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

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2400160 C 2013/12/17

(11)(21) 2 400 160

(12) BREVET CANADIEN
CANADIAN PATENT

(13) C

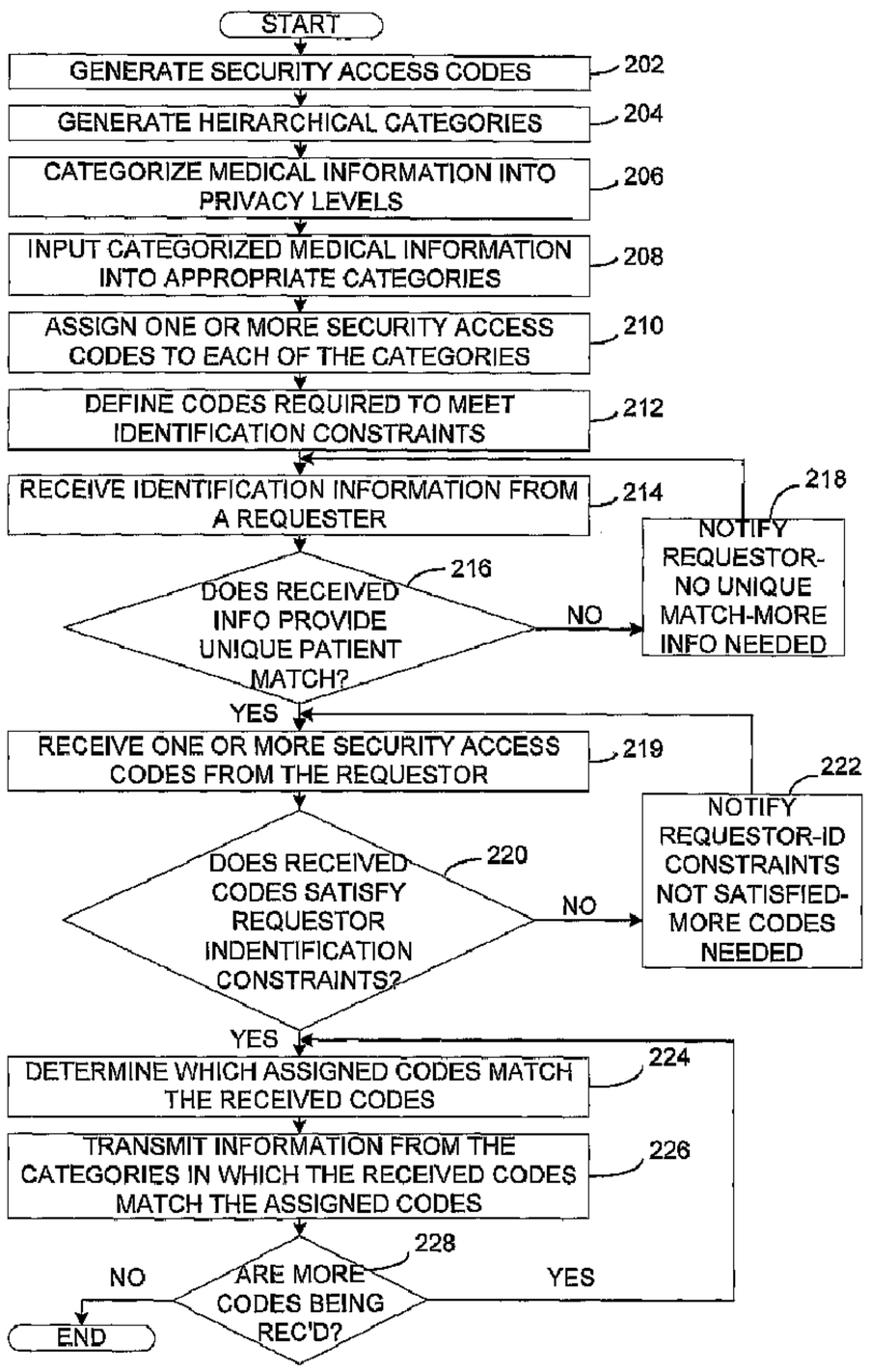
(86) Date de dépôt PCT/PCT Filing Date: 2001/02/22
(87) Date publication PCT/PCT Publication Date: 2001/08/30
(45) Date de délivrance/Issue Date: 2013/12/17
(85) Entrée phase nationale/National Entry: 2002/08/14
(86) N° demande PCT/PCT Application No.: US 2001/006001
(87) N° publication PCT/PCT Publication No.: 2001/063538
(30) Priorités/Priorities: 2000/02/22 (US60/183,857);
2000/04/25 (US09/557,724)

(51) Cl.Int./Int.Cl. *G06Q 50/22* (2012.01)
(72) Inventeur/Inventor:
SCHOENBERG, ROY, US

(73) Propriétaire/Owner:
TRIZETTO CORPORATION, US

(74) Agent: RICHES, MCKENZIE & HERBERT LLP

(54) Titre : PROCEDE ET SYSTEME DE TRANSMISSION D'INFORMATIONS MEDICALES
(54) Title: METHOD AND SYSTEM FOR DISTRIBUTING HEALTH INFORMATION



(57) Abrégé/Abstract:
A method of and system for distributing medical information for an individual over a communications network is disclosed. The method includes the steps of generating a plurality of security access codes (202), generating a plurality of hierarchical categories

CA 2400160 C 2013/12/17

(11)(21) 2 400 160

(13) C

(57) Abrégé(suite)/Abstract(continued):
ranging from a low security category to a high security category (204), categorizing the individual's medical information into privacy levels ranging from a least private level to a most private level (206), inputting the individual's categorized medical information into the plurality of hierarchical categories (208), the least private level being input into the low security category and the most private level being input into the high security category and assigning, to each of the categories, one or more of the access security codes, such that the medical information in each category will be released only if the assigned access security codes are received (214).

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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
30 August 2001 (30.08.2001)

PCT

(10) International Publication Number
WO 01/63538 A1

(51) International Patent Classification[7]: G06F 17/60

(21) International Application Number: PCT/US01/06001

(22) International Filing Date: 22 February 2001 (22.02.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/183,857 22 February 2000 (22.02.2000) US
09/557,724 25 April 2000 (25.04.2000) US

(71) Applicant: **CAREKEY.COM, INC.** [IL/US]; 5th floor, 137 Newbury Street, Boston, MA 02116 (US).

(72) Inventor: **SCHOENBERG, Roy**; 199 Massachusetts Avenue, Boston, MA 02115 (US).

(74) Agents: **LAPPIN, Mark, G.** et al.; McDermott, Will & Emery, 28 State Street, Boston, MA 02109-1775 (US).

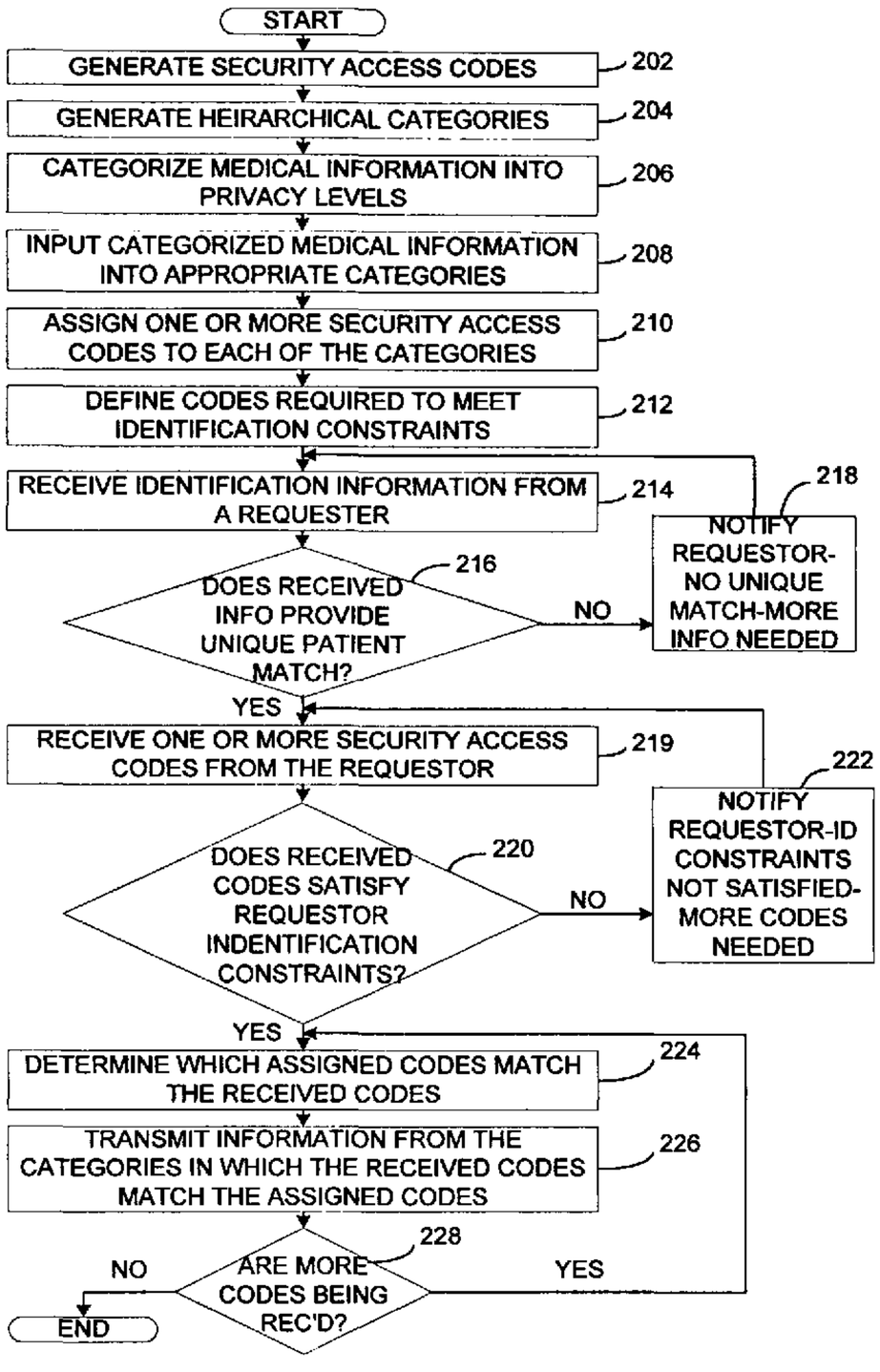
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE,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

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, 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, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

[Continued on next page]

(54) Title: METHOD AND SYSTEM FOR DISTRIBUTING HEALTH INFORMATION



(57) Abstract: A method of and system for distributing medical information for an individual over a communications network is disclosed. The method includes the steps of generating a plurality of security access codes (202), generating a plurality of hierarchical categories ranging from a low security category to a high security category (204), categorizing the individual's medical information into privacy levels ranging from a least private level to a most private level (206), inputting the individual's categorized medical information into the plurality of hierarchical categories (208), the least private level being input into the low security category and the most private level being input into the high security category and assigning, to each of the categories, one or more of the access security codes, such that the medical information in each category will be released only if the assigned access security codes are received (214).

WO 01/63538 A1

APPLICATION

FOR

UNITED STATES LETTERS PATENT

TO ALL WHOM IT MAY CONCERN:

Be it known that **Roy Schoenberg** has invented **METHOD AND SYSTEM FOR DISTRIBUTING HEALTH INFORMATION** of which the following description in connection with the accompanying drawings is a specification.

RELATED APPLICATIONS

This application claims the benefit of United States Provisional Patent Application No. 60/183,857, filed February 22, 2000.

FIELD OF THE INVENTION

This invention generally relates to a method of and system for distributing medical information over a communications network, and more specifically to a method of and system for assigning increasing levels of security to portions of an individual's medical records and linking each of the security levels to access security codes that must be supplied by the requestor of the medical information in order to access the medical records.

BACKGROUND OF THE INVENTION

When a patient is brought into a hospital for emergency care, it is very unlikely that the patient's medical record will be present in the hospital. A patient's medical record is very important, particularly in an emergency situation, as it typically contains information regarding the patient's blood type, allergies, medical history, etc. Typically, such records are at the location where the patient receives the majority of his or her medical care. In most cases, this is the location of the patient's primary care physician, thus making quick access to the record by the emergency care provider virtually impossible. Furthermore, even if the patient's medical record is accessible, it is likely that much of the information in the record is scattered between several archives in various locations; is obsolete, redundant or indecipherable to the extent that it does not benefit the patient at the point of care.

Presently, the transfer of patients' medical records between care providers is done in a number of different ways. Records can be transferred by phone, facsimile and overnight mail, however, these options are relatively slow, expensive and can be unreliable. The use of email for transferring medical records can be relatively simple and quick. However, email is typically too insecure for transferring the sensitive information contained in a patient's medical record, and information can only be exchanged between parties that are aware of each other's email addresses.

Smart cards, which contain memory devices in which a patient's medical data is stored, allow the patient to carry his or her medical records, thereby potentially enabling immediate access to the patient's record. However, the cards are easily lost or misplaced, thus endangering the securing of the record, and smart cards must be compatible with the smart card reader at a particular medical location, which may not always be the case. Furthermore, since the smart card must be physically present at the time the information is needed, remote consultation is impossible. For example, if an ambulance is bringing a patient to the hospital, the information contained in the smart card cannot be accessed by care providers at the hospital until the patient arrives. A further disadvantage of the above methods is that they generally do not permit only selective access to the patient's information, depending on the situation that has precipitated the need for the patient's medical data. For example, if the patient suffers a broken bone, while information regarding the patient's blood type and allergies might be necessary for the proper treatment of the injury, the patient's cardiological or serological data is not. None of the above methods can prevent unnecessary medical data from being divulged to the medical care provider, thus potentially risking the patient's privacy.

While the internet could be used to distribute medical records, there is presently no online system that is capable of securely distributing only the information from a patient's medical record that is necessary for the situation that has required access to the record.

Accordingly, it is an object of this invention to provide a method of and system for distributing medical information in which the medical care provider has quick access to a patient's medical record, but only to the information within the medical record that is necessary for the proper treatment of the patient at that time.

SUMMARY OF THE INVENTION

The present invention is directed to a method of and system for distributing medical information over a communications network. A patient's medical record is divided into a hierarchy of categories, each category having a level of privacy associated therewith which is greater than the previous level. The lowest level category could include information such as blood type and allergies, while a high-

level category could include the patient's HIV status. The patient constructs a list of access codes, wherein, the higher the level of the category, the more access codes are required to gain access to the category of the medical record. This enables the patient to control how much access to his or her medical records a particular medical care provider has, by selecting the access codes that are provided to the care provider.

According to the invention, a method for distributing health information includes the steps of generating health data representative of at least one set of health information for an individual, generating access priority data representative of an access priority associated with each of the at least one set of health information, the access priority being based on criteria for release authorization established by the individual, storing at a datastore the health data and associated access priority data and receiving from a requestor, by way of a communications network, a request for at least one of the sets of health information, the request including access data correlated to an access priority. The method further includes the steps of processing the access data to determine whether the access data corresponds to the access priority criteria for the requested health information. When the access data corresponds to the access criteria for the requested health information, the requested health information is transmitted to the requestor by way of the communications network, and, when the access data fails to correspond to the access criteria, access to the requestor to the health information is denied. The communications network may be the Internet and the transmitted health information may be encrypted.

According to another embodiment of the invention, a method of distributing medical information for an individual over a communications network includes the steps of generating a plurality of security access codes, generating a plurality of hierarchical categories, ranging from a low security category to a high security category, categorizing the individual's medical information into privacy levels ranging from a least private level to a most private level, inputting the individual's categorized medical information into the plurality of hierarchical categories, the least private level being input into the low security category and the most private level being input into the high security category and assigning, to each of the categories, one or more of the access security codes, such that the medical

information in each category will be released only if the assigned access security codes are received. The method further includes the steps of receiving, from a requestor, one or more of the access security codes over the communications network, determining whether the received access security codes match one or more of the assigned access security codes and transmitting, to the requestor over the communications network, the medical information in the categories in which the received security access codes match the assigned security access codes.

A system for distributing medical information for an individual over a communications network according to the present invention includes a server system including a computer processor and associated memory, the server system having database of a plurality or hierarchical categories for the individual, the categories ranging from a low security category to a high security category, each of the categories having medical information of the information contained therein. The medical information ranges from least private information to most private information, the least private medical information being contained in the low security category and the most private medical information being contained in the high security category. Each of the categories have one or more security access codes assigned thereto. The system further includes a request system including a computer processor and associated memory, the requestor system inputting one or more of the security access codes to the server system over the communications network and an access determining device for transmitting, to the requestor system, the medical information in each of the categories in which the input security access codes match the assigned security access codes. The system may further include a setup system, having a computer processor and associated memory, for inputting the medical information to the database. The security access codes may be defined by a user and assigned to the categories by the user through the setup system. More of the security access codes may be required to access the high security categories than the low security categories. The setup system and the requestor system may be the same system. The setup system may be coupleable to said network by a wired or a wireless connection, and may be a personal computer, an interactive television system, a personal digital assistant or a cellular telephone.

In another aspect, the present invention provides a method of distributing health information, comprising the steps of:

A. categorizing health information for an individual into a plurality of hierarchical sets of health information;

B. assigning by said individual access priority data representative of an access priority level to each of said plurality of sets of health information in said hierarchy, said access priority levels being based on differing criteria for release authorization for each of said plurality of sets of health information established by said individual;

C. storing, at a datastore, each of said plurality of sets of health information in said hierarchy and associated access priority data;

D. providing, by said individual to one or more requesters, access priority data corresponding to a desired level in said hierarchy;

E. receiving, from a requestor, by way of a communications network, a request for at least one of said plurality of sets of health information in said hierarchy, said request including access priority data correlated to an access priority level;

F. processing said access priority data to determine whether said access priority data corresponds to said access priority level for said requested health information; and

i. when said access priority data corresponds to said access priority level for said requested health information, transmitting said requested health information to said requester by way of said communications network, and

ii. when said access data fails to correspond to said access priority level, denying access to said requestor to said health information.

In another aspect, the present invention provides a method of distributing medical information for an individual over a communications network comprising the steps of:

A. generating a plurality of access security codes;

B. generating a plurality of hierarchical categories, ranging from a low security category to a high security category;

C. categorizing the individual's medical information into privacy levels ranging from a least private level to a most private level;

D. inputting the individual's categorized medical information into said plurality of hierarchical categories, said least private level being input into said low security category and said most private level being input into said high security category;

E. assigning by said individual, to each of said categories, one or more of said access security codes, such that said medical information in each category will be released only if the assigned access security codes are received;

F. providing, by said individual to one or more requestors, access priority data corresponding to a desired level in said hierarchy;

G. receiving, from a requestor, one or more of said access security codes over said communications network;

H. determining whether said received access security codes match one or more of said assigned access security codes; and

I. transmitting, to said requestor over said communications network, said medical information in said categories in which said received security access codes match said assigned security access codes.

In another aspect, the present invention provides a method of distributing personal information, comprising the steps of:

A. categorizing personal information for an individual into a plurality of hierarchical sets of personal information;

B. assigning, by said individual, access priority data representative of an access priority level to each of said plurality of sets of personal information in said hierarchy, said access priority levels being based on differing criteria for release authorization for each of said plurality of sets of personal information established by said individual;

C. storing, at a datastore, each of said plurality of sets of personal information in said hierarchy and associated access priority data; providing, by said individual to one or more requesters, access priority data corresponding to a desired level in said hierarchy;

D. receiving from a requestor, by way of a communications network, a request for at least one of said plurality of sets of personal information in said hierarchy, said request including access priority data correlated to an access priority level;

D. processing said access priority data to determine whether said access priority data corresponds to said access priority level for said requested personal information; and

(i.) when said access priority data corresponds to said access priority level for said requested personal information, transmitting said requested personal information to said requestor by way of said communications network, and

(ii.) when said access data fails to correspond to said access priority level, denying access to said requester to said personal information.

In another aspect, the present invention provides a method of distributing health information, comprising the steps of:

A. categorizing health information for an individual into a plurality of hierarchical sets of health information;

B. assigning by said individual access priority data representative of an access priority level to each of said plurality of sets of health information in said hierarchy, said access priority levels being based on differing criteria for release authorization for each of said plurality of sets of health information established by said individual;

C. storing, at a datastore, each of said plurality of sets of health information in said hierarchy and associated access priority data;

D. providing, by said individual to one or more requesters, access priority data corresponding to a desired level in said hierarchy;

E. receiving, from a requestor, by way of a communications network, a request for at least one of said plurality of sets of health information in said hierarchy, said

request including access priority data correlated to an access priority level;

F. processing said access priority data to determine whether said access priority data corresponds to said access priority level for said requested health information; and

i. when said access priority data corresponds to said access priority level for said requested health information, transmitting said requested health information to said requester by way of said communications network, and

ii. when said access data fails to correspond to said access priority level, denying access to said requestor to said health information.

In another aspect, the present invention provides a method of distributing medical information for an individual over a communications network comprising the steps of:

A. generating a plurality of access security codes;

B. generating a plurality of hierarchical categories, ranging from a low security category to a high security category;

C. categorizing the individual's medical information into privacy levels ranging from a least private level to a most private level;

D. inputting the individual's categorized medical information into said plurality of hierarchical categories, said least private level being input into said low security category and said most private level being input into said high security category;

E. assigning by said individual, to each of said categories, one or more of said access security codes, such that said medical information in each category will be released only if the assigned access security codes are received;

F. providing, by said individual to one or more requestors, access priority data corresponding to a desired level in said hierarchy;

G. receiving, from a requestor, one or more of said access security codes over said communications network;

H. determining whether said received access security codes match one or more of said assigned access security codes; and

I. transmitting, to said requestor over said communications network, said medical information in said categories in which said received security access codes match said assigned security access codes.

In another aspect, the present invention provides a system for distributing medical information for an individual over a communications network comprising: a server system including a computer processor and associated memory, said server system having database of a plurality of hierarchical categories for the individual, said categories ranging from a low security category to a high security category, each of said categories having medical information of said individual contained therein, said medical information ranging from least private information to most private information, said least private medical information being contained in said low security category and said most private medical information being contained in said high security category, each of said categories having one or more security access codes assigned thereto; a request system including a computer processor and associated memory, said request system inputting, in response to a requestor, one or more of said security access codes provided to said requestor by said individual, to said server system over said communications network; and an access determining device for transmitting, to said request system, the medical information in each of said categories in which said input security access codes match said assigned security access codes.

In a further aspect, the present invention provides a method of distributing personal information for an individual over a communications network comprising the steps of:

A. generating a plurality of access security codes;

B. generating a plurality of hierarchical categories, ranging from a low security category to a high security category;

C. categorizing the individual's personal information into privacy levels ranging from a least private level to a most private level;

D. inputting the individual's categorized personal information into said plurality of hierarchical categories, said least private level being input into said low security category and said most private level being input into said high security category;

E. assigning, by said individual, to each of said categories, one or more of said access security codes, such that said personal information in each category will be released only if the assigned access security codes are received;

F. providing by said individual to one or more requestors, access priority data corresponding to a desired level in said hierarchy;

G. receiving, from a requestor, one or more of said access security codes over said communications network;

H. determining whether said received access security codes match one or more of said assigned access security codes; and

I. transmitting, to said requestor over said communications network, said personal information in said categories in said hierarchy which said received security access codes match said assigned security access codes.

In yet another aspect, the present invention provides a system for distributing personal information for an individual over a communications network comprising: a server system including a computer processor and associated memory, said server system having database of a plurality of hierarchical categories for the individual, said categories ranging from a low security category to a high security category, each of said categories having personal information of said individual contained therein, said personal information ranging from least private information to most private information, said least private personal information being contained in said low security category and said most private personal information being contained in said high security category, each of said categories having one or more security access codes assigned thereto; a request system including a computer processor and associated memory, said request system inputting, in response to a requester, one or more of said security access codes provided to said requester by said individual, to said server system over said communications network; and an access determining device for transmitting, to said requests system, the personal information in each of said categories in which said input security access codes match said assigned security access codes.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other objects of this invention, the various features thereof, as well as the invention itself may be more fully understood from the following description when read together with the accompanying drawings in which:

FIG. 1 is a diagrammatic view of a system for distributing medical information in accordance with the present invention;

FIG. 2 is a flow diagram of a method of distributing medical information in accordance with the present invention;

FIG. 3 is a schematic diagram of an example medical record in accordance with the system of FIG. 1;

FIG. 4 is a diagrammatic view of another embodiment of the system of FIG. 1; and

FIG. 5 is a diagrammatic view of yet another embodiment of the system of FIG. 1.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention enables a medical care provider to have remote access to a patient's medical record, while also enabling the patient to dictate exactly how much information the medical care provider can access. FIG. 1 shows a diagram of a system 100 for distributing medical information in accordance with a preferred embodiment of the present invention. The system 100 includes setup system 110, server system 120 and request system 130 all connected to a common communications network 160. Preferably, the setup system 110, server system 120 and request system 130 can each be a personal computer such as an IBM PC or IBM PC compatible system or an APPLE® MacINTOSH® system or a more advanced computer system such as an Alpha-based computer system available from Compaq Computer Corporation or SPARC® Station computer system available from SUN Microsystems Corporation, although a main frame computer system can also be used. Preferably, the communications channel 160 is a TCP/IP-based network such as the Internet or an intranet, although almost any well known LAN, WAN or VPN technology can be used.

In one preferred embodiment of the invention, the setup system 110 and request system 130 are IBM PC compatible systems operating a Microsoft Windows® operating system and server system 120 is configured as a web server providing access to information such as web pages in HTML format via the HyperText Transport Protocol (http). The setup system 110 and request system 130 include software to allow viewing of web pages, commonly referred to as a web browser, thus being capable of accessing web pages located on server system 120. Furthermore, setup system 110, server system 120 and request system 130 include software for encrypting and decrypting data that is transmitted over the communications network 160. Alternatively, setup system 110 and request system 130 can be any wired or wireless device that can be connected to a communications network, such as an interactive television system, such as WEBTV, a personal digital assistant (PDA) or a cellular telephone. In this preferred embodiment, setup system 110 is located at the patient's primary care physician's office and request system 130 is located wherever access to a patient's medical record is required, such as in an emergency room, ambulance or another doctor's office.

FIG. 2 shows a flow diagram of the method of distributing medical information according to the present invention. First, the user of the setup system 110, FIG. 1, who can be the patient or the patient's physician, generates security access codes, step 202, which will provide varying access to the patient's medical records. Such security access codes can include demographic data such as the patient's name, birth date, social security number, address and phone number; non-demographic data such as a passport number and the patient's native language; physical attributes such as eye and hair color and scars or other identifying marks; and user-definable fields such as passwords. The user then generates hierarchical categories into which the patient's medical information will be stored, step 204. These categories range from a low security category, for information that the patient is less concerned about becoming known by a third party, to a high security category, for information that the patient is more concerned about becoming known by a third party. The patient and/or the patient's physician then determine the level of privacy that is desired for each piece of medical information in the patient's medical record, step 206. The least private level could include information such as

the patient's blood type and allergies. The most private level could include HIV data. Intermediate levels of privacy may include serology data, psychiatric data, cardiology data and genetic data. After the levels of privacy for each piece of the patient's medical information is determined, the information is transmitted from the setup system 110, FIG. 1, to the server system 120 over the communications network 160, and is stored in a database 122 of the server system 120, step 208. The patient then assigns one or more of the security access codes to each of the categories in the database 122, step 210. Preferably, security access codes that are easier to ascertain are assigned to low security categories, while security access codes that are more difficult to ascertain are assigned to high security categories. This allows the patient to more precisely control who has access to the categories, by enabling the patient to provide the security access codes for each of the categories only to medical personnel who have a "need-to-know" the particular information in each category.

As a further security measure, the patient can define which of the security access codes are necessary to be input by the requestor to identify the requestor as being authorized to access the patient's medical record, step 212. The security access code that will identify an authorized requestor is preferably a code that will not be easily guessed by an unauthorized requestor.

When a patient's medical record is needed, the requestor inputs to the server system 120, FIG. 1, through request system 130 and over network 160, any information that is known about the patient in order to identify the patient. While prior art systems require specific predetermined data to identify a patient, the present invention is capable of searching its database to identify the patient based on whatever information the requester can provide. Such information can include, but is not limited to, actual medical record numbers for a particular hospital, demographic data such as the patient's name, age and sex, information from a smart card that identifies the patient, retinal or iris scans and fingerprints. This flexible identification system enables the present invention to be used in conjunction with existing legacy systems. Since the database 122 may include medical records for a great number of patients, the server system 120 determines whether, based on the identification information input by the requestor, a unique patient match has been

achieved, step 216. In this embodiment, the identification information input by the requestor could also be the security access codes set up by the patient. If the identification information input by the requestor does not define a unique patient in the database, the server system notifies the requestor that more identification information is needed to establish a unique patient match, step 218. If the identification information provided by the requestor provides a unique patient match, step 216, the server system prompts the requestor to enter security access codes for the patient. The server system then receives one or more of the security access codes input to the server system by the requestor, step 219. The server system 120 determines whether the received security access codes satisfy the requestor identification constraints, step 220. If they do not, the system notifies the requestor that the identification constraints have not been satisfied, step 222. If the identification constraints have been satisfied, the server system determines which of the assigned access codes match the received access codes input by the requestor, step 224, and transmits, to the request system 130 over the network 160, the information from the categories in which the received security access codes match the assigned security access codes, step 226. If more of the security access codes are received from the requestor, step 228, the system returns to step 224 to determine which of the assigned codes match the received codes. If no more codes are received in step 228, the process is terminated.

An example of the above method will now be described with reference to FIG. 3, which is a schematic diagram of a patient's medical record in accordance to the present invention. Shown at 302 are the security access codes generated by the patient in step 202, FIG. 2. In this example, the security access codes are the patient's first and last name, zip code, eye color, street number, birth year and two passwords. Shown at 304 are the medical information categories generated by the patient in step 204. These categories include EKG data, allergies, blood type, serology, HIV and CD4. Shown at 306 are the security access codes that are assigned to the categories in step 210. As described above, security access codes that are easily ascertained are assigned to low security categories and security access codes that are more difficult to ascertain are assigned to high security categories. In this example, the patient's birth year is assigned to the blood type and allergies,

while the patient-defined passwords are assigned to the HIV and CD4 categories. In step 212, the patient's first name and zip code, block 308, are chosen by the patient as the identification constraints that must be input by the requestor to establish that he or she is authorized to access the patient's record. The following table shows the response of the system of the present invention to various combinations of security access codes received from the requestor:

If Requestor supplies this data	**System will respond with**
A. First Name unique	"More than one patient found, No match"
B. First Name, Birth Year, Key 1	"Unique Match, ID constraints not satisfied"
C. First Name, Zip Code, Birth Year	EKG data, Allergies, Blood Type
D. First Name, Zip Code, Birth Year, Password 1	EKG data, Allergies, Blood Type, Serology,
E. First Name, Zip Code, Birth Year, Password 1, Password 2	EKG data, Allergies, Blood Type, Serology, HIV
F. First Name, Zip Code, Birth Year, Password 1, Password 2, Eye Color	EKG data, Allergies, Blood Type, Serology, HIV, CD4

In case A, because only the first name of the patient has been received, the system determines that a unique patient match has not been found, step 216, and notifies the requestor that more identification information is needed, step 218. In case B, the first name, birth year and password 1 are received, step 214. Although these codes allow the system to identify the patient, step 216, the identification constraints (the first name and zip code) have not been satisfied, step 220. The system then notifies the requestor that more security access codes are needed to satisfy the identification constraints, step 222. In case C, the first name, zip code and birth year are received, step 214. A unique patient match is found, step 216 and the identification constraints are satisfied, step 220. The system then determines which of the assigned security access codes match the received security access

codes, step 224, and transmits the information in the matched categories to the requestor over the communications network, step 226. In case C, because the zip code and birth year are received, referring to block 306, FIG. 3, the EKG data, allergy information and blood type are transmitted from the database to the requestor. In case D, in addition to the first name zip code and birth year, password 1 is received, step 228. Accordingly, in addition to the information transmitted in case C, the patient's serology information is transmitted to the requestor. In case E, with the receipt of password 2, the patient's HIV data is transmitted and, in case F, the receipt of password 2 and the patient's eye color prompts the transmission of the patient's CD4 data, in addition to all of the information transmitted in the previous cases.

While the invention has been described as including one setup system that accesses the server system, a plurality of setup systems may be coupled to the server system in order to allow patients at different locations to access the server system. Such a configuration is illustrated in Fig. 4. In this system 300, in addition to the setup system 110, a second server system 140 is coupleable to the server system 120 over communications network 160 for the purpose distributing medical information in the manner described above. It will be understood that the server system can be accessed through any number of setup systems, and that any number of request systems may access the server system in the manner described above.

In another embodiment, shown at 400 in FIG. 5, system 115 can be used both as the setup system, for setting up the patient's medical record on the database 122 of server system 120, as described above, and as the request system for inputting security access codes and receiving medical information from the database 122 of the server system 120, as described above.

Accordingly, the present invention provides a method of and system for distributing medical information over a network in a granular manner, in which the patient determines the level of security associated with each category of his or her medical record. A requestor of the patient's medical information must input the proper security access codes to the system in order to obtain the information contained within the record. The system enables the patient to distribute the security access codes for the hierarchy of security categories to medical personnel on a

"need-to-know" basis, thus reducing the risk that highly private medical information be exposed to unauthorized personnel.

Attached as Appendix A is the Javascript and VBscript code used to implement the present invention and a sample database on which the invention can be used.

The invention may be embodied in other specific forms without departing from the spirit or essential characteristics thereof. The present embodiments are therefore to be considered in respects as illustrative and not restrictive, the scope of the invention being indicated by the appended claims rather than by the foregoing description, and all changes which come within the meaning and range of the equivalency of the claims are therefore intended to be embraced therein.

```
<!--#include file="adovbs.inc" -->
<%
dim a()
dim patient_id

'*****************************************************************************
*****************
' this sub loads the request_data table with the entries in the form
sub update_database_with_new_patient()
dim time_stamp
dim num_of_items
dim counter
time_stamp = cstr(now())       ' time stamp
counter = 0
length = cint(Request.Form("count"))       ' number of fields to extract from
request form

      ' create an entry in the patients table and get the new id
      set cnn1 = server.CreateObject("adodb.connection")
      cnn1.Open "filedsn=cardionet.dsn"
      set rst1 = server.CreateObject("adodb.recordset")
      rst1.ActiveConnection = cnn1
      rst1.CursorType = adOpenKeyset
      rst1.LockType = adLockOptimistic
      rst1.Source = "patients"
      rst1.open
      rst1.AddNew
            rst1("ref_data_1")=Request.Form("1")
            rst1("ref_data_2")=Request.Form("2")
            rst1("ref_data_3")=Request.ServerVariables("remote_addr")
      rst1.Update
      patient_id= rst1.Fields("patient_id")
      rst1.Close
      set rst1 = nothing

' add the pid_data for the new patient to the pid_data table
set rst2 = server.CreateObject("adodb.recordset")
rst2.ActiveConnection = cnn1
rst2.CursorType = adOpenKeyset
rst2.LockType = adLockOptimistic
rst2.Source = "pid_data"
rst2.Open
for x = 1 to length ' loop through items
      if len(cstr(Request.Form(cstr(x))))>0 then ' append only items that have a
value enetered
            counter = counter + 1
            redim preserve a(3,counter)

            ' redim works only on last dimension, note it is turned around 2,30
            a(0,counter) = cstr(x) ' item id
            a(1,counter) = lcase(trim(Request.Form(cstr(x)))) ' item value
            a(2,counter) = (Request.Form("c" & cstr(x))="on") ' item is a
required for identification


                  rst2.AddNew ' add a record for each entered item
                  rst2("patient_id") = cint(patient_id)
```

```
                    rst2("item_id") = cint(a(0,counter))
                    rst2("value") = cstr(a(1,counter))
                    rst2.update

                if a(2,counter) then 'add a record in id_pass id this is a required
item for identification
                    set rst3 = server.CreateObject("adodb.recordset")
                    rst3.ActiveConnection = cnn1
                    rst3.CursorType = adOpenKeyset
                    rst3.LockType = adLockOptimistic
                    rst3.Source = "id_pass"
                    rst3.Open
                    rst3.AddNew
                        rst3("patient_id")=cint(patient_id)
                        rst3("item_id")=cint(a(0,counter))
                    rst3.Update
                    rst3.Close
                end if

        end if
next
rst2.Close
cnn1.Close
set rst2=nothing
set cnn1=nothing
a(0,0) = cstr(counter)
end sub

%>

<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<TITLE></TITLE>
</HEAD>
<body  background=backgrnd.gif>
<%
call update_database_with_new_patient
Response.Write("Patient Added to Database")
%>

<P> </P>

</BODY>
</HTML>
```

```
<!--#include file="adovbs.inc" -->
<%
dim a()
dim patient_id

'*************************************************************************************
*****************
' this sub loads the request_data table with the entries in the form
sub update_database_with_new_patient()
dim time_stamp
dim num_of_items
dim counter
time_stamp = cstr(now())       ' time stamp
counter = 0
length = cint(Request.Form("count"))       ' number of fields to extract from
request form

      ' create an entry in the patients table and get the new id
      set cnn1 = server.CreateObject("adodb.connection")
      cnn1.Open "filedsn=cardionet.dsn"
      set rst1 = server.CreateObject("adodb.recordset")
      rst1.ActiveConnection = cnn1
      rst1.CursorType = adOpenKeyset
      rst1.LockType = adLockOptimistic
      rst1.Source = "patients"
      rst1.open
      rst1.AddNew
            rst1("ref_data_1")=Request.Form("1")
            rst1("ref_data_2")=Request.Form("2")
            rst1("ref_data_3")=Request.ServerVariables("remote_addr")
      rst1.Update
      patient_id= rst1.Fields("patient_id")
      rst1.Close
      set rst1 = nothing

' add the pid_data for the new patient to the pid_data table
set rst2 = server.CreateObject("adodb.recordset")
rst2.ActiveConnection = cnn1
rst2.CursorType = adOpenKeyset
rst2.LockType = adLockOptimistic
rst2.Source = "pid_data"
rst2.Open
for x = 1 to length ' loop through items
      if len(cstr(Request.Form(cstr(x))))>0 then ' append only items that have a
value entered
            counter = counter + 1
            redim preserve a(3,counter)

            ' redim works only on last dimension, note it is turned around 2,30
            a(0,counter) = cstr(x) ' item id
            a(1,counter) = lcase(trim(Request.Form(cstr(x)))) ' item value
            a(2,counter) = (Request.Form("c" & cstr(x))="on") ' item is a
required for identification

                  rst2.AddNew ' add a record for each entered item
                  rst2("patient_id") = cint(patient_id)
```

```
                    rst2("item_id") = cint(a(0,counter))
                    rst2("value") = cstr(a(1,counter))
                    rst2.update

            if a(2,counter) then 'add a record in id_pass id this is a required
item for identification
                    set rst3 = server.CreateObject("adodb.recordset")
                    rst3.ActiveConnection = cnn1
                    rst3.CursorType = adOpenKeyset
                    rst3.LockType = adLockOptimistic
                    rst3.Source = "id_pass"
                    rst3.Open
                    rst3.AddNew
                        rst3("patient_id")=cint(patient_id)
                        rst3("item_id")=cint(a(0,counter))
                rst3.Update
                rst3.Close
            end if

      end if
next
rst2.Close
cnn1.Close
set rst2=nothing
set cnn1=nothing
a(0,0) = cstr(counter)
end sub

%>

<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<TITLE></TITLE>
</HEAD>
<body  background=backgrnd.gif>
<%
call update_database_with_new_patient
Response.Write("Patient Added to Database")
%>

<P> </P>

</BODY>
</HTML>
```

```
<!--#include file="adovbs.inc" -->

<%

function item_locked(fld_name,fld_value,pid)
      set rst_chk=server.CreateObject("adodb.recordset")
      rst_chk.ActiveConnection = cnn1
      rst_chk.CursorType = adOpenKeyset
      rst_chk.LockType = adLockOptimistic

      sql="select distinct item_id from locks where (patient_id=" & cint(pid) &
" and " & fld_name & "=" & cint(fld_value)
      sql= sql & ") and item_id not in ( select distinct item_id from
request_data where user_ip='" & user_ip & "')"
      rst_chk.Source = sql
      rst_chk.Open
      if rst_chk.EOF then
            item_locked = false
      else
            item_locked = true
      end if
      'while not(rst_chk.EOF)
      '     Response.Write("<br>" & rst_chk.Fields("item_id") & "<br>")
      '     rst_chk.MoveNext
      'wend
      rst_chk.Close
      set rst_chk=nothing
end function

%>

<html><head></head><body  background=backgrnd.gif>
<strong> Observations : </strong>
<hr>
<table align=center border=3 bordercolor=#0000cd cellpadding=3 width=100%>
<tr><th>Category</th><th>Parameter</th><th>Observation</th></tr>

<%


user_ip=Request.ServerVariables("remote_addr")

' open the database connection
set cnn1 = server.CreateObject("adodb.connection")
cnn1.Open "filedsn=cardionet.dsn"

' extract the patient on which this ip is allowed to work on, from the online
table

set rst_pid=server.CreateObject("adodb.recordset")
rst_pid.ActiveConnection = cnn1
rst_pid.CursorType = adOpenKeyset
rst_pid.LockType = adLockOptimistic
rst_pid.Source = "select patient_id from online where user_ip='" & user_ip & "'"
rst_pid.Open
if not rst_pid.EOF then
      pid=cint(rst_pid.Fields("patient_id"))
```

```
end if


' loop through the categories
set rst_cat = server.CreateObject("adodb.recordset")
rst_cat.ActiveConnection = cnn1
rst_cat.CursorType = adOpenKeyset
rst_cat.LockType = adLockOptimistic
rst_cat.Source = "categories"
rst_cat.Open
while not rst_cat.EOF
      cat_id = cint(rst_cat.Fields("category_id"))
      cat_name= rst_cat.Fields("category_name")
      'Response.Write(cat_id & "-")
      'Response.Write(cat_name & ":")
      if item_locked("category_id",cat_id,pid) then  ' IF CATEGORY IS LOCKED
            Response.Write("<tr><td colspan=3 align=left>Category " & cat_name &
" is Locked</td></tr>")
      else
            ' loop through the parameters
            set rst_par = server.CreateObject("adodb.recordset")
            rst_par.ActiveConnection = cnn1
            rst_par.CursorType = adOpenKeyset
            rst_par.LockType = adLockOptimistic
            rst_par.Source = "select * from [parameters] where category_id=" &
cat_id
            rst_par.Open
            while not rst_par.EOF
                  par_id=rst_par.Fields("parameter_id")
                  par_name=rst_par.Fields("parameter_name")
                  'Response.Write(par_id & "-")
                  'Response.Write(par_name & ":")
                  if item_locked("parameter_id",par_id,pid) then ' IF PARAMETER
IS LOCKED
                        Response.Write("<tr><td colspan=3 align=left>" &
cat_name & " : " & par_name & " Parameter is Locked</td></tr>")
                  else
                        ' loop through the observations
                        set rst_obs = server.CreateObject("adodb.recordset")
                        rst_obs.ActiveConnection = cnn1
                        rst_obs.CursorType = adOpenKeyset
                        rst_obs.LockType = adLockOptimistic
                        rst_obs.Source = "select * from observations where
parameter_id=" & par_id & " and patient_id=" & pid
                        rst_obs.Open

                        while not rst_obs.EOF
                              obs_id=rst_obs.Fields("observation_id")
                              obs_value=rst_obs.Fields("value")
                              if item_locked("observation_id",obs_id,pid) then '
IF OBSERVATION IS LOCKED
                                    Response.Write("<tr><td colspan=3 align=left
>" & cat_name & " : " & par_name & " Observation is Locked</td></tr>")
                              else
                              Response.write("<tr><td>" & cat_name & "</td><td>"
& par_name & "</td><td>" & obs_value & "</td></tr>")
```

```
                    end if
                ' CLOSE OBSERVATION IF
                    rst_obs.MoveNext
                wend ' close observation loop

                rst_obs.Close
                set rst_obs=nothing
            end if
' CLOSE PARAMETER IF
            rst_par.MoveNext

        wend ' close parameter loop

        rst_par.Close
            set rst_par=nothing
            end if
CLOSE CATEGORY IF
            rst_cat.MoveNext
      wend ' close category loop

      rst_cat.Close
      set rst_cat=nothing


      Response.Write("</table></body></html>")
cnn1.Close
%>
```

```
<!--#include file="adovbs.inc" -->
<!--#include file="setodbc.inc" -->
<%

' get the category id and the observation_id from the request
cat_id=Request.QueryString("catid")
obx_id=Request.QueryString("obxid")

if cat_id=null or obx_id=null or cat_id="" or obx_id="" then
      Response.Redirect "id_input_page.asp","_top"
end if

' extract the patient on which this ip is allowed to work on, from the online
table
user_ip=Request.ServerVariables("remote_addr")
set rst_pid=server.CreateObject("adodb.recordset")
sql = "select patient_id from online where user_ip like '" & user_ip & "'"
rst_pid.Open sql,connobj,adOpenKeyset,adLockOptimistic
if not rst_pid.EOF then
      pid=cint(rst_pid.Fields("patient_id"))
      rst_pid.Close
else
      rst_pid.Close
      set rst_pid= nothing
      Response.Redirect("id_input_page.asp")
end if

' get the observation text
set rst_obx=server.CreateObject("adodb.recordset")
sql="select obx_text from observations where obx_id=" & obx_id & " and
patient_id=" & pid & " and category_id=" & cat_id
rst_obx.Open sql,connobj,adOpenKeyset,adLockOptimistic
if not rst_obx.EOF then
      obx_text=rst_obx.Fields(0)
end if
rst_obx.Close
set rst_obx=nothing
Response.Write(" Coding for observation: " & obx_text)

' get the observation codes
set rst_obc=server.CreateObject("adodb.recordset")
sql="select observation_coding.code_system_id,
observation_coding.code_id,coding_systems.coding_system_id,coding_systems.coding
_system_name from observation_coding inner join coding_systems on
observation_coding.code_system_id=coding_system_id where
observation_coding.observation_id=" & obx_id
'Response.Write("<br>" & sql & "<br>")
rst_obc.Open sql,connobj,adOpenKeyset,adLockOptimistic
while not rst_obc.EOF
      obc_text="<br> code system name : " & rst_obc.Fields("coding_system_name")
& " ... code :" & rst_obc.Fields("code_id") & " <br>"
      Response.Write(obc_text)
      rst_obc.MoveNext
wend
rst_obc.Close
set rst_obc=nothing
```

```
Response.End

%>

<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<TITLE></TITLE>
</HEAD>
<BODY>

<P> </P>

</BODY>
</HTML>
```

```
<HTML>
<HEAD>
<META http-equiv="Expires" content="Tue, 20 Aug 1996 14:25:27 GMT">
<TITLE></TITLE>
</HEAD>
<frameset rows="100,*">
      <frame name="top_frame" src="logo_page.asp" border=0>
      <frameset cols="35%,65%">
            <frame name="left_frame" src="skeleton.asp" border=1>
            <frame name="right_frame" src="view_patient.asp" border=1>
      </frameset>
</frameset>
</HTML>
```

```
<!--#include file="adovbs.inc" -->
<!--#include file="setodbc.inc" -->
<%

' get the category id from the left frame ( skeleton.asp )
cat_id=Request.QueryString("id")
cat_name=Request.QueryString("cat_name")
if cat_id=null or cat_id="" then
      Response.Redirect("id_input_page.asp")
end if


' extract the patient on which this ip is allowed to work on, from the online
table
user_ip=Request.ServerVariables("remote_addr")
set rst_pid=server.CreateObject("adodb.recordset")
sql = "select patient_id from online where user_ip like '" & user_ip & "'"
rst_pid.Open sql,connobj,adOpenKeyset,adLockOptimistic
if not rst_pid.EOF then
      pid=cint(rst_pid.Fields("patient_id"))
      rst_pid.Close
else
      rst_pid.Close
      set rst_pid= nothing
      Response.Redirect("id_input_page.asp")
end if

%>


<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<TITLE></TITLE>
</HEAD>
<BODY background=backgrnd.gif >
<%

rst_pid.Open "select value from pid_data where patient_id=" & pid & " and
item_id<3",connobj,adOpenDynamic,adLockOptimistic
if not rst_pid.EOF then
      rst_pid.Movelast
      Response.Write "<center>Patient: <strong>" & rst_pid.Fields(0)
      rst_pid.Movefirst
      Response.Write( " " & rst_pid.Fields(0) & "</strong>      
      Category: <strong>" & cat_name)

end if
rst_pid.Close
set rst_pid=nothing

Response.Write("<hr width=90% noshade>")



%>
```

```
<form action=process_new_observation.asp method=post name=new_obx>
<input type=hidden name=pid value=<%=pid%>>
<input type=hidden name=cat_id value=<%=cat_id%>>
<input type=hidden name=cat_name value=<%=cat_name%>>
<table cellspacing=2 cellpadding=4>
<tr><th width=10% > </th><th  width=20%
align=left>Observation:  </th><td width=60%><textarea name=obx_val
width=80 rows=3 cols=40></textarea></td><th width=10% > </th></tr>
<tr><th> </th><th align=left>Time:  </th><td><input name=obx_time
width=80 value="<%=FormatDateTime(now(),0)%>"></td></tr>
<!--   Getting observation system
<tr><th> </th><th align=left>Coding System:   </th><td><SELECT
id=select1 name=code_system>
<%
set rst2=server.CreateObject("adodb.recordset")
sql="select * from coding_systems"
rst2.Open sql,connobj,adOpenKeyset,adLockOptimistic
while not rst2.EOF
      Response.Write("<option value=" & rst2.Fields("coding_system_id") & ">" &
rst2.Fields("coding_system_name") & "</option>")
      rst2.MoveNext
wend
rst2.Close
set rst2=nothing
%>
</SELECT></td></tr>
<tr><th> </th><th align=left>Code Value:  </th><td><input
name=code_value width=40></td></tr>

-->
</table>
<br clear=all><center><input type=submit value="Register New
Observation"><br><br></center><hr width=90% noshade>
</form>
<%

'load any existing observations for this patient into the rst1 recordset
set rst1=server.CreateObject("adodb.recordset")
sql="select * from observations where patient_id=" & pid & " and category_id=" &
cat_id
rst1.Open sql,connobj,adOpenKeyset,adLockOptimistic
if rst1.EOF then
      Response.Write("<li> No Observations recorded in this category")
else
      %>
      <form name=edit_obx method=post
action=process_delete_observation.asp><center>
      <input type=hidden name=pid value=<%=pid%>>
      <input type=hidden name=cat_id value=<%=cat_id%>>
      <input type=hidden name=cat_name value=<%=cat_name%>>
      <table border=1 width=90% >
      <tr><th>Delete</th><th>Existing
Observations</th><th>Time</th><th>Coding</th></tr>
      <%
      while not rst1.EOF
            Response.Write ("<tr><td align=center><input type=checkbox name=c" &
rst1.Fields("obx_id") & "></input></td>")
```

```
            Response.Write ("<td>" & rst1.Fields("obx_text") & "</td>")
            Response.Write ("<td>" & rst1.Fields("obx_time") & "</td>" )
            'Response.Write ("<td><center><input type=button name=b" &
rst1.Fields("obx_id") & " value=""Edit..."" ></td></center></tr>" & chr(10))
            url="edit_codes_1.asp?obxid=" & cstr(rst1.Fields("obx_id")) &
"&catid=" & cat_id
            %>
            <td><center><input type=button name="b<%=rst1.Fields("obx_id")%>"
value="Edit..." language=javascript onclick="window.open('<%=url%>')">
            <%
            rst1.MoveNext
      wend
      %>
      </table><center><br clear=all><input type=submit value="Delete
Observations"></center>
      </form>
      <%
end if

rst1.Close
set rst1=nothing
%>

</BODY>
</HTML>
```

```
<!--#include file="adovbs.inc" -->
<!--#include file="setodbc.inc" -->

<% dim arr()
' (0,0) holds the number of items in the array
' (0,x) holds the item_id
' (1,x) holds the item_name
' (2,x) holds the previous value provided by the ip in previous sessions



'*************************************************************************************
**************
' this function loads the items and their names a blank array named arr()


Function get_pid_items()
'dim cnn
dim rst
dim counter

    set rst = server.CreateObject("adodb.recordset")


    'rst.Open sql,"FILEDSN=cardionet.dsn",adOpenDynamic
    sql="select * from pid_items"
    rst.Open sql,connobj,adOpenDynamic
    rst.MoveFirst
    counter = 0
    while not rst.EOF
            counter = counter + 1
            redim preserve arr(3,counter+1)
            arr(0,counter) = trim(cstr(rst.Fields("item_id").value))
            arr(1,counter) = trim(cstr(rst.Fields("item_name").value))
            rst.MoveNext
    wend
    rst.Movefirst
    if counter > 0 then
            ReDim arr(2,counter+1)
            for x = 1 to counter
                    arr(0,x) = Trim(CStr(rst.Fields("item_id").Value))
                    arr(1,x) = Trim(CStr(rst.Fields("item_name").Value))
                    rst.MoveNext
            next
            arr(0, 0) = CStr(counter)
            get_pid_items = true
      else
            get_pid_items = false
      end if
      rst.close
      Set rst= Nothing
      session.Abandon ()
End Function
```

```
'*****************************************************************************
**************
' this sub loads the array with the previous values entered by the same ip user
to the arr() array

sub load_existing_values()
user_ip = trim(cstr(Request.ServerVariables("remote_addr")))
set rst_1= server.CreateObject("adodb.recordset")
counter=cint(arr(0,0))
for x = 1 to counter
      sql="select item_value from request_data where ((user_ip like '" & user_ip
& "') and (item_id=" & cint(arr(0,x)) & "))"
      'Response.Write("<br>" & sql)
      rst_1.Open sql,connobj,adOpenKeyset,adLockOptimistic
      'Response.Write(rst_1.RecordCount )
      if not (rst_1.EOF) then
            arr(2,x)=rst_1.Fields("item_value")
            'Response.Write("hello " & arr(2,x))
      else
            arr(2,x)=null
      end if
      rst_1.Close
next
set rst_1=nothing
end sub



'*****************************************************************************
***********
' this sub generates the actual HTML grid with the values if they exist in the
arr() array

Sub export_html()
Dim x
    x=0
    items_per_id = arr
    length = cint(items_per_id(0, 0))
    Response.Write( "<center><strong><font size=4 color=#0000ff>CareKey Patient
Identification Page</font></strong><hr>")
    Response.Write("<form action=""process_id_input.asp"" method=post>")
    Response.Write("<input type=hidden name=""count"" value=""" & CStr(length) &
""""></input>")
    Response.Write("<table cellpadding=1 cellspacing=0 border=1>")
    Response.Write("<tr bgcolor=#00008b><th>  ID   </th><th><font
color=#ffff00>Item Name</th><th><font
color=#ffff00>Input</th><th> </th><th>  ID  </th><th><font
color=#ffff00>Item Name</th><th><font color=#ffff00>Input</th></tr>")
    For x = 1 To Int(length / 2)
        Response.Write("<tr><th>" & items_per_id(0,x) & "</th><td>" &
items_per_id(1,x) & "</td>")
        Response.Write("<td><input name=""" & items_per_id(0,x) & """ size=20
value=""" & items_per_id(2,x) & """></input></td ><td
bgcolor=#00008b> </td>")
        Response.Write("<th>" & items_per_id(0,x + Int(length / 2)) &
"</th><td>" & items_per_id(1,x + Int(length / 2)) & "</td>")
```

```
        Response.Write("<td><input name=""" & items_per_id(0,x + Int(length /
2)) & """ size=20></input></td><tr>")
    Next
    If (x - 1) * 2 < length Then
        Response.Write("<tr><th></th><td></td><td></td><td
bgcolor=MediumAquamarine></td><th>" & items_per_id(0,length) & "</th><td>" &
items_per_id(1,length) & "</td>")
        Response.Write("<td><input name=""" & items_per_id(0,length) & """
size=20></input></td></tr>")
    End If
    Response.Write("</table><br><input type=""submit"" value=""Attempt Unique
Identification""></form>")
End Sub

'*************************************************************************
*********
' HTML CODE STARTS HERE


%>
<html ><head><font size=3  face=sans-serif>
<META http-equiv="Expires" content="Tue, 20 Aug 1996 14:25:27 GMT">
<TITLE>Carekey Patient Identification Page</TITLE>
</HEAD>
<body  background=backgrnd.gif>
<%
if get_pid_items() then
      call load_existing_values
      call export_html
end if
connobj.close
set connobj=nothing
%>
<HR NOSHADE>
</BODY>
</HTML>
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 FINAL//EN">
<HTML>
<HEAD>
<script language="javascript" for="window" event="onload">

</script>
<font size=2  face=serif>
<script language="javascript">
var html_text='';
var toggle=false;
var depth=1000;
var next_sib;
var vpos=0;
var vmax=1;
var temp_node;
var xmldoc=new ActiveXObject("Microsoft.XMLDOM");
xmldoc.load("make_dtd.asp");
loaddoc();

function loaddoc ()
{
if (xmldoc.readyState=="4"){
      start();
      window.section.innerHTML=html_text;
      }
else window.setTimeout("loaddoc()" , 200 );
}

function start() {
      var root=xmldoc.documentElement;
      stratify(vpos,root);
      }


function stratify(vpos,node) {
      if (vpos<depth){

            node_name=node.nodeName;
            if (node_name != '#text')
                  node_att=node.attributes(0).text
            else
                  node_att=node.nodeName;
            node_text=node.text;
            if (node_text != '' || toggle ){
                  taber='';
                  for (x=0;x<vpos;x++)
                        taber=taber+'       ';

                  if (node_name=='obx'){
                        local_html=taber+'    <font
color="0000ff">'+node_text+'</font><br>'+String.fromCharCode(10)
                        html_text=html_text+local_html;
                        }
                  else {
                        if (node.hasChildNodes()){
                              local_html=taber+'<label
onclick=document.all("'+node_name+'").style.display="none">« </label>';
```

```
                        local_html=local_html+'<label
onclick=document.all("'+node_name+'").style.display="block">»</label>';
                    }
                    else
                        local_html=taber+'     ';

                    if (node_text !='')
                        local_html=local_html+'<strong><font
color="ff0000">'+node_att+'</font></strong><br>'
                    else
                        local_html=local_html+'<strong><font
color="808080">'+node_att+'</font></strong><br>';

                    local_html=local_html+'<div
id="'+node_name+'">'+String.fromCharCode(10);
                    html_text=html_text+local_html;
                    if (node.hasChildNodes()){
                        if (vmax<vpos)
                            vmax=vpos;
                        stratify(vpos+1,node.childNodes(0));
                        taber='';
                        for (x=0;x<vpos;x++)
                            taber=taber+'    ';
                    }
                html_text=html_text+'</div>'+String.fromCharCode(10);
                }

            }
            temp_node=node.nextSibling;
            if (temp_node != null && temp_node.nodeName != '#comment' )
                stratify(vpos,temp_node);
        }
}

function toggle_empty(){
        toggle=~toggle;
        html_text='';
        stratify(0,xmldoc.documentElement);
        window.section.innerHTML=html_text;
}


function display(flag){
        var root=xmldoc.documentElement;

        if (flag==1){
            show(root);
        }
        if (flag==2){
            hide(root);
        }
}

function hide(node){
            node_name=node.nodeName;
            if (node_name!='obx' && node_name!='#text'){
```

```
                if (document.all(node_name)!=null)
                    document.all(node_name).style.display="none";

            }
            if (node.hasChildNodes()){
                hide(node.childNodes(0));
            }
            temp_node=node.nextSibling;
            if (temp_node != null && temp_node.nodeName != '#comment' ){
                hide(temp_node);
            }
}

function show(node) {
    node_name=node.nodeName;
        if (node_name!='obx' && node_name!='#text'){

            if (document.all(node_name)!=null){
                document.all(node_name).style.display="block";
            }

        }
        if (node.hasChildNodes()){
            show(node.childNodes(0));
        }
        temp_node=node.nextSibling;
        if (temp_node != null && temp_node.nodeName != '#comment' ){
            show(temp_node);
        }
}

</script>
</HEAD>
<body background=backgrnd.gif>
<!--<body background=backgrnd.gif> -->
<center>
<input name="showit" value="Show All" type="button" onclick="display(1)"> 
<input name="hideit" value="Hide All" type="button" onclick="display(2)"> 
<input name="toggle" value="Toggle Empty" type="button"
onclick="toggle_empty()"> 
<input name="Id_Page" value="Back to ID page" type="button"
onclick="window.open('id_input_page.asp','_top')">
<input name="frames" value="Data Entry" type="button"
onclick="window.open('frame_set_1.asp','_top')">
</center>
<br><hr noshade width=80%><br>
<div id="section"></div>
</BODY>
</HTML>
```

```
<!--#include file="adovbs.inc" -->
<!--#include file="setodbc.inc" -->

<%


dim a()
dim user_id

'
*************************************************************************
*************
' this function creates the SQL that finds the unique match using the data in
the array
' returns negative number is no matches, 0 if many matches, patient_id if unique
match
Function get_pid_id()
dim rst
dim counter
    set rst = server.CreateObject("adodb.recordset")
    part0 = "select distinct patient_id from pid_data where ((1=1) "
    part1 = " and ( "
    part2 = " patient_id in (select patient_id from pid_data where item_id="
    part3 = " and value like '"
    part4 = "'))"
      part5 = ")"
      sql = part0
      for x = 1 to cint(a(0,0))
            sql = sql + part1 + part2 + a(0,x) + part3 + a(1,x) + part4
      next
      sql= sql + part5
      'Response.Write(sql)
    rst.Open sql,connobj
    'Response.Write(sql)
      if not rst.EOF then rst.MoveFirst
    counter = 0
    while not rst.EOF
            counter = counter + 1
            'Response.Write( "<br>patient found:" +
cstr(rst.Fields("patient_id").value)+ "<br>")
            rst.MoveNext
    wend
    'Response.Write("<br>counter:" & counter )
    if counter =1 then  ' patient matched
            rst.MoveFirst
            get_pid_id = rst.Fields("patient_id").value
      end if
      if counter < 1 then     ' no patients match
            get_pid_id = -1
      end if
      if counter >1 then      ' too many patients match
            get_pid_id = 0
      end if
    set rst= Nothing
      session.Abandon ()
End Function
```

```
'*****************************************************************************
****************
' this sub loads the request_data table with the entries in the form
sub update_database_with_user_request()
dim time_stamp
dim num_of_items
dim rst1
dim counter
user_id = trim(cstr(Request.ServerVariables("remote_addr"))) ' ip address
time_stamp = cstr(now())       ' time stamp
counter = 0
length = cint(Request.Form("count"))       ' number of fields to extract from
request form
' remove old entries for this ip
      set rst1 = server.CreateObject("adodb.recordset")
      rst1.CursorType= adOpenDynamic
      sql = "delete from request_data where user_ip like '" & user_id & "'" '
clear database from previous attempts
      rst1.open sql,connobj
      set rst1 = nothing

'set cnn1 = server.CreateObject("adodb.connection")
'cnn1.Open "filedsn=cardionet.dsn"
set rst2 = server.CreateObject("adodb.recordset")
rst2.ActiveConnection = connobj
rst2.CursorType = adOpenKeyset
rst2.LockType = adLockOptimistic
rst2.Source = "request_data"
rst2.Open
for x = 1 to length ' loop through items
      if len(cstr(Request.Form(cstr(x))))>0 then ' append only items that have a
value enetered
            counter = counter + 1
            redim preserve a(2,counter)
            ' redim works only on last dimension, note it is turned around 2,30
            a(0,counter) = cstr(x) ' item id
            a(1,counter) = lcase(trim(Request.Form(cstr(x)))) ' item value
                  rst2.AddNew ' add a record for each entered item
                  rst2("user_ip") = user_id
                  rst2("item_id") = a(0,counter)
                  rst2("item_value") = a(1,counter)
                  rst2("time_stamp") = time_stamp
                  rst2("session_id") = cstr(session.SessionID)
                  rst2.update
      end if
next
rst2.Close
'cnn1.Close
set rst2=nothing
'set cnn1=nothing
a(0,0) = cstr(counter)
end sub
```

```
'*************************************************************************************
**************
' this funciton checks to see if the items specified by the patient as required
to his
' identification are indeed provided. the list of those items are in table
id_pass and
' the items provided are in table request_data

function check_id_pass(pid)
      'set cnn4 = server.CreateObject("adodb.connection")
      'cnn4.Open "filedsn=cardionet.dsn"
      user_ip= trim(cstr(Request.ServerVariables("remote_addr")))
      set recset= server.CreateObject("adodb.recordset")
      recset.ActiveConnection = connobj
      recset.CursorType = adOpenKeyset
      recset.LockType = adLockOptimistic
      ' the following sql returns the item_id that were not provided and are
necessary for authorization
      sql0="select item_name from pid_items where item_id in "
      sql1="(select item_id from id_pass where (patient_id=" & pid & ")"
      sql2="and item_id not in "
      sql3="· (select item_id from request_data where user_ip like '" & user_ip &
"'))"

      recset.source=sql0 & sql1 & sql2 & sql3
      recset.Open

      if recset.RecordCount>0 then ' patient security specs are not met
            check_id_pass=0
            Response.Write("<hr><br><li><font size=2 color=#0000cd><strong> For
this patient, access will be authorized upon furnishing the following
information: </font>")
            'Response.Write(recset.RecordCount )
            while not recset.eof
                  Response.Write("<br><li>" & recset.Fields ("item_name"))
                  recset.movenext
            wend
            Response.Write("<br></font>")
      else
            check_id_pass=1 ' patient security is OK
      end if
      recset.close
'     cnn4.Close
      set recset=nothing
'     set cnn4=nothing

end function


'*************************************************************************************
*******
' this sub updates the online table with the access granted for this ip_user for
this patient
```

```
sub update_online (user, patient)
      set rst5 = server.CreateObject("adodb.recordset")
      'set cnn5 = server.CreateObject("adodb.connection")
      'cnn5.Open "filedsn=cardionet.dsn"
      sql="delete from online where user_ip like '" & user & "'"
      rst5.Open sql,connobj
      set rst6 = server.CreateObject("adodb.recordset")
      rst6.ActiveConnection = connobj
      rst6.CursorType = adOpenKeyset
      rst6.LockType = adLockOptimistic
      rst6.Source = "online"
      rst6.Open
      rst6.AddNew
            rst6("user_ip")=user
            rst6("patient_id")=patient
            rst6("time_stamp")=now()
      rst6.Update
      rst6.Close
'     cnn5.Close
      set rst6=nothing
      set rst5=nothing
'     set cnn5=nothing
end sub


'
*************************************************************************************
*******
' ********************** P R O G R A M       S T A R T
*************************************
'
*************************************************************************************
*******

call update_database_with_user_request
p = get_pid_id  ' calling the patient matching function

'Response.Write ("<hr>" & cstr(a(0,0)) &  "  items provided by ip user " &
user_id & "<br><hr>")
p_int=cint(p)
if p_int>0 then
      secure=check_id_pass(cstr(p))
      if secure=1 then
            update_Online user_id,p_int ' update the online table with the new
access provision
            'Response.Write("patient identified:" & p_int)
            'response.redirect("data_root.asp")
            'Response.Write("You now have access to patient " & p_int )
            Response.Redirect("view_patient.asp")
      else
            %>
            <html>
            <head>
            </head>
```

```
            <body background=backgrnd.gif>
            <br><center><hr noshade width=80%></center>
            <br clear=all>
            <%
            Response.Write("<br>Access requirements not met for the identified
patient: " & p)
      end if
end if

if p_int<0 then Response.Write ("<li>  No Matching patients found. Attempt
identification again using validated data only.")
if p_int=0 then Response.Write ("<li>  More than one patient found.Try
adding or changing patient information.")

'if p_int>0 then check_id_pass(cstr(p))
%>
<center><br><br>
<a href="id_input_page.asp">Back to Identification Page</a>
<P> </P>
</BODY>
</HTML>
```

```
<!--#include file="adovbs.inc" -->
<!--#include file="setodbc.inc" -->
<%

pid=Request.Form("pid")
obx_value=Request.Form("obx_val")
obx_time=Request.Form("obx_time")
code_system=Request.Form("code_system")
code_value=Request.Form("code_value")
cat_id=Request.Form("cat_id")
cat_name=Request.Form("cat_name")

if obx_value="" then
      Response.Redirect("get_data.asp?id=" & cat_id & "&cat_name=" & cat_name)
end if

if not(isdate(obx_time) ) then
      obx_time=now()
end if

'Response.Write("<br>" & pid & "<br>" & obx_value & "<br>" & obx_time & "<br>" &
code_system & "<br>" & code_value & "<br>" & cat_id )
'Response.End

set rst=server.CreateObject("adodb.recordset")
      rst.open "observations",connobj,adOpenKeyset,adLockOptimistic
      rst.AddNew
            rst("patient_id")=pid
            rst("obx_text")=obx_value
            rst("obx_time")=obx_time
            rst("category_id")=cat_id
      rst.Update
      obx_id= rst.Fields("obx_id")
      rst.Close
      ' this is part of the old interface where coding was entered as part of
the observation
      'rst.open "observation_coding",connobj,adOpenKeyset,adLockOptimistic
      'rst.AddNew
      '     rst("observation_id")=obx_id
      '     rst("code_system_id")=code_system
      '     rst("code_id")=code_value
      'rst.Update
      'rst.Close
set rst=nothing

Response.Redirect("get_data.asp?id=" & cat_id & "&cat_name=" & cat_name)

%>
```

```
<!--#include file="adovbs.inc" -->
<!--#include file="setodbc.inc" -->
<%

' extract the patient on which this ip is allowed to work on, from the online
table
user_ip=Request.ServerVariables("remote_addr")
set rst_pid=server.CreateObject("adodb.recordset")
sql = "select patient_id from online where user_ip like '" & user_ip & "'"
rst_pid.Open sql,connobj,adOpenKeyset,adLockOptimistic
if not rst_pid.EOF then
      pid=cint(rst_pid.Fields("patient_id"))
      rst_pid.Close
      set rst_pid= nothing
else
      rst_pid.Close
      set rst_pid= nothing
      Response.Redirect("id_input_page.asp")
end if
'*************************************************************************
****

' check if either a category or an observation is locked for this patient
function item_locked(fld_name,fld_value)
      ' fld_name should be either "observation_id" or "category_id"
      ' fld_value should contain the id of the item required weather an
observation or a category
      set rst_chk=server.CreateObject("adodb.recordset")

      sql="select distinct item_id from locks where (patient_id=" & cint(pid) &
" and " & fld_name & "=" & cint(fld_value)
      sql= sql & ") and item_id not in ( select distinct item_id from
request_data where user_ip like '" & user_ip & "')"

      rst_chk.Open sql,connobj,adOpenKeyset,adLockOptimistic
      if rst_chk.EOF then
            item_locked = false
      else
            item_locked = true
      end if
      rst_chk.Close
      set rst_chk=nothing
end function

'if item_locked("category_id",19)then
'     Response.Write("locked")
'     else
'     Response.Write("not locked")
'end if

'*************************************************************************
****
```

```
sub stratify_dtd(item_id,name,level)    ' generate the DTD from the table
categories
      set rst=server.CreateObject("adodb.recordset")
      sql="select * from categories where category_parent_id=" & item_id
      rst.Open sql,connobj,adOpenDynamic,adLockOptimistic
      tabber=""
      for x =1 to level
            tabber=tabber & "      "
      next
      if rst.EOF then
            Response.Write(chr(10) & tabber & "<!ELEMENT id" & item_id & "
(obx*)>")
            Response.Write(chr(10) & tabber & "<!ATTLIST id" & item_id & " name
CDATA """ & name & """>")
      else
            Response.Write(chr(10) & tabber & "<!ELEMENT id" & item_id & "
(obx*,")
            while not rst.EOF
                  local_id=rst.Fields("category_id")
                  Response.Write(   "id" & local_id & "*")
                  rst.MoveNext
                  if not rst.EOF then
                        Response.Write(", ")
                  end if
            wend
            Response.Write(" )>")
            Response.Write(chr(10) & tabber & "<!ATTLIST id" & item_id & " name
CDATA """ & name & """>")
            rst.MoveFirst

            level=level+1
            while not rst.eof
            '     Response.Write("   ,stratify level "  & level & ":" &
rst.Fields("category_id"))
                  cat_id=rst.Fields("category_id")
                  cat_name=rst.Fields("category_name")
                  call stratify_dtd(cat_id,cat_name,level) ' recursion origin
                  rst.MoveNext
            wend
            level=level-1
      end if
      rst.Close
      set rst=nothing
end sub
'*****************************************************************************
**********

sub stratify_xml(item_id,name,level)    ' produce the actual XML data
      tabber=""
      for x =1 to level
            tabber=tabber & "      "
      next

      ' check if this category is locked, in which case write the category name
as empty and exit this branch
      if item_locked("category_id",item_id) then
```

```
            Response.Write(chr(10) & tabber & "<id" & item_id & " name=""" &
name & "_Locked" & """/>")
            exit sub
      end if


      set rst=server.CreateObject("adodb.recordset")
      set data=server.CreateObject("adodb.recordset")
      sql="select * from categories where category_parent_id=" & item_id
      rst.Open sql,connobj,adOpenDynamic,adLockOptimistic

      ' check for data of this node
      sql="select * from observations where category_id=" & item_id & " and
patient_id=" & pid & " order by obx_time"
      data.Open sql,connobj,adOpenDynamic,adLockOptimistic

      if rst.EOF then ' if the parent node has no children... check for data....
            if data.EOF then ' if no data, close the node
                  Response.Write(chr(10) & tabber & "<id" & item_id & "
name=""" & name & """/>")
            else
                  Response.Write(chr(10) & tabber & "<id" & item_id & " name=""
& name & """>")
                  while not data.EOF ' if it has data, but no children, load it
and then close the node
                        obs_id=cint(data.Fields("obx_id"))
                        obs_txt=data.Fields("obx_text")
                        if item_locked("observation_id",obs_id) then
                              obs_txt="Observation Locked"
                        end if
                        Response.Write(chr(10) & tabber & "     " & "<obx
obx_id=""" & data.Fields("obx_id") & """>")
                        Response.Write(obs_txt & "</obx>")
                        data.MoveNext
                  wend
                  Response.Write(chr(10) & tabber & "</id" & item_id & ">")
            end if

      else  ' if the parent node has children... check for data....

            Response.Write(chr(10) & tabber & "<id" & item_id & " name=""" &
name & """>")
                  'print out all the obxs of the root element if they exist
                  while not data.EOF ' if it has data, print it.
                        obs_id=cint(data.Fields("obx_id"))
                        obs_txt=data.Fields("obx_text")
                        if item_locked("observation_id",obs_id) then
                              obs_txt=" Observation Locked "
                        end if
                        Response.Write(chr(10) & tabber & "     " & "<obx
obx_id=""" & data.Fields("obx_id") & """>")
                        Response.Write(obs_txt & "</obx>")
                        data.MoveNext
                  wend
                  data.close

            level=level+1
```

```
                ' and then drill down to the child nodes
                while not rst.eof
                        cat_id=rst.Fields("category_id")
                        cat_name=rst.Fields("category_name")
                        call stratify_xml(cat_id,cat_name,level) 'recursion origin
                        rst.MoveNext
                wend
                Response.Write(chr(10) & tabber & "</id" & item_id & ">")
                level=level-1
        end if
        rst.Close
        set rst=nothing
        set data=nothing
end sub

' --------- INIT XML PAGE WITH INTERNAL DTD ----------------------------
Response.Write("<?xml version=""1.0""?>" & chr(10))
Response.Write("<!DOCTYPE idl [" & chr(10))
Response.write("<!ELEMENT obx (#PCDATA)>" & chr(10))
Response.Write("<!ATTLIST obx" & chr(10))
Response.Write("        obx_id CDATA #REQUIRED" & chr(10))
Response.Write("        code_sys_id CDATA ""ICD9""" & chr(10))
Response.Write("        code_id CDATA ""134.18""" & chr(10))
Response.Write(">")
Response.Write(chr(10) & chr(10))

' ---------- CALL THE PROCEDURE THAT CREATES THE DTD ACCORDING TO THE DATABASE -
-----
call stratify_dtd (1,"xmlnet",0)
Response.Write(chr(10) & "]>" & chr(10))

' ---------- CALL THE PROCEDURE THAT GENERATES THE ACTUAL XML DATA DOCUMENT ---
------
Response.Write(chr(10) & chr(10))
call stratify_xml(1,"xmlnet",0)

%>
```

```
<!--#include file="adovbs.inc" -->
<!--#include file="setodbc.inc" -->
<%

pid=Request.Form("pid")
obx_value=Request.Form("obx_val")
obx_time=Request.Form("obx_time")
code_system=Request.Form("code_system")
code_value=Request.Form("code_value")
cat_id=Request.Form("cat_id")
cat_name=Request.Form("cat_name")

if obx_value="" then
      Response.Redirect("get_data.asp?id=" & cat_id & "&cat_name=" & cat_name)
end if

if not(isdate(obx_time) ) then
      obx_time=now()
end if

'Response.Write("<br>" & pid & "<br>" & obx_value & "<br>" & obx_time & "<br>" &
code_system & "<br>" & code_value & "<br>" & cat_id )
'Response.End

set rst=server.CreateObject("adodb.recordset")
      rst.open "observations",connobj,adOpenKeyset,adLockOptimistic
      rst.AddNew
            rst("patient_id")=pid
            rst("obx_text")=obx_value
            rst("obx_time")=obx_time
            rst("category_id")=cat_id
      rst.Update
      obx_id= rst.Fields("obx_id")
      rst.Close
      ' this is part of the old interface where coding was entered as part of
the observation
      'rst.open "observation_coding",connobj,adOpenKeyset,adLockOptimistic
      'rst.AddNew
      '     rst("observation_id")=obx_id
      '     rst("code_system_id")=code_system
      '     rst("code_id")=code_value
      'rst.Update
      'rst.Close
set rst=nothing

Response.Redirect("get_data.asp?id=" & cat_id & "&cat_name=" & cat_name)

%>
```

```
<HTML>
<HEAD>
<META NAME="CareKey System Demo" Content="Center for Clinical Computing, Harvard
Medical School">
<TITLE>CareKey Demo</TITLE>
</HEAD>
<BODY background=backgrnd.gif>
<font size=5 color=#00008b><strong><Center>
<a href="id_input_page.asp" >
<img name="img1" src="bidmc-banner.gif" border=0 width=90% >
</a>
</BODY>
</HTML>
```

```
<!--#include file="adovbs.inc" -->
<!--#include file="setodbc.inc" -->
<%

sub stratify_dtd(item_id,name,level)    ' generate the DTD from the table
categories
      set rst=server.CreateObject("adodb.recordset")
      sql="select * from categories where category_parent_id=" & item_id
      rst.Open sql,connobj,adOpenDynamic,adLockOptimistic
      tabber=""
      for x =1 to level
            tabber=tabber & "      "
      next
      if rst.EOF then
            Response.Write(chr(10) & tabber & "<!ELEMENT id" & item_id & "
(obx*)>")
            Response.Write(chr(10) & tabber & "<!ATTLIST id" & item_id )
            Response.Write(chr(10) & tabber & " name CDATA """ & name & """")
            Response.Write(chr(10) & tabber & " cat_id CDATA """ & item_id &
""">")
            Response.Write(chr(10))
      else
            Response.Write(chr(10) & tabber & "<!ELEMENT id" & item_id & "
(obx*,")
            while not rst.EOF
                  local_id=rst.Fields("category_id")
                  Response.Write(   "id" & local_id & "*")
                  rst.MoveNext
                  if not rst.EOF then
                        Response.Write(", ")
                  end if
            wend
            Response.Write(" )>")

            Response.Write(chr(10) & tabber & "<!ATTLIST id" & item_id )
            Response.Write(chr(10) & tabber & " name CDATA """ & name & """")
            Response.Write(chr(10) & tabber & " cat_id CDATA """ & item_id &
""">")
            Response.Write(chr(10))



            'Response.Write(chr(10) & tabber & "<!ATTLIST id" & item_id & " name
CDATA """ & name & """>")
            rst.MoveFirst

            level=level+1
            while not rst.eof
            '     Response.Write("   ,stratify level "  & level & ":" &
rst.Fields("category_id"))
                  cat_id=rst.Fields("category_id")
                  cat_name=rst.Fields("category_name")
                  call stratify_dtd(cat_id,cat_name,level) ' recursion origin
                  rst.MoveNext
            wend
            level=level-1
```

```
        end if
        rst.Close
        set rst=nothing
end sub
'*************************************************************************
***********

sub stratify_xml(item_id,name,level)    ' produce the actual XML data
        tabber=""
        for x =1 to level
                tabber=tabber & "       "
        next

        set rst=server.CreateObject("adodb.recordset")
        set data=server.CreateObject("adodb.recordset")
        sql="select * from categories where category_parent_id=" & item_id
        rst.Open sql,connobj,adOpenDynamic,adLockOptimistic


        if rst.EOF then ' if the parent node has no children...
                        Response.Write(chr(10) & tabber & "<id" & item_id & "
name=""" & name & """/>")
        else  ' if the parent node has children... check for data....

                Response.Write(chr(10) & tabber & "<id" & item_id & " name=""" &
name & """>")
                'print out all the obxs of the root element if they exist
                level=level+1
                ' and then drill down to the child nodes
                while not rst.eof
                        cat_id=rst.Fields("category_id")
                        cat_name=rst.Fields("category_name")
                        call stratify_xml(cat_id,cat_name,level) 'recursion origin
                        rst.MoveNext
                wend
                Response.Write(chr(10) & tabber & "</id" & item_id & ">")
                level=level-1
        end if
        rst.Close
        set rst=nothing
        set data=nothing
end sub

' --------- INIT XML PAGE WITH INTERNAL DTD ----------------------------
Response.Write("<?xml version=""1.0""?>" & chr(10))
Response.Write("<!DOCTYPE id1 [" & chr(10))
Response.write("<!ELEMENT obx (#PCDATA)>" & chr(10))

Response.Write(chr(10) & chr(10))

' ---------- CALL THE PROCEDURE THAT CREATES THE DTD ACCORDING TO THE DATABASE -
-----
call stratify_dtd (1,"xmlnet",0)
Response.Write(chr(10) & "]>" & chr(10))

' ----------- CALL THE PROCEDURE THAT GENERATES THE ACTUAL XML DATA DOCUMENT ---
------
```

```
Response.Write(chr(10) & chr(10))
call stratify_xml(1,"xmlnet",0)



%>
```

```
<!--#include file="adovbs.inc" -->
<!--#include file="setodbc.inc" -->
<%
dim arr()
counter=0
pid=Request.Form("pid")
cat_id=Request.Form("cat_id")
cat_name=Request.Form("cat_name")

set rst1=server.CreateObject("adodb.recordset")
sql="select obx_id from observations where patient_id=" & pid & " and
category_id=" & cat_id
rst1.Open sql,connobj,adOpenDynamic,adLockOptimistic
while not rst1.EOF
      obxid=rst1.Fields("obx_id")
      cname="c" & obxid
      checkid=Request.Form(cname)
      if checkid<>"" then
            counter=counter+1
            redim preserve arr(counter+1)
            arr(counter)=obxid
            'Response.Write(obxid & "<br>")
            rst1.Delete adAffectCurrent
            rst1.MovePrevious
      end if
      rst1.MoveNext
wend
'Response.Write("<br>" & pid & "<br>" & cat_id )
rst1.Close
sql=""
sql="delete from observation_coding where observation_id in ("
for x = 1 to counter
      sql=sql & arr(x) & ","
next
sql=sql & "0)"
'Response.Write(sql)
rst1.open sql,connobj,adOpenDynamic,adLockOptimistic
' recordset closes automatically after deletion
set rst1=nothing
Response.Redirect("get_data.asp?id=" & cat_id & "&cat_name=" & cat_name)

%>
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 FINAL//EN">
<HTML>
<HEAD>
<font size=2  face=serif>

<script language="javascript">
var html_text='';
var toggle=false;
var depth=1000;
var next_sib;
var vpos=0;
var vmax=1;
var temp_node;
var cat_id;
var xmldoc=new ActiveXObject("Microsoft.XMLDOM");
xmldoc.load("prepare_skeleton.asp");
loaddoc();

function loaddoc () {
      if (xmldoc.readyState=="4"){
            start();
            window.section.innerHTML=html_text;
            }
      else window.setTimeout("loaddoc()" , 1000 );
}

function start() {
      var root=xmldoc.documentElement;
      stratify(vpos,root);
      }


function stratify(vpos,node) {
      if (vpos<depth){
            node_name=node.nodeName;
            if (node_name != '#text')
                  node_att=node.attributes(0).text
            else
                  node_att=node.nodeName;

            node_text=node.text;
            taber='';
            for (x=0;x<vpos;x++)
                  taber=taber+'       ';

            if (node.hasChildNodes()){

                  local_html=taber+'<label
onclick=document.all("'+node_name+'").style.display="none">« </label>';
                  local_html=local_html+'<label
onclick=document.all("'+node_name+'").style.display="block">» </label>';
            }
            else
                  local_html=taber+'     ';

            cat_id=node.attributes(1).text;
```

```
            local_html=local_html+'<strong><font
color="#0000ff">'+node_att+'  <a  target="right_frame"
href="get_data.asp?id='+cat_id+'&cat_name='+node_att+'">»</a></font></strong><br
>';
            local_html=local_html+'<div
id="'+node_name+'">'+String.fromCharCode(10);
            html_text=html_text+local_html;
            if (node.hasChildNodes()){
                  if (vmax<vpos)
                        vmax=vpos;
                  stratify(vpos+1,node.childNodes(0));
                  taber='';
                  for (x=0;x<vpos;x++)
                        taber=taber+'    ';
            }
            html_text=html_text+'</div>'+String.fromCharCode(10);
            temp_node=node.nextSibling;
            if (temp_node != null && temp_node.nodeName != '#comment' )
                  stratify(vpos,temp_node);
      }
}

function display(flag){
      var root=xmldoc.documentElement;

      if (flag==1){
            show(root);
      }
      if (flag==2){
            hide(root);
      }
}

function hide(node) {
            node_name=node.nodeName;
            document.all(node_name).style.display="none";
            if (node.hasChildNodes()){
                  hide(node.childNodes(0));
            }
            temp_node=node.nextSibling;
            if (temp_node != null && temp_node.nodeName != '#comment' )
                  hide(temp_node);
}

function show(node) {
            node_name=node.nodeName;
            document.all(node_name).style.display="block";
            if (node.hasChildNodes()){
                  show(node.childNodes(0));
            }
            temp_node=node.nextSibling;
            if (temp_node != null && temp_node.nodeName != '#comment' )
                  show(temp_node);
}

</script>
</HEAD>
```

```
<body  background=backgrnd.gif>
<center>
<input name="showit" value="Show All" type="button" onclick="display(1)"> 
<input name="hideit" value="Hide All" type="button" onclick="display(2)"> 
<input name="view" value="View" type="button" language=javascript
onclick="window.open('view_patient.asp','right_frame')">

</center>
<br><hr noshade width=80%><br>
<div id="section"></div>
</BODY>
</HTML>
```

```
<%
'--------------------------------------------------------------------
' Microsoft ADO
'
' (c) 1996-1998 Microsoft Corporation.  All Rights Reserved.
'
'
'
' ADO constants include file for VBScript
'
'--------------------------------------------------------------------

'---- CursorTypeEnum Values ----
Const adOpenForwardOnly = 0
Const adOpenKeyset = 1
Const adOpenDynamic = 2
Const adOpenStatic = 3

'---- CursorOptionEnum Values ----
Const adHoldRecords = &H00000100
Const adMovePrevious = &H00000200
Const adAddNew = &H01000400
Const adDelete = &H01000800
Const adUpdate = &H01008000
Const adBookmark = &H00002000
Const adApproxPosition = &H00004000
Const adUpdateBatch = &H00010000
Const adResync = &H00020000
Const adNotify = &H00040000
Const adFind = &H00080000

'---- LockTypeEnum Values ----
Const adLockReadOnly = 1
Const adLockPessimistic = 2
Const adLockOptimistic = 3
Const adLockBatchOptimistic = 4

'---- ExecuteOptionEnum Values ----
Const adRunAsync = &H00000010
Const adAsyncExecute = &H00000010
Const adAsyncFetch = &H00000020
Const adAsyncFetchNonBlocking = &H00000040
Const adExecuteNoRecords = &H00000080

'---- ConnectOptionEnum Values ----
Const adAsyncConnect = &H00000010

'---- ObjectStateEnum Values ----
Const adStateClosed = &H00000000
Const adStateOpen = &H00000001
Const adStateConnecting = &H00000002
Const adStateExecuting = &H00000004
Const adStateFetching = &H00000008

'---- CursorLocationEnum Values ----
Const adUseServer = 2
Const adUseClient = 3
```

```
'---- DataTypeEnum Values ----
Const adEmpty = 0
Const adTinyInt = 16
Const adSmallInt = 2
Const adInteger = 3
Const adBigInt = 20
Const adUnsignedTinyInt = 17
Const adUnsignedSmallInt = 18
Const adUnsignedInt = 19
Const adUnsignedBigInt = 21
Const adSingle = 4
Const adDouble = 5
Const adCurrency = 6
Const adDecimal = 14
Const adNumeric = 131
Const adBoolean = 11
Const adError = 10
Const adUserDefined = 132
Const adVariant = 12
Const adIDispatch = 9
Const adIUnknown = 13
Const adGUID = 72
Const adDate = 7
Const adDBDate = 133
Const adDBTime = 134
Const adDBTimeStamp = 135
Const adBSTR = 8
Const adChar = 129
Const adVarChar = 200
Const adLongVarChar = 201
Const adWChar = 130
Const adVarWChar = 202
Const adLongVarWChar = 203
Const adBinary = 128
Const adVarBinary = 204
Const adLongVarBinary = 205
Const adChapter = 136
Const adFileTime = 64
Const adDBFileTime = 137
Const adPropVariant = 138
Const adVarNumeric = 139

'---- FieldAttributeEnum Values ----
Const adFldMayDefer = &H00000002
Const adFldUpdatable = &H00000004
Const adFldUnknownUpdatable = &H00000008
Const adFldFixed = &H00000010
Const adFldIsNullable = &H00000020
Const adFldMayBeNull = &H00000040
Const adFldLong = &H00000080
Const adFldRowID = &H00000100
Const adFldRowVersion = &H00000200
Const adFldCacheDeferred = &H00001000
Const adFldKeyColumn = &H00008000

'---- EditModeEnum Values ----
```

```
Const adEditNone = &H0000
Const adEditInProgress = &H0001
Const adEditAdd = &H0002
Const adEditDelete = &H0004

'---- RecordStatusEnum Values ----
Const adRecOK = &H0000000
Const adRecNew = &H0000001
Const adRecModified = &H0000002
Const adRecDeleted = &H0000004
Const adRecUnmodified = &H0000008
Const adRecInvalid = &H0000010
Const adRecMultipleChanges = &H0000040
Const adRecPendingChanges = &H0000080
Const adRecCanceled = &H0000100
Const adRecCantRelease = &H0000400
Const adRecConcurrencyViolation = &H0000800
Const adRecIntegrityViolation = &H0001000
Const adRecMaxChangesExceeded = &H0002000
Const adRecObjectOpen = &H0004000
Const adRecOutOfMemory = &H0008000
Const adRecPermissionDenied = &H0010000
Const adRecSchemaViolation = &H0020000
Const adRecDBDeleted = &H0040000

'---- GetRowsOptionEnum Values ----
Const adGetRowsRest = -1

'---- PositionEnum Values ----
Const adPosUnknown = -1
Const adPosBOF = -2
Const adPosEOF = -3

'---- enum Values ----
Const adBookmarkCurrent = 0
Const adBookmarkFirst = 1
Const adBookmarkLast = 2

'---- MarshalOptionsEnum Values ----
Const adMarshalAll = 0
Const adMarshalModifiedOnly = 1

'---- AffectEnum Values ----
Const adAffectCurrent = 1
Const adAffectGroup = 2
Const adAffectAll = 3
Const adAffectAllChapters = 4

'---- ResyncEnum Values ----
Const adResyncUnderlyingValues = 1
Const adResyncAllValues = 2

'---- CompareEnum Values ----
Const adCompareLessThan = 0
Const adCompareEqual = 1
Const adCompareGreaterThan = 2
Const adCompareNotEqual = 3
```

```
Const adCompareNotComparable = 4

'---- FilterGroupEnum Values ----
Const adFilterNone = 0
Const adFilterPendingRecords = 1
Const adFilterAffectedRecords = 2
Const adFilterFetchedRecords = 3
Const adFilterPredicate = 4
Const adFilterConflictingRecords = 5

'---- SearchDirectionEnum Values ----
Const adSearchForward = 1
Const adSearchBackward = -1

'---- PersistFormatEnum Values ----
Const adPersistADTG = 0
Const adPersistXML = 1
Const adPersistHTML = 2

'---- StringFormatEnum Values ----
Const adStringXML = 0
Const adStringHTML = 1
Const adClipString = 2

'---- ADCPROP_UPDATECRITERIA_ENUM Values ----
Const adCriteriaKey = 0
Const adCriteriaAllCols = 1
Const adCriteriaUpdCols = 2
Const adCriteriaTimeStamp = 3

'---- ADCPROP_ASYNCTHREADPRIORITY_ENUM Values ----
Const adPriorityLowest = 1
Const adPriorityBelowNormal = 2
Const adPriorityNormal = 3
Const adPriorityAboveNormal = 4
Const adPriorityHighest = 5

'---- ConnectPromptEnum Values ----
Const adPromptAlways = 1
Const adPromptComplete = 2
Const adPromptCompleteRequired = 3
Const adPromptNever = 4

'---- ConnectModeEnum Values ----
Const adModeUnknown = 0
Const adModeRead = 1
Const adModeWrite = 2
Const adModeReadWrite = 3
Const adModeShareDenyRead = 4
Const adModeShareDenyWrite = 8
Const adModeShareExclusive = &Hc
Const adModeShareDenyNone = &H10

'---- IsolationLevelEnum Values ----
Const adXactUnspecified = &Hffffffff
Const adXactChaos = &H00000010
Const adXactReadUncommitted = &H00000100
```

```
Const adXactBrowse = &H00000100
Const adXactCursorStability = &H00001000
Const adXactReadCommitted = &H00001000
Const adXactRepeatableRead = &H00010000
Const adXactSerializable = &H00100000
Const adXactIsolated = &H00100000

'---- XactAttributeEnum Values ----
Const adXactCommitRetaining = &H00020000
Const adXactAbortRetaining = &H00040000

'---- PropertyAttributesEnum Values ----
Const adPropNotSupported = &H0000
Const adPropRequired = &H0001
Const adPropOptional = &H0002
Const adPropRead = &H0200
Const adPropWrite = &H0400

'---- ErrorValueEnum Values ----
Const adErrInvalidArgument = &Hbb9
Const adErrNoCurrentRecord = &Hbcd
Const adErrIllegalOperation = &Hc93
Const adErrInTransaction = &Hcae
Const adErrFeatureNotAvailable = &Hcb3
Const adErrItemNotFound = &Hcc1
Const adErrObjectInCollection = &Hd27
Const adErrObjectNotSet = &Hd5c
Const adErrDataConversion = &Hd5d
Const adErrObjectClosed = &He78
Const adErrObjectOpen = &He79
Const adErrProviderNotFound = &He7a
Const adErrBoundToCommand = &He7b
Const adErrInvalidParamInfo = &He7c
Const adErrInvalidConnection = &He7d
Const adErrNotReentrant = &He7e
Const adErrStillExecuting = &He7f
Const adErrOperationCancelled = &He80
Const adErrStillConnecting = &He81
Const adErrNotExecuting = &He83
Const adErrUnsafeOperation = &He84

'---- ParameterAttributesEnum Values ----
Const adParamSigned = &H0010
Const adParamNullable = &H0040
Const adParamLong = &H0080

'---- ParameterDirectionEnum Values ----
Const adParamUnknown = &H0000
Const adParamInput = &H0001
Const adParamOutput = &H0002
Const adParamInputOutput = &H0003
Const adParamReturnValue = &H0004

'---- CommandTypeEnum Values ----
Const adCmdUnknown = &H0008
Const adCmdText = &H0001
Const adCmdTable = &H0002
```

```
Const adCmdStoredProc = &H0004
Const adCmdFile = &H0100
Const adCmdTableDirect = &H0200

'---- EventStatusEnum Values ----
Const adStatusOK = &H0000001
Const adStatusErrorsOccurred = &H0000002
Const adStatusCantDeny = &H0000003
Const adStatusCancel = &H0000004
Const adStatusUnwantedEvent = &H0000005

'---- EventReasonEnum Values ----
Const adRsnAddNew = 1
Const adRsnDelete = 2
Const adRsnUpdate = 3
Const adRsnUndoUpdate = 4
Const adRsnUndoAddNew = 5
Const adRsnUndoDelete = 6
Const adRsnRequery = 7
Const adRsnResynch = 8
Const adRsnClose = 9
Const adRsnMove = 10
Const adRsnFirstChange = 11
Const adRsnMoveFirst = 12
Const adRsnMoveNext = 13
Const adRsnMovePrevious = 14
Const adRsnMoveLast = 15

'---- SchemaEnum Values ----
Const adSchemaProviderSpecific = -1
Const adSchemaAsserts = 0
Const adSchemaCatalogs = 1
Const adSchemaCharacterSets = 2
Const adSchemaCollations = 3
Const adSchemaColumns = 4
Const adSchemaCheckConstraints = 5
Const adSchemaConstraintColumnUsage = 6
Const adSchemaConstraintTableUsage = 7
Const adSchemaKeyColumnUsage = 8
Const adSchemaReferentialContraints = 9
Const adSchemaTableConstraints = 10
Const adSchemaColumnsDomainUsage = 11
Const adSchemaIndexes = 12
Const adSchemaColumnPrivileges = 13
Const adSchemaTablePrivileges = 14
Const adSchemaUsagePrivileges = 15
Const adSchemaProcedures = 16
Const adSchemaSchemata = 17
Const adSchemaSQLLanguages = 18
Const adSchemaStatistics = 19
Const adSchemaTables = 20
Const adSchemaTranslations = 21
Const adSchemaProviderTypes = 22
Const adSchemaViews = 23
Const adSchemaViewColumnUsage = 24
Const adSchemaViewTableUsage = 25
Const adSchemaProcedureParameters = 26
```

```
Const adSchemaForeignKeys = 27
Const adSchemaPrimaryKeys = 28
Const adSchemaProcedureColumns = 29
Const adSchemaDBInfoKeywords = 30
Const adSchemaDBInfoLiterals = 31
Const adSchemaCubes = 32
Const adSchemaDimensions = 33
Const adSchemaHierarchies = 34
Const adSchemaLevels = 35
Const adSchemaMeasures = 36
Const adSchemaProperties = 37
Const adSchemaMembers = 38

'---- SeekEnum Values ----
Const adSeekFirstEQ = &H1
Const adSeekLastEQ = &H2
Const adSeekGE = &H4
Const adSeekGT = &H8
Const adSeekLE = &H10
Const adSeekLT = &H20
%>
```

What is claimed is:

1. A method of distributing health information, comprising the steps of:

A. categorizing health information for an individual into a plurality of hierarchical sets of health information;

B. assigning by said individual access priority data representative of an access priority level to each of said plurality of sets of health information in said hierarchy, said access priority levels being based on differing criteria for release authorization for each of said plurality of sets of health information established by said individual;

C. storing, at a datastore, each of said plurality of sets of health information in said hierarchy and associated access priority data;

D. providing, by said individual to one or more requesters, access priority data corresponding to a desired level in said hierarchy;

E. receiving, from a requestor, by way of a communications network, a request for at least one of said plurality of sets of health information in said hierarchy, said request including access priority data correlated to an access priority level;

F. processing said access priority data to determine whether said access priority data corresponds to said access priority level for said requested health information; and

i. when said access priority data corresponds to said access priority level for said requested health information, transmitting said requested health information to said requester by way of said communications network, and

ii. when said access data fails to correspond to said access priority level, denying access to said requestor to said health information.

2. The method according to claim 1, wherein said communications network is the Internet.

3. The method according to claim 1, wherein said transmitted health information is encrypted.

4. The method of distributing medical according to claim 2 further comprising the step of designating certain of said access priority data as identification constraints which must be received in step D before access to said medical information is granted.

5. A method of distributing medical information for an individual over a communications network comprising the steps of:

A. generating a plurality of access security codes;

B. generating a plurality of hierarchical categories, ranging from a low security category to a high security category;

C. categorizing the individual's medical information into privacy levels ranging from a least private level to a most private level;

D. inputting the individual's categorized medical information into said plurality of hierarchical categories, said least private level being input into said low security category and said most private level being input into said high security category;

E. assigning by said individual, to each of said categories, one or more of said access security codes, such that said medical information in each category will be released only if the assigned access security codes are received;

F. providing, by said individual to one or more requestors, access priority data corresponding to a desired level in said hierarchy;

G. receiving, from a requestor, one or more of said access security codes over said communications network;

H. determining whether said received access security codes match one or more of said assigned access security codes; and

I. transmitting, to said requestor over said communications network, said medical information in said categories in which said received security access codes match said assigned security access codes.

6. The method of distributing medical information for an individual over a network according to claim 5, wherein said communications network is the internet.

7. The method of distributing medical information for an individual over a network according to claim 6, wherein said released medical information is encrypted.

8. The method of distributing medical information for an individual over a network according to claim 6 further comprising the step of designating certain of said security access codes as identification constraints which must be received in step F before access to said medical information is granted.

9. The method of distributing medical information for an individual over a network according to claim 6 wherein, prior to step F, identification information is received from the requestor, said identification information being for identifying the individual.

10. The method of distributing medical information for an individual over a network according to claim 9 wherein said identification information is selected from the group consisting of the individual's medical record numbers, demographic data, information from a smart card that identifies the patient, retinal scans, iris scans and fingerprints.

11. The method of distributing medical information for an individual over a network according to claim 9 wherein said identification information is any information about the individual which is available to said requester.

12. A system for distributing medical information for an individual over a communications network comprising:

a server system including a computer processor and associated memory, said server system having database of a plurality of hierarchical categories for the individual, said categories ranging from a low security category to a high security category, each of said categories having medical information of said individual contained therein, said medical information ranging from least private information to most private information, said least private medical information being contained in said low security category and said most private medical information being contained in said high security category, each of said categories having one or more security access codes assigned thereto;

a request system including a computer processor and associated memory, said request system inputting, in response to a requestor, one or more of said security access codes provided to said requestor by said individual, to said server system over said communications network; and

an access determining device for transmitting, to said request system, the medical information in each of said categories in which said input security access codes match said assigned security access codes.

13. The system according to claim 12 wherein said communications network is the internet.

14. The system according to claim 13, further including a setup system, including a computer processor and associated memory, for inputting said medical information to said database.

15. The system according to claim 14 wherein said security access codes are defined by a user of the system for distributing medical information and are assigned to said categories by said user through said setup system.

16. The system according to claim 13 wherein more of said security access codes are required to access said high security categories than said low security categories.

17. The system according to claim 13 wherein said setup system and said requester system are the same system.

18. The system of claim 13 wherein said request system is coupleable to said network by wired connection.

19. The system of claim 18 wherein said request system is selected from the group consisting of a personal computer, an interactive television system, a personal digital assistant and a cellular telephone.

20. The system of claim 13 wherein said request system is coupleable to said network by a wireless connection.

21. The system of claim 20 wherein said request system is selected from the group consisting of a personal computer, an interactive television system, a personal digital assistant and a cellular telephone.

22. The system of claim 14 wherein said setup system is coupleable to said network by a wired connection.

23. The system of claim 22 wherein said setup system is selected from the group consisting of a personal computer, an interactive television system, a personal digital assistant and a cellular telephone.

24. The system of claim 14 wherein said setup system is coupleable to said network by a wireless connection.

25. The system of claim 24 wherein said setup system is selected from the group consisting of a personal computer, an interactive television system, a personal digital assistant and a cellular telephone.

26. A method of distributing personal information, comprising the steps of:

A. categorizing personal information for an individual into a plurality of hierarchical sets of personal information;

B. assigning, by said individual, access priority data representative of an access priority level to each of said plurality of sets of personal information in said hierarchy, said access priority levels being based on differing criteria for release authorization for each of said plurality of sets of personal information established by said individual;

C. storing, at a datastore, each of said plurality of sets of personal information in said hierarchy and associated access priority data;

D. providing, by said individual to one or more requesters, access priority data corresponding to a desired level in said hierarchy;

E. receiving from a requestor, by way of a communications network, a request for at least one of said plurality of sets of personal information in said hierarchy, said request including access priority data correlated to an access priority level;

F. processing said access priority data to determine whether said access priority data corresponds to said access priority level for said requested personal information; and

i. when said access priority data corresponds to said access priority level for said requested personal information, transmitting said requested personal information to said requestor by way of said communications network, and

ii. when said access data fails to correspond to said access priority level, denying access to said requester to said personal information.

27. A method of distributing personal information for an individual over a communications network comprising the steps of:

A. generating a plurality of access security codes;

B. generating a plurality of hierarchical categories, ranging from a low security category to a high security category;

C. categorizing the individual's personal information into privacy levels ranging from a least private level to a most private level;

D. inputting the individual's categorized personal information into said plurality of hierarchical categories, said least private level being input into said low security category and said most private level being input into said high security category;

E. assigning, by said individual, to each of said categories, one or more of said access security codes, such that said personal information in each category will be released only if the assigned access security codes are received;

F. providing by said individual to one or more requestors, access priority data corresponding to a desired level in said hierarchy;

G. receiving, from a requestor, one or more of said access security codes over said communications network;

H. determining whether said received access security codes match one or more of said assigned access security codes; and

I. transmitting, to said requestor over said communications network, said personal information in said categories in said hierarchy which said received security access codes match said assigned security access codes.

28. The method of distributing personal information for an individual over a network according to claim 27 further comprising the step of designating certain of said security access codes as identification constraints which must be received in step F before access to said personal information is granted.

29. The method of distributing personal information for an individual over a network according to claim 28 wherein, prior to step F, identification information is received from the requester, said identification information being for identifying the individual.

30. The method of distributing personal information for an individual over a network according to claim 29 wherein said identification information is selected from the group consisting of the individual's demographic data, information from a smart card that identifies the patient, retinal scans, iris scans and fingerprints.

31. A system for distributing personal information for an individual over a communications network comprising:

a server system including a computer processor and associated memory, said server system having database of a plurality of hierarchical categories for the individual, said categories ranging from a low security category to a high security category, each of said categories having personal information of said individual contained therein, said personal information ranging from least private information to most private information, said least private personal information being contained in said low security category and said most private personal information being contained in said high security category, each of said categories having one or more security access codes assigned thereto;

a request system including a computer processor and associated memory, said request system inputting, in response to a requester, one or more of said security access

codes provided to said requester by said individual, to said server system over said communications network; and

an access determining device for transmitting, to said requests system, the personal information in each of said categories in which said input security access codes match said assigned security access codes.

32. The system according to claim 31, further including a setup system, including a computer processor and associated memory, for inputting said personal information to said database.

33. The system according to claim 32 wherein said security access codes are defined by said individual and are assigned to said categories by said individual through said setup system.

34. The system according to claim 31 wherein more of said security access codes are required to access said high security categories than said low security categories.

35. The system according to claim 32 wherein said setup system and said request system are the same system.

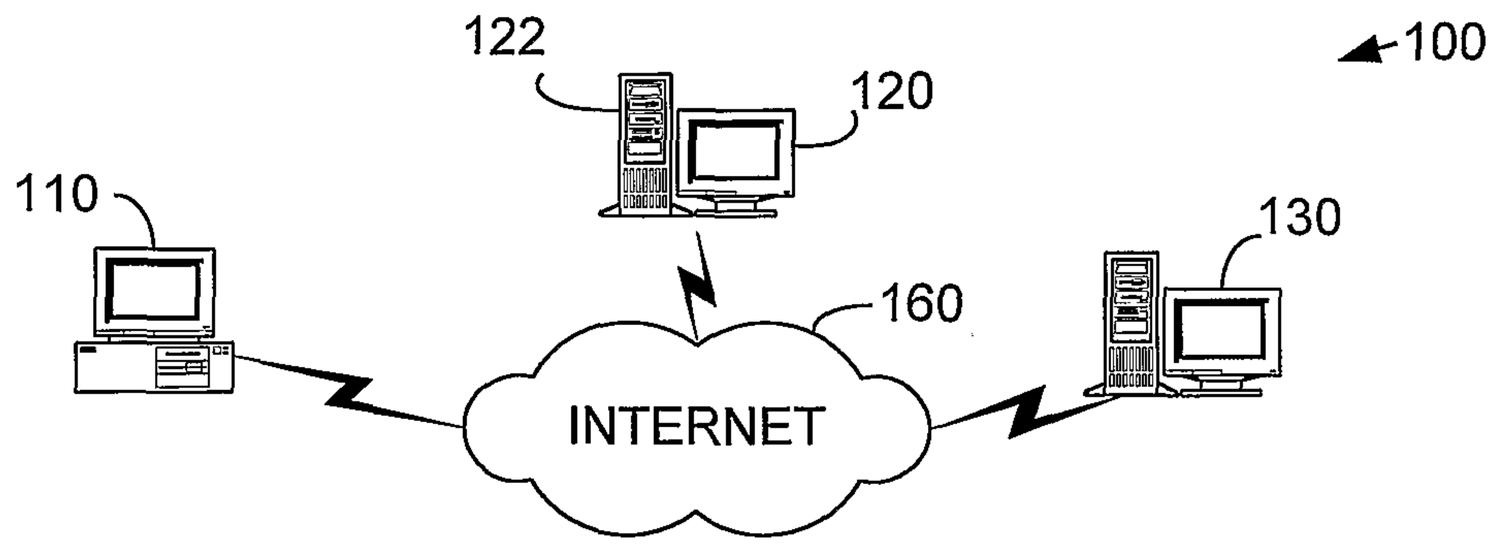


FIG.1

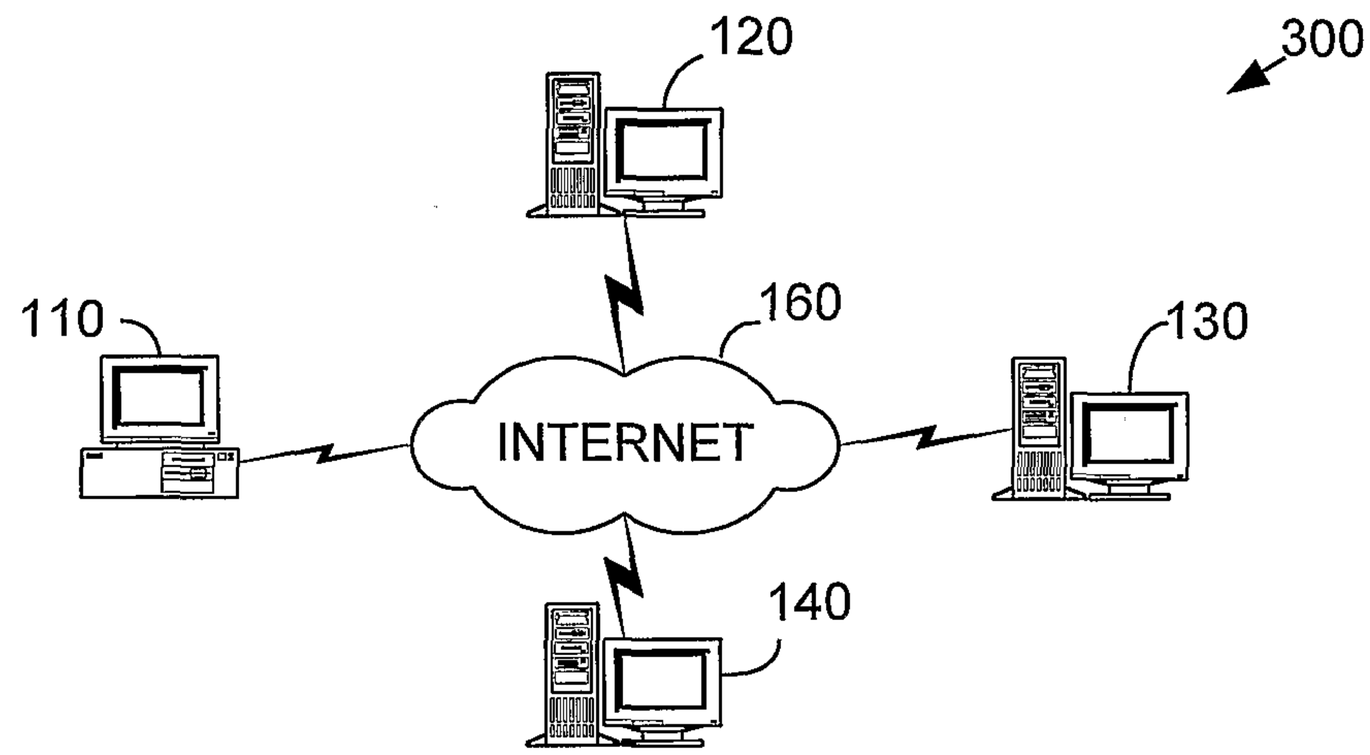


FIG.4

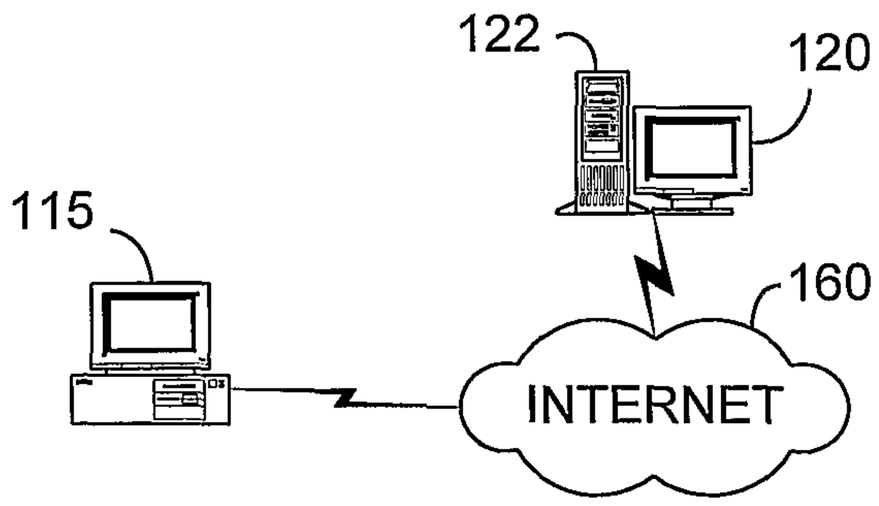


FIG.5

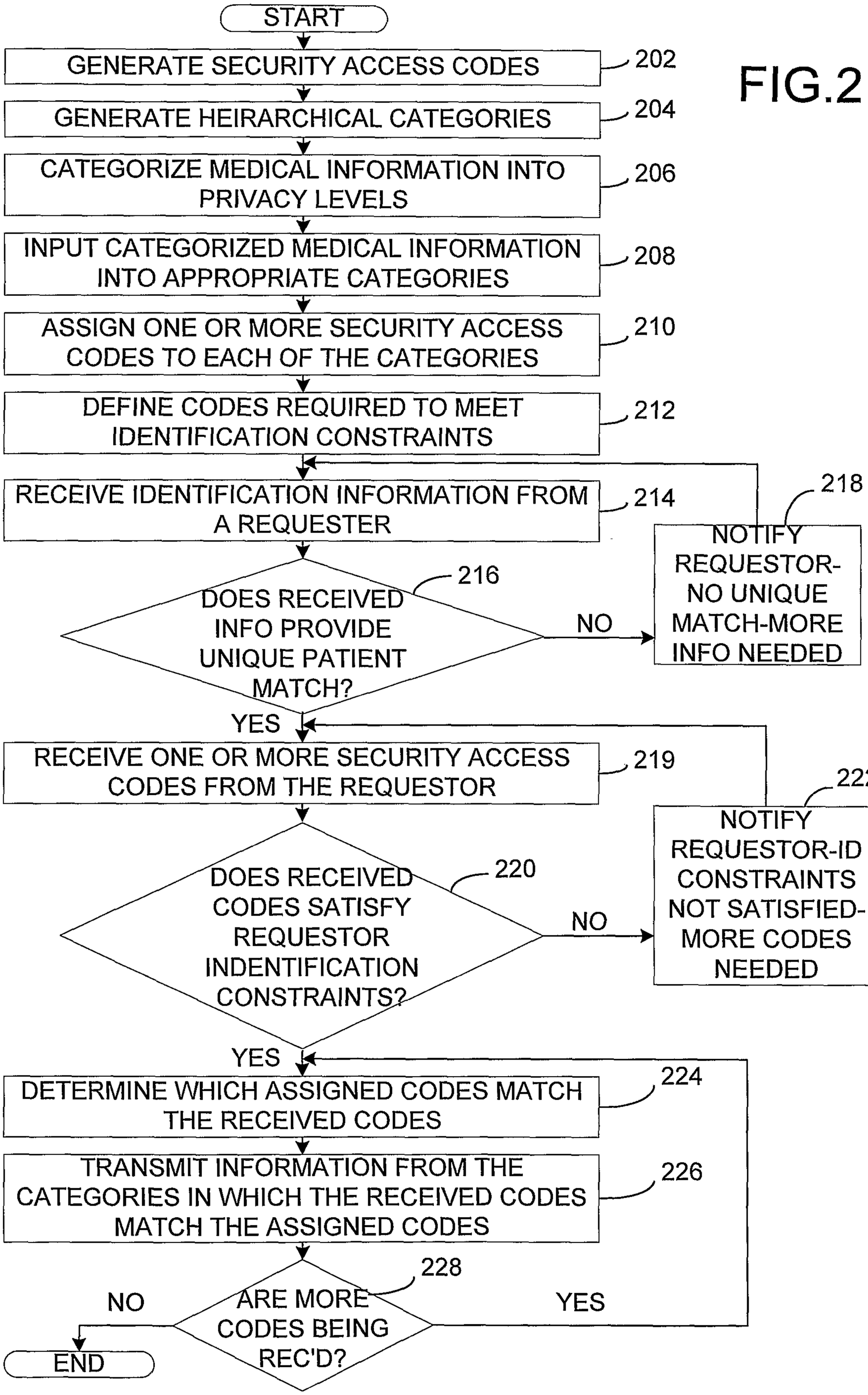

START
GENERATE SECURITY ACCESS CODES
202
GENERATE HEIRARCHICAL CATEGORIES
204
CATEGORIZE MEDICAL INFORMATION INTO PRIVACY LEVELS
206
INPUT CATEGORIZED MEDICAL INFORMATION INTO APPROPRIATE CATEGORIES
208
ASSIGN ONE OR MORE SECURITY ACCESS CODES TO EACH OF THE CATEGORIES
210
DEFINE CODES REQUIRED TO MEET IDENTIFICATION CONSTRAINTS
212
RECEIVE IDENTIFICATION INFORMATION FROM A REQUESTER
214
DOES RECEIVED INFO PROVIDE UNIQUE PATIENT MATCH?
216
NO
NOTIFY REQUESTOR-NO UNIQUE MATCH-MORE INFO NEEDED
218
YES
RECEIVE ONE OR MORE SECURITY ACCESS CODES FROM THE REQUESTOR
219
DOES RECEIVED CODES SATISFY REQUESTOR INDENTIFICATION CONSTRAINTS?
220
NO
NOTIFY REQUESTOR-ID CONSTRAINTS NOT SATISFIED-MORE CODES NEEDED
222
YES
DETERMINE WHICH ASSIGNED CODES MATCH THE RECEIVED CODES
224
TRANSMIT INFORMATION FROM THE CATEGORIES IN WHICH THE RECEIVED CODES MATCH THE ASSIGNED CODES
226
ARE MORE CODES BEING REC'D?
228
NO
YES
END

FIG.2

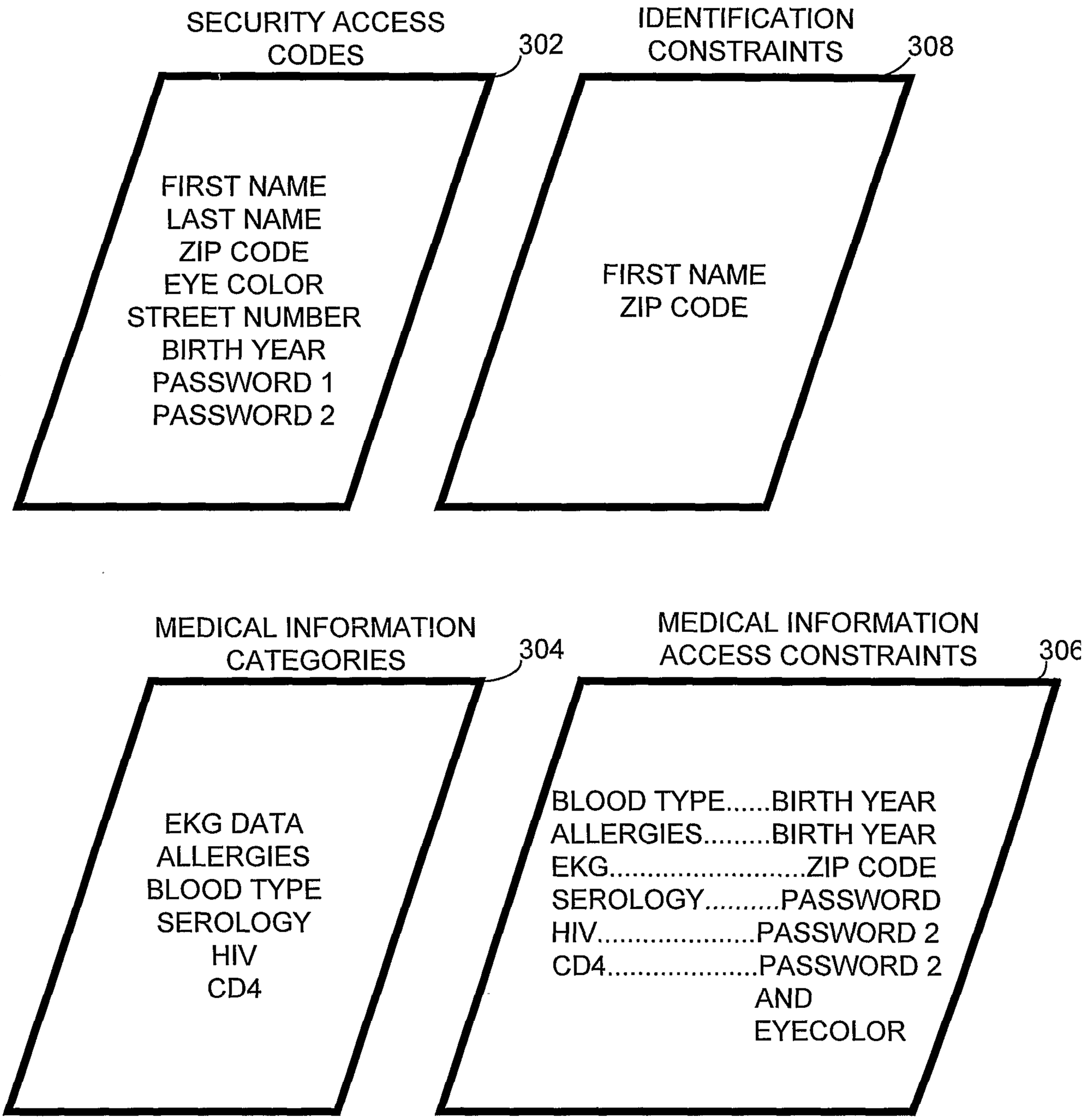
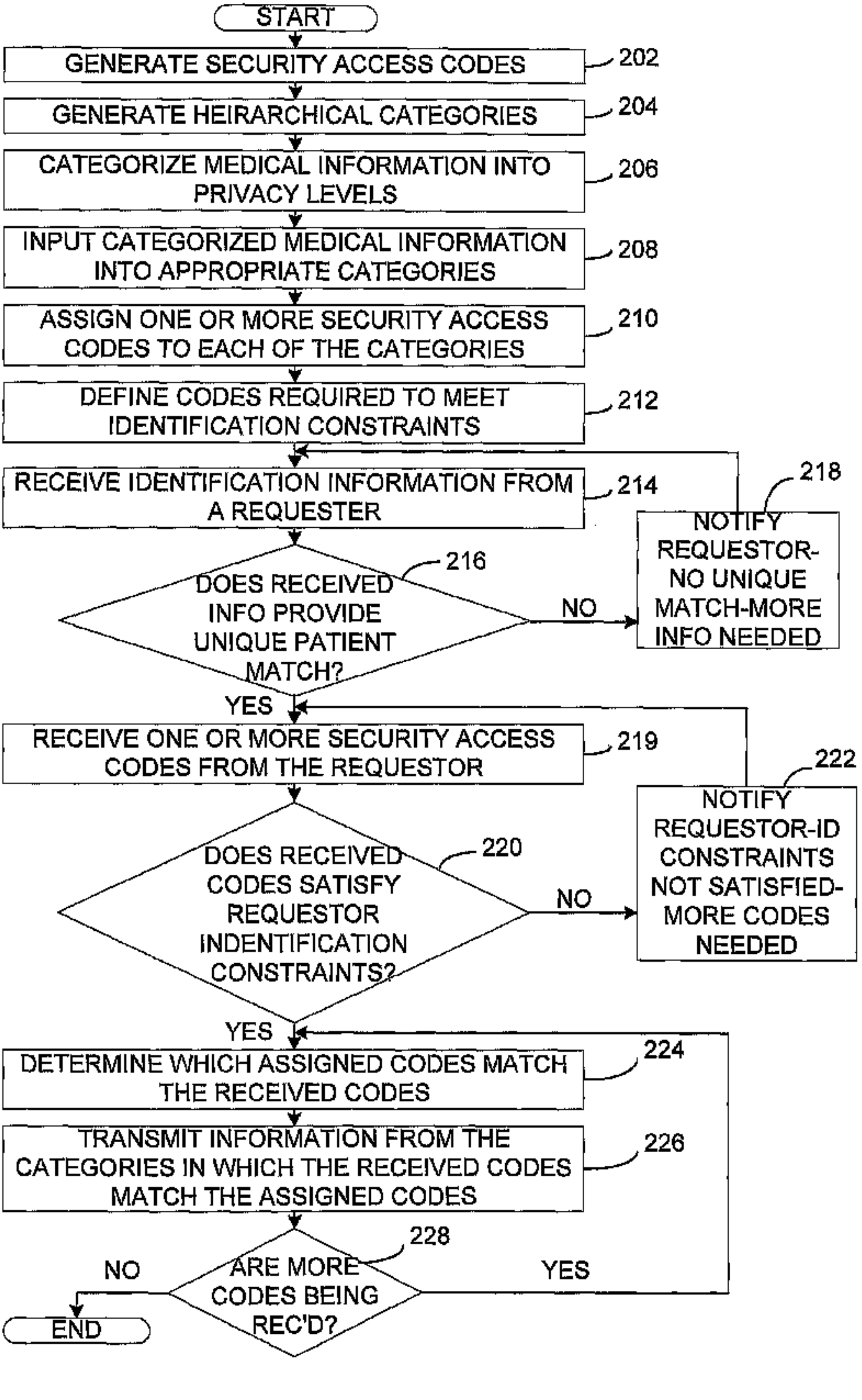

SECURITY ACCESS CODES
302
FIRST NAME
LAST NAME
ZIP CODE
EYE COLOR
STREET NUMBER
BIRTH YEAR
PASSWORD 1
PASSWORD 2
IDENTIFICATION CONSTRAINTS
308
FIRST NAME
ZIP CODE
MEDICAL INFORMATION CATEGORIES
304
EKG DATA
ALLERGIES
BLOOD TYPE
SEROLOGY
HIV
CD4
MEDICAL INFORMATION ACCESS CONSTRAINTS
306
BLOOD TYPE.....BIRTH YEAR
ALLERGIES.....BIRTH YEAR
EKG.....ZIP CODE
SEROLOGY.....PASSWORD
HIV.....PASSWORD 2
CD4.....PASSWORD 2
AND
EYECOLOR

FIG.3


START
GENERATE SECURITY ACCESS CODES
202
GENERATE HEIRARCHICAL CATEGORIES
204
CATEGORIZE MEDICAL INFORMATION INTO PRIVACY LEVELS
206
INPUT CATEGORIZED MEDICAL INFORMATION INTO APPROPRIATE CATEGORIES
208
ASSIGN ONE OR MORE SECURITY ACCESS CODES TO EACH OF THE CATEGORIES
210
DEFINE CODES REQUIRED TO MEET IDENTIFICATION CONSTRAINTS
212
RECEIVE IDENTIFICATION INFORMATION FROM A REQUESTER
214
218
NOTIFY REQUESTOR-NO UNIQUE MATCH-MORE INFO NEEDED
216
DOES RECEIVED INFO PROVIDE UNIQUE PATIENT MATCH?
NO
YES
RECEIVE ONE OR MORE SECURITY ACCESS CODES FROM THE REQUESTOR
219
222
NOTIFY REQUESTOR-ID CONSTRAINTS NOT SATISFIED-MORE CODES NEEDED
220
DOES RECEIVED CODES SATISFY REQUESTOR INDENTIFICATION CONSTRAINTS?
NO
YES
DETERMINE WHICH ASSIGNED CODES MATCH THE RECEIVED CODES
224
TRANSMIT INFORMATION FROM THE CATEGORIES IN WHICH THE RECEIVED CODES MATCH THE ASSIGNED CODES
226
228
ARE MORE CODES BEING REC'D?
NO
YES
END