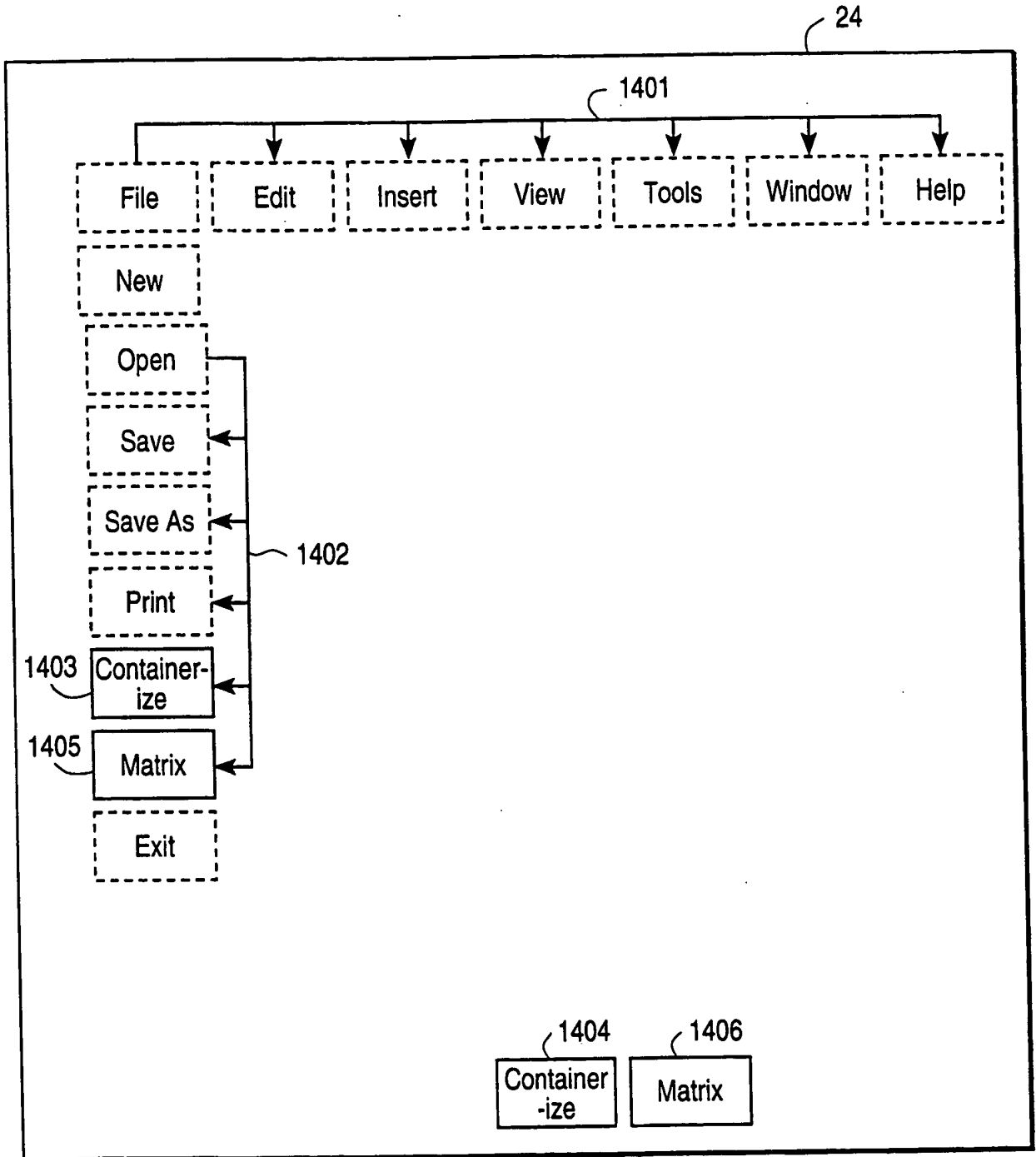


FIG. 17

INTERNATIONAL SEARCH REPORT

International application No.  
PCT/US99/01988

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :G06F 17/30, 3/14  
US CL :Please See Extra Sheet.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : Please See Extra Sheet.

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched  
MICROSOFT COMPUTER DICTIONARY

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)  
APS, PRO-QUEST

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5,768,510 A (GISH et al) 16 June 1998, column 5.	1-36
A	US 5,848,246 A (GISH et al) 08 December 1998, column 5.	1-36

Further documents are listed in the continuation of Box C.  See patent family annex.

* Special categories of cited documents:	*T* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
*A* document defining the general state of the art which is not considered to be of particular relevance	*X* document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
*E* earlier document published on or after the international filing date	*Y* document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
*L* document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	*Z* document member of the same patent family
*O* document referring to an oral disclosure, use, exhibition or other means	
*P* document published prior to the international filing date but later than the priority date claimed	

Date of the actual completion of the international search

03 JUNE 1999

Date of mailing of the international search report

15 JUN 1999

Name and mailing address of the ISA/US  
Commissioner of Patents and Trademarks  
Box PCT  
Washington, D.C. 20231

Facsimile No. (703) 305-3230

Authorized officer

RUAY LIAN HO

Telephone No. (703) 305-3834

INTERNATIONAL SEARCH REPORT

International application No.  
PCT/US99/01988

A. CLASSIFICATION OF SUBJECT MATTER:

US CL :

707/1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 100, 101, 102, 103, 104, 200, 201, 202, 203, 204, 205, 206; 709/202, 203, 218, 228; 713/200, 201

B. FIELDS SEARCHED

Minimum documentation searched

Classification System: U.S.

707/1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 100, 101, 102, 103, 104, 200, 201, 202, 203, 204, 205, 206; 709/202, 203, 218, 228; 713/200, 201



UNITED STATES PATENT AND TRADEMARK OFFICE

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

Table with 5 columns: APPLICATION NO., FILING DATE, FIRST NAMED INVENTOR, ATTORNEY DOCKET NO., CONFIRMATION NO.

11/280,700 11/14/2005 Michael De Angelo 17776-002US4 9680

26181 7590 03/17/2008
FISH & RICHARDSON P.C.
PO BOX 1022
MINNEAPOLIS, MN 55440-1022

Table with 1 column: EXAMINER

NGUYEN, CAM LINH T

Table with 2 columns: ART UNIT, PAPER NUMBER

2161

Table with 2 columns: MAIL DATE, DELIVERY MODE

03/17/2008 PAPER

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

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

<b>Office Action Summary</b>	<b>Application No.</b> 11/280,700	<b>Applicant(s)</b> ANGELO, MICHAEL DE	
	<b>Examiner</b> CAM-LINH NGUYEN	<b>Art Unit</b> 2161	

**-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --**

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 1 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

- 1)  Responsive to communication(s) filed on 17 February 2006.
- 2a)  This action is **FINAL**.
- 2b)  This action is non-final.
- 3)  Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

- 4)  Claim(s) 1-30 is/are pending in the application.
  - 4a) Of the above claim(s) \_\_\_\_\_ is/are withdrawn from consideration.
- 5)  Claim(s) \_\_\_\_\_ is/are allowed.
- 6)  Claim(s) \_\_\_\_\_ is/are rejected.
- 7)  Claim(s) \_\_\_\_\_ is/are objected to.
- 8)  Claim(s) 1-30 are subject to restriction and/or election requirement.

**Application Papers**

- 9)  The specification is objected to by the Examiner.
- 10)  The drawing(s) filed on \_\_\_\_\_ is/are: a)  accepted or b)  objected to by the Examiner.  
 Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).  
 Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11)  The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

- 12)  Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
  - a)  All    b)  Some \*    c)  None of:
    - 1.  Certified copies of the priority documents have been received.
    - 2.  Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
    - 3.  Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

\* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

- 1)  Notice of References Cited (PTO-892)
- 2)  Notice of Draftsperson's Patent Drawing Review (PTO-948)
- 3)  Information Disclosure Statement(s) (PTO/SB/08)  
 Paper No(s)/Mail Date \_\_\_\_\_.
- 4)  Interview Summary (PTO-413)  
 Paper No(s)/Mail Date \_\_\_\_\_.
- 5)  Notice of Informal Patent Application
- 6)  Other: \_\_\_\_\_.

## DETAILED ACTION

### *Election/Restrictions*

1. Restriction to one of the following inventions is required under 35 U.S.C. 121:
  - I. Claims 1 – 18, 21 – 24, and 30, drawn to a method for searching data, classified in class 707, subclass 3.
  - II. Claims 19 – 20, 25 - 29, drawn to a method for update database, classified in class 707, subclass 100.

The inventions are distinct, each from the other because of the following reasons:

2. Inventions I and II are unrelated. Inventions are unrelated if it can be shown that they are not disclosed as capable of use together and they have different designs, modes of operation, and effects (MPEP § 802.01 and § 806.06). In the instant case, the invention I and II are classified in two different subclass and is not related to each other. Furthermore, the inventions as claimed do not encompass overlapping subject matter and there is nothing of record to show them to be obvious variants.

3. Restriction for examination purposes as indicated is proper because all these inventions listed in this action are independent or distinct for the reasons given above and there would be a serious search and examination burden if restriction were not required because one or more of the following reasons apply:

- (a) the inventions have acquired a separate status in the art in view of their different classification;

Art Unit: 2161

- (b) the inventions have acquired a separate status in the art due to their recognized divergent subject matter;
- (c) the inventions require a different field of search (for example, searching different classes/subclasses or electronic resources, or employing different search queries);
- (d) the prior art applicable to one invention would not likely be applicable to another invention;
- (e) the inventions are likely to raise different non-prior art issues under 35 U.S.C. 101 and/or 35 U.S.C. 112, first paragraph.

**Applicant is advised that the reply to this requirement to be complete must include (i) an election of a invention to be examined even though the requirement may be traversed (37 CFR 1.143) and (ii) identification of the claims encompassing the elected invention.**

The election of an invention may be made with or without traverse. To reserve a right to petition, the election must be made with traverse. If the reply does not distinctly and specifically point out supposed errors in the restriction requirement, the election shall be treated as an election without traverse. Traversal must be presented at the time of election in order to be considered timely. Failure to timely traverse the requirement will result in the loss of right to petition under 37 CFR 1.144. If claims are added after the election, applicant must indicate which of these claims are readable on the elected invention.

If claims are added after the election, applicant must indicate which of these claims are readable upon the elected invention.

Should applicant traverse on the ground that the inventions are not patentably distinct, applicant should submit evidence or identify such evidence now of record showing the

Art Unit: 2161

inventions to be obvious variants or clearly admit on the record that this is the case. In either instance, if the examiner finds one of the inventions unpatentable over the prior art, the evidence or admission may be used in a rejection under 35 U.S.C. 103(a) of the other invention.

4. Any inquiry concerning this communication or earlier communications from the examiner should be directed to CAM-LINH NGUYEN whose telephone number is (571)272-4024. The examiner can normally be reached on Monday-Friday.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Apu Mofiz can be reached on (571) 272-4080. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

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

/CamLinh Nguyen/

Primary Examiner, Art Unit 2161



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

<b>REQUEST FOR WITHDRAWAL AS ATTORNEY OR AGENT AND CHANGE OF CORRESPONDENCE ADDRESS</b>	Application Number	11/280,700
	Filing Date	November 14, 2005
	First Named Inventor	Michael De Angelo
	Art Unit	2161
	Examiner Name	Not Assigned
	Attorney Docket Number	PATT-002/04US

**To: Commissioner for Patents**  
**P.O. Box 1450**  
**Alexandria, VA 22313-1450**

Please withdraw me as attorney or agent for the above identified patent application, and  
 all the attorneys/agents of record.

the attorneys/agents (with registration numbers) listed on the attached paper(s), or

the attorneys/agents associated with Customer Number

NOTE: This box can only be checked when the power of attorney of record in the application is to all the practitioners associated with a customer number.

The reasons for this request are: **Disengaging due to lack of payment.**


**CORRESPONDENCE ADDRESS**

1.  The correspondence address is NOT affected by this withdrawal.

2.  Change the correspondence address and direct all future correspondence to:

The address associated with Customer Number:

**OR**

<input checked="" type="checkbox"/> Firm or Individual Name	Michael De Angelo, Pattern Intelligence, Inc.		
Address	6796 Giovanetti Road		
City	Forestville	State CA	Zip 95472
Country	U.S.A.		
Telephone	Direct: (760) 799-0379	Email ab9ac99@yahoo.com	
Signature			Registration No. 33,885
Name	William S. Galliani	Telephone No. 650-843-5000	
Date	May 8, 2008		

*NOTE: Withdrawal is effective when approved rather than when received. Unless there are at least 30 days between approval of withdrawal and the expiration date of a time period for response or possible extension period, the request to withdraw is normally disapproved.*

This collection of information is required by 37 CFR 1.36. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

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

## Electronic Acknowledgement Receipt

<b>EFS ID:</b>	3277585
<b>Application Number:</b>	11280700
<b>International Application Number:</b>	
<b>Confirmation Number:</b>	9680
<b>Title of Invention:</b>	System and method for creating and manipulating information containers with dynamic registers
<b>First Named Inventor/Applicant Name:</b>	Michael De Angelo
<b>Customer Number:</b>	26181
<b>Filer:</b>	William S. Galliani/Gina Luna
<b>Filer Authorized By:</b>	William S. Galliani
<b>Attorney Docket Number:</b>	17776-002US4
<b>Receipt Date:</b>	08-MAY-2008
<b>Filing Date:</b>	14-NOV-2005
<b>Time Stamp:</b>	20:08:22
<b>Application Type:</b>	Utility under 35 USC 111(a)

### Payment information:

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

### File Listing:

Document Number	Document Description	File Name	File Size(Bytes) /Message Digest	Multi Part /.zip	Pages (if appl.)
1	Change of Address	Request_For_Withdrawal.pdf	141917 <small>8067042c6f967c6fd65a3157004ae56d ea06eb19</small>	no	1

### Warnings:

### Information:

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

**New Applications Under 35 U.S.C. 111**

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

**National Stage of an International Application under 35 U.S.C. 371**

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

**New International Application Filed with the USPTO as a Receiving Office**

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

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

Appl. No. : 11/280,700 Confirmation No.: 9680  
Applicant : Michael DeAngelo  
Filing Date : November 14, 2005  
Title : SYSTEM AND METHOD FOR CREATING AND MANIPULATING INFORMATION  
CONTAINERS WITH DYNAMIC REGISTERS  
Group Art Unit : 2161  
Examiner : Cam Linh T Nguyen  
Docket No. : 20933-4003  
Customer No. : 34313

Commissioner for Patents  
Mail Stop Amendment  
PO Box 1450  
Alexandria, VA 22313-1450

**AMENDMENT AND RESPONSE TO RESTRICTION REQUIREMENT**

Sir:

In response to the Office Action dated March 17, 2008, please amend the above-identified application as follows:

Recitation of the pending claims is reflected in the listing of claims, which begins on page 2 of this paper.

Remarks begin on page 7 of this paper.

Applicant : Michael DeAngelo  
Appl. No. : 11/280,700  
Examiner : Cam Linh T Nguyen  
Docket No. : 20933-4003

### AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions and listings of claims in the application:

1. (Original) A method comprising:  
receiving a search query;  
searching container registers encapsulated and logically defined in a plurality of containers to identify one or more containers responsive to the search query; and  
providing a list characterizing the identified containers.
2. (Original) A method as in claim 1, when the received search query comprises a labeled data tree having at least one parent-child relationship.
3. (Original) A method as in claim 1, further comprising: providing information identifying containers that have previously been used to respond to one or more processed queries that are substantially similar to the search query.
4. (Original) A method as in claim 1, wherein the provided information is stored in one or more search templates.
5. (Original) A method as in claim 1, further comprising: providing information identifying substantially similar search phrases, search templates, or labeled data trees that have previously been used to respond to one or more processed queries that are substantially similar to the search query.
6. (Original) A method as in claim 5, further comprising:  
receiving a selection of one of the substantially similar search phrases; and  
providing a list of previously identified containers associated with the selected search phrase.
7. (Original) A method as in claim 1, wherein the list provides a title of each identified

Applicant : Michael DeAngelo  
Appl. No. : 11/280,700  
Examiner : Cam Linh T Nguyen  
Docket No. : 20933-4003

container and a short description of its contents.

8. (Original) A method as in claim 1, further comprising: receiving a container search level parameter; and wherein the searching content and container registers only searches within container levels associated with the container search level parameter.

9. (Original) A method as in claim 1, further comprising: receiving a container search level parameter; and wherein the list of identified containers only comprises containers associated with the container search level parameter.

10. (Original) A method as in claim 1, wherein the searching further comprises: encapsulating the search query into a search container.

11. (Original) A method as in claim 10, wherein the searching further comprises:  
receiving, by a gateway, the search container;  
storing, by the gateway, data contained within a register of the search container; and  
determining whether any registers of containers accessible via the gateway are associated with the register of the search container.

12. (Original) A method as in claim 11, further comprising:  
generating a new gateway; and  
associating the container with the new gateway.

13. (Original) A method as in claim 11, further comprising: periodically aggregating the contents of registers in a plurality of gateways to characterize a plurality of containers coupled thereto.

14. (Original) A method as in claim 11, wherein the contents of the registers in each of the plurality of gateways comprise at least one metric chosen from a group comprising: frequency of access of the gateway, grade of access of the gateway, description of users that have accessed the

Applicant : Michael DeAngelo  
Appl. No. : 11/280,700  
Examiner : Cam Linh T Nguyen  
Docket No. : 20933-4003

gateway, an identity of containers that have accessed the gateway, parameters associated with the gateway register, and historically accumulated register data.

15. (Original) A method as in claim 11, further comprising: monitoring transactions involving one or more gateways or containers.

16. (Original) A method as in claim 15, further comprising: generating new containers based on the monitored transactions.

17. (Original) A method as in claim 15, wherein the transactions are based on each instance a gateway or container passes through another gateway or container.

18. (Original) A method comprising:

receiving a search query;

polling a plurality of gateways to identify registers encapsulated therein, the registers relating to one or more containers logically defining data contained therein associated with the search query, wherein the containers are coupled to the gateways; and

providing a list characterizing the identified containers.

19-20 (Canceled)

21. (Original) A computer program product, tangibly embodied on computer-readable media, comprising instructions operable to cause a data processing apparatus to:

receive a search query;

search content and container registers encapsulated and logically defined in a plurality of containers to identify one or more containers associated with the search query; and

Applicant : Michael DeAngelo  
Appl. No. : 11/280,700  
Examiner : Cam Linh T Nguyen  
Docket No. : 20933-4003

provide a list characterizing the identified containers.

22. (Original) A computer program product, tangibly embodied on computer-readable media, comprising instructions operable to cause a data processing apparatus to:

receive a search query;

poll a plurality of gateways to identify registers encapsulated therein, the registers relating to one or more containers logically defining data contained therein associated with the search query, wherein the containers are coupled to the gateways; and

provide a list characterizing the identified containers.

23. (Original) An apparatus comprising:

means for receiving a search query;

means for searching content and container registers encapsulated and logically defined in a plurality of containers to identify one or more containers associated with the search query; and

means for providing a list characterizing the identified containers.

24. (Original) An apparatus comprising:

means for receiving a search query;

means for polling a plurality of gateways to identify registers encapsulated therein, the registers relating to one or more containers logically defining data contained therein associated with the search query, wherein the containers are coupled to the gateways; and

means for providing a list characterizing the identified containers.

25-29. (Canceled)

30. (Original) A method comprising:

receiving a search query;



Applicant : Michael DeAngelo  
Appl. No. : 11/280,700  
Examiner : Cam Linh T Nguyen  
Docket No. : 20933-4003

searching container registers encapsulated and logically defined in a plurality of containers to identify one or more search query templates; and

providing a list characterizing the identified one or more search query templates to formulate subsequent search queries.

Applicant : Michael DeAngelo  
Appl. No. : 11/280,700  
Examiner : Cam Linh T Nguyen  
Docket No. : 20933-4003

**REMARKS**

**Election/Restriction**

Claims 1-30 are pending in this application. Applicant has been requested to select one of the distinct species of the claimed invention:

Group I, drawn to a method for searching data (claims 1-18, 21-24, and 30); and

Group II, drawn to a method for update database (claims 19-20 and 25-29).

Applicant hereby elects to prosecute, without traverse, claims 1-18, 21-24 and 30 (Group I).

Based on this election, please cancel claims 19-20, 25-29 without prejudice.

Applicant hereby submits in compliance with 37 C.F.R. §1.48(b) that there has been no change of inventorship by electing claims 1-18, 21-24 and 30 and canceling claims 19-20, 25-29.

An action on the merits with regard to the elected claims is respectfully requested.

Should the Examiner have any questions or comments on the application, the Examiner should feel free to contact the undersigned via telephone.

Applicant : Michael DeAngelo  
Appl. No. : 11/280,700  
Examiner : Cam Linh T Nguyen  
Docket No. : 20933-4003

The Commissioner is authorized to charge Orrick, Herrington & Sutcliffe LLP's Deposit Account No. 15-0665 for any fees required under 37 C.F.R. §§ 1.16 and 1.17 that are not covered, in whole or in part, by a check enclosed herewith and to credit any overpayments to said Deposit Account 15-0665.

Respectfully submitted,  
ORRICK, HERRINGTON & SUTCLIFFE LLP

Dated: August 7, 2008

By: Sanjeet Dutta  
Sanjeet K. Dutta  
Reg. No. 46,145

Orrick, Herrington & Sutcliffe LLP  
4 Park Plaza, Suite 1600  
Irvine, CA 92614-2558  
Tel. (650)614-7647  
Fax: (650) 614-7401

## Electronic Patent Application Fee Transmittal

<b>Application Number:</b>	11280700			
<b>Filing Date:</b>	14-Nov-2005			
<b>Title of Invention:</b>	System and method for creating and manipulating information containers with dynamic registers			
First Named Inventor/Applicant Name:	Michael De Angelo			
<b>Filer:</b>	Jeffrey A. Miller/Rita Hernandez			
<b>Attorney Docket Number:</b>	17776-002US4			
Filed as Small Entity				
<b>Utility Filing Fees</b>				
<b>Description</b>	<b>Fee Code</b>	<b>Quantity</b>	<b>Amount</b>	<b>Sub-Total in USD(\$)</b>
<b>Basic Filing:</b>				
<b>Pages:</b>				
<b>Claims:</b>				
<b>Miscellaneous-Filing:</b>				
<b>Petition:</b>				
<b>Patent-Appeals-and-Interference:</b>				
Post-Allowance-and-Post-Issuance:				
<b>Extension-of-Time:</b>				
Extension - 4 months with \$0 paid	2254	1	820	820

Description	Fee Code	Quantity	Amount	Sub-Total in USD(\$)
<b>Miscellaneous:</b>				
<b>Total in USD (\$)</b>				<b>820</b>

## Electronic Acknowledgement Receipt

<b>EFS ID:</b>	3743937
<b>Application Number:</b>	11280700
<b>International Application Number:</b>	
<b>Confirmation Number:</b>	9680
<b>Title of Invention:</b>	System and method for creating and manipulating information containers with dynamic registers
<b>First Named Inventor/Applicant Name:</b>	Michael De Angelo
<b>Customer Number:</b>	26181
<b>Filer:</b>	Jeffrey A. Miller/Rita Hernandez
<b>Filer Authorized By:</b>	Jeffrey A. Miller
<b>Attorney Docket Number:</b>	17776-002US4
<b>Receipt Date:</b>	07-AUG-2008
<b>Filing Date:</b>	14-NOV-2005
<b>Time Stamp:</b>	13:29:42
<b>Application Type:</b>	Utility under 35 USC 111(a)

### Payment information:

Submitted with Payment	yes
Payment Type	Deposit Account
Payment was successfully received in RAM	\$ 820
RAM confirmation Number	9905
Deposit Account	150665
Authorized User	

The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows:

Charge any Additional Fees required under 37 C.F.R. Section 1.16 (National application filing, search, and examination fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.17 (Patent application and reexamination processing fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.19 (Document supply fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.21 (Miscellaneous fees and charges)

### File Listing:

Document Number	Document Description	File Name	File Size(Bytes) /Message Digest	Multi Part /.zip	Pages (if appl.)
1	Power of Attorney	Power4003.pdf	47227 7da0d0d266b9e393de6370dd4476d85 ec84efd6	no	1

### Warnings:

### Information:

2		4003Resp_to_RR_08_07_08.pdf	285465 7f95d93745f34460e4bac682a071b43c ca3d2cfe	yes	10
---	--	-----------------------------	--	-----	----

### Multipart Description/PDF files in .zip description

Document Description	Start	End
Extension of Time	1	2
Response to Election / Restriction Filed	3	10

### Warnings:

### Information:

3	Fee Worksheet (PTO-06)	fee-info.pdf	8188 f838c6cc59a65cbd14ff91c3b488348aa aa91802	no	2
---	------------------------	--------------	--	----	---

### Warnings:

### Information:

**Total Files Size (in bytes):** 340880

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

#### New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

#### National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

#### New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

<b>REVOCATION OF POWER OF ATTORNEY WITH NEW POWER OF ATTORNEY, AND STATEMENT UNDER 37 CFR 3.73(b)</b>	<b>Application Number</b>	11/280,700
	<b>Filing Date</b>	November 14, 2005
	<b>First Named Inventor</b>	Michael De Angelo
	<b>Group Art Unit</b>	2161
	<b>Examiner Name</b>	Cam Linh T Nguyen
	<b>Confirmation No.</b>	9680
	<b>Attorney Dkt Number</b>	20933-4003

I hereby revoke all previous powers of attorney or authorizations of agent given in the above-identified application.

I hereby appoint the practitioners at 34313  
Customer Number

as my/our attorney(s) or agent(s) to prosecute the application identified above, and to transact all business in the United States Patent and Trademark Office connected therewith.

Please change the correspondence address for the above-identified application to the above-mentioned Customer Number:

I am the Assignee of record of the entire interest. See 37 CFR 3.71.

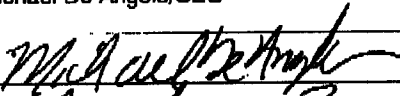
The following statement is made under 37 CFR 3.73(b):

Pattern Intelligence, a corporation, states that it is the assignee of the entire right, title, and interest in the above identified patent application/patent by virtue of an assignment from the inventor(s) of the patent application/patent identified above.

1. From: Michael De Angelo To: Pattern Intelligence, Inc. The document was recorded on February 16, 2006 in the United States Patent and Trademark Office at Reel 017269 Frame 0568, or for which a copy thereof is attached.  
And

The undersigned (whose title is supplied below) is authorized to act on behalf of the assignee.

**SIGNATURE of Applicant or Assignee of Record**

Name	Michael De Angelo/CEO
Signature	
Date	August 2008

NOTE: Signatures of all the inventor(s) or assignees of record of the entire interest or their representative(s) are required. Submit multiple forms if more than one signature is required, see below\*.

\*Total of 1 forms are submitted.



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

Appl. No. : 11/280,700 Confirmation No.: 9680  
 Applicant : Michael DeAngelo  
 Filing Date : November 14, 2005  
 Title : SYSTEM AND METHOD FOR CREATING AND MANIPULATING INFORMATION  
 CONTAINERS WITH DYNAMIC REGISTERS  
 Group Art Unit : 2161  
 Examiner : Cam Linh T Nguyen  
 Docket No. : 20933-4003  
 Customer No. : 34313

Commissioner for Patents  
 Mail Stop Amendment  
 PO Box 1450  
 Alexandria, VA 22313-1450

REQUEST FOR EXTENSION OF TIME

This is a request under the provisions of 37 CFR § 1.136(a) to extend the period for filing a reply in the above-identified application.

The requested extension and fees are as followings:

EXTENSION (months)	FEE FOR SMALL ENTITY	FEE FOR OTHER THAN SMALL ENTITY
<input type="checkbox"/> one month	\$60.00	\$120.00
<input type="checkbox"/> two months	\$230.00	460.00
<input type="checkbox"/> three months	\$525.00	\$1,050.00
<input checked="" type="checkbox"/> four months	\$820.00	\$1,640.00
<input type="checkbox"/> five months	\$1,115.00	\$2,230.00
	<b>Fee</b>	<b>\$820.00</b>

Applicant claims small entity status. See 37 CFR 1.27.

Applicant : Michael DeAngelo  
Appl. No. : 11/280,700  
Examiner : Cam Linh T Nguyen  
Docket No. : 20933-4003

Method of Payment of fees:

- A check in the amount of \_\_\_\_\_ is enclosed to cover the above fee(s).
- Charge Orrick's Deposit Account No. **15-0665** in the amount of **\$820.00**
- The Commissioner is authorized to charge Orrick's Deposit Account No. **15-0665** for any fees required under 37 CFR §§ 1.16, 1.17 and 1.445 that are not covered, in whole or in part, by a check enclosed herewith and to credit any overpayments to said Deposit Account No. **15-0665**.

I am the

- applicant/inventor
- assignee of record of the entire interest. See 38 CFR 3.71. Statement under 37 CFR 3.373(b) is enclosed.
- Attorney or agent of record. Registration No. 46,145.
- Attorney or agent under 37 CFR 1.34. Registration No. \_\_\_\_\_.

Respectfully submitted,  
ORRICK, HERRINGTON & SUTCLIFFE LLP

Dated: August 7, 2008

By: Sanjeet Dutta  
Sanjeet K. Dutta  
Reg. No. 46,145

Four Park Plaza, Suite 1600  
Irvine, California 92614-2558  
(650) 614-7660



COOLEY GODWARD KRONISH LLP  
ATTN: PATENT GROUP, SUITE 1100  
777 - 6<sup>TH</sup> STREET, NW  
WASHINGTON DC 20001

COPY MAILED

SEP 17 2008

In re Application of	:	
Michael DEANGELO	:	
Application No. 11/280,700	:	DECISION ON PETITION
Filed: November 14, 2005	:	TO WITHDRAW
Attorney Docket No. 17776-002US4	:	FROM RECORD

This is a decision on the Request to Withdraw as attorney or agent under 37 C.F.R. § 1.36(b) filed May 8, 2008.

The request is **NOT APPROVED**.

A review of the file record indicates that William S. Galliani does not have power of attorney in this patent application. See 37 C.F.R. § 10.40. Accordingly, the request to withdraw under 37 C.F.R. § 1.36(b) is not applicable.

All future communications from the Office will continue to be directed to the below-listed address until otherwise properly notified by the applicant.

Telephone inquires concerning this decision should be directed to the undersigned at 571-272-6735.

Diane Goodwyn  
Petitions Examiner  
Office of Petitions

cc: FISH & RICHARDSON PC  
PO BOX 1022  
MINNEAPOLIS MN 55440-1022



UNITED STATES PATENT AND TRADEMARK OFFICE

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

Table with 5 columns: APPLICATION NO., FILING DATE, FIRST NAMED INVENTOR, ATTORNEY DOCKET NO., CONFIRMATION NO.
11/280,700 11/14/2005 Michael De Angelo 17776-002US4 9680

34313 7590 10/07/2008
ORRICK, HERRINGTON & SUTCLIFFE, LLP
IP PROSECUTION DEPARTMENT
4 PARK PLAZA
SUITE 1600
IRVINE, CA 92614-2558

EXAMINER

NGUYEN, CAM LINH T

ART UNIT PAPER NUMBER

2161

MAIL DATE DELIVERY MODE

10/07/2008

PAPER

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

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



Art Unit: 2161

### **DETAILED ACTION**

1. This Office Action is responsive to communication filed on 08/07/2008.
2. Claims 1 – 18, 21 – 24, and 30 are currently pending.

### ***Election/Restrictions***

3. Applicant's election without traverse of group I (claims 1 – 18, 21 – 24, and 30) in the reply filed on 08/07/2008 is acknowledged.
4. Claims 19, 20, 25 - 29 withdrawn from further consideration pursuant to 37 CFR 1.142(b) as being drawn to a nonelected species, there being no allowable generic or linking claim. Election was made **without** traverse in the reply filed on 08/07/2008.

### ***Information Disclosure Statement***

5. The information disclosure statement (IDS) submitted on 05/03/2006 is in compliance with the provisions of 37 CFR 1.97. Accordingly, the information disclosure statement is being considered by the examiner.

### ***Claim Rejections - 35 USC § 102***

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

A person shall be entitled to a patent unless –

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

Art Unit: 2161

7. Claims 1 – 2, 4, 7 - 18, 21 – 24, and 30 are rejected under 35 U.S.C. 102(e) as being anticipated by Donohue et al (U.S. 5,987,480).

◆As per claims 1, 21, 23, 30,

Donohue discloses a method, system comprising:

- “Receiving a search query” corresponds to the step of receiving the URL request (Fig. 3A, step 48, col. 10, lines 31 – 32).
- “Searching container registers encapsulated and logically defined in a plurality of containers to identify one or more containers responsive to the search query” corresponds to the step of searching for the name and value pair and the corresponding template using the information in the URL (col. 10, lines 32 – 42). The name and value pair and the template in this case correspond to the “container register” (col. 5, lines 26 - 35).
- “Providing a list characterizing the identified containers” corresponds to the step of delivery the document (Fig. 3C, step 82, col. 13, lines 11 – 14).

◆As per claim 2,

- “The received search query comprises a labeled data tree having at least one parent-child relationship” See col.5, lines 28 – 29 where the “labeled data tree” corresponds to the hierarchical directory structure.

◆As per claim 4,

- “The provided information is stored in one or more search templates” See Fig. 3A- 3C.

◆As per claim 7,

- “The list provides a title of each identified container and a short description of its contents” See col. 13, lines 11 – 21.

Art Unit: 2161

◆As per claims 8 - 9,

- Receiving a container search level parameter; and wherein the searching content and container registers only searches within container levels associated with the container search level parameter” See col. 13, lines 22 – 26.

◆As per claim 10,

- “Encapsulating the search query into a search container” See col. 8, lines 3 – 9.

◆As per claims 11 - 17,

- “receiving, by a gateway, the search container; storing, by the gateway, data contained within a register of the search container; and determining whether any registers of containers accessible via the gateway are associated with the register of the search container” See col. 9, lines 54 – col. 10, lines 3.

◆As per claims 18, 22, 24,

Claims 18, 22, 24, are rejected based on the rejection of claims 1 and 11.

### ***Claim Rejections - 35 USC § 103***

8. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all

obviousness rejections set forth in this Office action:

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

9. Claims 3, 5 – 6 are rejected under 35 U.S.C. 103(a) as being unpatentable over Donohue et al (U.S. 5,987,480) in view of Walker et al (U.S. 5,862,223).

◆As per claims 3, 5 – 6,



Art Unit: 2161

Donohue discloses a method for searching document using name value pair and the template to populate the document. Donohue does not clearly disclose that “Providing information identifying containers that have previously been used to respond to one or more processed queries that are substantially similar to the search query” or “providing information identifying substantially similar search phrases, search templates, or labeled data trees that have previously been used to respond to one or more processed queries that are substantially similar to the search query” and “receiving a selection of one of the substantially similar search phrases; and providing a list of previously identified containers associated with the selected search phrase”.

However, storing and reusing the history transaction is well known in the art. Walker provided an example. Walker teaches an expertise system and method comprising the teaching of storing user request and corresponding expert answer in database (Fig. 2, 265, 270). When a new request comes, the system will search for duplicate requests (Fig. 7, step 710), and provide the user with the corresponding answer (Fig. 7, step 720, 740). These teachings are similar to the instant claim invention.

It would have been obvious to one with ordinary skill in the art at the time the invention was made to apply the teaching of Walker into the invention of Donohue sine both inventions were available and the combination would reduce the time searching for information and improve the system performance.

### ***Conclusion***

10. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Art Unit: 2161

- Marc Krellenstein (U.S. 5,924,090) discloses a method and apparatus for searching a database of records.
- Ito et al (U.S. 6,725,251) discloses a local file transfer method and system.
- Balogh et al (U.S. 5,493,677) discloses method for retrieving of digital images with evoked suggestion set captions and natural language interface.

11. Any inquiry concerning this communication or earlier communications from the examiner should be directed to CamLinh Nguyen whose telephone number is (571) 272 - 4024.

The examiner can normally be reached on Monday-Friday.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Apu Mofiz can be reached on (571) 272 - 4080. The fax phone number for the organization where this application or proceeding is assigned is 571 - 273 - 8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/CamLinh Nguyen/  
Primary Examiner, Art Unit 2161

<b>Notice of References Cited</b>	Application/Control No. 11/280,700	Applicant(s)/Patent Under Reexamination ANGELO, MICHAEL DE	
	Examiner CAM-LINH NGUYEN	Art Unit 2161	Page 1 of 1

**U.S. PATENT DOCUMENTS**

*	Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
*	A US-5,987,480 A	11-1999	Donohue et al.	715/207
*	B US-5,862,223 A	01-1999	Walker et al.	705/50
*	C US-5,924,090 A	07-1999	Krellenstein, Marc F.	707/5
*	D US-6,725,251 B2	04-2004	Ito et al.	709/203
*	E US-5,493,677 A	02-1996	Balogh et al.	707/104.1
	F US-			
	G US-			
	H US-			
	I US-			
	J US-			
	K US-			
	L US-			
	M US-			


**FOREIGN PATENT DOCUMENTS**

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

**NON-PATENT DOCUMENTS**

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

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

<b>Index of Claims</b>  	<b>Application/Control No.</b>  11280700	<b>Applicant(s)/Patent Under Reexamination</b>  ANGELO, MICHAEL DE
	<b>Examiner</b>  CAM-LINH NGUYEN	<b>Art Unit</b>  2161

✓	<b>Rejected</b>
=	<b>Allowed</b>


-	<b>Cancelled</b>
÷	<b>Restricted</b>

N	<b>Non-Elected</b>
I	<b>Interference</b>

A	<b>Appeal</b>
O	<b>Objected</b>

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

CLAIM		DATE							
Final	Original	10/06/2008							
	1	✓							
	2	✓							
	3	✓							
	4	✓							
	5	✓							
	6	✓							
	7	✓							
	8	✓							
	9	✓							
	10	✓							
	11	✓							
	12	✓							
	13	✓							
	14	✓							
	15	✓							
	16	✓							
	17	✓							
	18	✓							
	19	-							
	20	-							
	21	✓							
	22	✓							
	23	✓							
	24	✓							
	25	-							
	26	-							
	27	-							
	28	-							
	29	-							
	30	✓							

<b>Search Notes</b>  	<b>Application/Control No.</b>  11280700	<b>Applicant(s)/Patent Under Reexamination</b>  ANGELO, MICHAEL DE
	<b>Examiner</b>  CAM-LINH NGUYEN	<b>Art Unit</b>  2161

<b>SEARCHED</b>			
<b>Class</b>	<b>Subclass</b>	<b>Date</b>	<b>Examiner</b>

<b>SEARCH NOTES</b>		
<b>Search Notes</b>	<b>Date</b>	<b>Examiner</b>
EAST search	10/4/08	Linh

<b>INTERFERENCE SEARCH</b>			
<b>Class</b>	<b>Subclass</b>	<b>Date</b>	<b>Examiner</b>

--	--



UNITED STATES PATENT AND TRADEMARK OFFICE

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

BIB DATA SHEET

CONFIRMATION NO. 9680

<b>SERIAL NUMBER</b> 11/280,700	<b>FILING or 371(c) DATE</b> 11/14/2005 <b>RULE</b>	<b>CLASS</b> 707	<b>GROUP ART UNIT</b> 2161	<b>ATTORNEY DOCKET NO.</b> 17776-002US4	
<b>APPLICANTS</b> Michael De Angelo, Palm Springs, CA; <b>** CONTINUING DATA *****</b> This application is a CON of 09/284,113 04/07/1999 PAT 7,010,536 which is a 371 of PCT/US99/01988 01/28/1999 which claims benefit of 60/073,209 01/30/1998 <b>** FOREIGN APPLICATIONS *****</b> <b>** IF REQUIRED, FOREIGN FILING LICENSE GRANTED ** ** SMALL ENTITY **</b> 12/12/2005					
Foreign Priority claimed <input type="checkbox"/> Yes <input checked="" type="checkbox"/> No 35 USC 119(a-d) conditions met <input type="checkbox"/> Yes <input checked="" type="checkbox"/> No Verified and /CAM-LINH T NGUYEN/ Acknowledged Examiner's Signature	<input type="checkbox"/> Met after Allowance LN Initials	<b>STATE OR COUNTRY</b> CA	<b>SHEETS DRAWINGS</b> 30	<b>TOTAL CLAIMS</b> 30 <sup>23</sup>	<b>INDEPENDENT CLAIMS</b> 12 <sup>7</sup>
<b>ADDRESS</b> ORRICK, HERRINGTON & SUTCLIFFE, LLP IP PROSECUTION DEPARTMENT 4 PARK PLAZA SUITE 1600 IRVINE, CA 92614-2558 UNITED STATES					
<b>TITLE</b> System and method for creating and manipulating information containers with dynamic registers					
<b>FILING FEE RECEIVED</b> 1715	FEES: Authority has been given in Paper No. _____ to charge/credit DEPOSIT ACCOUNT No. _____ for following:		<input type="checkbox"/> All Fees <input type="checkbox"/> 1.16 Fees (Filing) <input type="checkbox"/> 1.17 Fees (Processing Ext. of time) <input type="checkbox"/> 1.18 Fees (Issue) <input type="checkbox"/> Other _____ <input type="checkbox"/> Credit		

## EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	19122	gateway same (register\$3 or subscrib\$4)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 20:43
L2	20262	gate\$way\$1 same (register\$3 or subscrib\$4)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 20:43
L3	1853	gate\$way\$1 same (register\$3 or subscrib\$4) same (container\$1 or folder\$1 or file\$1 or object\$1)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 20:44
L4	829	search\$3 near4 (register\$3 or subscrib\$4) near4 (container\$1 or folder\$1 or file\$1 or object\$1)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 20:44
L5	22	3 and 4	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 20:44
L6	2	5 and @AD<"19980130"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 20:45
L7	208	(search\$3 near4 (register\$3 or subscrib\$4) near4 (container\$1 or folder\$1 or file\$1 or object\$1)).ab.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 20:46
L8	71	(search\$3 near4 (register\$3 or subscrib\$4) near4 (container\$1 or folder\$1 or file\$1 or object\$1)).clm.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 20:46
L9	10	7 and 8	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 20:46
L10	1	9 and @AD<"19980130"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 20:46
L11	5421	(search\$3 near4 (container\$1 or folder\$1 or file\$1 or object\$1)).clm.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 20:47

L12	11680	(search\$3 near4 (container\$1 or folder\$1 or file\$1 or object \$1)).ab.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 20:47
L13	1352	11 and 12	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 20:47
L14	69	4 and 13	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 20:47
L15	13	14 and @AD<"19980130"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 20:48
L16	94	search\$3 near3 query near3 templat\$3	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 20:56
L17	83	16 and (container\$1 or object \$1 or file\$1 or folder\$1)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 20:56
L18	4	16 and ((container\$1 or object\$1 or file\$1 or folder \$1) near3 register\$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 20:57
L19	1	16 and ((container\$1 or folder\$1) near3 register\$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 20:57
L20	26	16 and (container\$1 or folder \$1)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 20:58
L21	0	20 and @AD<"19980130"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 20:58
L22	36	16 and register\$3	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 20:58
L23	0	22 and @AD<"19980130"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 20:58



L24	4	16 and @AD<"19980130"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 20:58
L25	4121	search\$3 near5 templat\$3	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 21:01
L26	506	25 and container\$1	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 21:01
L27	242	26 and register\$3	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 21:01
L28	9	27 and @AD<"19980130"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 21:02
L29	9	28 not 24	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 21:02
L30	2290	(expert\$3 same system).ti.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 21:09
L31	4	30 and walker.in.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 21:09
L32	4	31 and templat\$3	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 21:11
L33	639	(search\$3 near5 templat\$3). ab.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 21:17
L34	434	(search\$3 near5 templat\$3). clm.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 21:17
L35	102	33 and 34	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 21:17

L36	12	35 and @AD<"19980130"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 21:18
L37	3951	search\$3 near5 (meta\$data or metadata)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 21:21
L38	26509	descript\$3 near3 content\$1	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 21:22
L39	772	37 and 38	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 21:22
L40	32	39 and 25	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 21:22
L41	0	40 and @AD<"19980130"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 21:22
L42	0	39 and tempalt\$3	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 21:23
L43	240	39 and templat\$3	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 21:23
L44	181	43 and register\$3	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 21:23
L45	0	44 and @AD<"19980130"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 21:23
L46	718	(search\$3 near5 (meta\$data or metadata)).ab.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 21:23
L47	681	(search\$3 near5 (meta\$data or metadata)).clm.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 21:23

L48	16	43 and 46	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 21:24
L49	0	48 and @AD<"19980130"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 21:24
L50	195	37 and 46 and 47	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 21:24
L51	1	50 and @AD<"19980130"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 21:25
L52	1	39 and @AD<"19980130"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 21:28
L53	1204	37 and (46 or 47)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 21:29
L54	9	53 and @AD<"19980130"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2008/10/03 21:29

10/3/08 9:45:16 PM

C:\Documents and Settings\CNguyen11\My Documents\EASTWorkspaces\11095443.wsp

JPW



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant : Michael De Angelo  
Serial No. : 11/280,700  
Filed : November 14, 2005  
Title : SYSTEM AND METHOD FOR CREATING AND MANIPULATING INFORMATION CONTAINERS WITH DYNAMIC REGISTER

Art Unit : 2161  
Examiner : Unknown

**MAIL STOP AMENDMENT**

Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

INFORMATION DISCLOSURE STATEMENT

Applicants request consideration of the references listed on the attached PTO-1449 form. Under 37 C.F.R. § 1.98 (a)(2)(ii), only copies of foreign patent documents and/or non-patent literature are enclosed. Copies of any listed U.S. patents or U.S. patent application publications can be provided upon request.

This statement is being filed within three months of the filing date of the application or before the receipt of a first Office Action on the merits. Please apply any charges or credits to Deposit Account No. 06-1050.

Respectfully submitted,

Carl A. Kukkonen, III  
Reg. No. 42,773

Date: May 1, 2006

Fish & Richardson P.C.  
12390 El Camino Real  
San Diego, California 92130  
Telephone: (858) 678-5070  
Facsimile: (858) 678-5099  
10625442.doc

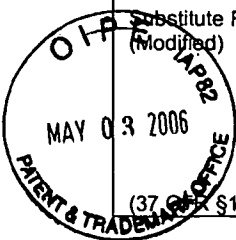
CERTIFICATE OF MAILING BY FIRST CLASS MAIL

I hereby certify under 37 CFR §1.8(a) that this correspondence is being deposited with the United States Postal Service as first class mail with sufficient postage on the date indicated below and is addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

May 1, 2006  
Date of Deposit

Teresa Salazar-Fischer  
Signature

Teresa Salazar-Fischer  
Typed or Printed Name of Person Signing Certificate



Substitute Form PTO-1449 (Modified) <b>Information Disclosure Statement                  by Applicant</b> (Use several sheets if necessary) (37 C.F.R. § 1.98(b))	U.S. Department of Commerce Patent and Trademark Office	Attorney's Docket No. 17776-002US4	Application No. 11/280,700
		Applicant Michael De Angelo	
		Filing Date November 14, 2005	Group Art Unit 2161

U.S. Patent Documents							
Examiner Initial	Desig. ID	Document Number	Publication Date	Patentee	Class	Subclass	Filing Date If Appropriate
/LN/	AA	5,664,208	09/02/97	Pavley, et al.			
/LN/	AB	5,768,510	06/16/98	Gish			
/LN/	AC	5,815,665A	09/98	Teper, et al.			
/LN/	AD	5,848,246	12/08/98	Gish			
/LN/	AE	6,016,495	01/00	Mc Keehan, et al.			
/LN/	AF	6,075,791 A	06/00	Chiussi, et al.			
/LN/	AG	6,154,782 A	11/00	Kawaguchi, et al.			
/LN/	AH	6,173,280 B1	01/01	Ramkumar, et al.			
/LN/	AI	6,198,738B1	03/01	Chang, et al.			
/LN/	AJ	6,351,745	02/02	Itakura, et al.			

Foreign Patent Documents or Published Foreign Patent Applications								
Examiner Initial	Desig. ID	Document Number	Publication Date	Country or Patent Office	Class	Subclass	Translation	
							Yes	No
/LN/	AK	WO 98/02831	01/22/98	PCT				
/LN/	AL	WO 99/39285	08/05/99	PCT				

Other Documents (include Author, Title, Date, and Place of Publication)		
Examiner Initial	Desig. ID	Document
	AM	none

Examiner Signature /Cam Linh Nguyen/	Date Considered 10/05/2008
EXAMINER: Initials citation considered. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.	



UNITED STATES PATENT AND TRADEMARK OFFICE

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

APPLICATION NUMBER	FILING OR 371(C) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
11/280,700	11/14/2005	Michael De Angelo	17776-002US4

**CONFIRMATION NO. 9680**

**POA ACCEPTANCE LETTER**



34313  
ORRICK, HERRINGTON & SUTCLIFFE, LLP  
IP PROSECUTION DEPARTMENT  
4 PARK PLAZA  
SUITE 1600  
IRVINE, CA 92614-2558

Date Mailed: 01/13/2009

**NOTICE OF ACCEPTANCE OF POWER OF ATTORNEY**

This is in response to the Power of Attorney filed 08/07/2008.

The Power of Attorney in this application is accepted. Correspondence in this application will be mailed to the above address as provided by 37 CFR 1.33.

/cyhall/

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



UNITED STATES PATENT AND TRADEMARK OFFICE

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

APPLICATION NUMBER	FILING OR 371(C) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
11/280,700	11/14/2005	Michael De Angelo	17776-002US4

**CONFIRMATION NO. 9680**

**POWER OF ATTORNEY NOTICE**

26181  
FISH & RICHARDSON P.C.  
PO BOX 1022  
MINNEAPOLIS, MN 55440-1022



Date Mailed: 01/13/2009

**NOTICE REGARDING CHANGE OF POWER OF ATTORNEY**

This is in response to the Power of Attorney filed 08/07/2008.

- The Power of Attorney to you in this application has been revoked by the assignee who has intervened as provided by 37 CFR 3.71. Future correspondence will be mailed to the new address of record(37 CFR 1.33).

/eyhall/

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

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

Appl. No. : 11/280,700 Confirmation No.: 9680  
Applicant : Michael De Angelo  
Filing Date : November 14, 2005  
Title : SYSTEM AND METHOD FOR CREATING AND MANIPULATING  
INFORMATION CONTAINERS WITH DYNAMIC REGISTERS  
Group Art Unit : 2161  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003  
Customer No. : 34313

**Mail Stop: Amendment**  
Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

**RESPONSE TO OFFICE ACTION**

Sir:

In response to the Office Action dated October 7, 2008 please amend the above-identified application as follows:

**Amendments to the Claims** are reflected in the listing of claims, which begins on page 2 of this paper.

**Remarks** begin on page 9 of this paper.



Applicant : Michael De Angelo  
Appl. No. : 11/280,700  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003

### **AMENDMENTS TO THE CLAIMS**

This listing of claims will replace all prior versions and listings of claims in the application:

#### **LISTING OF CLAIMS**

1. (Currently Amended) A method comprising:  
receiving a search query;  
searching container registers encapsulated and logically defined in a plurality of  
containers to identify one or more containers responsive to the search query, the  
container registers having defined therein data comprising historical data  
associated with interactions of the one or more containers with other containers  
from the one or more containers; and  
providing a list characterizing the identified containers.
2. (Original) A method as in claim 1, when the received search query comprises  
a labeled data tree having at least one parent-child relationship.
3. (Original) A method as in claim 1, further comprising: providing information  
identifying containers that have previously been used to respond to one or more  
processed queries that are substantially similar to the search query.
4. (Original) A method as in claim 1, wherein the provided information is stored  
in one or more search templates.

Applicant : Michael De Angelo  
Appl. No. : 11/280,700  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003

5. (Original) A method as in claim 1, further comprising: providing information identifying substantially similar search phrases, search templates, or labeled data trees that have previously been used to respond to one or more processed queries that are substantially similar to the search query.
  
6. (Original) A method as in claim 5, further comprising:  
receiving a selection of one of the substantially similar search phrases; and  
providing a list of previously identified containers associated with the selected search phrase.
  
7. (Original) A method as in claim 1, wherein the list provides a title of each identified container and a short description of its contents.
  
8. (Original) A method as in claim 1, further comprising: receiving a container search level parameter; and wherein the searching content and container registers only searches within container levels associated with the container search level parameter.
  
9. (Original) A method as in claim 1, further comprising: receiving a container search level parameter; and wherein the list of identified containers only comprises containers associated with the container search level parameter.

Applicant : Michael De Angelo  
Appl. No. : 11/280,700  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003

10. (Original) A method as in claim 1, wherein the searching further comprises:  
encapsulating the search query into a search container.
  
11. (Original) A method as in claim 10, wherein the searching further comprises:  
receiving, by a gateway, the search container;  
storing, by the gateway, data contained within a register of the search container; and  
determining whether any registers of containers accessible via the gateway are associated  
with the register of the search container.
  
12. (Original) A method as in claim 11, further comprising:  
generating a new gateway; and  
associating the container with the new gateway.
  
13. (Original) A method as in claim 11, further comprising: periodically  
aggregating the contents of registers in a plurality of gateways to characterize a  
plurality of containers coupled thereto.
  
14. (Original) A method as in claim 11, wherein the contents of the registers in  
each of the plurality of gateways comprise at least one metric chosen from a  
group comprising: frequency of access of the gateway, grade of access of the  
gateway, description of users that have accessed the gateway, an identity of  
containers that have accessed the gateway, parameters associated with the  
gateway register, and historically accumulated register data.

Applicant : Michael De Angelo  
Appl. No. : 11/280,700  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003

15. (Original) A method as in claim 11, further comprising: monitoring transactions involving one or more gateways or containers.
16. (Original) A method as in claim 15, further comprising: generating new containers based on the monitored transactions.
17. (Original) A method as in claim 15, wherein the transactions are based on each instance a gateway or container passes through another gateway or container.
18. (Currently Amended) A method comprising:  
receiving a search query;  
polling a plurality of gateways to identify registers encapsulated therein, the registers relating to one or more containers logically defining data contained therein associated with the search query, the one or more containers having container registers defined therein, the container registers containing data comprising historical data associated with interactions of the one or more containers with other containers from the one or more containers, wherein the containers are coupled to the gateways; and  
providing a list characterizing the identified containers.
- 19-20. (Previously Canceled)

Applicant : Michael De Angelo  
Appl. No. : 11/280,700  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003

21. (Currently Amended) A computer program product, tangibly embodied on computer-readable media, comprising instructions operable to cause a data processing apparatus to:

receive a search query;

search content and container registers encapsulated and logically defined in a plurality of containers to identify one or more containers associated with the search query, the container registers having defined therein data comprising historical data associated with interactions of the one or more containers with other containers from the one or more containers; and

provide a list characterizing the identified containers.

22. (Currently Amended) A computer program product, tangibly embodied on computer-readable media, comprising instructions operable to cause a data processing apparatus to:

receive a search query;

poll a plurality of gateways to identify registers encapsulated therein, the registers relating to one or more containers logically defining data contained therein associated with the search query, the one or more containers having container registers defined therein, the container registers containing data comprising historical data associated with interactions of the one or more containers with other containers from the one or more containers, wherein the containers are coupled to the gateways; and

provide a list characterizing the identified containers.

Applicant : Michael De Angelo  
Appl. No. : 11/280,700  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003

23. (Currently Amended) An apparatus comprising:

means for receiving a search query;

means for searching content and container registers encapsulated and logically defined in

a plurality of containers to identify one or more containers associated with the

search query, the container registers having defined therein data comprising

historical data associated with interactions of the one or more containers with

other containers from the one or more containers; and

means for providing a list characterizing the identified containers.

24. (Currently Amended) An apparatus comprising:

means for receiving a search query;

means for polling a plurality of gateways to identify registers encapsulated therein, the

registers relating to one or more containers logically defining data contained

therein associated with the search query, the one or more containers having

container registers defined therein, the container registers containing data

comprising historical data associated with interactions of the one or more

containers with other containers from the one or more containers, wherein the

containers are coupled to the gateways; and

means for providing a list characterizing the identified containers.

25-29. (Previously Canceled)

Applicant : Michael De Angelo  
Appl. No. : 11/280,700  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003

30. (Currently Amended) A method comprising:

receiving a search query;

searching container registers encapsulated and logically defined in a plurality of

containers to identify one or more search query templates, the container registers

having defined therein data comprising historical data associated with interactions

of the one or more containers with other containers from the one or more

containers; and

providing a list characterizing the identified one or more search query templates to

formulate subsequent search queries.

Applicant : Michael De Angelo  
Appl. No. : 11/280,700  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003

### **REMARKS**

Claims 1-18, 21-24, and 30 are pending in the present application.

Claims 1, 2, 4, 7-18, 21-24, and 30 have been rejected under 35 U.S.C. §102(e) as being anticipated by U.S. Patent No. 5,987,480 to Donohue, *et al.* (“Donohue”).

Claims 3, 5-6 have been rejected under 35 U.S.C. §103(a) as being unpatentable over Donohue in view of U.S. Patent No. 5,862,223 to Walker, *et al.* (“Walker”).

Claims 1, 18, 21, 22, 23, 24, and 30 have been amended. It is respectfully submitted that no new matter has been added.

Reconsideration of the application as amended herein is respectfully requested.

### **Rejections under 35 U.S.C. §102(e)**

Claims 1, 2, 4, 7-18, 21-24, and 30 are rejected under 35 U.S.C. §102(e) as being anticipated by Donohue. The applicant respectfully disagrees and submits that Claims 1, 2, 4, 7-18, 21-24, and 30 are not anticipated by Donohue.

Regarding the rejection of independent Claims 1, 21, 23, and 30, the Examiner states:

Donohue discloses a method, system comprising: ...“Searching container registers encapsulated and logically defined in a plurality of containers to identify one or more containers responsive to the search query” corresponds to the step of searching for the name and value pair and the corresponding template using the information in the URL (col. 10, lines 32-42). The name and value pair and the template in this case correspond to the “container register” (col. 5, lines 26-35).

10/07/08 Office Action, page 3. The Examiner also rejects independent Claims 18, 22, and 24 based on the rejection of Claim 1. 10/07/08 Office Action, page 4. The applicant respectfully disagrees for the following reasons.



Applicant : Michael De Angelo  
Appl. No. : 11/280,700  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003

Donohue discloses a system wherein document templates can be searched, retrieved, and populated with customized content for a particular user. Donohue, column 4, lines 14-20. The document templates are stored on a web server in a hierarchical directory structure, and may be retrieved through user search requests. Donohue, column 5, lines 26-35. User data for populating the customized documents is stored in a data source on the web server in a form representing names and values. Donohue, column 5, lines 26-55 and column 10 lines 45-50. A user registers and must log in to search for documents, and once a document is selected a script loads the name and value information for the given user. Donohue, column 10, lines 31-40. A template parsing function then steps through the user data and appropriately populates the document until the end of the template has been reached. Donohue, column 10, lines 51-65.

In contrast, Claim 1 is amended to recite:

**container registers encapsulated and logically defined in a plurality of containers...the container registers having defined therein data comprising historical data associated with interactions...with other containers** from the one or more containers

Claim 1, emphasis added. Because Donohue does not teach or suggest the features as recited in Claim 1, the applicants respectfully submit that Claim 1 and Claims 2-17 which depend from Claim 1 are not anticipated by Donohue under 35 U.S.C. §102(e).

Claim 18 is amended to recite:

**the one or more containers having container registers defined therein, the container registers containing data comprising historical data associated with interactions** of the one or more containers **with other containers** from the one or more containers

Applicant : Michael De Angelo  
Appl. No. : 11/280,700  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003

Claim 18, emphasis added. Similarly, these features of Claim 18 are neither taught nor suggested by Donohue, therefore the applicants respectfully submit that Claim 18 is not anticipated by Donohue under 35 U.S.C. §102(e).

Claim 21 is amended to recite:

**container registers encapsulated and logically defined in a plurality of containers...the container registers having defined therein data comprising historical data associated with interactions...with other containers** from the one or more containers

Claim 21, emphasis added. Similarly, these features of Claim 21 are neither taught nor suggested by Donohue, therefore the applicants respectfully submit that Claim 21 is not anticipated by Donohue under 35 U.S.C. §102(e).

Claim 22 is amended to recite:

**the one or more containers having container registers defined therein, the container registers containing data comprising historical data associated with interactions** of the one or more containers **with other containers** from the one or more containers

Claim 22, emphasis added. Similarly, these features of Claim 22 are neither taught nor suggested by Donohue, therefore the applicants respectfully submit that Claim 22 is not anticipated by Donohue under 35 U.S.C. §102(e).

Claim 23 is amended to recite:

**container registers encapsulated and logically defined in a plurality of containers...the container registers having defined therein data comprising historical data associated with interactions...with other containers** from the one or more containers

Claim 23, emphasis added. Similarly, these features of Claim 23 are neither taught nor suggested by Donohue, therefore the applicants respectfully submit that Claim 23 is not anticipated by Donohue under 35 U.S.C. §102(e).

Applicant : Michael De Angelo  
Appl. No. : 11/280,700  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003

Claim 24 is amended to recite:

**the one or more containers having container registers defined therein, the container registers containing data comprising historical data associated with interactions of the one or more containers with other containers from the one or more containers**

Claim 24, emphasis added. Similarly, these features of Claim 24 are neither taught nor suggested by Donohue, therefore the applicants respectfully submit that Claim 24 is not anticipated by Donohue under 35 U.S.C. §102(e).

Claim 30 is amended to recite:

**container registers encapsulated and logically defined in a plurality of containers...the container registers having defined therein data comprising historical data associated with interactions...with other containers from the one or more containers**

Claim 30, emphasis added. Similarly, these features of Claim 30 are neither taught nor suggested by Donohue, therefore the applicants respectfully submit that Claim 30 is not anticipated by Donohue under 35 U.S.C. §102(e).

**Rejections under 35 U.S.C. §103(a)**

Claims 3, 5-6 are rejected under 35 U.S.C. §103(a) as being unpatentable over Donohue in view of Walker. The applicant respectfully disagrees and submits that Claims 3, 5-6 are patentable over Donohue in view of Walker.

The Examiner states:

Donohue does not clearly disclose that “Providing information identifying containers that have previously been used to respond to one or more processed queries that are substantially similar to the search query”... Walker teaches an expertise system and method comprising the teaching of storing user request and corresponding expert answer in database (Fig. 2, 265, 270).

Applicant : Michael De Angelo  
Appl. No. : 11/280,700  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003

10/07/08 Office Action, page 5. The applicant respectfully disagrees for the following reasons.

Claims 3, 5 and 6 depend from independent Claim 1, which is amended to recite:  
**container registers encapsulated and logically defined in a plurality of containers...the container registers having defined therein data comprising historical data associated with interactions...with other containers** from the one or more containers

Claim 1, emphasis added. As previously discussed with regard to the rejection of Claim 1, Donohue does not teach or suggest such a feature. Walker does not provide such a teaching either. Instead, Walker discloses a system wherein users may pay an expert to receive answers to questions over the internet. Walker, column 6, lines 56-66. Answers to previously asked questions are stored in a database to possibly eliminate the need for an expert to duplicate work should a user be willing to accept an old answer. Walker, column 19, lines 66-67 and column 10 lines 1-9.

Because neither Donohue nor Walker, alone or in combination, teach or suggest such features, the applicants respectfully submit that Claims 3, 5, and 6 are patentable over Donohue in view of Walker under 35 U.S.C. §103(a).

Applicant : Michael De Angelo  
Appl. No. : 11/280,700  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003

**CONCLUSION**

In view of the foregoing, it is believed that all claims now pending (1) are in proper form, (2) are neither obvious nor anticipated by the relied upon art of record, and (3) are in condition for allowance. A Notice of Allowance is earnestly solicited at the earliest possible date. If the Examiner believes that a telephone conference would be useful in moving the application forward to allowance, the Examiner is encouraged to contact the undersigned at (650) 614-7400. If there are any additional charges, please charge Deposit Account No. 15-0665.

Respectfully submitted,

ORRICK, HERRINGTON & SUTCLIFFE  
LLP

Dated: 3/5/09

By: Sanjeet K. Dutta  
Sanjeet K. Dutta  
Reg. No. 46,145

Orrick, Herrington & Sutcliffe LLP  
4 Park Plaza, Suite 1600  
Irvine, California 92614-2558  
(650) 614-7647 (telephone)  
(650) 614-7401 (facsimile)

## Electronic Patent Application Fee Transmittal

<b>Application Number:</b>	11280700
<b>Filing Date:</b>	14-Nov-2005
<b>Title of Invention:</b>	System and method for creating and manipulating information containers with dynamic registers
<b>First Named Inventor/Applicant Name:</b>	Michael De Angelo
<b>Filer:</b>	Sanjeet Kumar Dutta
<b>Attorney Docket Number:</b>	17776-002US4

Filed as Small Entity

### Utility under 35 USC 111(a) Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD(\$)
<b>Basic Filing:</b>				
<b>Pages:</b>				
<b>Claims:</b>				
<b>Miscellaneous-Filing:</b>				
<b>Petition:</b>				
<b>Patent-Appeals-and-Interference:</b>				
<b>Post-Allowance-and-Post-Issuance:</b>				
<b>Extension-of-Time:</b>				
Extension - 2 months with \$0 paid	2252	1	245	245

Description	Fee Code	Quantity	Amount	Sub-Total in USD(\$)
<b>Miscellaneous:</b>				
<b>Total in USD (\$)</b>				<b>245</b>

## Electronic Acknowledgement Receipt

<b>EFS ID:</b>	4912836
<b>Application Number:</b>	11280700
<b>International Application Number:</b>	
<b>Confirmation Number:</b>	9680
<b>Title of Invention:</b>	System and method for creating and manipulating information containers with dynamic registers
<b>First Named Inventor/Applicant Name:</b>	Michael De Angelo
<b>Customer Number:</b>	34313
<b>Filer:</b>	Sanjeet Kumar Dutta
<b>Filer Authorized By:</b>	
<b>Attorney Docket Number:</b>	17776-002US4
<b>Receipt Date:</b>	05-MAR-2009
<b>Filing Date:</b>	14-NOV-2005
<b>Time Stamp:</b>	16:27:05
<b>Application Type:</b>	Utility under 35 USC 111(a)

### Payment information:

Submitted with Payment	yes
Payment Type	Deposit Account
Payment was successfully received in RAM	\$245
RAM confirmation Number	2053
Deposit Account	150665
Authorized User	

The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows:

Charge any Additional Fees required under 37 C.F.R. Section 1.17 (Patent application and reexamination processing fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.19 (Document supply fees)



Charge any Additional Fees required under 37 C.F.R. Section 1.21 (Miscellaneous fees and charges)

**File Listing:**

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1		4003Resp_3_5_09.pdf	150333 <small>756df0e904adaedb644846db77324cff58ecd56c</small>	yes	16
<b>Multipart Description/PDF files in .zip description</b>					
	<b>Document Description</b>		<b>Start</b>		<b>End</b>
	Extension of Time		1		2
	Supplemental Response or Supplemental Amendment		3		16
<b>Warnings:</b>					
<b>Information:</b>					
2	Fee Worksheet (PTO-06)	fee-info.pdf	29875 <small>d85504a1244faddd4f3549f4f9b3c0fe50fb934a</small>	no	2
<b>Warnings:</b>					
<b>Information:</b>					
<b>Total Files Size (in bytes):</b>			180208		

**This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.**

**New Applications Under 35 U.S.C. 111**

**If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.**

**National Stage of an International Application under 35 U.S.C. 371**

**If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.**

**New International Application Filed with the USPTO as a Receiving Office**

**If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appl. No. : 11/280,700 Confirmation No.: 9680  
 Applicant : Michael De Angelo  
 Filing Date : November 14, 2005  
 Title : SYSTEM AND METHOD FOR CREATING AND MANIPULATING  
 INFORMATION CONTAINERS WITH DYNAMIC REGISTERS  
 Group Art Unit : 2161  
 Examiner : Nguyen, Cam Linh T  
 Docket No. : 20933-4003  
 Customer No. : 34313

PETITION FOR EXTENSION OF TIME UNDER 37 CFR 1.136(a)

Commissioner for Patents  
 P.O. Box 1450  
 Alexandria, VA 22313-1450

Sir:

This is a request under the provisions of 37 CFR § 1.136(a) to extend the period for filing a reply in the above-identified application.

The requested extension and fees are as followings: (check time period desired below):

EXTENSION (months)	FEE FOR SMALL ENTITY	FEE FOR OTHER THAN SMALL ENTITY
<input type="checkbox"/> one month	\$65.00	\$130.00
<input checked="" type="checkbox"/> two months	\$245.00	\$490.00
<input type="checkbox"/> three months	\$555.00	\$1,110.00
<input type="checkbox"/> four months	\$865.00	\$1,730.00
<input type="checkbox"/> five months	\$1,080.00	\$2,250.00
	<b>Fee</b>	<b>\$245.00</b>

Applicant claims small entity status. See 37 CFR 1.27.

Method of Payment of fees:

- A check in the amount of \_\_\_\_\_ is enclosed to cover the above fee(s).
- Charge Orrick's Deposit Account No. **15-0665** in the amount of **\$245.00**

- The Commissioner is authorized to charge Orrick's Deposit Account No. **15-0665** for any fees required under 37 CFR §§ 1.16, 1.17 and 1.445 that are not covered, in whole or in part, by a check enclosed herewith and to credit any overpayments to said Deposit Account No. **15-0665**.

I am the

- applicant/inventor
- assignee of record of the entire interest. See 38 CFR 3.71. Statement under 37 CFR 3.373(b) is enclosed.
- Attorney or agent of record. Registration No. 46,145
- Attorney or agent under 37 CFR 1.34. Registration No. \_\_\_\_\_.

Respectfully submitted,

ORRICK, HERRINGTON & SUTCLIFFE LLP

Dated: 3/5/09

By: Sanjeet Dutta  
Sanjeet K. Dutta  
Reg. No. 46,145

Four Park Plaza, Suite 1600  
Irvine, California 92614-2558  
(650) 614-7647

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

<b>PATENT APPLICATION FEE DETERMINATION RECORD</b> Substitute for Form PTO-875	Application or Docket Number <b>11/280,700</b>	Filing Date <b>11/14/2005</b>	<input type="checkbox"/> To be Mailed
---	---	----------------------------------	---------------------------------------

APPLICATION AS FILED – PART I			OTHER THAN SMALL ENTITY			
	(Column 1)	(Column 2)	SMALL ENTITY <input checked="" type="checkbox"/>	OR		
FOR	NUMBER FILED	NUMBER EXTRA	RATE (\$)	FEE (\$)	RATE (\$)	FEE (\$)
<input type="checkbox"/> BASIC FEE <small>(37 CFR 1.16(a), (b), or (c))</small>	N/A	N/A	N/A		N/A	
<input type="checkbox"/> SEARCH FEE <small>(37 CFR 1.16(k), (l), or (m))</small>	N/A	N/A	N/A		N/A	
<input type="checkbox"/> EXAMINATION FEE <small>(37 CFR 1.16(o), (p), or (q))</small>	N/A	N/A	N/A		N/A	
TOTAL CLAIMS <small>(37 CFR 1.16(i))</small>	minus 20 =	*	X \$ =	OR	X \$ =	
INDEPENDENT CLAIMS <small>(37 CFR 1.16(h))</small>	minus 3 =	*	X \$ =		X \$ =	
<input type="checkbox"/> APPLICATION SIZE FEE <small>(37 CFR 1.16(s))</small>	If the specification and drawings exceed 100 sheets of paper, the application size fee due is \$250 (\$125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).					
<input type="checkbox"/> MULTIPLE DEPENDENT CLAIM PRESENT <small>(37 CFR 1.16(j))</small>						
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL		TOTAL	

APPLICATION AS AMENDED – PART II					OTHER THAN SMALL ENTITY			
	(Column 1)	(Column 2)	(Column 3)					
AMENDMENT	<b>03/05/2009</b>	CLAIMS REMAINING AFTER AMENDMENT	HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA	RATE (\$)	ADDITIONAL FEE (\$)	RATE (\$)	ADDITIONAL FEE (\$)
	Total <small>(37 CFR 1.16(i))</small>	* 23	Minus ** 30	= 0	X \$26 =	0	OR	X \$ =
	Independent <small>(37 CFR 1.16(h))</small>	* 7	Minus *** 12	= 0	X \$110 =	0	OR	X \$ =
	<input type="checkbox"/> Application Size Fee <small>(37 CFR 1.16(s))</small>						OR	
	<input type="checkbox"/> FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM <small>(37 CFR 1.16(j))</small>						OR	
					TOTAL ADD'L FEE	0	OR	TOTAL ADD'L FEE

	(Column 1)	(Column 2)	(Column 3)					
AMENDMENT		CLAIMS REMAINING AFTER AMENDMENT	HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA	RATE (\$)	ADDITIONAL FEE (\$)	RATE (\$)	ADDITIONAL FEE (\$)
	Total <small>(37 CFR 1.16(i))</small>	*	Minus **	=	X \$ =		OR	X \$ =
	Independent <small>(37 CFR 1.16(h))</small>	*	Minus ***	=	X \$ =		OR	X \$ =
	<input type="checkbox"/> Application Size Fee <small>(37 CFR 1.16(s))</small>						OR	
	<input type="checkbox"/> FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM <small>(37 CFR 1.16(j))</small>						OR	
					TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE

\* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.  
 \*\* If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".  
 \*\*\* If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".  
 The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

Legal Instrument Examiner:  
 /MYRTLE B. LEIGH/

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**  
 If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.



UNITED STATES PATENT AND TRADEMARK OFFICE

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

Table with 5 columns: APPLICATION NO., FILING DATE, FIRST NAMED INVENTOR, ATTORNEY DOCKET NO., CONFIRMATION NO.
11/280,700 11/14/2005 Michael De Angelo 17776-002US4 9680

34313 7590 05/01/2009
ORRICK, HERRINGTON & SUTCLIFFE, LLP
IP PROSECUTION DEPARTMENT
4 PARK PLAZA
SUITE 1600
IRVINE, CA 92614-2558

EXAMINER

NGUYEN, CAM LINH T

ART UNIT PAPER NUMBER

2161

MAIL DATE DELIVERY MODE

05/01/2009

PAPER

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

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



Art Unit: 2161

**DETAILED ACTION**

1. This Office Action is responsive to communication filed on 03/05/2009.
2. Applicant's amendments to claims 1 – 18, 21 – 24, and 30 are acknowledged.

Consequently, claims 1 – 18, 21 – 24, and 30 are currently pending.

***Claim Rejections - 35 USC § 101***

3. 35 U.S.C. 101 reads as follows:

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

4. Claims 1 – 18 and 30 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

Claims 1, 18, and 30 are directed to a method for searching a container. However, the claim lack of a physical hardware to carry out the invention.

Based on the Office's guidance to Examiners is that a § 101 process must (1) be tied to another statutory class (such as a particular apparatus) or (2) transform underlying subject matter (such as an article or materials) to a different state or thing. Thus, to qualify as a § 101 statutory process, the claim should positively recite the other statutory class (the thing or product) to which it is tied, for example, by identifying the apparatus that accomplishes the method steps, or positively recite the subject matter that is being transformed, for example, by identifying the material that is being changed to a different state.

Art Unit: 2161

***Claim Rejections - 35 USC § 103***

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

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

6. Claims 1 – 2, 4, 7 - 18, 21 – 24, and 30 are rejected under 35 U.S.C. 103(a) as being unpatentable over Donohue et al (U.S. 5,987,480) in view of Nizzari et al (U.S. 6,014,647).

◆As per claims 1, 21, 23, 30,

Donohue discloses a method, system comprising:

- “Receiving a search query” corresponds to the step of receiving the URL request (Fig. 3A, step 48, col. 10, lines 31 – 32).
- “Searching container registers encapsulated and logically defined in a plurality of containers to identify one or more containers responsive to the search query” corresponds to the step of searching for the name and value pair and the corresponding template using the information in the URL (col. 10, lines 32 – 42). The name and value pair and the template in this case correspond to the “container register” (col. 5, lines 26 - 35).
- “Providing a list characterizing the identified containers” corresponds to the step of delivery the document (Fig. 3C, step 82, col. 13, lines 11 – 14).

Donohue does not clearly disclose that “the container registers having defined therein data comprising historical data associated with interactions of the one or more containers with other containers from the one or more containers”.



Art Unit: 2161

However these differences are only found in the nonfunctional descriptive material and are not functionally involved in the steps recited. Thus, this descriptive material will not distinguish the claimed invention from the prior art in terms of patentability, *see In re Gulack*, 703 F.2d 1381, 1385, 217 USPQ 401,404 (Fed. Cir. 1983); *In re Lowry*, 32 F.3d 1579, 32 USPQ2d 1031 (Fed. Cir.1994).

In addition, Nizzari, on the other hand, also teaches "historical data associated with interaction of the one or more container" by using a interaction tracking system that records all the interaction of a user (container) and stores it in a database (abstract, Fig. 1, Fig. 4, col. 3, lines 10 – 18, col. 4, lines 15 – 18, col. 8, lines 33 – 56 of Nizzari).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to include different type of data because such data does not functionally relate to the steps in the method claimed and because the subjective interpretation of the data does not patentably distinguish the claimed invention.

It also would have been obvious to a person of ordinary skill in the art at the time the invention was made to include the teaching of Nizzari into the invention of Donohue since both inventions was available at the time and the combination would provide the user more data types in the container and decreasing the searching time for the user.

◆As per claim 2,

- “The received search query comprises a labeled data tree having at least one parent-child relationship” See col.5, lines 28 – 29 of Donohue where the “labeled data tree” corresponds to the hierarchical directory structure.

Art Unit: 2161

◆As per claim 4,

- “The provided information is stored in one or more search templates” See Fig. 3A- 3C of Donohue.

◆As per claim 7,

- “The list provides a title of each identified container and a short description of its contents” See col. 13, lines 11 – 21 of Donohue.

◆As per claims 8 - 9,

- Receiving a container search level parameter; and wherein the searching content and container registers only searches within container levels associated with the container search level parameter” See col. 13, lines 22 – 26 of Donohue.

◆As per claim 10,

- “Encapsulating the search query into a search container” See col. 8, lines 3 – 9 of Donohue.

◆As per claims 11 - 17,

- “receiving, by a gateway, the search container; storing, by the gateway, data contained within a register of the search container; and determining whether any registers of containers accessible via the gateway are associated with the register of the search container” See col. 9, lines 54 – col. 10, lines 3 of Donohue.

◆As per claims 18, 22, 24,

Claims 18, 22, 24, are rejected based on the rejection of claims 1 and 11.

Art Unit: 2161

7. Claims 3, 5 – 6 are rejected under 35 U.S.C. 103(a) as being unpatentable over Donohue et al (U.S. 5,987,480) in view of Nizzari et al (U.S. 6,014,647) as applied to claims above, further in view of Walker et al (U.S. 5,862,223).

◆As per claims 3, 5 – 6,

Donohue/Nizzari discloses a method for searching document using name value pair and the template to populate the document. Donohue does not clearly disclose that “Providing information identifying containers that have previously been used to respond to one or more processed queries that are substantially similar to the search query” or “providing information identifying substantially similar search phrases, search templates, or labeled data trees that have previously been used to respond to one or more processed queries that are substantially similar to the search query” and “receiving a selection of one of the substantially similar search phrases; and providing a list of previously identified containers associated with the selected search phrase”.

However, storing and reusing the history transaction is well known in the art. Walker provided an example. Walker teaches an expertise system and method comprising the teaching of storing user request and corresponding expert answer in database (Fig. 2, 265, 270). When a new request comes, the system will search for duplicate requests (Fig. 7, step 710), and provide the user with the corresponding answer (Fig. 7, step 720, 740). These teachings are similar to the instant claim invention.

It would have been obvious to one with ordinary skill in the art at the time the invention was made to apply the teaching of Walker into the invention of Donohue/Nizzari sine both

Art Unit: 2161

inventions were available and the combination would reduce the time searching for information and improve the system performance.

*Response to Arguments*

8. Applicant's arguments with respect to claims 1 – 18, 21 – 24, and 30 have been considered but are moot in view of the new ground(s) of rejection.

*Conclusion*

9. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

10. Any inquiry concerning this communication or earlier communications from the examiner should be directed to CAM-LINH NGUYEN whose telephone number is (571) 272 - 4024. The examiner can normally be reached on Monday-Friday.

Art Unit: 2161

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Apu Mofiz can be reached on (571) 272 - 4080. The fax phone number for the organization where this application or proceeding is assigned is 571 - 273 - 8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/CamLinh Nguyen/  
Primary Examiner, Art Unit 2161

<b>Notice of References Cited</b>	Application/Control No. 11/280,700	Applicant(s)/Patent Under Reexamination ANGELO, MICHAEL DE	
	Examiner CAM-LINH NGUYEN	Art Unit 2161	Page 1 of 1

**U.S. PATENT DOCUMENTS**

*	Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
*	A US-6,014,647 A	01-2000	Nizzari et al.	705/39
	B US-			
	C US-			
	D US-			
	E US-			
	F US-			
	G US-			
	H US-			
	I US-			
	J US-			
	K US-			
	L US-			
	M US-			

**FOREIGN PATENT DOCUMENTS**

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

**NON-PATENT DOCUMENTS**

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

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

## EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L17	1977	(interaction\$1 near4 (folder \$1 or container\$1))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 20:27
L18	603	17 and @AD<"19980130"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 20:28
L19	58	18 and (register\$3 or registry or registration)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 20:28
L20	1	19 and template\$1	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 20:29
L21	8	19 and "707"/\$.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 20:30
L22	41562	(interaction\$1 near4 (data or information or histor\$3))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 20:34
L23	12	18 and 22	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 20:34
L24	9	23 not 21	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 20:34
S36	99459	(history or historical\$2) near3 (data or information or record\$1)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 13:42
S37	36094	register\$3 near3 (container \$1 or folder\$1 or file\$1)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 13:43
S38	261	S36 same S37	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 13:43

S39	480	search\$3 near4 register\$3 near3 (container\$1 or folder\$1 or file\$1)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 13:44
S40	2	S36 same S39	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 13:44
S41	78	S38 and @AD<"19980130"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 13:45
S42	2	S41 and "707"/\$.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 13:45
S43	691	((history or historical\$2) near3 (data or information or record\$1) near4 interaction \$1	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 13:46
S44	3	S37 same S43	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 13:46
S45	25	S37 and S43	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 13:46
S46	0	S45 and @AD<"19980130"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 13:46
S47	4	((history or historical\$2) near3 (data or information or record\$1) near4 interaction \$1) same (container\$1 or folder\$1)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 13:47
S48	52	((history or historical\$2) near3 (data or information or record\$1) near4 interaction \$1).ab.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 13:49
S49	25	S43 and @AD<"19980130"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 13:49
S50	4	((register\$3 or registration) near4 (history or historical \$2) near3 (data or information or record\$1) near4 interaction\$1	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 13:53




S51	15	(register\$3 or registration) same ((history or historical \$2) near3 (data or information or record\$1) near4 interaction\$1)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 13:54
S52	11	S51 not S50	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 13:54
S53	1	S52 and @AD<"19980130"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 13:54
S54	245	(register\$3 or registration) same ((history or historical \$2) same (data or information or record\$1) same interaction\$1)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 13:56
S55	3	S39 and S54	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 13:56
S56	6	S54 and @AD<"19980130"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 13:56
S57	6	(register\$3 or registration) near4 (data or information or record\$1) near4 interaction \$1 near4 (file\$1 or folder\$1 or container\$1)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 14:55
S58	20	(register\$3 or registration) near4 interaction\$1 near4 (file\$1 or folder\$1 or container\$1)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 15:18
S59	1	S58 and @AD<"19980130"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 15:18
S60	52	S43 and S48	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 15:22
S61	6	S60 and @AD<"19980130"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 15:22
S62	4881	interaction\$1 near4 (file\$1 or folder\$1 or container\$1)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 15:26

S63	57	S62 same register\$3	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 15:26
S64	9	S63 and @AD<"19980130"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 15:26
S65	1453	S62 and register\$3	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 15:28
S66	1534	S62 and (registration or register\$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 15:28
S67	184	S66 and @AD<"19980130"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 15:28
S68	175	S67 not S64	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 15:28
S69	41	(S36 or S37) and S68	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 15:29
S70	484	(interaction\$1 near4 (file\$1 or folder\$1 or container\$1)). ab.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 15:54
S71	62	S70 and (register\$3 or registration or registry)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 15:55
S72	7	S71 and @AD<"19980130"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 15:55
S73	456	(interaction\$1 near4 (file\$1 or folder\$1 or container\$1)). clm.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 15:56
S74	85	S73 and @AD<"19980130"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 15:56

S75	14	S74 and register\$3	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 15:56
S76	12	S75 not S72	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/04/29 15:57

4/29/09 8:52:34 PM

C:\Documents and Settings\CNguyen11\My Documents\EAST\Workspaces\11095443.wsp

<b>Index of Claims</b>  	<b>Application/Control No.</b>  11280700	<b>Applicant(s)/Patent Under Reexamination</b>  ANGELO, MICHAEL DE
	<b>Examiner</b>  CAM-LINH NGUYEN	<b>Art Unit</b>  2161

✓	<b>Rejected</b>
=	<b>Allowed</b>


-	<b>Cancelled</b>
÷	<b>Restricted</b>

N	<b>Non-Elected</b>
I	<b>Interference</b>

A	<b>Appeal</b>
O	<b>Objected</b>

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

CLAIM		DATE							
Final	Original	10/06/2008	04/29/2009						
	1	✓	✓						
	2	✓	✓						
	3	✓	✓						
	4	✓	✓						
	5	✓	✓						
	6	✓	✓						
	7	✓	✓						
	8	✓	✓						
	9	✓	✓						
	10	✓	✓						
	11	✓	✓						
	12	✓	✓						
	13	✓	✓						
	14	✓	✓						
	15	✓	✓						
	16	✓	✓						
	17	✓	✓						
	18	✓	✓						
	19	-	-						
	20	-	-						
	21	✓	✓						
	22	✓	✓						
	23	✓	✓						
	24	✓	✓						
	25	-	-						
	26	-	-						
	27	-	-						
	28	-	-						
	29	-	-						
	30	✓	✓						

<b>Search Notes</b>  	<b>Application/Control No.</b>  11280700	<b>Applicant(s)/Patent Under Reexamination</b>  ANGELO, MICHAEL DE
	<b>Examiner</b>  CAM-LINH NGUYEN	<b>Art Unit</b>  2161

<b>SEARCHED</b>			
<b>Class</b>	<b>Subclass</b>	<b>Date</b>	<b>Examiner</b>

<b>SEARCH NOTES</b>		
<b>Search Notes</b>	<b>Date</b>	<b>Examiner</b>
EAST search	10/4/08	Linh
EAST search	4/29/09	Linh

<b>INTERFERENCE SEARCH</b>			
<b>Class</b>	<b>Subclass</b>	<b>Date</b>	<b>Examiner</b>

--	--

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

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

Appl. No. : 11/280,700 Confirmation No.: 9680  
Applicant : Michael DeAngelo  
Filing Date : November 14, 2005  
Title : SYSTEM AND METHOD FOR CREATING AND MANIPULATING  
INFORMATION CONTAINERS WITH DYNAMIC REGISTERS  
Group Art Unit : 2161  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003  
Customer No. : 34313

Commissioner for Patents  
Mail Stop RCE  
P.O. Box 1450  
Alexandria, VA 22313-1450

**REQUEST FOR CONTINUED EXAMINATION (RCE) TRANSMITTAL**

I. Submission required under 37 CFR § 1.114

- A.  Previously submitted
1.  Consider the amendment(s)/reply under 37 CFR § 1.116 previously filed on \_\_\_\_\_
  2.  Consider the arguments in the Appeal Brief or Reply Brief previously filed on \_\_\_\_\_
  3.  Other \_\_\_\_\_
- B.  Enclosed
1.  Amendment/Reply
  2.  Affidavit(s)/Declarations(s)
  3.  Information Disclosure Statement (IDS) with copies of non U.S. references
  4.  Other –

Applicant : DeAngelo, Michael  
 Appl. No. : 11/280,700  
 Examiner : Nguyen, Cam Linh T  
 Docket No. : 20933-4003

II. Miscellaneous

- A.  Suspension of action on the above-identified application is requested under 37 CFR § 1.103(c) for a period of \_\_\_\_\_ months. (Period of suspension shall not exceed 3 months; fee under 37 CFR § 1.17(i) required.)
- B.  Other \_\_\_\_\_

III. Fees

- A.  The Commissioner is hereby authorized to charge the following fees, or credit any overpayments, to Deposit Account No. 15-0665

1.  RCE fees (37 CFR §1.17(e):

RCE Fee						\$810.00
	Claims filed or remaining after amendment		Highest number previously paid for			
Total Claims	23	-	30	= 0	x \$52.00	\$00.00
Independent Claims	7	-	12	= 0	x \$220.00	\$00.00
<input checked="" type="checkbox"/> Reduction by ½ for Filing by Small Entity. Note 37 CFR §§ 1.9, 1.27, 1.28.						\$405.00
<b>TOTAL OF ABOVE CALCULATIONS</b>						<b>\$405.00</b>

2.  Extension of time fee (37 CFR §§ 1.136 and 1.17)

EXTENSION (months)	FEE FOR SMALL ENTITY	FEE FOR OTHER THAN SMALL ENTITY
<input type="checkbox"/> one month	\$65.00	\$130.00
<input type="checkbox"/> two months	\$245.00	490.00
<input checked="" type="checkbox"/> three months	\$555.00	\$1,110.00
<input type="checkbox"/> four months	\$865.00	\$1,730.00
<input type="checkbox"/> five months	\$1,175.00	\$2,350.00
	<b>Fee</b>	<b>\$555.00</b>

2.  Other \_\_\_\_\_

- B.  Check in the amount of \$\_\_\_\_\_ is enclosed

Applicant : DeAngelo, Michael  
Appl. No. : 11/280,700  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003

- C.  The Commissioner is authorized to charge Orrick, Herrington & Sutcliffe LLP's Deposit Account No. **15-0665** in the amount of \$960.00 to cover the filing fee and to credit any overpayments to said Deposit Account **15-0665**

Respectfully submitted,  
ORRICK, HERRINGTON & SUTCLIFFE LLP

Dated: November 2, 2009

By: Sanjeet Dutta  
Sanjeet K. Dutta  
Reg. No. 46,145

Orrick, Herrington & Sutcliffe LLP  
4 Park Plaza, Suite 1600  
Irvine, CA 92614-2558  
Tel. 650 614-7400  
Fax: 949-567-6710  
Customer Number: 34313



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

Appl. No. : 11/280,700 Confirmation No.: 9680  
Applicant : Michael De Angelo  
Filing Date : November 14, 2005  
Title : SYSTEM AND METHOD FOR CREATING AND MANIPULATING  
INFORMATION CONTAINERS WITH DYNAMIC REGISTERS  
Group Art Unit : 2161  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003  
Customer No. : 34313

**Mail Stop: Amendment**  
Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

**RESPONSE TO OFFICE ACTION**

Sir:

In response to the Office Action dated May 1, 2009 please amend the above-identified application as follows:

**Amendments to the Claims** are reflected in the listing of claims, which begins on page 2 of this paper.

**Remarks** begin on page 11 of this paper.

Applicant : Michael De Angelo  
Appl. No. : 11/280,700  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003

### **AMENDMENTS TO THE CLAIMS**

This listing of claims will replace all prior versions and listings of claims in the application:

#### **LISTING OF CLAIMS**

1. (Currently Amended) A computer-implemented method comprising:  
receiving a search query;  
searching first container registers encapsulated and logically defined in a plurality of  
containers to identify ~~one or more~~ identified containers responsive to the search  
query, the container registers having defined therein data comprising historical  
data associated with interactions of the ~~one or more~~ identified containers with  
other containers from the ~~one or more~~ plurality of containers, wherein searching  
the first container registers comprises searching the historical data;  
encapsulating the identified containers in a new container;  
updating second container registers of the identified containers with data associated with  
interactions of the identified containers with the new container; and  
providing a list characterizing the identified containers.
2. (Currently Amended) A computer-implemented method as in claim 1, when the  
received search query comprises a labeled data tree having at least one parent-  
child relationship.

Applicant : Michael De Angelo  
Appl. No. : 11/280,700  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003

3. (Currently Amended) A computer-implemented method as in claim 1, further comprising: providing information identifying containers that have previously been used to respond to one or more processed queries that are substantially similar to the search query.
4. (Currently Amended) A computer-implemented method as in claim 1, wherein the provided information is stored in one or more search templates.
5. (Currently Amended) A computer-implemented method as in claim 1, further comprising: providing information identifying substantially similar search phrases, search templates, or labeled data trees that have previously been used to respond to one or more processed queries that are substantially similar to the search query.
6. (Currently Amended) A computer-implemented method as in claim 5, further comprising:  
  
receiving a selection of one of the substantially similar search phrases; and  
  
providing a list of previously identified containers associated with the selected search phrase.
7. (Currently Amended) A computer-implemented method as in claim 1, wherein the list provides a title of each identified container and a short description of its contents.

Applicant : Michael De Angelo  
Appl. No. : 11/280,700  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003

8. (Currently Amended) A computer-implemented method as in claim 1, further comprising: receiving a container search level parameter; and wherein the searching content and container registers only searches within container levels associated with the container search level parameter.
  
9. (Currently Amended) A computer-implemented method as in claim 1, further comprising: receiving a container search level parameter; and wherein the list of identified containers only comprises containers associated with the container search level parameter.
  
10. (Currently Amended) A computer-implemented method as in claim 1, wherein the searching further comprises: encapsulating the search query into a search container.
  
11. (Currently Amended) A computer-implemented method as in claim 10, wherein the searching further comprises:  
  
receiving, by a gateway, the search container;  
  
storing, by the gateway, data contained within a register of the search container; and  
  
determining whether any registers of containers accessible via the gateway are associated  
  
with the register of the search container.

Applicant : Michael De Angelo  
Appl. No. : 11/280,700  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003

12. (Currently Amended) A computer-implemented method as in claim 11, further comprising:

generating a new gateway; and

associating the container with the new gateway.

13. (Currently Amended) A computer-implemented method as in claim 11, further comprising: periodically aggregating the contents of registers in a plurality of gateways to characterize a plurality of containers coupled thereto.

14. (Currently Amended) A computer-implemented method as in claim 11, wherein the contents of the registers in each of the plurality of gateways comprise at least one metric chosen from a group comprising: frequency of access of the gateway, grade of access of the gateway, description of users that have accessed the gateway, an identity of containers that have accessed the gateway, parameters associated with the gateway register, and historically accumulated register data.

15. (Currently Amended) A computer-implemented method as in claim 11, further comprising: monitoring transactions involving one or more gateways or containers.

16. (Currently Amended) A computer-implemented method as in claim 15, further comprising: generating new containers based on the monitored transactions.

Applicant : Michael De Angelo  
Appl. No. : 11/280,700  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003

17. (Currently Amended) A computer-implemented method as in claim 15, wherein the transactions are based on each instance a gateway or container passes through another gateway or container.

18. (Currently Amended) A computer-implemented method comprising:  
receiving a search query;  
polling a plurality of gateways to identify registers encapsulated therein, the gateways having a plurality of containers coupled thereto, the identified registers relating to one or more identified containers logically defining data contained therein associated with the search query, the one or more identified containers having container registers defined therein, the container registers containing data comprising historical data associated with interactions of the one or more identified containers with other containers from the one or more plurality of containers, wherein the containers are coupled to the gateways, wherein polling the plurality of gateways comprises searching the historical data;  
encapsulating the identified containers in a new container;  
updating the container registers of the identified containers with data associated with interactions of the identified containers with the new container; and  
providing a list characterizing the identified containers.

19-20. (Canceled)

Applicant : Michael De Angelo  
Appl. No. : 11/280,700  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003

21. (Currently Amended) A computer program product, tangibly embodied on computer-readable media, comprising instructions operable to cause a data processing apparatus to:

receive a search query;

search content and first container registers encapsulated and logically defined in a plurality of containers to identify ~~one or more~~ identified containers associated with the search query, the first container registers having defined therein data comprising historical data associated with interactions of the ~~one or more~~ identified containers with other containers from the ~~one or more~~ plurality of containers, wherein searching the first container registers comprises searching the historical data;

encapsulate the identified containers in a new container;

update second container registers of the identified containers with data associated with interactions of the identified containers with the new container; and

provide a list characterizing the identified containers.

22. (Currently Amended) A computer program product, tangibly embodied on computer-readable media, comprising instructions operable to cause a data processing apparatus to:

receive a search query;

Applicant : Michael De Angelo  
Appl. No. : 11/280,700  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003

poll a plurality of gateways to identify registers encapsulated therein, the gateways  
having a plurality of containers coupled thereto, the identified registers relating to  
one or more identified containers logically defining data contained therein  
associated with the search query, the ~~one or more~~ identified containers having  
container registers defined therein, the container registers containing data  
comprising historical data associated with interactions of the ~~one or more~~  
identified containers with other containers from the ~~one or more~~ plurality of  
containers, ~~wherein the containers are coupled to the gateways, wherein polling~~  
the plurality of gateways comprises searching the historical data;  
encapsulate the identified containers in a new container;  
update container registers of the identified containers with data associated with  
interactions of the identified containers with the new container; and  
provide a list characterizing the identified containers.

23. (Currently Amended) An apparatus comprising:

means for receiving a search query;

means for searching content and first container registers encapsulated and logically  
defined in a plurality of containers to identify ~~one or more~~ identified containers  
associated with the search query, the first container registers having defined  
therein data comprising historical data associated with interactions of the ~~one or~~  
~~more~~ identified containers with other containers from the ~~one or more~~ plurality  
containers, wherein searching container registers comprises searching the  
historical data;



Applicant : Michael De Angelo  
Appl. No. : 11/280,700  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003

means for encapsulating the identified containers in a new container;

means for updating second container registers of the identified containers with data

associated with interactions of the identified containers with the new container;

and

means for providing a list characterizing the identified containers.

24. (Currently Amended) An apparatus comprising:

means for receiving a search query;

means for polling a plurality of gateways to identify identified registers encapsulated therein, the gateways having a plurality of containers coupled thereto, the identified registers relating to one or more identified containers logically defining data contained therein associated with the search query, the one or more identified containers having container registers defined therein, the container registers containing data comprising historical data associated with interactions of the one or more identified containers with other containers from the one or more plurality of containers, wherein the containers are coupled to the gateways, wherein polling the plurality of gateways comprises searching the historical data;

means for encapsulating the identified containers in a new container;

means for updating the container registers of the identified containers with data

associated with interactions of the identified containers with the new container;

and

means for providing a list characterizing the identified containers.

Applicant : Michael De Angelo  
Appl. No. : 11/280,700  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003

25-29. (Canceled)

30. (Currently Amended) A computer-implemented method comprising:

receiving a search query;

searching first container registers encapsulated and logically defined in a plurality of

containers to identify ~~one or more~~ search query templates encapsulated in

identified containers, the first container registers having defined therein data

comprising historical data associated with interactions of the ~~one or more~~

identified containers with other containers from the ~~one or more~~ plurality of

containers, wherein searching container registers comprises searching the

historical data;

encapsulating the identified containers in a new container;

updating second container registers of the identified containers with data associated with

interactions of the identified containers with the new container; and

providing a list characterizing the identified one or more search query templates to

formulate subsequent search queries.

Applicant : Michael De Angelo  
Appl. No. : 11/280,700  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003

**REMARKS**

Claims 1-18, 21-24, and 30 are pending in the present application.

Claims 1-18 and 30 have been rejected under 35 U.S.C. §101 as being directed to non-statutory subject matter.

Claims 1, 2, 4, 7-18, 21-24, and 30 have been rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 5,987,480 to Donohue, *et al.* (“Donohue”) in view of U.S. Patent No. 6,014,647 to Nizzari, *et al.* (“Nizzari”).

Claims 3, 5-6 have been rejected under 35 U.S.C. §103(a) as being unpatentable over Donohue in view of Nizzari, and further in view of U.S. Patent No. 5,862,223 to Walker, *et al.* (“Walker”).

Claims 1-18, 21-24, and 30 have been amended. It is respectfully submitted that no new matter has been added.

Reconsideration of the application as amended herein is respectfully requested.

**Rejections under 35 U.S.C. §101**

Claims 1-18 and 30 are rejected under 35 U.S.C. §101 as being directed to non-statutory subject matter. The Examiner states, “Claims 1, 18, and 30 are directed to a method for searching a container. However, the claim[s] lack of a physical hardware to carry out the invention.” Claims 1-18 and 30 are amended to recite “computer-implemented...”, and therefore the applicant respectfully submits the Examiner's rejections under 35 U.S.C. §101 are overcome.

Applicant : Michael De Angelo  
Appl. No. : 11/280,700  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003

**Rejections under 35 U.S.C. §103(a)**

Claims 1, 2, 4, 7-18, 21-24, and 30 have been rejected under 35 U.S.C. §103(a) as being unpatentable over Donohue in view of Nizzari. The applicant respectfully disagrees and submits that Claims 1, 2, 4, 7-18, 21-24, and 30 are patentable over Donohue in view of Nizzari.

Regarding the rejection of independent Claims 1, 21, 23, and 30, the Examiner states:

Donohue does not clearly disclose that “the container registers having defined therein data comprising historical data associated with interactions of the one or more containers...” However these differences are only found in the nonfunctional descriptive material and are not functionally involved in the steps recited. Thus, this descriptive material will not distinguish the claimed invention from the prior art in terms of patentability.

5/1/09 Office Action, pp. 3-4. Independent Claims 1, 21, 23, and 30 are amended to recite, “**searching the historical data**” which is functionally descriptive material.

Therefore the applicant respectfully submits that the Examiner's rejections are overcome.

Regarding the rejection of Claim 1, the Examiner also states:

Nizzari, on the other hand, also teaches “historical data associated with interaction with one or more container” by using a interaction tracking system that records all the interaction of a user (container) and stores it in a database.

5/1/09 Office Action, pp. 3-4. The applicant respectfully disagrees for the following reasons.

Claim 1 recites:

Applicant : Michael De Angelo  
Appl. No. : 11/280,700  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003

encapsulating the identified containers in a new container; updating second container registers of the identified containers with data associated with interactions of the identified containers with the new container

Claim 1. As acknowledged by the Examiner, Donohue does not teach or suggest “historical interaction data.” 5/1/09 Office Action, page 3. Nizzari does not provide such a teaching either. Nizzari describes processing interactions between a customer and a business. Nizzari, Column 1, lines 63-65. An interaction with a customer can include an operator speaking to a customer, and can be automated. Nizzari, Column 2, lines 30-33. An agent can retrieve a history of recent transactions for a particular customer to customize service for the customer or answer customer questions. Nizzari, Column 8, lines 57-68. In other words, Nizzari describes storing interactions between a customer and a business, and the ability to retrieve information related to the interactions. This is not “encapsulating the identified containers in a new container; updating second container registers of the identified containers with data associated with interactions of the identified containers with the new container” which is recited in Claim 1. Because neither Donohue nor Nizzari, alone or in combination, teach or suggest such a feature, the applicants respectfully submit that Claim 1 and Claims 2-17 which depend from Claim 1 are patentable under 35 U.S.C. §103(a) over Dononue in view of Nizzari.

The Examiner rejected Claim 18 for the same reasons as Claim 1. Applicant respectfully disagrees for the following reasons.

Claim 18 recites:

encapsulating the identified containers in a new container; updating the container registers of the identified containers with data associated with interactions of the identified containers with the new container

Applicant : Michael De Angelo  
Appl. No. : 11/280,700  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003

Claim 18. Similarly, these features of Claim 18 are neither taught nor suggested by Donohue, therefore the applicants respectfully submit that Claim 18 is patentable under 35 U.S.C. §103(a) over Donohue in view of Nizzari.

The Examiner rejected Claim 21 for the same reasons as Claim 1. Applicant respectfully disagrees for the following reasons.

Claim 21 recites:

encapsulate the identified containers in a new container; update second container registers of the identified containers with data associated with interactions of the identified containers with the new container

Claim 21. Similarly, these features of Claim 21 are neither taught nor suggested by Donohue, therefore the applicants respectfully submit that Claim 21 is patentable under 35 U.S.C. §103(a) over Donohue in view of Nizzari.

The Examiner rejected Claim 22 for the same reasons as Claim 1. Applicant respectfully disagrees for the following reasons.

Claim 22 recites:

encapsulate the identified containers in a new container; update container registers of the identified containers with data associated with interactions of the identified containers with the new container

Claim 22. Similarly, these features of Claim 22 are neither taught nor suggested by Donohue, therefore the applicants respectfully submit that Claim 22 is patentable under 35 U.S.C. §103(a) over Donohue in view of Nizzari.

The Examiner rejected Claim 23 for the same reasons as Claim 1. Applicant respectfully disagrees for the following reasons.

Claim 23 recites:

means for encapsulating the identified containers in a new container;

Applicant : Michael De Angelo  
Appl. No. : 11/280,700  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003

means for updating second container registers of the identified containers with data associated with interactions of the identified containers with the new container

Claim 23. Similarly, these features of Claim 23 are neither taught nor suggested by Donohue, therefore the applicants respectfully submit that Claim 23 is patentable under 35 U.S.C. §103(a) over Donohue in view of Nizzari.

The Examiner rejected Claim 24 for the same reasons as Claim 1. Applicant respectfully disagrees for the following reasons.

Claim 24 recites:

means for encapsulating the identified containers in a new container;  
means for updating the container registers of the identified containers with data associated with interactions of the identified containers with the new container

Claim 24. Similarly, these features of Claim 24 are neither taught nor suggested by Donohue, therefore the applicants respectfully submit that Claim 24 is patentable under 35 U.S.C. §103(a) over Donohue in view of Nizzari.

The Examiner rejected Claim 30 for the same reasons as Claim 1. Applicant respectfully disagrees for the following reasons.

Claim 30 recites:

encapsulating the identified containers in a new container; updating second container registers of the identified containers with data associated with interactions of the identified containers with the new container

Claim 30. Similarly, these features of Claim 30 are neither taught nor suggested by Donohue, therefore the applicants respectfully submit that Claim 30 is patentable under 35 U.S.C. §103(a) over Donohue in view of Nizzari.

Applicant : Michael De Angelo  
Appl. No. : 11/280,700  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003

Claims 3, 5-6 are rejected under 35 U.S.C. §103(a) as being unpatentable over Donohue in view of Nizzari and further in view of Walker. The applicant respectfully disagrees and submits that Claims 3, 5-6 are patentable over Donohue in view of Walker.

The Examiner states:

Donohue does not clearly disclose that "Providing information identifying containers that have previously been used to respond to one or more processed queries that are substantially similar to the search query"... Walker teaches an expertise system and method comprising the teaching of storing user request and corresponding expert answer in database (Fig. 2, 265, 270).

05/01/09 Office Action, page 6. The applicant respectfully disagrees for the following reasons.

Claims 3, 5 and 6 depend from independent Claim 1, which recites:

encapsulating the identified containers in a new container;  
updating second container registers of the identified containers with data associated with interactions of the identified containers with the new container

Claim 1. As previously discussed with regard to the rejection of Claim 1, neither Donohue nor Nizzari teach or suggest such a feature. Walker does not provide such a teaching either. Instead, Walker discloses a system wherein users may pay an expert to receive answers to questions over the internet. Walker, column 6, lines 56-66. Answers to previously asked questions are stored in a database to possibly eliminate the need for an expert to duplicate work should a user be willing to accept an old answer. Walker, column 19, lines 66-67 and column 10 lines 1-9. This is not "encapsulating the identified containers in a new container; updating second container registers of the identified containers with data associated with interactions of the identified containers with the new container" which is recited in Claim 1. Because neither Donohue, Nizzari, nor Walker,



Applicant : Michael De Angelo  
Appl. No. : 11/280,700  
Examiner : Nguyen, Cam Linh T  
Docket No. : 20933-4003

alone or in combination, teach or suggest such features, the applicants respectfully submit that Claims 3, 5, and 6 are patentable over Donohue in view of Nizzari and further in view of Walker under 35 U.S.C. §103(a).

### CONCLUSION

In view of the foregoing, it is believed that all claims now pending (1) are in proper form, (2) are neither obvious nor anticipated by the relied upon art of record, and (3) are in condition for allowance. A Notice of Allowance is earnestly solicited at the earliest possible date. If the Examiner believes that a telephone conference would be useful in moving the application forward to allowance, the Examiner is encouraged to contact the undersigned at (650) 614-7400. If there are any additional charges, please charge Deposit Account No. 15-0665.

Respectfully submitted,

ORRICK, HERRINGTON & SUTCLIFFE LLP

Dated: 11/2/09

By: Sanjeet Dutta  
Sanjeet K. Dutta  
Reg. No. 46,145

Orrick, Herrington & Sutcliffe LLP  
4 Park Plaza, Suite 1600  
Irvine, California 92614-2558  
(650) 614-7647 (telephone)  
(650) 614-7401 (facsimile)

## Electronic Patent Application Fee Transmittal

<b>Application Number:</b>	11280700
<b>Filing Date:</b>	14-Nov-2005
<b>Title of Invention:</b>	System and method for creating and manipulating information containers with dynamic registers
<b>First Named Inventor/Applicant Name:</b>	Michael De Angelo
<b>Filer:</b>	Hanbum Cho/Dana Zottola
<b>Attorney Docket Number:</b>	20933-4003

Filed as Small Entity

### Utility under 35 USC 111(a) Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD(\$)
<b>Basic Filing:</b>				
<b>Pages:</b>				
<b>Claims:</b>				
<b>Miscellaneous-Filing:</b>				
<b>Petition:</b>				
<b>Patent-Appeals-and-Interference:</b>				
<b>Post-Allowance-and-Post-Issuance:</b>				
<b>Extension-of-Time:</b>				
Extension - 3 months with \$0 paid	2253	1	555	555

Description	Fee Code	Quantity	Amount	Sub-Total in USD(\$)
<b>Miscellaneous:</b>				
Request for continued examination	2801	1	405	405
<b>Total in USD (\$)</b>				<b>960</b>

## Electronic Acknowledgement Receipt

<b>EFS ID:</b>	6378382
<b>Application Number:</b>	11280700
<b>International Application Number:</b>	
<b>Confirmation Number:</b>	9680
<b>Title of Invention:</b>	System and method for creating and manipulating information containers with dynamic registers
<b>First Named Inventor/Applicant Name:</b>	Michael De Angelo
<b>Customer Number:</b>	34313
<b>Filer:</b>	Hanbum Cho/Dana Zottola
<b>Filer Authorized By:</b>	Hanbum Cho
<b>Attorney Docket Number:</b>	20933-4003
<b>Receipt Date:</b>	02-NOV-2009
<b>Filing Date:</b>	14-NOV-2005
<b>Time Stamp:</b>	21:10:04
<b>Application Type:</b>	Utility under 35 USC 111(a)

### Payment information:

Submitted with Payment	yes
Payment Type	Deposit Account
Payment was successfully received in RAM	\$960
RAM confirmation Number	5474
Deposit Account	150665
Authorized User	

The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows:

Charge any Additional Fees required under 37 C.F.R. Section 1.17 (Patent application and reexamination processing fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.21 (Miscellaneous fees and charges)

<b>File Listing:</b>					
<b>Document Number</b>	<b>Document Description</b>	<b>File Name</b>	<b>File Size(Bytes)/ Message Digest</b>	<b>Multi Part /.zip</b>	<b>Pages (if appl.)</b>
1		4003RespFinal_wRCE_wExt_11_2_09.pdf	653950  cab2d92469a0a8e1e90419cca1f157f7bf0e ba6	yes	20
<b>Multipart Description/PDF files in .zip description</b>					
<b>Document Description</b>			<b>Start</b>	<b>End</b>	
Request for Continued Examination (RCE)			1	3	
Amendment After Final			4	20	
<b>Warnings:</b>					
<b>Information:</b>					
2	Fee Worksheet (PTO-875)	fee-info.pdf	32056  c63a3896f34735675671456c689b6bb2da8 31baf	no	2
<b>Warnings:</b>					
<b>Information:</b>					
<b>Total Files Size (in bytes):</b>			686006		
<p><b>This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.</b></p> <p><b><u>New Applications Under 35 U.S.C. 111</u></b>  <b>If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.</b></p> <p><b><u>National Stage of an International Application under 35 U.S.C. 371</u></b>  <b>If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.</b></p> <p><b><u>New International Application Filed with the USPTO as a Receiving Office</u></b>  <b>If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.</b></p>					

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

<b>PATENT APPLICATION FEE DETERMINATION RECORD</b> Substitute for Form PTO-875	Application or Docket Number <b>11/280,700</b>	Filing Date <b>11/14/2005</b>	<input type="checkbox"/> To be Mailed
---	---	----------------------------------	---------------------------------------

APPLICATION AS FILED – PART I			OTHER THAN SMALL ENTITY				
	(Column 1)	(Column 2)	SMALL ENTITY <input checked="" type="checkbox"/>	OR			
FOR	NUMBER FILED	NUMBER EXTRA	RATE (\$)	FEE (\$)	OR	RATE (\$)	FEE (\$)
<input type="checkbox"/> BASIC FEE <small>(37 CFR 1.16(a), (b), or (c))</small>	N/A	N/A	N/A			N/A	
<input type="checkbox"/> SEARCH FEE <small>(37 CFR 1.16(k), (l), or (m))</small>	N/A	N/A	N/A			N/A	
<input type="checkbox"/> EXAMINATION FEE <small>(37 CFR 1.16(o), (p), or (q))</small>	N/A	N/A	N/A			N/A	
TOTAL CLAIMS <small>(37 CFR 1.16(i))</small>	minus 20 =	*	X \$ =		OR	X \$ =	
INDEPENDENT CLAIMS <small>(37 CFR 1.16(h))</small>	minus 3 =	*	X \$ =			X \$ =	
<input type="checkbox"/> APPLICATION SIZE FEE <small>(37 CFR 1.16(s))</small>	If the specification and drawings exceed 100 sheets of paper, the application size fee due is \$250 (\$125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).						
<input type="checkbox"/> MULTIPLE DEPENDENT CLAIM PRESENT <small>(37 CFR 1.16(j))</small>							
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL			TOTAL	

APPLICATION AS AMENDED – PART II					OTHER THAN SMALL ENTITY				
	(Column 1)	(Column 2)	(Column 3)						
AMENDMENT	11/02/2009	CLAIMS REMAINING AFTER AMENDMENT	HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA	RATE (\$)	ADDITIONAL FEE (\$)	OR	RATE (\$)	ADDITIONAL FEE (\$)
	Total <small>(37 CFR 1.16(i))</small>	* 23	Minus ** 30	= 0	X \$26 =	0	OR	X \$ =	
	Independent <small>(37 CFR 1.16(h))</small>	* 7	Minus *** 12	= 0	X \$110 =	0	OR	X \$ =	
	<input type="checkbox"/> Application Size Fee <small>(37 CFR 1.16(s))</small>								
	<input type="checkbox"/> FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM <small>(37 CFR 1.16(j))</small>						OR		
					TOTAL ADD'L FEE	0	OR	TOTAL ADD'L FEE	

APPLICATION AS AMENDED – PART II					OTHER THAN SMALL ENTITY				
	(Column 1)	(Column 2)	(Column 3)						
AMENDMENT		CLAIMS REMAINING AFTER AMENDMENT	HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA	RATE (\$)	ADDITIONAL FEE (\$)	OR	RATE (\$)	ADDITIONAL FEE (\$)
	Total <small>(37 CFR 1.16(i))</small>	*	Minus **	=	X \$ =		OR	X \$ =	
	Independent <small>(37 CFR 1.16(h))</small>	*	Minus ***	=	X \$ =		OR	X \$ =	
	<input type="checkbox"/> Application Size Fee <small>(37 CFR 1.16(s))</small>								
	<input type="checkbox"/> FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM <small>(37 CFR 1.16(j))</small>						OR		
					TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

\* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.  
 \*\* If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".  
 \*\*\* If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".  
 The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

Legal Instrument Examiner:  
 /KIM WATSON SAUNDERS/

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**  
 If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.



NOTICE OF ALLOWANCE AND FEE(S) DUE

34313 7590 12/03/2009

ORRICK, HERRINGTON & SUTCLIFFE, LLP
IP PROSECUTION DEPARTMENT
4 PARK PLAZA
SUITE 1600
IRVINE, CA 92614-2558

EXAMINER: NGUYEN, CAM LINH T
ART UNIT: 2161
PAPER NUMBER:
DATE MAILED: 12/03/2009

Table with 5 columns: APPLICATION NO., FILING DATE, FIRST NAMED INVENTOR, ATTORNEY DOCKET NO., CONFIRMATION NO.

11/280,700 11/14/2005 Michael De Angelo 20933-4003 9680

TITLE OF INVENTION: SYSTEM AND METHOD FOR CREATING AND MANIPULATING INFORMATION CONTAINERS WITH DYNAMIC REGISTERS

Table with 7 columns: APPLN. TYPE, SMALL ENTITY, ISSUE FEE DUE, PUBLICATION FEE DUE, PREV. PAID ISSUE FEE, TOTAL FEE(S) DUE, DATE DUE

nonprovisional YES \$755 \$300 \$0 \$1055 03/03/2010

THE APPLICATION IDENTIFIED ABOVE HAS BEEN EXAMINED AND IS ALLOWED FOR ISSUANCE AS A PATENT. PROSECUTION ON THE MERITS IS CLOSED. THIS NOTICE OF ALLOWANCE IS NOT A GRANT OF PATENT RIGHTS. THIS APPLICATION IS SUBJECT TO WITHDRAWAL FROM ISSUE AT THE INITIATIVE OF THE OFFICE OR UPON PETITION BY THE APPLICANT. SEE 37 CFR 1.313 AND MPEP 1308.

THE ISSUE FEE AND PUBLICATION FEE (IF REQUIRED) MUST BE PAID WITHIN THREE MONTHS FROM THE MAILING DATE OF THIS NOTICE OR THIS APPLICATION SHALL BE REGARDED AS ABANDONED. THIS STATUTORY PERIOD CANNOT BE EXTENDED. SEE 35 U.S.C. 151. THE ISSUE FEE DUE INDICATED ABOVE DOES NOT REFLECT A CREDIT FOR ANY PREVIOUSLY PAID ISSUE FEE IN THIS APPLICATION. IF AN ISSUE FEE HAS PREVIOUSLY BEEN PAID IN THIS APPLICATION (AS SHOWN ABOVE), THE RETURN OF PART B OF THIS FORM WILL BE CONSIDERED A REQUEST TO REAPPLY THE PREVIOUSLY PAID ISSUE FEE TOWARD THE ISSUE FEE NOW DUE.

HOW TO REPLY TO THIS NOTICE:

I. Review the SMALL ENTITY status shown above.

If the SMALL ENTITY is shown as YES, verify your current SMALL ENTITY status:

A. If the status is the same, pay the TOTAL FEE(S) DUE shown above.

B. If the status above is to be removed, check box 5b on Part B - Fee(s) Transmittal and pay the PUBLICATION FEE (if required) and twice the amount of the ISSUE FEE shown above, or

If the SMALL ENTITY is shown as NO:

A. Pay TOTAL FEE(S) DUE shown above, or

B. If applicant claimed SMALL ENTITY status before, or is now claiming SMALL ENTITY status, check box 5a on Part B - Fee(s) Transmittal and pay the PUBLICATION FEE (if required) and 1/2 the ISSUE FEE shown above.

II. PART B - FEE(S) TRANSMITTAL, or its equivalent, must be completed and returned to the United States Patent and Trademark Office (USPTO) with your ISSUE FEE and PUBLICATION FEE (if required). If you are charging the fee(s) to your deposit account, section "4b" of Part B - Fee(s) Transmittal should be completed and an extra copy of the form should be submitted. If an equivalent of Part B is filed, a request to reapply a previously paid issue fee must be clearly made, and delays in processing may occur due to the difficulty in recognizing the paper as an equivalent of Part B.

III. All communications regarding this application must give the application number. Please direct all communications prior to issuance to Mail Stop ISSUE FEE unless advised to the contrary.

IMPORTANT REMINDER: Utility patents issuing on applications filed on or after Dec. 12, 1980 may require payment of maintenance fees. It is patentee's responsibility to ensure timely payment of maintenance fees when due.

**PART B - FEE(S) TRANSMITTAL**

**Complete and send this form, together with applicable fee(s), to: Mail Mail Stop ISSUE FEE  
 Commissioner for Patents  
 P.O. Box 1450  
 Alexandria, Virginia 22313-1450  
 or Fax (571)-273-2885**

**INSTRUCTIONS:** This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Use Block 1 for any change of address)

34313 7590 12/03/2009

**ORRICK, HERRINGTON & SUTCLIFFE, LLP**  
 IP PROSECUTION DEPARTMENT  
 4 PARK PLAZA  
 SUITE 1600  
 IRVINE, CA 92614-2558

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

**Certificate of Mailing or Transmission**

I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (571) 273-2885, on the date indicated below.

(Depositor's name)
(Signature)
(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
-----------------	-------------	----------------------	---------------------	------------------

11/280,700 11/14/2005 Michael De Angelo 20933-4003 9680

TITLE OF INVENTION: SYSTEM AND METHOD FOR CREATING AND MANIPULATING INFORMATION CONTAINERS WITH DYNAMIC REGISTERS

APPLN. TYPE	SMALL ENTITY	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
-------------	--------------	---------------	---------------------	----------------------	------------------	----------

nonprovisional YES \$755 \$300 \$0 \$1055 03/03/2010

EXAMINER	ART UNIT	CLASS-SUBCLASS
----------	----------	----------------

NGUYEN, CAM LINH T 2161 707-003000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).

- Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.
- "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. **Use of a Customer Number is required.**

2. For printing on the patent front page, list

- (1) the names of up to 3 registered patent attorneys or agents OR, alternatively, 1 \_\_\_\_\_
- (2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed. 2 \_\_\_\_\_
- 3 \_\_\_\_\_

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.11. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE (B) RESIDENCE: (CITY AND STATE OR COUNTRY)

Please check the appropriate assignee category or categories (will not be printed on the patent) :  Individual  Corporation or other private group entity  Government

4a. The following fee(s) are submitted:

- Issue Fee
- Publication Fee (No small entity discount permitted)
- Advance Order - # of Copies \_\_\_\_\_

4b. Payment of Fee(s); (Please first reapply any previously paid issue fee shown above)

- A check is enclosed.
- Payment by credit card. Form PTO-2038 is attached.
- The Director is hereby authorized to charge the required fee(s), any deficiency, or credit any overpayment, to Deposit Account Number \_\_\_\_\_ (enclose an extra copy of this form).

5. Change in Entity Status (from status indicated above)

- a. Applicant claims SMALL ENTITY status. See 37 CFR 1.27.
- b. Applicant is no longer claiming SMALL ENTITY status. See 37 CFR 1.27(g)(2).

NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

Authorized Signature \_\_\_\_\_ Date \_\_\_\_\_

Typed or printed name \_\_\_\_\_ Registration No. \_\_\_\_\_

This collection of information is required by 37 CFR 1.311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, Virginia 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.

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





UNITED STATES PATENT AND TRADEMARK OFFICE

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

Table with 5 columns: APPLICATION NO., FILING DATE, FIRST NAMED INVENTOR, ATTORNEY DOCKET NO., CONFIRMATION NO.
11/280,700 11/14/2005 Michael De Angelo 20933-4003 9680

34313 7590 12/03/2009

ORRICK, HERRINGTON & SUTCLIFFE, LLP
IP PROSECUTION DEPARTMENT
4 PARK PLAZA
SUITE 1600
IRVINE, CA 92614-2558

EXAMINER

NGUYEN, CAM LINH T

ART UNIT PAPER NUMBER

2161

DATE MAILED: 12/03/2009

Determination of Patent Term Adjustment under 35 U.S.C. 154 (b)

(application filed on or after May 29, 2000)

The Patent Term Adjustment to date is 227 day(s). If the issue fee is paid on the date that is three months after the mailing date of this notice and the patent issues on the Tuesday before the date that is 28 weeks (six and a half months) after the mailing date of this notice, the Patent Term Adjustment will be 227 day(s).

If a Continued Prosecution Application (CPA) was filed in the above-identified application, the filing date that determines Patent Term Adjustment is the filing date of the most recent CPA.

Applicant will be able to obtain more detailed information by accessing the Patent Application Information Retrieval (PAIR) WEB site (http://pair.uspto.gov).

Any questions regarding the Patent Term Extension or Adjustment determination should be directed to the Office of Patent Legal Administration at (571)-272-7702. Questions relating to issue and publication fee payments should be directed to the Customer Service Center of the Office of Patent Publication at 1-(888)-786-0101 or (571)-272-4200.

**Notice of Allowability**

<b>Application No.</b> 11/280,700	<b>Applicant(s)</b> ANGELO, MICHAEL DE	
<b>Examiner</b> CAM-LINH NGUYEN	<b>Art Unit</b> 2161	

**-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--**

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

- 1.  This communication is responsive to amendment filed on 11/02/09.
- 2.  The allowed claim(s) is/are 1-18,21-24 and 30.
- 3.  Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
  - a)  All   b)  Some\*   c)  None   of the:
    - 1.  Certified copies of the priority documents have been received.
    - 2.  Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
    - 3.  Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

\* Certified copies not received: \_\_\_\_\_.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application.

**THIS THREE-MONTH PERIOD IS NOT EXTENDABLE.**

- 4.  A SUBSTITUTE OATH OR DECLARATION must be submitted. Note the attached EXAMINER'S AMENDMENT or NOTICE OF INFORMAL PATENT APPLICATION (PTO-152) which gives reason(s) why the oath or declaration is deficient.
  - 5.  CORRECTED DRAWINGS ( as "replacement sheets") must be submitted.
    - (a)  including changes required by the Notice of Draftsperson's Patent Drawing Review ( PTO-948) attached
      - 1)  hereto or 2)  to Paper No./Mail Date \_\_\_\_\_.
    - (b)  including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No./Mail Date \_\_\_\_\_.
- Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet. Replacement sheet(s) should be labeled as such in the header according to 37 CFR 1.121(d).**
- 6.  DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

**Attachment(s)**

- 1.  Notice of References Cited (PTO-892)
- 2.  Notice of Draftsperson's Patent Drawing Review (PTO-948)
- 3.  Information Disclosure Statements (PTO/SB/08),  
Paper No./Mail Date \_\_\_\_\_
- 4.  Examiner's Comment Regarding Requirement for Deposit  
of Biological Material
- 5.  Notice of Informal Patent Application
- 6.  Interview Summary (PTO-413),  
Paper No./Mail Date 12/02/09.
- 7.  Examiner's Amendment/Comment
- 8.  Examiner's Statement of Reasons for Allowance
- 9.  Other \_\_\_\_\_.

/CamLinh Nguyen/  
Primary Examiner, Art Unit 2161

<b>Examiner-Initiated Interview Summary</b>	<b>Application No.</b> 11/280,700	<b>Applicant(s)</b> ANGELO, MICHAEL DE	
	<b>Examiner</b> CAM-LINH NGUYEN	<b>Art Unit</b> 2161	

**All Participants:**

(1) CAM-LINH NGUYEN.

(2) SANJEET K. DUTTA.

**Status of Application:** \_\_\_\_\_

(3) \_\_\_\_\_.

(4) \_\_\_\_\_.

**Date of Interview:** 2 December 2009

**Time:** \_\_\_\_\_

**Type of Interview:**

- Telephonic  
 Video Conference  
 Personal (Copy given to:  Applicant  Applicant's representative)

Exhibit Shown or Demonstrated:  Yes  No

If Yes, provide a brief description:

**Part I.**

Rejection(s) discussed:

*101 Rejection*

Claims discussed:

*1, 18, and 30*

Prior art documents discussed:

*None*

**Part II.**

**SUBSTANCE OF INTERVIEW DESCRIBING THE GENERAL NATURE OF WHAT WAS DISCUSSED:**

*Examiner suggested some amendments to overcome the 101 rejection (by replacing "searching, using the computer, first container registers ...). The Applicant agreed and let the Examiner to do it in the Examiner amendment.*

**Part III.**

- It is not necessary for applicant to provide a separate record of the substance of the interview, since the interview directly resulted in the allowance of the application. The examiner will provide a written summary of the substance of the interview in the Notice of Allowability.  
 It is not necessary for applicant to provide a separate record of the substance of the interview, since the interview did not result in resolution of all issues. A brief summary by the examiner appears in Part II above.

/CamLinh Nguyen/  
Primary Examiner, Art Unit 2161

(Applicant/Applicant's Representative Signature – if appropriate)

Art Unit: 2161

### DETAILED ACTION

1. This Office Action is responsive to communication filed on 11/02/2009 and the result of interview on 12/02/2009.
2. Claims 1 – 18, 21 – 24, and 30 are currently pending.
3. Claims 1 – 18, 21 – 24, 30 are renumbered as 1 – 23 respectively.

### EXAMINER'S AMENDMENT

4. An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

Authorization for this examiner's amendment was given in a telephone interview with Sanjeet K. Dutta on 12/02/2009.

The application has been amended as follows:

◆ REPLACE claim 1 as follow:

-- 1. (Currently Amended) A computer-implemented method comprising:

receiving a search query;

searching, using the computer, first container registers encapsulated and logically defined

in a plurality of containers to identify identified containers responsive to the search query,

the container registers having defined therein data comprising historical data associated

with interactions of the identified containers with other containers from the plurality of

containers, wherein searching the first container registers comprises searching the

Art Unit: 2161

historical data; encapsulating the identified containers in a new container; updating second container registers of the identified containers with data associated with interactions of the identified containers with the new container; and  
providing a list characterizing the identified containers. - -

◆ REPLACE claim 18 as follow:

- - 18. (Currently Amended) A computer-implemented method comprising:  
receiving a search query;  
polling, using the computer, a plurality of gateways to identify registers encapsulated therein, the gateways having a plurality of containers coupled thereto, the identified registers relating to identified containers logically defining data contained therein associated with the search query, the identified containers having container registers defined therein, the container registers containing data comprising historical data associated with interactions of the identified containers with other containers from the plurality of containers, wherein polling the plurality of gateways comprises searching the historical data;

encapsulating the identified containers in a new container;  
updating the container registers of the identified containers with data associated with interactions of the identified containers with the new container; and  
providing a list characterizing the identified containers. - -

◆ REPLACE claim 30 as follow:

- - 30. (Currently Amended) A computer-implemented method comprising:  
receiving a search query;

Art Unit: 2161

Searching, using the computer, first container registers encapsulated and logically defined in a plurality of containers to identify search query templates encapsulated in identified containers, the first container registers having defined therein data comprising historical data associated with interactions of the identified containers with other containers from the plurality of containers, wherein searching container registers comprises searching the historical data; encapsulating the identified containers in a new container; updating second container registers of the identified containers with data associated with interactions of the identified containers with the new container; and providing a list characterizing the identified one or more search query templates to formulate subsequent search queries. - -

***Allowable Subject Matter***

5. Claims 1 – 18, 21 – 24, 30 (original) are allowed.
6. The following is an examiner’s statement of reasons for allowance: in independent claims 1, 18, 21 – 24, and 30, a method and system comprising the teaching of “searching container registers comprises searching the historical data; encapsulating the identified containers in a new container; updating second container registers of the identified containers with data associated with interactions of the identified containers with the new container”, taken with the other limitations of the claim, were not disclosed by, would not have been obvious over, nor otherwise fairly disclosed by the prior art of record.
7. The dependent claims, being further limiting, definite and fully enabled by the Specification, are also allowed.

Art Unit: 2161

Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled “Comments on Statement of Reasons for Allowance.”


***Conclusion***

8. Any inquiry concerning this communication or earlier communications from the examiner should be directed to CamLinh Nguyen whose telephone number is (571) 272 - 4024. The examiner can normally be reached on Monday-Friday.

If attempts to reach the examiner by telephone are unsuccessful, the examiner’s supervisor, Apu Mofiz can be reached on (571) 272 - 4080. The fax phone number for the organization where this application or proceeding is assigned is 571 – 273 - 8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/CamLinh Nguyen/  
Primary Examiner, Art Unit 2161


<b>Issue Classification</b> 	<b>Application/Control No.</b> 11280700	<b>Applicant(s)/Patent Under Reexamination</b> ANGELO, MICHAEL DE
	<b>Examiner</b> CAM-LINH NGUYEN	<b>Art Unit</b> 2161

ORIGINAL				INTERNATIONAL CLASSIFICATION									
CLASS		SUBCLASS		CLAIMED				NON-CLAIMED					
707		758		G	0	6	F	17 / 30 (2006.01.01)					
<b>CROSS REFERENCE(S)</b>													
CLASS	SUBCLASS (ONE SUBCLASS PER BLOCK)												
707	768												

<input type="checkbox"/> <b>Claims renumbered in the same order as presented by applicant</b> <input type="checkbox"/> <b>CPA</b> <input type="checkbox"/> <b>T.D.</b> <input type="checkbox"/> <b>R.1.47</b>															
Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original
1	1	17	17												
2	2	18	18												
3	3		19												
4	4		20												
5	5	19	21												
6	6	20	22												
7	7	21	23												
8	8	22	24												
9	9		25												
10	10		26												
11	11		27												
12	12		28												
13	13		29												
14	14	23	30												
15	15														
16	16														

NONE		<b>Total Claims Allowed:</b>	
		23	
(Assistant Examiner)	(Date)	O.G. Print Claim(s)	O.G. Print Figure
/CAM-LINH NGUYEN/ Primary Examiner.Art Unit 2161	12/02/2009	1	7
(Primary Examiner)	(Date)		



<b>Index of Claims</b> 	<b>Application/Control No.</b> 11280700	<b>Applicant(s)/Patent Under Reexamination</b> ANGELO, MICHAEL DE
	<b>Examiner</b> CAM-LINH NGUYEN	<b>Art Unit</b> 2161

✓	<b>Rejected</b>
=	<b>Allowed</b>


-	<b>Cancelled</b>
÷	<b>Restricted</b>

<b>N</b>	<b>Non-Elected</b>
<b>I</b>	<b>Interference</b>

<b>A</b>	<b>Appeal</b>
<b>O</b>	<b>Objected</b>

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

CLAIM		DATE							
Final	Original	10/06/2008	04/29/2009	12/02/2009					
1	1	✓	✓	=					
2	2	✓	✓	=					
3	3	✓	✓	=					
4	4	✓	✓	=					
5	5	✓	✓	=					
6	6	✓	✓	=					
7	7	✓	✓	=					
8	8	✓	✓	=					
9	9	✓	✓	=					
10	10	✓	✓	=					
11	11	✓	✓	=					
12	12	✓	✓	=					
13	13	✓	✓	=					
14	14	✓	✓	=					
15	15	✓	✓	=					
16	16	✓	✓	=					
17	17	✓	✓	=					
18	18	✓	✓	=					
	19	-	-	-					
	20	-	-	-					
19	21	✓	✓	=					
20	22	✓	✓	=					
21	23	✓	✓	=					
22	24	✓	✓	=					
	25	-	-	-					
	26	-	-	-					
	27	-	-	-					
	28	-	-	-					
	29	-	-	-					
23	30	✓	✓	=					

<b>Search Notes</b>  	<b>Application/Control No.</b>  11280700	<b>Applicant(s)/Patent Under Reexamination</b>  ANGELO, MICHAEL DE
	<b>Examiner</b>  CAM-LINH NGUYEN	<b>Art Unit</b>  2161

<b>SEARCHED</b>			
<b>Class</b>	<b>Subclass</b>	<b>Date</b>	<b>Examiner</b>
707	751,755,758,768	12/01/2009	Linh

<b>SEARCH NOTES</b>		
<b>Search Notes</b>	<b>Date</b>	<b>Examiner</b>
EAST search	10/4/08	Linh
EAST search	4/29/09	Linh
EAST update search	12/01/2009	Linh
Interference search	12/01/2009	Linh

<b>INTERFERENCE SEARCH</b>			
<b>Class</b>	<b>Subclass</b>	<b>Date</b>	<b>Examiner</b>
707	751,755,758,768	12/01/2009	Linh

--	--



UNITED STATES PATENT AND TRADEMARK OFFICE

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

BIB DATA SHEET

CONFIRMATION NO. 9680

<b>SERIAL NUMBER</b> 11/280,700	<b>FILING or 371(c) DATE</b> 11/14/2005 <b>RULE</b>	<b>CLASS</b> 707	<b>GROUP ART UNIT</b> 2161	<b>ATTORNEY DOCKET NO.</b> 20933-4003	
<b>APPLICANTS</b> Michael De Angelo, Palm Springs, CA; <b>** CONTINUING DATA *****</b> This application is a CON of 09/284,113 04/07/1999 PAT 7,010,536 which is a 371 of PCT/US99/01988 01/28/1999 which claims benefit of 60/073,209 01/30/1998 <b>** FOREIGN APPLICATIONS *****</b> <b>** IF REQUIRED, FOREIGN FILING LICENSE GRANTED ** ** SMALL ENTITY **</b> 12/12/2005					
Foreign Priority claimed <input type="checkbox"/> Yes <input checked="" type="checkbox"/> No 35 USC 119(a-d) conditions met <input type="checkbox"/> Yes <input checked="" type="checkbox"/> No Verified and /CAM-LINH T NGUYEN/ Acknowledged Examiner's Signature	<input type="checkbox"/> Met after Allowance LN Initials	<b>STATE OR COUNTRY</b> CA	<b>SHEETS DRAWINGS</b> 30	<b>TOTAL CLAIMS</b> 30 23	<b>INDEPENDENT CLAIMS</b> 42 7
<b>ADDRESS</b> ORRICK, HERRINGTON & SUTCLIFFE, LLP IP PROSECUTION DEPARTMENT 4 PARK PLAZA SUITE 1600 IRVINE, CA 92614-2558 UNITED STATES					
<b>TITLE</b> System and method for creating and manipulating information containers with dynamic registers					
<b>FILING FEE RECEIVED</b> 1715	FEES: Authority has been given in Paper No. _____ to charge/credit DEPOSIT ACCOUNT No. _____ for following:		<input type="checkbox"/> All Fees <input type="checkbox"/> 1.16 Fees (Filing) <input type="checkbox"/> 1.17 Fees (Processing Ext. of time) <input type="checkbox"/> 1.18 Fees (Issue) <input type="checkbox"/> Other _____ <input type="checkbox"/> Credit		

## EAST Search History

## EAST Search History (Prior Art)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L11	2	((container\$1 or folder\$1) same register\$1 same (historical or history) same interaction\$1).clm.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/12/01 15:49
L12	0	11 and @AD<"19980130"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/12/01 15:50
L13	48711	(707/1 or 707/2 or 707/3 or 707/4 or 707/5 or 707/6 or 707/9 or 707/10 or 707/100 or 707/101).ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/12/01 15:51
L14	2	11 and 13	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/12/01 15:52
L15	0	((container\$1 or folder\$1) same register\$1 same (historical or history) same interaction\$1 same search \$3).clm.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/12/01 15:52
L16	0	((container\$1 or folder\$1) same (historical or history) same interaction\$1 same search\$3).clm.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/12/01 15:52
L17	1	( register\$1 same (historical or history) same interaction \$1 same search\$3).clm.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/12/01 15:53
L18	0	13 and 17 and @AD<"19980130"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/12/01 15:53
L19	0	((container\$1 or folder\$1) same register\$1 same interaction\$1 same search \$3).clm.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/12/01 15:54
L20	1	((container\$1 or folder\$1) same register\$1 same (historical or history) same search\$3).clm.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/12/01 15:54

L21	0	13 and 20 and @AD<"19980130"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2009/12/01 15:55
-----	---	---------------------------------	--	----	-----	---------------------

12/1/09 3:55:28 PM

C:\Documents and Settings\CNguyen11\My Documents\EAST\Workspaces\template.wsp

**PART B - FEE(S) TRANSMITTAL**

Complete and send this form, together with applicable fee(s), to: **Mail** Mail Stop ISSUE FEE  
 Commissioner for Patents  
 P.O. Box 1450  
 Alexandria, Virginia 22313-1450  
 or **Fax** (571)-273-2885

**INSTRUCTIONS:** This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Use Block 1 for any change of address)

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

34313 7590 12/03/2009

**ORRICK, HERRINGTON & SUTCLIFFE, LLP**  
 IP PROSECUTION DEPARTMENT  
 4 PARK PLAZA  
 SUITE 1600  
 IRVINE, CA 92614-2558

**Certificate of Mailing or Transmission**

I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (571) 273-2885, on the date indicated below.

_____ (Depositor's name)
_____ (Signature)
_____ (Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/280,700	11/14/2005	Michael De Angelo	20933-4003	9680

TITLE OF INVENTION: SYSTEM AND METHOD FOR CREATING AND MANIPULATING INFORMATION CONTAINERS WITH DYNAMIC REGISTERS

APPLN. TYPE	SMALL ENTITY	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	YES	\$755	\$300	\$0	\$1055	03/03/2010

EXAMINER	ART UNIT	CLASS-SUBCLASS
NGUYEN, CAM LINH T	2161	707-003000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).  
 Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.  
 "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. Use of a Customer Number is required.

2. For printing on the patent front page, list  
 (1) the names of up to 3 registered patent attorneys or agents OR, alternatively,  
 (2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.

**Orrick, Herrington & Sutcliffe LLP**  
 1 \_\_\_\_\_  
 2 \_\_\_\_\_  
 3 \_\_\_\_\_

**3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)**

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.11. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE: **Incandescent, Inc.** (B) RESIDENCE: (CITY and STATE OR COUNTRY) **San Rafael, California**

Please check the appropriate assignee category or categories (will not be printed on the patent):  Individual  Corporation or other private group entity  Government

**4a. The following fee(s) are submitted:**

Issue Fee  
 Publication Fee (No small entity discount permitted)  
 Advance Order - # of Copies \_\_\_\_\_

**4b. Payment of Fee(s): (Please first reapply any previously paid issue fee shown above)**

A check is enclosed.  
 Payment by credit card. Form PTO-2038 is attached.  
 The Director is hereby authorized to charge the required fee(s), any deficiency, or credit any overpayment, to Deposit Account Number **15-0665** (enclose an extra copy of this form).

**5. Change in Entity Status (from status indicated above)**

a. Applicant claims SMALL ENTITY status. See 37 CFR 1.27.  b. Applicant is no longer claiming SMALL ENTITY status. See 37 CFR 1.27(g)(2).

NOTE: The issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant, a registered attorney or agent, or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

Authorized Signature Sanjeet Dutta  
 Typed or printed name **Sanjeet K. Dutta**

Date **March 3, 2010**  
 Registration No. **46,145**

This collection of information is required by 37 CFR 1.311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, Virginia 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.

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

## Electronic Patent Application Fee Transmittal

<b>Application Number:</b>	11280700
<b>Filing Date:</b>	14-Nov-2005
<b>Title of Invention:</b>	SYSTEM AND METHOD FOR CREATING AND MANIPULATING INFORMATION CONTAINERS WITH DYNAMIC REGISTERS
<b>First Named Inventor/Applicant Name:</b>	Michael De Angelo
<b>Filer:</b>	Jeffrey A. Miller/Rita Hernandez
<b>Attorney Docket Number:</b>	20933-4003

Filed as Small Entity

### Utility under 35 USC 111(a) Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD(\$)
<b>Basic Filing:</b>				
<b>Pages:</b>				
<b>Claims:</b>				
<b>Miscellaneous-Filing:</b>				
<b>Petition:</b>				
<b>Patent-Appeals-and-Interference:</b>				
<b>Post-Allowance-and-Post-Issuance:</b>				
Utility Appl issue fee	2501	1	755	755
Publ. Fee- early, voluntary, or normal	1504	1	300	300

Description	Fee Code	Quantity	Amount	Sub-Total in USD(\$)
<b>Extension-of-Time:</b>				
<b>Miscellaneous:</b>				
<b>Total in USD (\$)</b>				<b>1055</b>



## Electronic Acknowledgement Receipt

<b>EFS ID:</b>	7129042
<b>Application Number:</b>	11280700
<b>International Application Number:</b>	
<b>Confirmation Number:</b>	9680
<b>Title of Invention:</b>	SYSTEM AND METHOD FOR CREATING AND MANIPULATING INFORMATION CONTAINERS WITH DYNAMIC REGISTERS
<b>First Named Inventor/Applicant Name:</b>	Michael De Angelo
<b>Customer Number:</b>	34313
<b>Filer:</b>	Jeffrey A. Miller/Rita Hernandez
<b>Filer Authorized By:</b>	Jeffrey A. Miller
<b>Attorney Docket Number:</b>	20933-4003
<b>Receipt Date:</b>	03-MAR-2010
<b>Filing Date:</b>	14-NOV-2005
<b>Time Stamp:</b>	11:26:49
<b>Application Type:</b>	Utility under 35 USC 111(a)

### Payment information:

Submitted with Payment	yes
Payment Type	Deposit Account
Payment was successfully received in RAM	\$1055
RAM confirmation Number	8026
Deposit Account	150665
Authorized User	

The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows:

Charge any Additional Fees required under 37 C.F.R. Section 1.16 (National application filing, search, and examination fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.17 (Patent application and reexamination processing fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.19 (Document supply fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.21 (Miscellaneous fees and charges)

### File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Issue Fee Payment (PTO-85B)	4003_03_03_10.pdf	462590 a231673c72173e3d2243bef35c7ff640559e ebe7	no	1

### Warnings:

### Information:

2	Fee Worksheet (PTO-875)	fee-info.pdf	31971 5f72c262e71aa64a4b6596cd37fcb19e86e3 4805	no	2
---	-------------------------	--------------	---	----	---

### Warnings:

### Information:

**Total Files Size (in bytes):** 494561

**This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.**

#### **New Applications Under 35 U.S.C. 111**

**If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.**

#### **National Stage of an International Application under 35 U.S.C. 371**

**If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.**

#### **New International Application Filed with the USPTO as a Receiving Office**

**If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.**



APPLICATION NO.	ISSUE DATE	PATENT NO.	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/280,700	04/20/2010	7702682	20933-4003	9680

34313 7590 03/31/2010  
 ORRICK, HERRINGTON & SUTCLIFFE, LLP  
 IP PROSECUTION DEPARTMENT  
 4 PARK PLAZA  
 SUITE 1600  
 IRVINE, CA 92614-2558

### ISSUE NOTIFICATION

The projected patent number and issue date are specified above.

**Determination of Patent Term Adjustment under 35 U.S.C. 154 (b)**  
 (application filed on or after May 29, 2000)

The Patent Term Adjustment is 579 day(s). Any patent to issue from the above-identified application will include an indication of the adjustment on the front page.

If a Continued Prosecution Application (CPA) was filed in the above-identified application, the filing date that determines Patent Term Adjustment is the filing date of the most recent CPA.

Applicant will be able to obtain more detailed information by accessing the Patent Application Information Retrieval (PAIR) WEB site (<http://pair.uspto.gov>).

Any questions regarding the Patent Term Extension or Adjustment determination should be directed to the Office of Patent Legal Administration at (571)-272-7702. Questions relating to issue and publication fee payments should be directed to the Application Assistance Unit (AAU) of the Office of Data Management (ODM) at (571)-272-4200.

APPLICANT(s) (Please see PAIR WEB site <http://pair.uspto.gov> for additional applicants):

Michael De Angelo, Palm Springs, CA;

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

Page 1 of 1

PATENT NO. : 7,702,682
APPLICATION NO. : 11/280,700
ISSUE DATE : April 20, 2010
INVENTOR(S) : Michael De Angelo

It is certified that an error appears or errors appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

IN THE SPECIFICATION

Column 18, line 22, replace the number 10 with -110---

Column 23, line 5, replace the number 1 with -11--

MAILING ADDRESS OF SENDER:

PATENT NO. 7,702,682

No. of additional copies

Sanjeet K. Dutta
Orrick, Herrington & Sutcliffe LLP
4 Park Plaza, Suite 1600
Irvine, CA 92614

=> 0

This collection of information is required by 37 CFR 1.322, 1.323, and 1.324. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 USC 122 and 37 CFR 1.14. This collection is estimated to take 1.0 hour to complete, including gathering, preparing and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Attention Certificate of Corrections Branch, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

OHS West:260679567.1
15367-4021 RIH/RIH

## Electronic Acknowledgement Receipt

<b>EFS ID:</b>	7656199
<b>Application Number:</b>	11280700
<b>International Application Number:</b>	
<b>Confirmation Number:</b>	9680
<b>Title of Invention:</b>	SYSTEM AND METHOD FOR CREATING AND MANIPULATING INFORMATION CONTAINERS WITH DYNAMIC REGISTERS
<b>First Named Inventor/Applicant Name:</b>	Michael De Angelo
<b>Customer Number:</b>	34313
<b>Filer:</b>	Jeffrey A. Miller/Rita Hernandez
<b>Filer Authorized By:</b>	Jeffrey A. Miller
<b>Attorney Docket Number:</b>	20933-4003
<b>Receipt Date:</b>	20-MAY-2010
<b>Filing Date:</b>	14-NOV-2005
<b>Time Stamp:</b>	17:57:35
<b>Application Type:</b>	Utility under 35 USC 111(a)

### Payment information:

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

### File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Request for Certificate of Correction	4003_Cert_of_Correction.pdf	197031 <small>019d8214271dd067558b086014175703de76a3f7</small>	no	1

### Warnings:

### Information:

**This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.**

**New Applications Under 35 U.S.C. 111**

**If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.**

**National Stage of an International Application under 35 U.S.C. 371**

**If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.**

**New International Application Filed with the USPTO as a Receiving Office**

**If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.**

UNITED STATES PATENT AND TRADEMARK OFFICE  
**CERTIFICATE OF CORRECTION**

PATENT NO. : 7,702,682 B2  
APPLICATION NO. : 11/280700  
DATED : April 20, 2010  
INVENTOR(S) : Michael De Angelo

Page 1 of 1

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

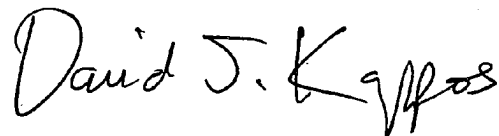
**IN THE SPECIFICATION**

Column 18, line 22, replace the number 10 with --110--

Column 23, line 5, replace the number 1 with --11--

Signed and Sealed this

Twenty-ninth Day of June, 2010

A handwritten signature in black ink that reads "David J. Kappos". The signature is written in a cursive, slightly slanted style.

David J. Kappos  
*Director of the United States Patent and Trademark Office*

**PATENT ASSIGNMENT**

Electronic Version v1.1  
 Stylesheet Version v1.1

<b>SUBMISSION TYPE:</b>	NEW ASSIGNMENT
<b>NATURE OF CONVEYANCE:</b>	ASSIGNMENT
<b>CONVEYING PARTY DATA</b>	
<b>Name</b>	<b>Execution Date</b>
Incandescent, Inc.	07/11/2012
<b>RECEIVING PARTY DATA</b>	
<b>Name:</b>	Evolutionary Intelligence, LLC
<b>Street Address:</b>	c/o Gutride Safier LLP, 835 Douglass Street
<b>City:</b>	San Francisco
<b>State/Country:</b>	CALIFORNIA
<b>Postal Code:</b>	94114
<b>PROPERTY NUMBERS Total: 3</b>	
<b>Property Type</b>	<b>Number</b>
Patent Number:	7010536
Patent Number:	7702682
Patent Number:	7873682
<b>CORRESPONDENCE DATA</b>	
<b>Fax Number:</b>	4154496469
<i>Correspondence will be sent via US Mail when the fax attempt is unsuccessful.</i>	
<b>Phone:</b>	415-789-6390
<b>Email:</b>	todd@gutridesafier.com
<b>Correspondent Name:</b>	Todd Kennedy
<b>Address Line 1:</b>	835 Douglass Street
<b>Address Line 4:</b>	San Francisco, CALIFORNIA 94114
<b>NAME OF SUBMITTER:</b>	Todd Kennedy
	This document serves as an Oath/Declaration (37 CFR 1.63).
<b>Total Attachments: 2</b>	
source=patent assignment executed and notarized#page1.tif	
source=patent assignment executed and notarized#page2.tif	

OP \$120.00 7010536



## PATENT ASSIGNMENT

Inventor: Michael De Angelo  
U.S. Patent Nos.: 7,010,536; 7,702,682; 7,873,682  
Application Nos.: 09/284,113; 11/280,700; 12/691,425  
Filing Dates: 1/28/99; 11/14/05; 1/21/10  
Titles: "System and method for creating and manipulating information containers with dynamic registers"

WHEREAS, Incandescent, Inc., a Delaware corporation having an office and place of business at 100 Pine Street, San Francisco, California 94111 (hereinafter referred to as "ASSIGNOR") is the current assignee of the U.S. Patents listed above (hereinafter referred to as the "PATENTS"); and

WHEREAS, Evolutionary Intelligence, LLC, a Delaware limited liability company having an office and place of business at 100 Pine Street, Suite 500, San Francisco, California 94111 (hereinafter referred to as "ASSIGNEE"), is desirous of acquiring the entire right, title and interest throughout the world in and to the PATENTS and in and to any letters patent that may be granted therefor in the United States and in any and all foreign countries.

NOW, THEREFORE, in exchange for good and valuable consideration, the receipt and sufficiency of which is hereby acknowledged, ASSIGNOR hereby assigns and transfers unto ASSIGNEE, the entire right, title and interest throughout the world in and to the PATENTS, the inventions described therein, and any and all letters patent which may be granted or have been granted, including all potential, existing, and future causes of action and associated past, present and future damages, for said inventions in the United States of America and its territorial possessions and in any and all foreign countries, and in any and all divisions, reissues, continuations, continuations-in-part, and certificates of correction thereof, including the right to file domestic and foreign applications directly in the name of ASSIGNEE and to claim priority rights deriving from the applications leading to the PATENTS, said inventions, PATENTS, and all other letters patent deriving from said invention to be held and enjoyed by ASSIGNEE and its successors and assigns for their use and benefit and of their successors and assigns as fully and entirely as the same would have been held and enjoyed by ASSIGNOR had this assignment not been made.

ASSIGNOR hereby authorizes and requests the Commissioner of Patents and Trademarks to issue all letters patent on said inventions to ASSIGNEE. ASSIGNOR warrants that ASSIGNOR is the rightful owner of the PATENTS and said invention, and that there are no others who could make a claim against the rights being assigned, and that the rights being assigned are subsisting and are not assigned, licensed, or otherwise diluted in any way. ASSIGNOR further agrees to execute and deliver any further papers and do such other acts as may be necessary and proper to vest full title in and to the PATENTS in the ASSIGNEE. ASSIGNOR further agrees to execute all instruments and documents required for the making and prosecution of foreign and domestic applications for letters patent on said inventions, and for litigation regarding said letters patent. ASSIGNOR also covenants that, if there are any disputes, actions, litigations, trials, or any other

challenges related to the rights being assigned, then ASSIGNOR shall assist ASSIGNEE to the best of ASSIGNOR's ability.

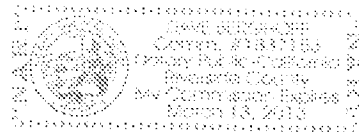
IN WITNESS WHEREOF, ASSIGNOR has caused these presents to be duly executed in a manner appropriate thereto this 9th day of July, 2012.

**ASSIGNOR:**  
Incandescent, Inc.

By: *Michael De Angelo*  
Michael De Angelo  
CEO

Signed before me this 11<sup>th</sup> day of July, 2012.

*Tom Beyliff*  
Notary Public



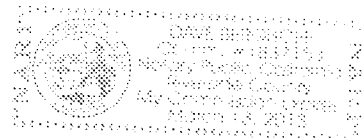
**ASSIGNEE hereby accepts this assignment.**

**ASSIGNEE:**  
Evolutionary Intelligence, LLC

By: *Michael De Angelo*  
Michael De Angelo  
Manager

Signed before me this 11<sup>th</sup> day of July, 2012.

*Tom Beyliff*  
Notary Public



AO 120 (Rev. 08/10)

<b>TO:</b> <b>Mail Stop 8</b> <b>Director of the U.S. Patent and Trademark Office</b> <b>P.O. Box 1450</b> <b>Alexandria, VA 22313-1450</b>	<b>REPORT ON THE</b> <b>FILING OR DETERMINATION OF AN</b> <b>ACTION REGARDING A PATENT OR</b> <b>TRADEMARK</b>
--	---

In Compliance with 35 U.S.C. § 290 and/or 15 U.S.C. § 1116 you are hereby advised that a court action has been filed in the U.S. District Court Eastern District of Texas on the following

Trademarks or  Patents. (  the patent action involves 35 U.S.C. § 292.):

DOCKET NO. 6:12-cv-794	DATE FILED 10/17/2012	U.S. DISTRICT COURT Eastern District of Texas
PLAINTIFF Evolutionary Intelligence, LLC		DEFENDANT Yelp, Inc.
PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK
1 7,010,536	3/7/2006	Evolutionary Intelligence, LLC
2 7,702,682	4/20/2010	Evolutionary Intelligence, LLC
3		
4		
5		

In the above—entitled case, the following patent(s)/ trademark(s) have been included:

DATE INCLUDED	INCLUDED BY <input type="checkbox"/> Amendment <input type="checkbox"/> Answer <input type="checkbox"/> Cross Bill <input type="checkbox"/> Other Pleading		
PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK	
1			
2			
3			
4			
5			

In the above—entitled case, the following decision has been rendered or judgement issued:

DECISION/JUDGEMENT
--------------------

CLERK	(BY) DEPUTY CLERK	DATE
-------	-------------------	------

Copy 1—Upon initiation of action, mail this copy to Director    Copy 3—Upon termination of action, mail this copy to Director  
 Copy 2—Upon filing document adding patent(s), mail this copy to Director    Copy 4—Case file copy



AO 120 (Rev. 08/10)

<b>TO: Mail Stop 8 Director of the U.S. Patent and Trademark Office P.O. Box 1450 Alexandria, VA 22313-1450</b>	<b>REPORT ON THE FILING OR DETERMINATION OF AN ACTION REGARDING A PATENT OR TRADEMARK</b>
---	---

In Compliance with 35 U.S.C. § 290 and/or 15 U.S.C. § 1116 you are hereby advised that a court action has been filed in the U.S. District Court Eastern District of Texas on the following

Trademarks or  Patents. (  the patent action involves 35 U.S.C. § 292.):

DOCKET NO. 6:12-cv-784	DATE FILED 10/17/2012	U.S. DISTRICT COURT Eastern District of Texas
PLAINTIFF Evolutionary Intelligence, LLC		DEFENDANT Facebook, Inc.
PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK
1 7,010,536	3/7/2006	Evolutionary Intelligence, LLC
2 7,702,682	4/20/2010	Evolutionary Intelligence, LLC
3		
4		
5		

In the above—entitled case, the following patent(s)/ trademark(s) have been included:

DATE INCLUDED	INCLUDED BY <input type="checkbox"/> Amendment <input type="checkbox"/> Answer <input type="checkbox"/> Cross Bill <input type="checkbox"/> Other Pleading		
PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK	
1			
2			
3			
4			
5			

In the above—entitled case, the following decision has been rendered or judgement issued:

DECISION/JUDGEMENT
--------------------

CLERK	(BY) DEPUTY CLERK	DATE
-------	-------------------	------

Copy 1—Upon initiation of action, mail this copy to Director    Copy 3—Upon termination of action, mail this copy to Director  
Copy 2—Upon filing document adding patent(s), mail this copy to Director    Copy 4—Case file copy

AO 120 (Rev. 08/10)

<b>TO:</b> <b>Mail Stop 8</b> <b>Director of the U.S. Patent and Trademark Office</b> <b>P.O. Box 1450</b> <b>Alexandria, VA 22313-1450</b>	<b>REPORT ON THE</b> <b>FILING OR DETERMINATION OF AN</b> <b>ACTION REGARDING A PATENT OR</b> <b>TRADEMARK</b>
--	---

In Compliance with 35 U.S.C. § 290 and/or 15 U.S.C. § 1116 you are hereby advised that a court action has been filed in the U.S. District Court Eastern District of Texas on the following

Trademarks or  Patents. (  the patent action involves 35 U.S.C. § 292.):

DOCKET NO. 6:12-cv-785	DATE FILED 10/17/2012	U.S. DISTRICT COURT Eastern District of Texas
PLAINTIFF Evolutionary Intelligence, LLC		DEFENDANT Foursquare Labs, Inc.
PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK
1 7,010,536	3/7/2006	Evolutionary Intelligence, LLC
2 7,702,682	4/20/2010	Evolutionary Intelligence, LLC
3		
4		
5		

In the above—entitled case, the following patent(s)/ trademark(s) have been included:

DATE INCLUDED	INCLUDED BY <input type="checkbox"/> Amendment <input type="checkbox"/> Answer <input type="checkbox"/> Cross Bill <input type="checkbox"/> Other Pleading		
PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK	
1			
2			
3			
4			
5			

In the above—entitled case, the following decision has been rendered or judgement issued:

DECISION/JUDGEMENT
--------------------

CLERK	(BY) DEPUTY CLERK	DATE
-------	-------------------	------

Copy 1—Upon initiation of action, mail this copy to Director    Copy 3—Upon termination of action, mail this copy to Director  
 Copy 2—Upon filing document adding patent(s), mail this copy to Director    Copy 4—Case file copy



AO 120 (Rev. 08/10)

<b>TO: Mail Stop 8</b> <b>Director of the U.S. Patent and Trademark Office</b> <b>P.O. Box 1450</b> <b>Alexandria, VA 22313-1450</b>	<b>REPORT ON THE</b> <b>FILING OR DETERMINATION OF AN</b> <b>ACTION REGARDING A PATENT OR</b> <b>TRADEMARK</b>
---	---

In Compliance with 35 U.S.C. § 290 and/or 15 U.S.C. § 1116 you are hereby advised that a court action has been filed in the U.S. District Court Eastern District of Texas on the following

Trademarks or  Patents. (  the patent action involves 35 U.S.C. § 292.):

DOCKET NO. 6:12-cv-789	DATE FILED 10/17/2012	U.S. DISTRICT COURT Eastern District of Texas
PLAINTIFF Evolutionary Intelligence, LLC		DEFENDANT Livingsocial, Inc.
PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK
1 7,010,536	3/7/2006	Evolutionary Intelligence, LLC
2 7,702,682	4/20/2010	Evolutionary Intelligence, LLC
3		
4		
5		

In the above—entitled case, the following patent(s)/ trademark(s) have been included:

DATE INCLUDED	INCLUDED BY <input type="checkbox"/> Amendment <input type="checkbox"/> Answer <input type="checkbox"/> Cross Bill <input type="checkbox"/> Other Pleading		
PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK	
1			
2			
3			
4			
5			

In the above—entitled case, the following decision has been rendered or judgement issued:

DECISION/JUDGEMENT
--------------------

CLERK	(BY) DEPUTY CLERK	DATE
-------	-------------------	------

Copy 1—Upon initiation of action, mail this copy to Director    Copy 3—Upon termination of action, mail this copy to Director  
 Copy 2—Upon filing document adding patent(s), mail this copy to Director    Copy 4—Case file copy



AO 120 (Rev. 08/10)

TO: <b>Mail Stop 8</b> <b>Director of the U.S. Patent and Trademark Office</b> <b>P.O. Box 1450</b> <b>Alexandria, VA 22313-1450</b>	<b>REPORT ON THE</b> <b>FILING OR DETERMINATION OF AN</b> <b>ACTION REGARDING A PATENT OR</b> <b>TRADEMARK</b>
---	---

In Compliance with 35 U.S.C. § 290 and/or 15 U.S.C. § 1116 you are hereby advised that a court action has been filed in the U.S. District Court Eastern District of Texas on the following

Trademarks or  Patents. (  the patent action involves 35 U.S.C. § 292.):

DOCKET NO. 6:12-cv-790	DATE FILED 10/17/2012	U.S. DISTRICT COURT Eastern District of Texas
PLAINTIFF Evolutionary Intelligence, LLC		DEFENDANT Millennial Media, Inc.
PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK
1 7,010,536	3/7/2006	Evolutionary Intelligence, LLC
2 7,702,682	4/20/2010	Evolutionary Intelligence, LLC
3		
4		
5		

In the above—entitled case, the following patent(s)/ trademark(s) have been included:

DATE INCLUDED	INCLUDED BY <input type="checkbox"/> Amendment <input type="checkbox"/> Answer <input type="checkbox"/> Cross Bill <input type="checkbox"/> Other Pleading		
PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK	
1			
2			
3			
4			
5			

In the above—entitled case, the following decision has been rendered or judgement issued:

DECISION/JUDGEMENT
--------------------

CLERK	(BY) DEPUTY CLERK	DATE
-------	-------------------	------

Copy 1—Upon initiation of action, mail this copy to Director Copy 3—Upon termination of action, mail this copy to Director  
 Copy 2—Upon filing document adding patent(s), mail this copy to Director Copy 4—Case file copy

AO 120 (Rev. 08/10)

<b>TO:</b> <b>Mail Stop 8</b> <b>Director of the U.S. Patent and Trademark Office</b> <b>P.O. Box 1450</b> <b>Alexandria, VA 22313-1450</b>	<b>REPORT ON THE</b> <b>FILING OR DETERMINATION OF AN</b> <b>ACTION REGARDING A PATENT OR</b> <b>TRADEMARK</b>
--	---

In Compliance with 35 U.S.C. § 290 and/or 15 U.S.C. § 1116 you are hereby advised that a court action has been filed in the U.S. District Court Eastern District of Texas on the following

Trademarks or  Patents. (  the patent action involves 35 U.S.C. § 292.):

DOCKET NO. 6:12-cv-791	DATE FILED 10/17/2012	U.S. DISTRICT COURT Eastern District of Texas
PLAINTIFF Evolutionary Intelligence, LLC		DEFENDANT Sprint Nextel Corporation, Sprint Communications Company L.P., Sprint Spectrum, L.P., Sprint Solutions, Inc.
PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK
1 7,010,536	3/7/2006	Evolutionary Intelligence, LLC
2 7,702,682	4/20/2010	Evolutionary Intelligence, LLC
3		
4		
5		

In the above—entitled case, the following patent(s)/ trademark(s) have been included:

DATE INCLUDED	INCLUDED BY <input type="checkbox"/> Amendment <input type="checkbox"/> Answer <input type="checkbox"/> Cross Bill <input type="checkbox"/> Other Pleading		
PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK	
1			
2			
3			
4			
5			

In the above—entitled case, the following decision has been rendered or judgement issued:

DECISION/JUDGEMENT
--------------------

CLERK	(BY) DEPUTY CLERK	DATE
-------	-------------------	------

Copy 1—Upon initiation of action, mail this copy to Director    Copy 3—Upon termination of action, mail this copy to Director  
 Copy 2—Upon filing document adding patent(s), mail this copy to Director    Copy 4—Case file copy

AO 120 (Rev. 08/10)

<b>TO:</b> <b>Mail Stop 8</b> <b>Director of the U.S. Patent and Trademark Office</b> <b>P.O. Box 1450</b> <b>Alexandria, VA 22313-1450</b>	<b>REPORT ON THE</b> <b>FILING OR DETERMINATION OF AN</b> <b>ACTION REGARDING A PATENT OR</b> <b>TRADEMARK</b>
--	---

In Compliance with 35 U.S.C. § 290 and/or 15 U.S.C. § 1116 you are hereby advised that a court action has been filed in the U.S. District Court Eastern District of Texas on the following

Trademarks or  Patents. (  the patent action involves 35 U.S.C. § 292.):

DOCKET NO. 6:12-cv-792	DATE FILED 10/17/2012	U.S. DISTRICT COURT Eastern District of Texas
PLAINTIFF Evolutionary Intelligence, LLC		DEFENDANT Twitter, Inc.
PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK
1 7,010,536	3/7/2006	Evolutionary Intelligence, LLC
2 7,702,682	4/20/2010	Evolutionary Intelligence, LLC
3		
4		
5		

In the above—entitled case, the following patent(s)/ trademark(s) have been included:

DATE INCLUDED	INCLUDED BY <input type="checkbox"/> Amendment <input type="checkbox"/> Answer <input type="checkbox"/> Cross Bill <input type="checkbox"/> Other Pleading		
PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK	
1			
2			
3			
4			
5			

In the above—entitled case, the following decision has been rendered or judgement issued:

DECISION/JUDGEMENT
--------------------

CLERK	(BY) DEPUTY CLERK	DATE
-------	-------------------	------

Copy 1—Upon initiation of action, mail this copy to Director    Copy 3—Upon termination of action, mail this copy to Director  
 Copy 2—Upon filing document adding patent(s), mail this copy to Director    Copy 4—Case file copy