US 20030074660A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2003/0074660 A1

McCormack et al. (43) Pub. Date: Apr. 17, 2003

(54) SYSTEM METHOD AND APPARATUS FOR PORTABLE DIGITAL IDENTITY

(75) Inventors: **Jonathan I. McCormack**, Sunnyvale, CA (US); **Venkatachary Srinivasan**, Sunnyvale, CA (US); **Hari Vasudev**, Milpitas, CA (US); **Raymond Drewery**, Woodside, CA (US)

Correspondence Address:
ALLAN J. JACOBSON
13310 Summit Square Center
Route 413 & Doublewoods Road
Langhorne, PA 19047 (US)

(73) Assignee: **Liberate Technologies**

(21) Appl. No.: **09/977,085**

(22) Filed: **Oct. 12, 2001**

Publication Classification

(51) **Int. Cl.**[7] **H04N 7/16**; H04N 7/173; G06F 15/16

(52) **U.S. Cl.** **725/2**; 725/110; 709/219

(57) **ABSTRACT**

Two-way digital media devices typically store digital identifying data that identify the user to providers of content and interactive data. In the case of a Web browser of a personal computer, the digital identity is stored in the form of a plurality of cookies that are used by respective web sites to personalize the web site experience for each particular user. When a user is at a different computer, the digital identifying data is not available. In addition, other types of interactive devices, such as CATV settop boxes, cell phones, PDAs and the like, may not have enough non-volatile memory (persistent storage) to store the digital identifying data. In order to provide users with a portable digital identity, a digital identity server is provided as a server node on the Internet, which retrieves digital identifying data and downloads such digital identifying data to any device upon request. In such manner, the user's digital identity is portable and available at any computer or other digital device that is being used. The system digital identity server permits devices without sufficient non-volatile memory storage to download a digital identity for temporary storage in volatile memory, thereby providing a digital identity in devices without non-volatile memory.

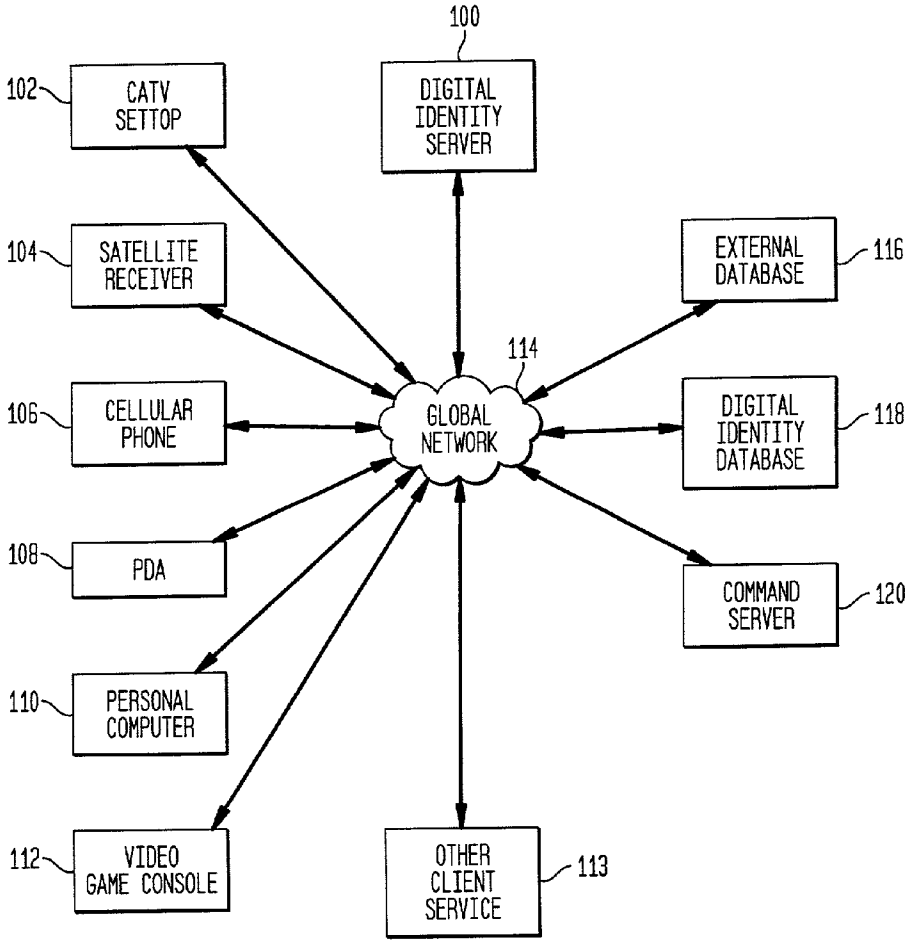


FIG. 1A

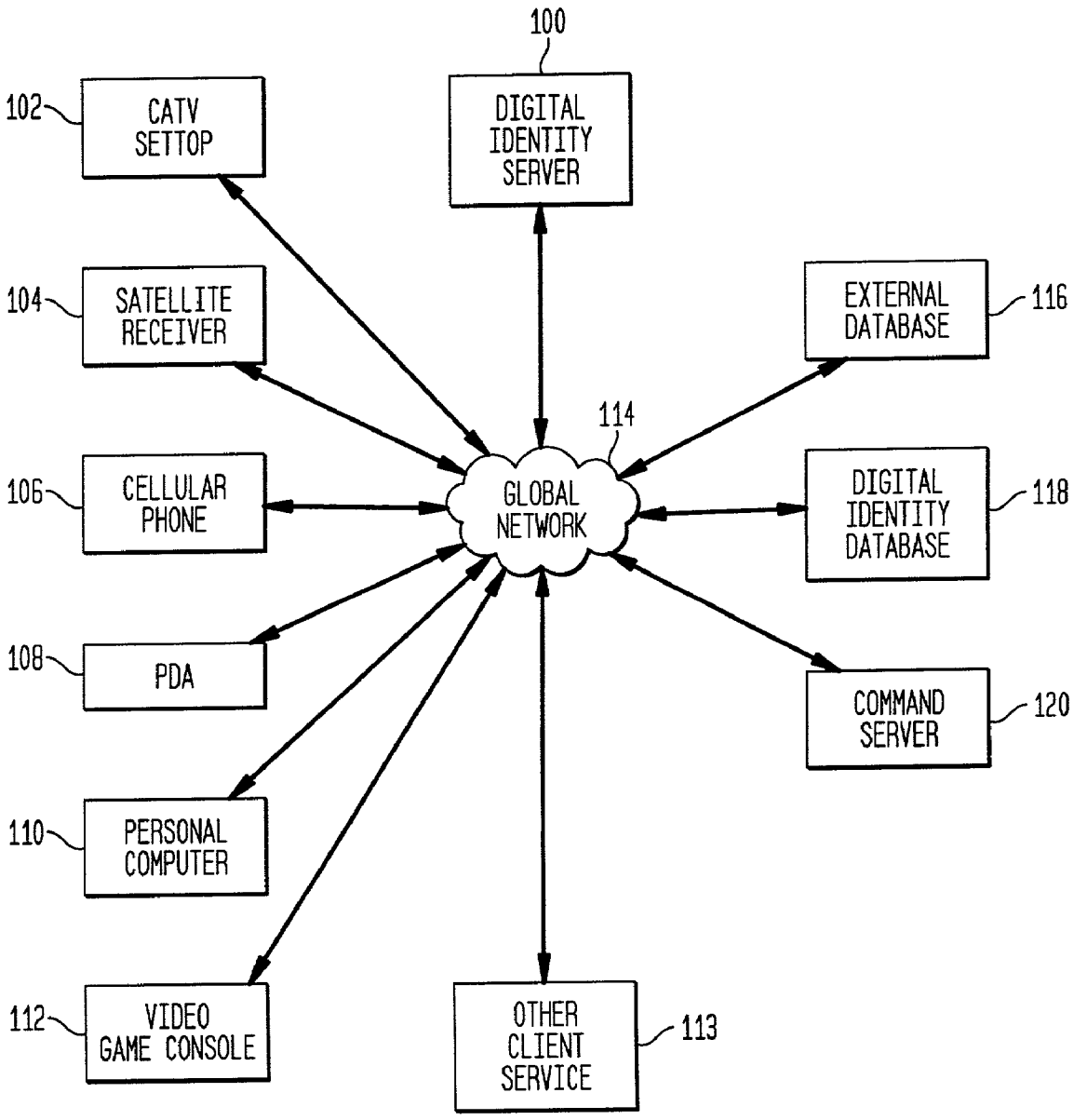
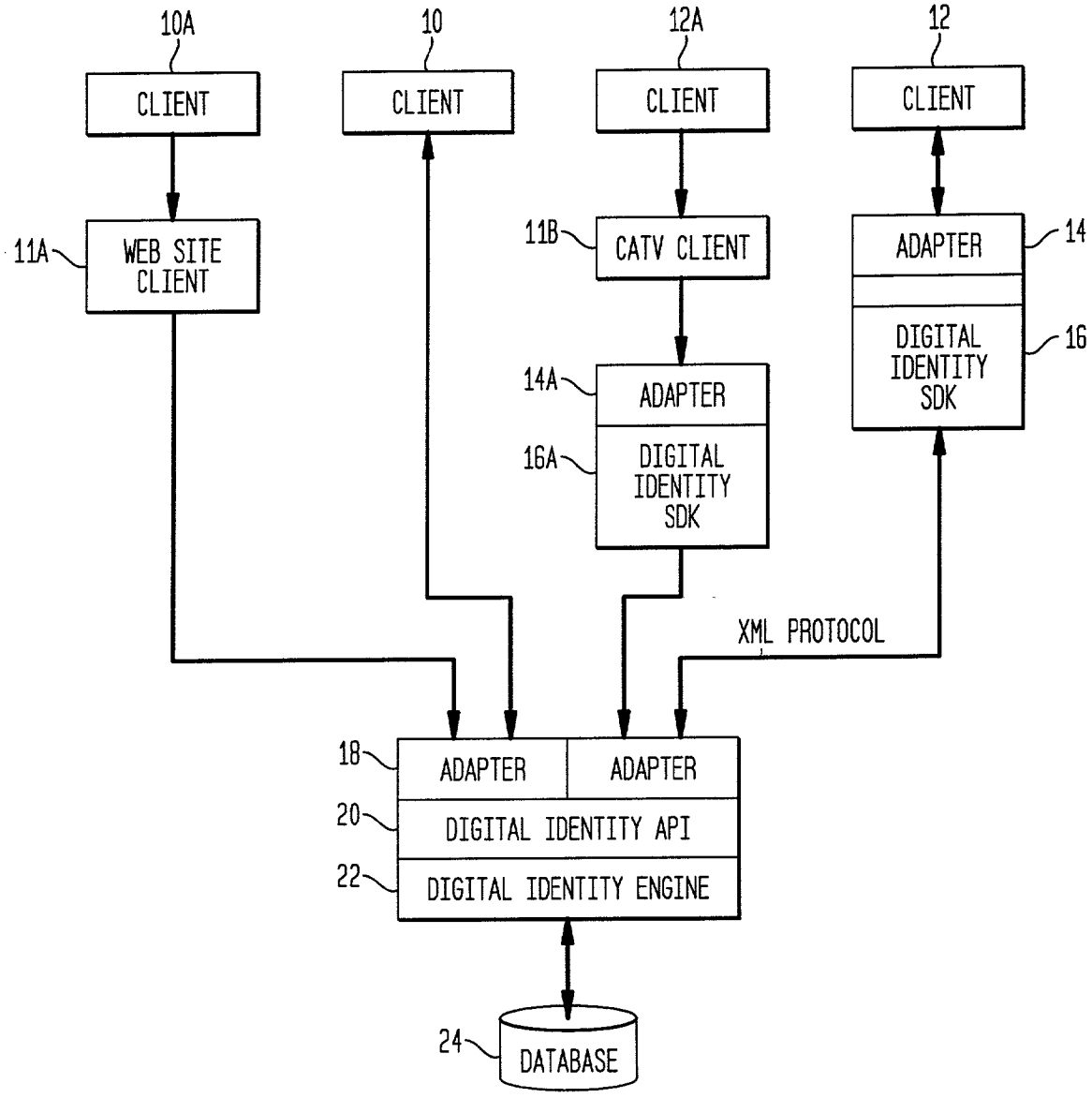

100
DIGITAL IDENTITY SERVER
102
CATV SETTOP
104
SATELLITE RECEIVER
106
CELLULAR PHONE
108
PDA
110
PERSONAL COMPUTER
112
VIDEO GAME CONSOLE
114
GLOBAL NETWORK
116
EXTERNAL DATABASE
118
DIGITAL IDENTITY DATABASE
120
COMMAND SERVER
113
OTHER CLIENT SERVICE

FIG. 1B


10A
CLIENT
10
CLIENT
12A
CLIENT
12
CLIENT
11A
WEB SITE CLIENT
11B
CATV CLIENT
ADAPTER
14
DIGITAL IDENTITY SDK
16
14A
ADAPTER
16A
DIGITAL IDENTITY SDK
XML PROTOCOL
18
ADAPTER
ADAPTER
20
DIGITAL IDENTITY API
22
DIGITAL IDENTITY ENGINE
24
DATABASE

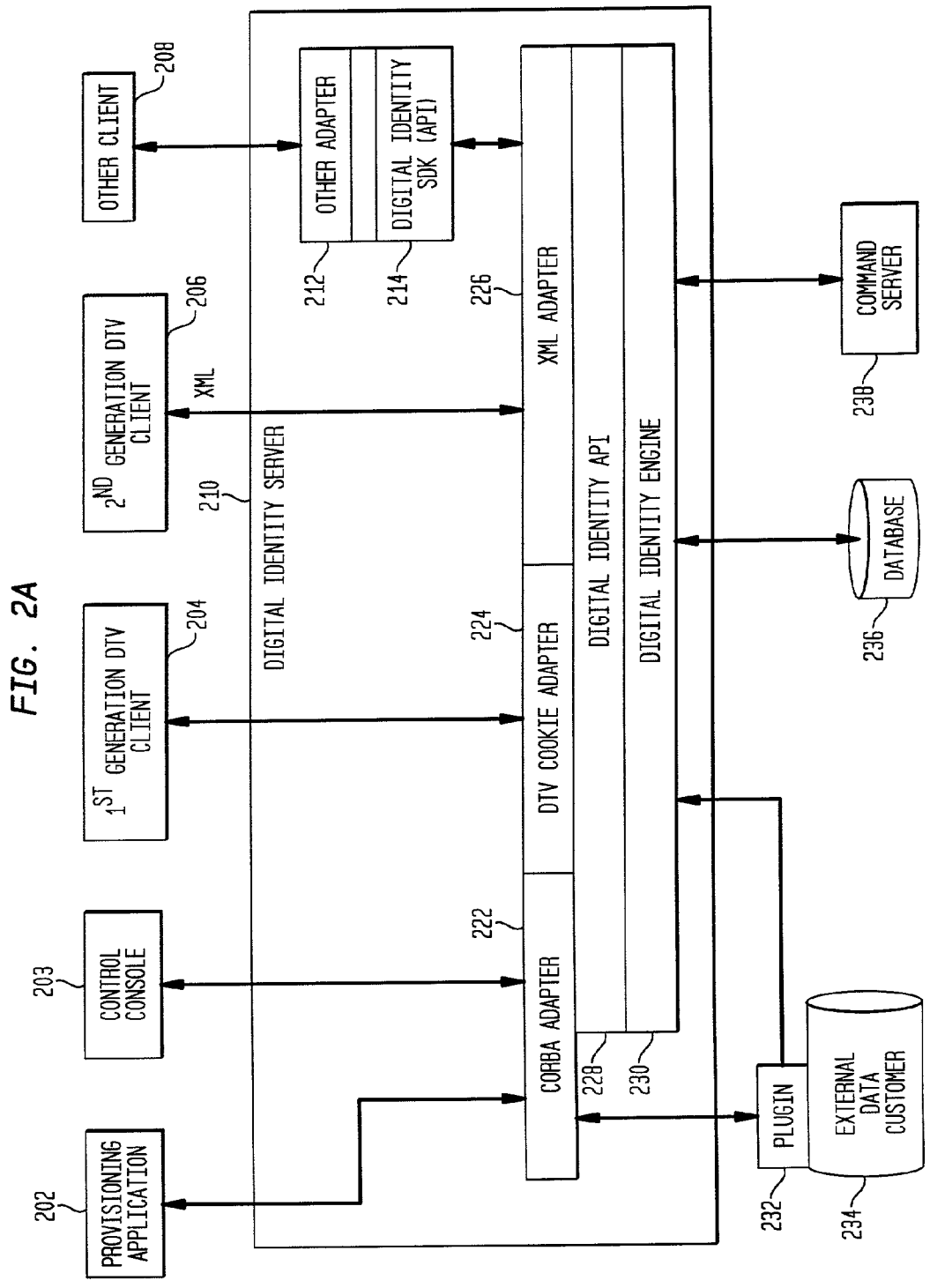

PROVISIONING APPLICATION
202
CONTROL CONSOLE
203
1ST GENERATION DTV CLIENT
204
2ND GENERATION DTV CLIENT
206
OTHER CLIENT
208
DIGITAL IDENTITY SERVER
210
XML
OTHER ADAPTER
212
DIGITAL IDENTITY SDK (API)
214
CORBA ADAPTER
222
DTV COOKIE ADAPTER
224
XML ADAPTER
226
DIGITAL IDENTITY API
228
DIGITAL IDENTITY ENGINE
230
PLUGIN
232
EXTERNAL DATA CUSTOMER
234
DATABASE
236
COMMAND SERVER
238

FIG. 2A

FIG. 2B

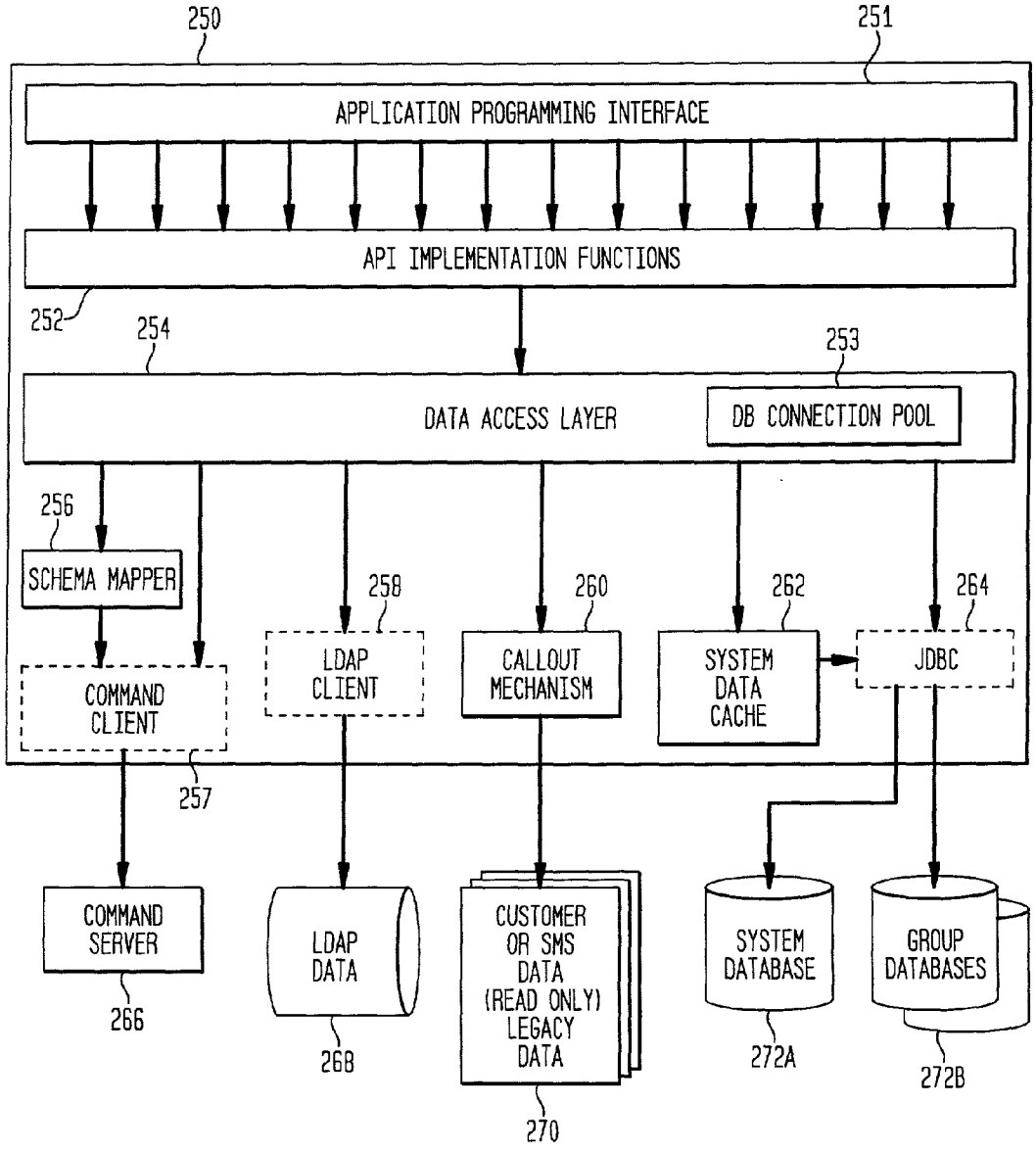

250
251
APPLICATION PROGRAMMING INTERFACE
API IMPLEMENTATION FUNCTIONS
252
254
253
DATA ACCESS LAYER
DB CONNECTION POOL
256
SCHEMA MAPPER
258
260
262
264
LDAP CLIENT
CALLOUT MECHANISM
SYSTEM DATA CACHE
JDBC
COMMAND CLIENT
257
COMMAND SERVER
266
LDAP DATA
268
CUSTOMER OR SMS DATA (READ ONLY) LEGACY DATA
270
SYSTEM DATABASE
272A
GROUP DATABASES
272B

FIG. 2C

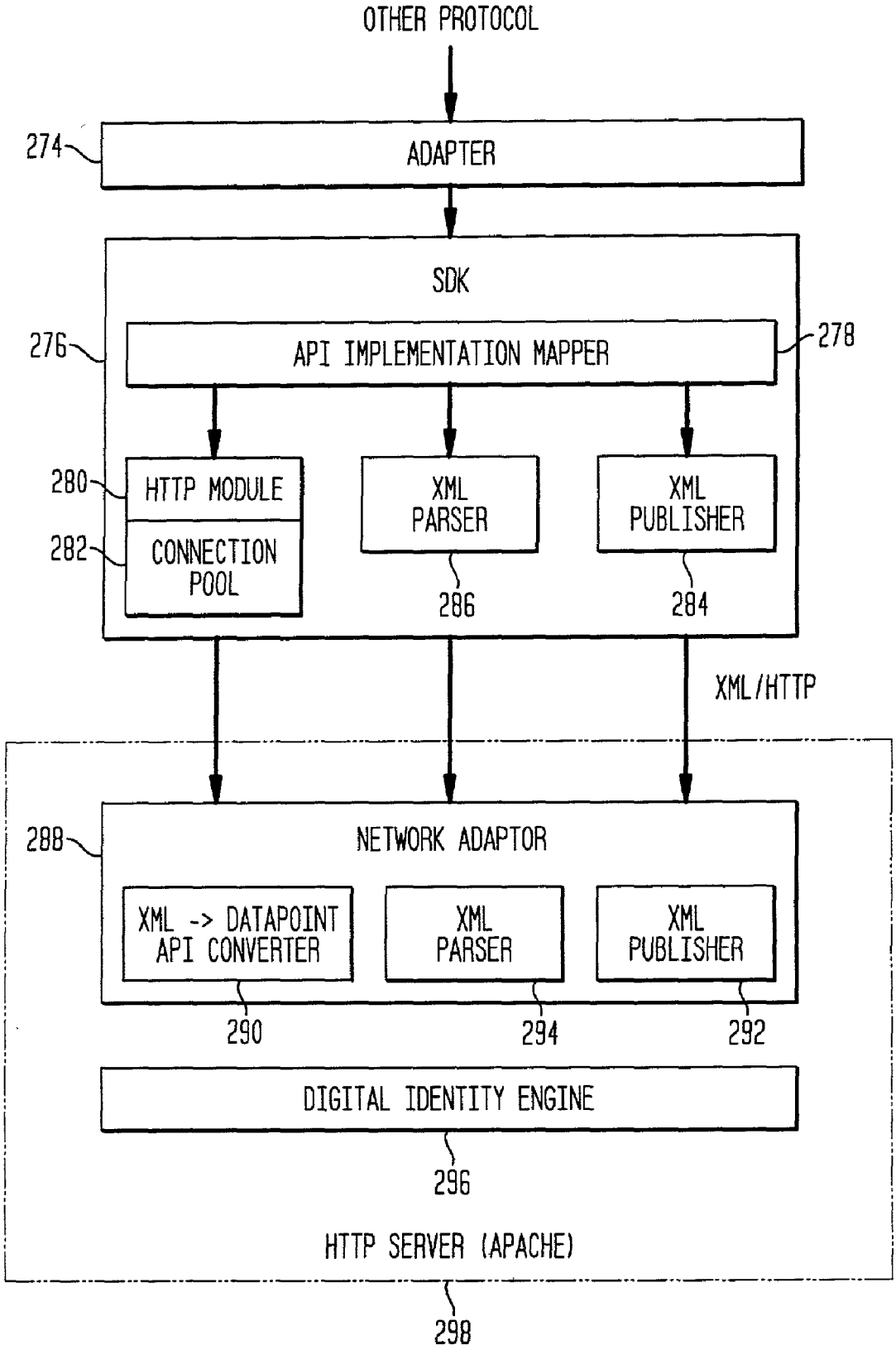

OTHER PROTOCOL
274
ADAPTER
SDK
276
API IMPLEMENTATION MAPPER
278
280
HTTP MODULE
282
CONNECTION POOL
XML PARSER
286
XML PUBLISHER
284
XML/HTTP
288
NETWORK ADAPTOR
XML -> DATAPOINT API CONVERTER
XML PARSER
XML PUBLISHER
290
294
292
DIGITAL IDENTITY ENGINE
296
HTTP SERVER (APACHE)
298

FIG. 3

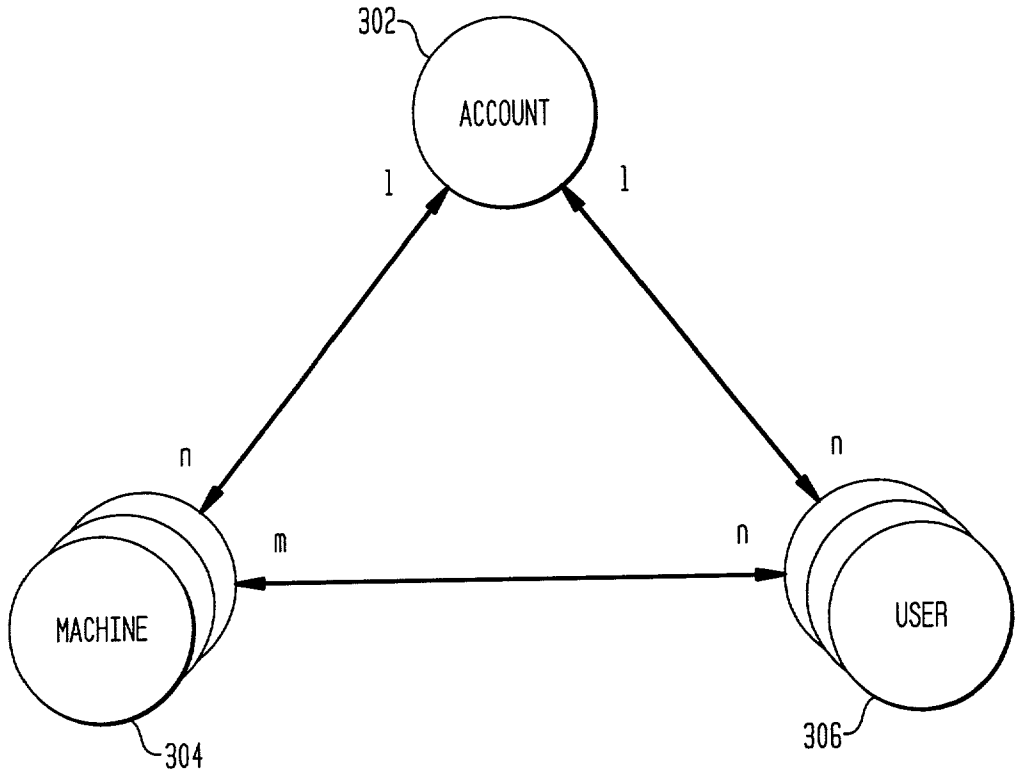

302
ACCOUNT
1
1
n
n
m
n
MACHINE
USER
304
306

FIG. 4

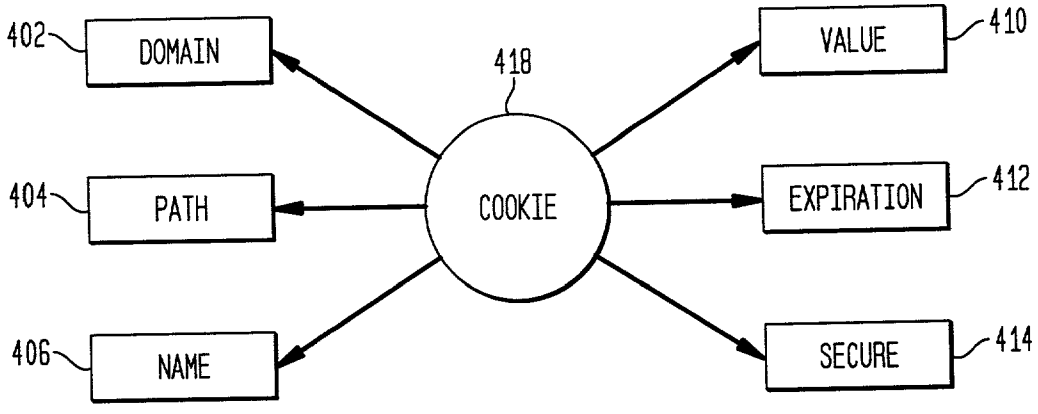

402
DOMAIN
418
VALUE
410
404
PATH
COOKIE
EXPIRATION
412
406
NAME
SECURE
414

FIG. 5

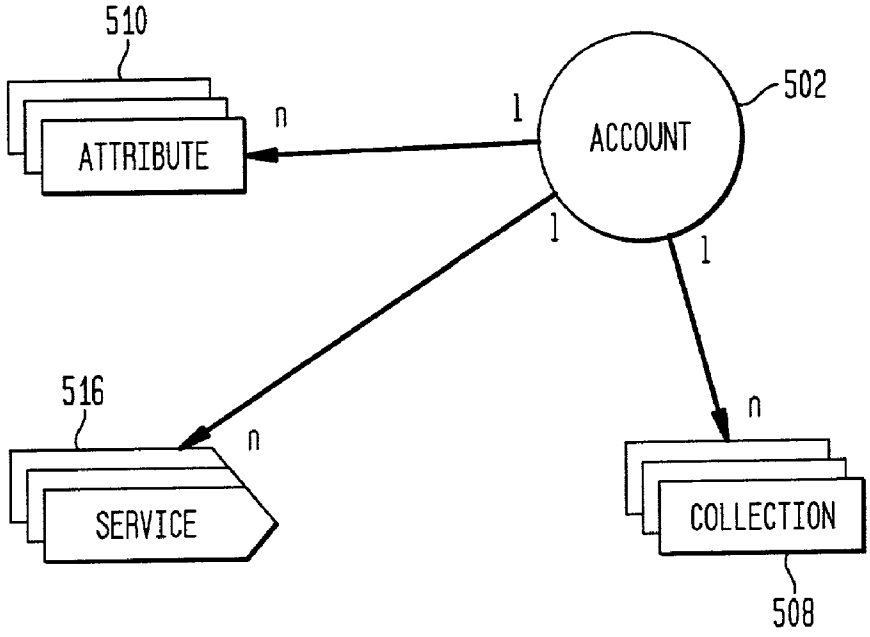

510
ATTRIBUTE
n
1
ACCOUNT
502
1
1
516
n
SERVICE
n
COLLECTION
508

FIG. 6

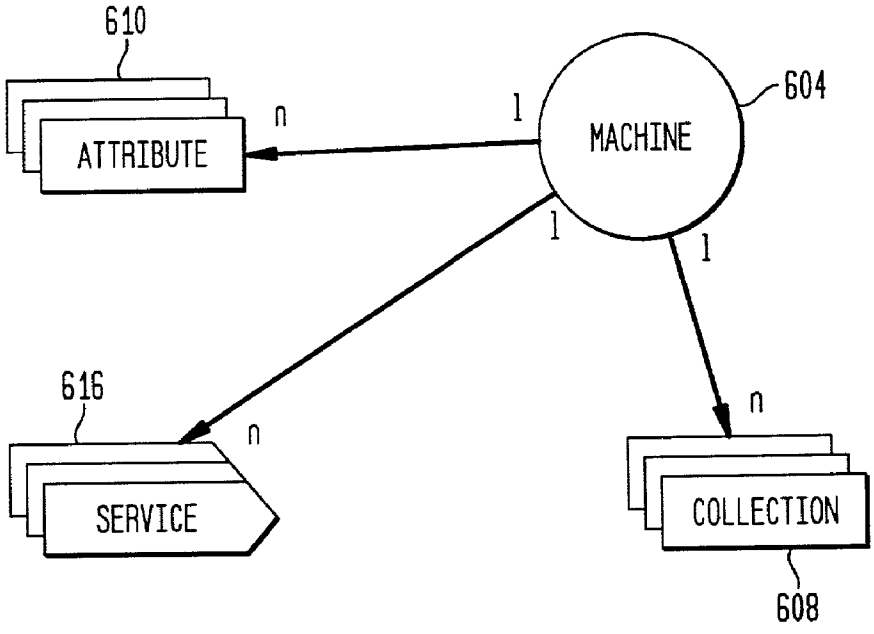

610
ATTRIBUTE
n
1
MACHINE
604
1
1
616
n
SERVICE
n
COLLECTION
608

FIG. 7

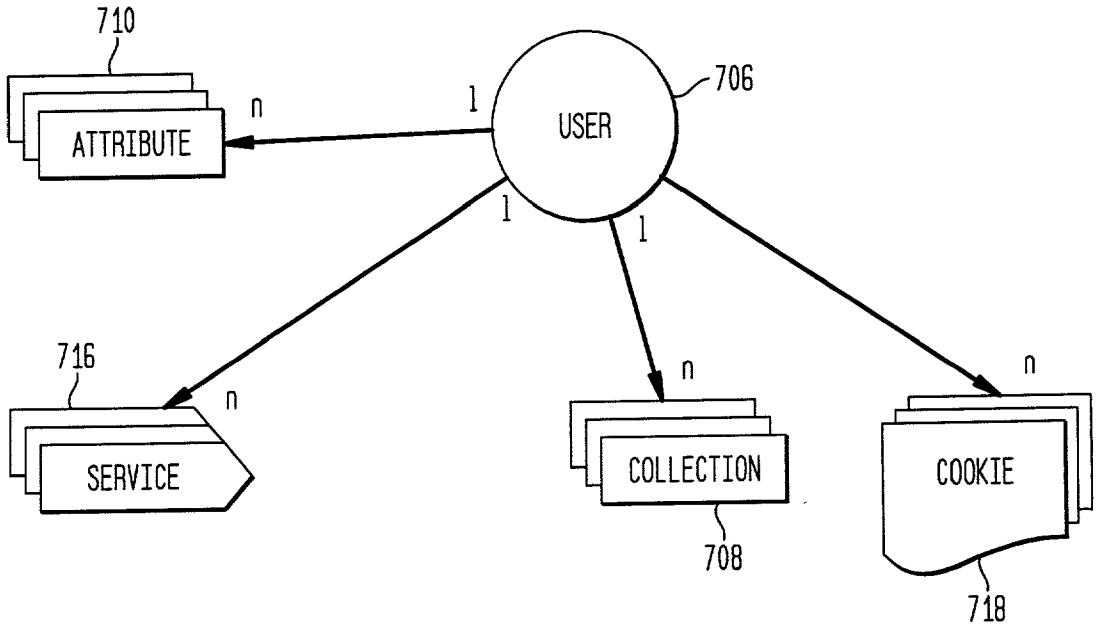

710
ATTRIBUTE
n
1
USER
706
1
1
716
n
SERVICE
n
COLLECTION
708
n
COOKIE
718

FIG. 8

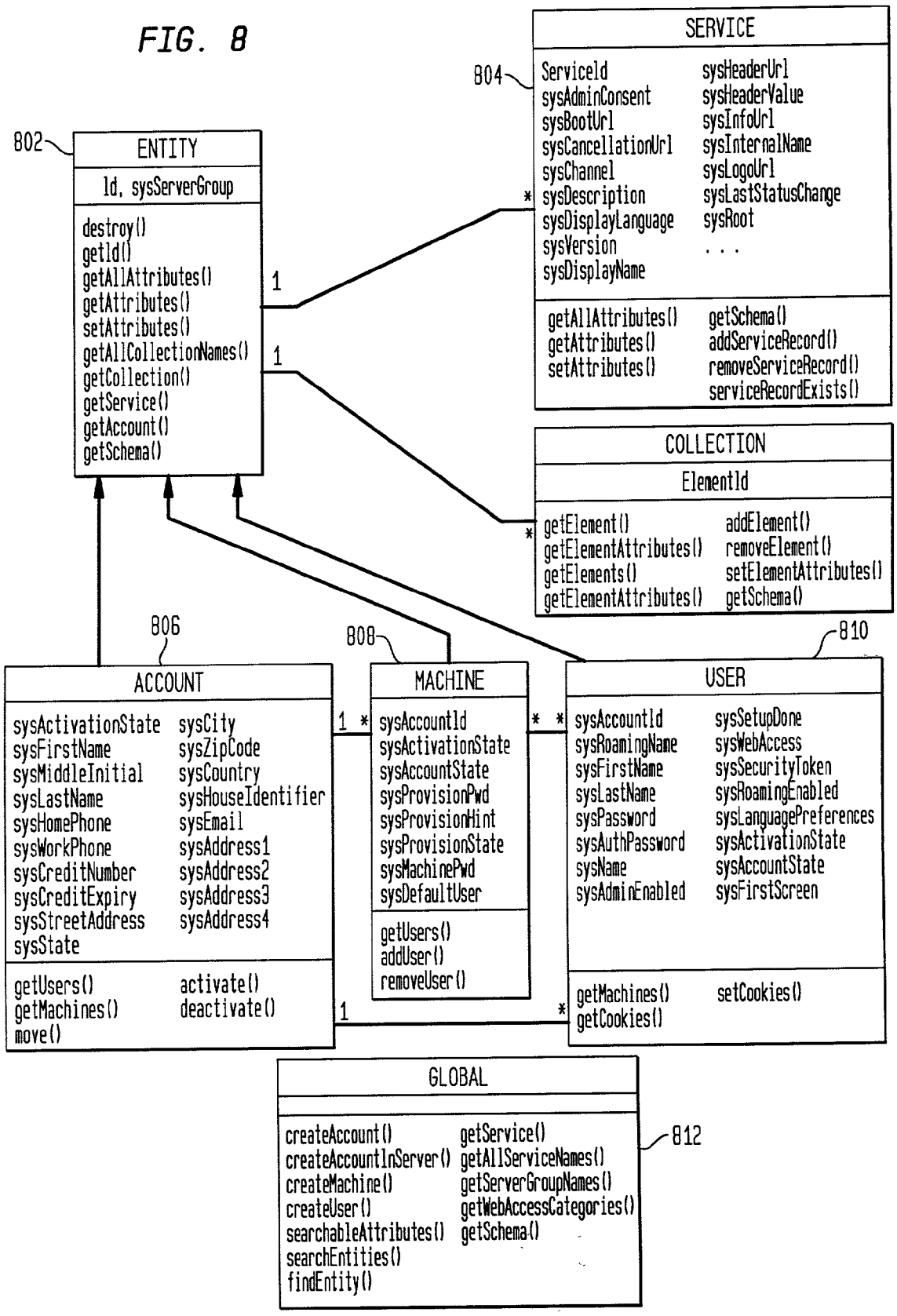
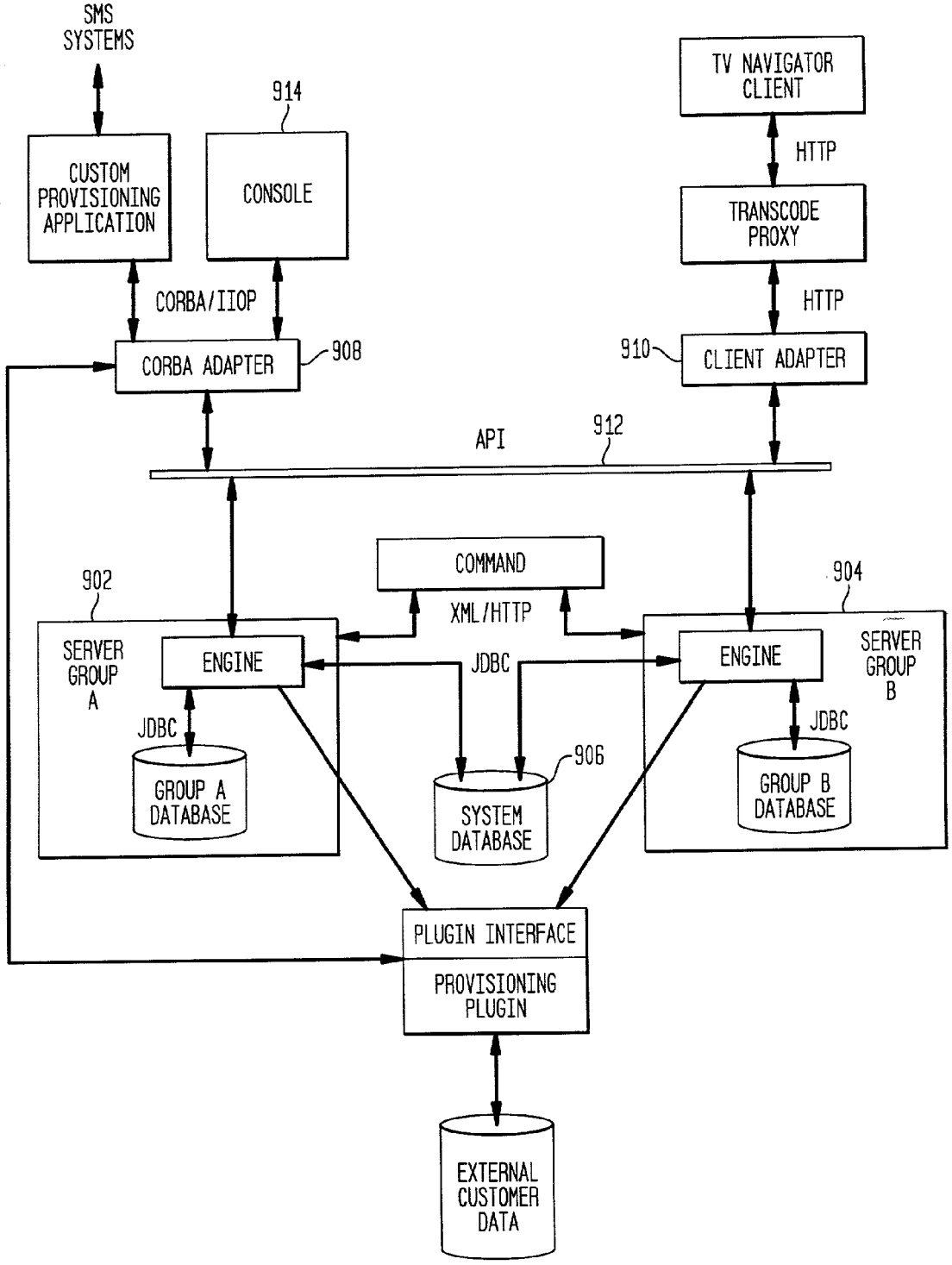

SERVICE
804
ServiceId
sysAdminConsent
sysBootUrl
sysCancellationUrl
sysChannel
sysDescription
sysDisplayLanguage
sysVersion
sysDisplayName
sysHeaderUrl
sysHeaderValue
sysInfoUrl
sysInternalName
sysLogoUrl
sysLastStatusChange
sysRoot
. . .
getAllAttributes()
getAttributes()
setAttributes()
getSchema()
addServiceRecord()
removeServiceRecord()
serviceRecordExists()
802
ENTITY
Id, sysServerGroup
destroy()
getId()
getAllAttributes()
getAttributes()
setAttributes()
getAllCollectionNames()
getCollection()
getService()
getAccount()
getSchema()
1
*
1
*
COLLECTION
ElementId
getElement()
getElementAttributes()
getElements()
getElementAttributes()
addElement()
removeElement()
setElementAttributes()
getSchema()
806
ACCOUNT
sysActivationState
sysFirstName
sysMiddleInitial
sysLastName
sysHomePhone
sysWorkPhone
sysCreditNumber
sysCreditExpiry
sysStreetAddress
sysState
sysCity
sysZipCode
sysCountry
sysHouseIdentifier
sysEmail
sysAddress1
sysAddress2
sysAddress3
sysAddress4
getUsers()
getMachines()
move()
activate()
deactivate()
808
MACHINE
sysAccountId
sysActivationState
sysAccountState
sysProvisionPwd
sysProvisionHint
sysProvisionState
sysMachinePwd
sysDefaultUser
getUsers()
addUser()
removeUser()
810
USER
sysAccountId
sysRoamingName
sysFirstName
sysLastName
sysPassword
sysAuthPassword
sysName
sysAdminEnabled
sysSetupDone
sysWebAccess
sysSecurityToken
sysRoamingEnabled
sysLanguagePreferences
sysActivationState
sysAccountState
sysFirstScreen
getMachines()
getCookies()
setCookies()
1
*
*
*
1
*
GLOBAL
createAccount()
createAccountInServer()
createMachine()
createUser()
searchableAttributes()
searchEntities()
findEntity()
getService()
getAllServiceNames()
getServerGroupNames()
getWebAccessCategories()
getSchema()
812

FIG. 9


SMS SYSTEMS
914
TV NAVIGATOR CLIENT
CUSTOM PROVISIONING APPLICATION
CONSOLE
HTTP
TRANSCODE PROXY
CORBA/IIOP
HTTP
CORBA ADAPTER
908
910
CLIENT ADAPTER
API
912
COMMAND
902
904
XML/HTTP
SERVER GROUP A
ENGINE
JDBC
ENGINE
SERVER GROUP B
JDBC
JDBC
GROUP A DATABASE
906
SYSTEM DATABASE
GROUP B DATABASE
PLUGIN INTERFACE
PROVISIONING PLUGIN
EXTERNAL CUSTOMER DATA

SYSTEM METHOD AND APPARATUS FOR PORTABLE DIGITAL IDENTITY

FIELD OF THE INVENTION

[0001] The present invention relates to the field of portable electronic devices and networked electronic communication.

BACKGROUND OF THE INVENTION

[0002] In using any kind of networked communication device, each user has a user profile that defines a personalized electronic environment for that particular communication device.

[0003] For example, while surfing the Internet on a personal computer, each Web site supplies individual browsers with unique cookies that store identifying information in the user's Internet browser. When visiting specific web sites, the cookies stored in the browser identify each individual to the web site. Based on the information stored in the browser's cookies, each Web site personalizes its contents according to the individual Internet browser. An individual may select bookmarks or favorite Web sites, which are stored on the individual's personal computer persistent memory (hard drive) and which further customize the electronic environment to suit the user.

[0004] As another example of a personalized identity in a networked electronic environment, in a cable television (CATV) system, each subscriber or household subscribes to different programming. Some subscribers have basic cable, while other subscribers have basic cable plus some premium channels. Each member of the household generally has different favorite channels. Additionally, some cable households have broadband Internet access via CATV using a cable modem. In such case, the CATV settop, in communication with a computer at the CATV headend, acts as an Internet browser. However CATV settop boxes tend to have limited computing capabilities, and in particular, CATV settop boxes tend to have limited persistent memory in which to store personalized information such as cookies.

[0005] As another example of a personalized identity in a networked electronic environment, a wireless cellular telephone system provides different subscribers with different calling plans specifically selected by the subscriber. Some cellular telephone networks also offer Internet connectivity to their customers. However, as in the case of CATV settop boxes, cellular phones tend to have limited capabilities as Internet browsers, and in particular, tend to have limited persistent memory storage for personalized information such as cookies.

[0006] When away from the home environment, the electronic environment is different from that at home. For example, using a settop box in a hotel room means that the subscriber typically does not have access to familiar television programs or customized Internet interface. Borrowing or renting a cellular telephone means that the subscriber may not have access to his regular calling plan, is unable to receive incoming calls using his regular phone number, and does not have the customized Internet environment as he would at home. Generally, when a traveler is surfing the Internet at a remote location, whether it be a computer, a remote CATV settop or a cellular phone away from home, Web sites that normally have customized content suited to the traveler, will not recognize the individual Internet browser while at such remote communications device.

SUMMARY OF THE INVENTION

[0007] The present invention is embodied in a digital identity server operating as a node on a distributed computing network such as the Internet.

[0008] In accordance with the present invention, a user enters unique identifying information into an electronic communication device. For example, an email address is unique to each person and entering an email address uniquely identifies the individual to the electronic communication device. Furthermore, entering a password in addition to an email address authenticates the user.

[0009] In order to personalize the electronic environment, the electronic communication device forwards the user's unique identifying information to the digital identity server via the Internet. The nature and capabilities of the given remote electronic communication device is also forwarded to the digital identity server. The digital identity server responds by transmitting a digital identity corresponding to the user to the given electronic communication device. The received digital identity defines the services for which the user is authorized (e.g. in the case of a CATV subscriber, the received digital identity includes a list of the premium channels that the subscriber is authorized to view). Furthermore, the digital identity server provides a level of functionality suited to the capabilities of the particular electronic communication device. The electronic communication device receives the digital identity of the user and personalizes its operation to suit the user.

[0010] The electronic communication device that retrieves the digital identity of the user and personalizes its operation to suit the user may be a CATV settop box, a cellular telephone, a computer, a video game console, an Internet access terminal, a payphone, a vending machine or any present or future electronic communication device. In such manner, the digital identity of the user is made portable, and follows the user wherever the user may go providing the user with the same electronic environment accessible from any electronic communication device.

[0011] In addition, the portable digital identity system of the present invention facilitates the use of simplified electronic communication devices. In particular, electronic communication devices accessing the user's portable digital identity may be implemented using very thin client software and/or without persistent data storage, making the electronic communication device smaller, lighter and less expensive.

[0012] The portable digital identity includes email preferences, email address book and email client software settings, thereby presenting to the user a seamless and consistent email environment. The portable digital identity includes TV preferences, permitting the user to watch his favorite shows and have his premium channels be available from any hotel room. The portable digital identity includes preferred credit card and shipping information (work and home addresses), as well as preferred carriers and other preferences to facilitate eCommerce applications.

[0013] The portable digital identity of the present invention permits the user to access familiar services across different platforms. For example, the present invention permits cross platform authorization (e.g. TV to PC) so that the user may access HBO on a PC. That is, an HBO subscriber may not only access HBO from any remote CATV settop, but may also access HBO on a PC (to the extent that HBO is available via a broadband Internet connection).

[0014] The portable digital identity of the present invention includes demographic profiles and marketing data on user's activities and preferences across devices and in real time. In such manner, for example, an online newspaper viewed at a remote location, contains articles of interest to the subscriber and presents demographically targeted advertising tailored to the interests of the subscriber.

[0015] The portable identity of the present invention may be used to control TV appliances. For example, if a user normally records a given television show on a weekly basis, that knowledge can be used to construct a profile of the user that is then mapped to the user's portable digital identity and stored. The profile may then be used by other applications to target application content at the user the suits his or her interests.

BRIEF DESCRIPTION OF THE DRAWINGS

[0016] **FIG. 1A** is a block diagram of a system including a portable digital identity server and a plurality of information appliances forming a global computer network in accordance with the present invention.

[0017] **FIG. 1B** is a block diagram of a system embodying a portable digital identity server in accordance with the present invention.

[0018] **FIG. 2A** is a block diagram further detailing the system architecture of a system embodying a portable digital identity server in accordance with the present invention.

[0019] **FIG. 2B** is a block diagram further detailing the implementation of the data access function of a system embodied in a portable digital identity server in accordance with the present invention.

[0020] **FIG. 2C** is a block diagram further detailing the implementation of an other network adapter for use with a portable digital identity server in accordance with the present invention.

[0021] **FIG. 3** is a block diagram of three top-level software objects: account, machine and user for use in conjunction with the present invention.

[0022] **FIG. 4** is a block diagram of the attributes of an Internet browser cookie for use in conjunction with the present invention.

[0023] **FIG. 5** is a block diagram of an account data object for storing an account's properties and billing information in accordance with the present invention.

[0024] **FIG. 6** is a block diagram of a machine data object to representing in interactive device for use in accordance with the present invention.

[0025] **FIG. 7** is a block diagram illustrating a user data object representing a user of an interactive device or application in accordance with the present invention.

[0026] **FIG. 8** is a hierarchical representation of the digital identity object model embodying the present invention.

[0027] **FIG. 9** is a system block diagram illustrating the scalability of the present invented system using server groups.

DETAILED DESCRIPTION

[0028] As shown in **FIG. 1A**, the digital identity server **100** communicates over a global computer network **114** (such as would include the Internet as part of the overall global network) with a plurality of user devices. Examples of user devices include a CATV settop **102**, a satellite receiver **104**, a cellular phone **106**, a personal digital assistant (PDA) **108**, personal computer **110**, video game console **112** or other client device **113**. Each user device connects to the Internet though another communication network. For example, a CATV settop box **102** is coupled through a CATV system, while a personal computer **110** is typically coupled through the public switched telephone network. The digital identity server **100** also communicates via the Internet **114** with a digital identity database **118**, an external database **116** as well as a command server **120**.

[0029] In operation, a user identifies himself to a user device **102, 104, 106, 108, 110, 112**. The user device **102, 104, 106, 108, 110, 112** requests a digital identity from the digital identity server **100**. In response, the digital identity server **100** communicates with the command server **120** to determine the nature and characteristics of the requesting user device **102, 104, 106, 108, 110, 112**. The digital identity server **100** then retrieves the digital identity information from either the system database **118** or an external database **116** and downloads it to the requesting user device **102, 104, 106, 108, 110, 112**. After the initial download of a digital identity to user device, the digital identity server **100** need not be involved except to download changes to update a user's digital identity.

[0030] In such manner, the digital identity server **100** mediates the user's access to applications/services and data from a variety of user devices ranging from a digital set-top box **102** to a portable personal digital assistant **108** to a mobile/cellular phone **106** to a game console **112** to a personal computer **110**.

[0031] Applications include Video-on-demand (VOD), gaming applications such as multi-player games, email, instant messaging, chat and broadcast based enhanced TV applications. In video-on demand for example, a viewer may pause a movie at a given point. The pause point of the movie becomes part of the viewer's digital identity. When the viewer returns to watch the rest of the movie, the viewer's downloaded digital identity contains the pause point. As a result, the viewer is able to continue watch the remainder of the movie at a later time from any CATV settop **102** communicating with the digital identity server **100**.

[0032] Additional applications include using an e-wallet (where the e-wallet for a user is tied to his digital identity and stored in the system) and delivering targeted advertising applications (where the profile of the user describing his interests and past behavior is tied to his digital identity and stored in the system).

[0033] The range of data contained in the portable digital identity includes the user's properties such as his preferences regarding the use of the device in question, his favorites data including the list of favorite applications and favorite Internet sites. The data includes the user's cookies which facilitate access to internet sites, and the set of applications/services that the user may access including the properties of the user for a specific application/service.

[0034] The digital identity server **100** retrieves configuration information from the command server **120** about various types or classes of devices within the system, and applies configuration information as a filter when returning digital identity data back to the user device. The set of applications and the generic user properties as well as application specific user properties are tailored take into

account the processing power, network bandwidth and memory footprint capabilities of the communications device currently in use by the user. Thus, when the user is on a powerful communications device such as a personal computer **110**, the list of applications available to such user includes the full set of allowed or subscribed applications. When the user is on a less powerful device such as a set-top box **102** or a personal digital assistant **108**, the list of applications available to a user will typically include only a lesser permissible subset of applications.

[**0035**] The digital identity of the user remains the same regardless of the device that the user is using at any point in time. Having a consistent digital identity retrievable at any point by Internet access, allows the user to access his applications/services and data in a seamless and transparent fashion. Thus, even while "roaming" i.e. moving between multiple devices in his home, or to a device at a remote location such as a digital set-top box or game console at a friend's home, or using a cellular phone/pager in his car, the user experiences a consistent electronic environment.

[**0036**] Associated with the notion of a portable digital identity is the notion of a General Services Architecture. The General Services Architecture defines and describes the model that allows the use of applications/services and associated data by the user from their various devices. In particular, the General Services Architecture includes a user account that defines the applications and services to which the user subscribes.

[**0037**] The digital identity server **100** and a General Services Architecture allows the service provider/operator to dynamically define and add new services/applications into their server-side infrastructure. Services are available dynamically to users based on a configurable policy that can be customized to suit the business needs of the specific network operator or service provider.

[**0038**] Furthermore, access to the service can be controlled at a very granular level all the way down to a specific device and a specific user. User A may subscribe to the video on demand (VOD) service, but User B may not be allowed access to the service or even be allowed to subscribe to the service at all. If the user is a subscriber to a specific service he may not be able to access the VOD service unless he is on a device that is actually capable of running that service as is determined by the digital identity server dynamically.

[**0039**] The digital identity of the user as implemented by the digital identity server **100** includes a rich object oriented programming model that provides high reliability and high availability and scales to millions of users. The digital identity server **100** has easy extensibility to new client devices **113** and new server platforms and also provides for easy integration with existing stores **116** of user information being maintained by service providers and operators elsewhere on the Internet.

[**0040**] The overall digital identity system design is a four-tier architecture of clients **10**, **10**A, **12**, **12**A, adapters **18**, **14**, engine **22** with application programming interfaces **20** (APIs) and database **24** shown in **FIG. 1B**. The digital identity server **100** provides a mechanism to provide connectors to different devices **10**, **12** (where client software resides) that can be hooked into the internal core digital identity engine **22**. Such connectors are referred to herein as adapters **18**, **14**.

[**0041**] A digital identity software development kit (SDK) **16** permits other clients **12** to write specialized adapters **14**. The specialized adapter **14** is a protocol translator written by an other client **12** using the digital identity SDK **16** that uses standardized XML protocol to communicate with a standard digital identity adapter to the digital identity engine **22**.

[**0042**] Client software may reside in devices other than user devices **10**, **12**. In particular, a CATV system **11**B can be a CATV client. In such case, digital identity software development kit (SDK) **16**A permits a specialized adapter **14**A to be written that uses standardized XML protocol to communicate with a standard digital identity adapter to the digital identity engine **22**. As another example, a web site can be a web site client **11**A communicating with the digital identity engine **22** via a standard adapters **18**.

[**0043**] The digital identity engine **22** is the component that handles all access to all data that adapters **10**, **10**A, **12**, **12**A (and thus clients) is stored on the server side. The core engine **22** and its digital identity APIs **20** is written in Java to take advantage of Java Database Connectivity (JDBC) as the primary mechanism for accessing the digital identity data.

[**0044**] Application programming interfaces (APIs) are available as part of the TV Navigator platform (including client-side JavaScript and Java APIs) and as part of the Connect Suite platform (the server-side Java based, XML based and CORBA based APIs) that allow applications to be authored on top of the digital identity platform. CORBA, an acronym for Common Object Request Broker Architecture, is a type of object-oriented programming language system.

[**0045**] The digital identity server (**100** in **FIG. 1**) further provides yet another interface that can be implemented by third parties in order to write specialized plug-ins (**223** in **FIG. 2A**) to the digital identity server **100**. Specialized plug-ins are used to access (in a transparent manner) information residing in external systems (**234** in **FIG. 2A**) and including the legacy billing and SMS systems of the CATV operator (or other service provider).

[**0046**] The portable digital identity server **18**, **20**, **22** of **FIG. 1B** is shown in further detail in **FIG. 2A** (**210**). The digital identity Engine **230** provides an application programming interface (API) **228** to client adapter writers. The digital identity API is implemented as an efficient means for adapters to name, store, and control access to user data. The design relies on a relational database to provide the storage and indexing of user data.

[**0047**] The digital identity engine **230** is what implements the API that adapters **222**, **224**, **226** use to perform operations on data. Adapters are software components that communicate with clients. Various adapters are developed for the various clients that digital identity server **210** supports. For example, standard adapters include a CORBA adapter, a digital television and cookie adapter **224** and an XML adapter **226**. An other adapter **212** is created using the software development kit **214**.

[**0048**] Various client software **202**, **203**, **204**, **206** and **208** communicates with the digital identity server **210** via a corresponding adapter. For example, provisioning application **202** and a digital identity control console **203** interface through a CORBA adapter **222**. A first generation digital television client **204** (using a proprietary protocol) interfaces through a digital television and cookie adapter **224**. A next generation digital television client communicates with the digital identity server **210** through an XML adapter **226** (using a standard version of Extensible Markup language or XML) as would an other adapter **212**. CORBA Clients use

their own protocol, notably CORBA IIOP in COBRA adapter **222**, rather than XML/HTTP in adapter **226**.

[**0049**] Similar to the operation of **FIG. 1A**, the command server **238** in **FIG. 2A** provides data on the nature and characteristics of the requesting user device. The digital identity server **210** then retrieves the digital identity information from either the system database **236** or an external database **234** and downloads it to the requesting user device.

[**0050**] As shown in **FIG. 2B**, the digital identity engine **250** includes API implementation functions **252** responsive to the application programming interface **251**. The digital identity engine **250** further comprises a data access layer **254** responsive to the API functions **252** to perform all of the mechanisms for abstracting or accessing data out of the backend (data storage).

[**0051**] The API implementation **252** communicates only with the data access layer **254** and not directly with the various back-end data access functions such as Lightweight Directory Access protocol (LDAP) **258**, **268**, Java Database Connectivity (JDBC) **264**, schema mapper **266**, callout mechanism **260** and system data cache **262**. (Liberate Technologies, 2 Circle Star Way, San Carlos, Calif. 94070). The design is quite general, in the sense that adding a new mechanism for accessing data would require no changes to the API implementation functions **252** or the adapters (**18** in **FIG. 1B**).

[**0052**] A data access layer **254** in the digital identity engine **230** provides the following functionality:

[**0053**] 1. Pools connections **253** to all data sources, including Group databases **272**B, System database **272**A, and Lightweight Directory Access protocol (LDAP) server(s) **268**.

[**0054**] 2. Dynamically updates and manage the relational database schema

[**0055**] 3. Use the schema mapper **256** and system data cache **262** to implement its operations and abstract from the API implementation **252** how data is accessed and where it is stored, hiding the fact that the data is distributed.

[**0056**] 4. Provide the implicit mapping from the schema of the objects defined in XML to the database schema

[**0057**] The Supported Objects are given below. These objects are the same as the objects defined in the XML Protocol as used in the digital identity server.

[**0058**] /account (primary object)

[**0059**] /user (primary object)

[**0060**] /machine (primary object)

[**0061**] /account/users (relationship—not extensible)

[**0062**] /account/machines (relationship—not extensible)

[**0063**] /machine/users (relationship—not extensible)

[**0064**] /user/cookies (cookies—not extensible)

[**0065**] /user/services (services)

[**0066**] /machine/services (services)

[**0067**] /account/services (services)

[**0068**] /user/favorites (collection object)

[**0069**] /user/addressbook (collection object)

[**0070**] . . . (etc)

[**0071**] When a new type of collection object is introduced (such as Address Book), no new API functions are needed. Only a new object, and its associated XML schema are needed. The data access layer **254** maintains objects and their mappings to physical tables automatically, so that no code has to change at the data access layer **254** when a new object is created. The same is true for new attributes of existing objects.

[**0072**] API Implementation

[**0073**] The API implementation **252** calls only the various parts of the data access layer **254** to perform the functions it needs, it does not call any other pieces, nor does it access any database (**268**, **270**, **272**A, **272**B) directly. The data access **254** layer maps each specific digital identity API **251** call into the (more general) data access call. For example, a CreateEntity API function calls the generic "create" or "set" method in the data access layer **254**, after setting up all the right parameters, and the connection. Similarly, a GetCookies or GetProperties API function, calls a generic "get" method, after setting up all the parameters for each type of object (see object list above) to get the data from the database.

[**0074**] Schema Mapper **256**

[**0075**] The Schema Mapper component **256** maintains the configuration of the XML schema objects, and their underlying physical tables. It also provides the data access layer **254** with a way to easily determine how to access a given piece of data. Furthermore, the schema mapper **256** stores XML schema blobs in the Command Server **266**, allowing the customer to extend and control the schema dynamically (through GUI tools). Finally, the schema mapper **256** stores information about where attributes reside (Database, external Lightweight Directory Access protocol (LDAP), etc).

[**0076**] System Data Cache **262**

[**0077**] The system data cache **262** minimizes the need to access the System database **272**A, since it is a global bottleneck. It further stores servergroup information for users/machines/accounts in a data cache **262** so that trips to the System database **272**A are eliminated whenever possible. The system data cache **262** further provides a way for the API implementation functions **252** to efficiently discover the user/machine/account relationships. Finally, the system data cache **262** ensures that the in-memory cache is kept consistent with the database, given that there may be multiple digital identity servers **250** behind a load balancer, and the servers need to appear stateless. The function of providing consistent state conditions across multiple digital identity servers is accomplished by allowing communication between multiple digital identity servers for notification purposes when an entity is deleted or moved.

[**0078**] Scalability

[**0079**] Each digital identity server **250** has the ability to connect to any datasource in the site, including all Group databases **272**B, the System database **272**A, and all external customer data (Lightweight Directory Access protocol (LDAP) **268**, SMS **270**, etc). However, each digital identity server **250** has the notion of a home server group, namely a group database **272**B to which it is "tightly" bound, either through physical locality, or through logical locality (i.e. it expects to service a certain subset of users/machines/accounts in the normal case). The digital identity server **250** optimizes its access to its own home server group **272**B as much as possible.

[0080] The digital identity server **250** provides less optimized access to other server groups' data for administration functions (such as MoveEntity) and to external customer data. The digital identity server **250** has the option to provide access to all functions on other server groups, if the adapters/Clients do not wish to connect to another digital identity server which is non-optimized. One digital identity server is able to service several adapters simultaneously. Load Balancing (when possible) is done between the clients and the adapters. In the alternative, load balancing may be done between the adapters and the digital identity Engine through the network adapter

[0081] Call-outs to Legacy Data

[0082] The call-out component **260** retrieves data on a read-only basis from external (operator-managed) datastores **270**. The customer data retrieval typically occurs as part of an operation like getProperties in the API. Use of the call-out **260** is dictated by the Schema Mapper **256**, which notes where and how individual properties can be retrieved. The call-outs are used only for primitive properties of Entities, but may be generalized to apply to other data (like Services or Collections) as well. Data structures are discussed in conjunction with **FIG. 3** through **FIG. 7**.

[0083] As a part of any call-out, the entityid must be converted into an ID that is meaningful for the external datastore. The conversion may involve a call into the system database **272**A or user database, to retrieve other properties. The conversion activity is performed either in the data access layer, or within the specific modules that perform the call-out.

[0084] Similarly, propertyNames needs to be converted into external attribute names; this information is generally available through the command server **266**.

[0085] When the call-out returns, the returned data is merged into the result set that is returned from digital identity (typically a sequence of PropertyNameValues), and is indistinguishable from other data.

EXTERNAL DATA—LEGACY DATASTORES

[0086] There are two call-outs shown in **FIG. 2**B: SMS (subscriber management system) **260** uses a general function call; Lightweight Directory Access protocol (LDAP) **258** is a special case for which higher-level support is provided. Both of these call outs are examples of the external data i.e. legacy datstores (**234** in **FIG. 2A**).

[0087] SMS Call-out

[0088] The SMS callout mechanism **260** uses a function call to a customer-provided routine. Typically, the SMS module is called with an entityid and one or more propertyNames. The SMS callout resolves the Id (convert to an external id), and then makes a function call to the external routine. In a Java implementation, this routine is typically provided as a .jar file, loaded as a plugin. The argument list for external function includes at least external entity ID and propertyName(s).

[0089] Lightweight Directory Access Protocol (LDAP) Call-out

[0090] The Lightweight Directory Access protocol (LDAP) call-out provides high-level support for retrieving data from a Lightweight Directory Access protocol (LDAP) repository.

[0091] The data access layer calls the Lightweight Directory Access protocol (LDAP) module, supplying information such as the entityid and propertyName(s). The Lightweight Directory Access protocol (LDAP) call-out converts the entityId into an Lightweight Directory Access protocol (LDAP) Distinguished Name (possibly using information from the System DB or User DB), and converts the propertyNames into Lightweight Directory Access protocol (LDAP) attribute names (possibly using configuration parameters).

[0092] Using configuration parameters, it then forms and executes a complete Lightweight Directory Access protocol (LDAP) call to retrieve the data, such as an Lightweight Directory Access protocol (LDAP) URL, and processes the result set.

[0093] In a Java implementation, this Lightweight Directory Access protocol (LDAP) client can be implemented on Java Naming Directory Interface (JNDI). Most call-outs use the DirContext.getAttributes() method, to retrieve a set of Lightweight Directory Access protocol (LDAP) attribute values, which are merged into the digital identity result set.

[0094] Besides being easier to use than the more general call-out mechanism, the Lightweight Directory Access protocol (LDAP) module enables the data access layer **254** to pool **253** Lightweight Directory Access protocol (LDAP) connections, as it also does for Java Database Connectivity (JDBC) connections.

[0095] Digital Identity Software Development Kit

[0096] The diagram in **FIG. 2C** shows the sub-components of the Software Development Kit (SDK) and network adapter **288**. The Software Development Kit (SDK) **276** and network adapters **288** both convert between the digital identity API and the network protocol (XML/HTTP).

[0097] The SDKs implement the digital identity API, hiding implementation details behind a standard interface. SDKs are delivered as libraries, which are used by customers who build out-of-process (external) adapters.

[0098] SDKs must be written in the same language as the corresponding adapters, so separate SDKs are required for each language in which adapters are written. SDKs are required for both Java and C/C++. Java adapters may be able to run natively (in-process), if their client-server protocol allows it. In-process adapters do not require SDKs.

[0099] The primary function of the SDK is to convert digital identity API calls into network-based communication with the digital identity server. A simplified process description is:

[0100] 1. Call XML Publisher **284** to convert API command and data into XML.

[0101] 2. Call HTTP module **280** to establish communication with server through connection pool **282**; send XML request; receive response.

[0102] 3. Call XML Parser **286** to parse response; convert to API data structures; return to caller.

[0103] Digital Identity Network Adapter **288**

[0104] The network adapter **288** runs in the same process as the digital identity server, and services out-of-process adapters. The job of the network adapter **288** is the complement of the SDK **276**; it converts XML/HTTP requests back into digital identity to API function calls. In this sense, the digital identity server is using its native network protocol as

a sort of RPC mechanism for the external adapters to make calls to the digital identity engine. The extensible markup language (XML) is actually very well suited for this purpose.

[0105] Similar conversions to/from XML (shown as XML Publisher **284** and XML Parser **286**) are performed in both the SDK and network adapter. These conversions share the same technology base, especially for XML parsing **286**.

[0106] Adapters

[0107] Each adapter's **274** primary function is to translate the communication that it receives from its client in its native protocol to the Liberate digital identity API. Clients make requests to adapters to perform certain operations such as getting and setting of data. These requests are decoded and handled by the adapter, and translated into digital identity API calls; data returned from the API calls is encoded and sent to the client. As mentioned, adapters may run in-process or out-of-process; the API is identical in either case.

[0108] Compared to in-process adapters, external ones offer advantages (independence of programming language; stability of external process) and disadvantages (potential performance penalty of extra network hop). In general, in-process adapters should be used where possible, for performance reasons.

[0109] Adapters that rely on HTTP use the same mechanism for decoding and handling the network traffic as the digital identity network adapter (namely an HTTP server such as Apache) **298**. A goal of the digital identity server is to use a single adapter to service all XML requests, be they from in-process or out-of-process adapters. To facilitate unification, a compatible XML format is adopted.

[0110] CORBA and LDAP adapters

[0111] The digital identity server does not provide native interfaces for CORBA or Lightweight Directory Access protocol (LDAP). Clients that use these protocols require special-purpose adapters. The adapters implement the appropriate Server type (CORBA or LDAP), but use digital identity protocols on the back end.

[0112] CORBA

[0113] The CORBA adapter is needed to support the User Data Manager, which is a CORBA Client (currently implemented as a Java applet). Like all CORBA servers, it supports an interface defined in an IDL. IDL has currently been defined for the digital identity engine consisting of about 40 operations, defined on 7 interfaces (object classes). This supports a particular object model, which has since been extended for digital identity. In order to make the new digital identity features accessible through CORBA, the IDL is extended. The adapter translates IDL calls into digital identity API calls.

[0114] The CORBA adapter may run either in-process or out-of-process with respect to the digital identity Engine. An in-process implementation links the digital identity Engine into the CORBA Server as a library, which is different from the method of running other in-process adapters, which are accessed through a web server interface. The out-of-process implementation uses the SDK and network adapter.

[0115] The Digital Identity Object Model

[0116] As shown in **FIG. 3**, there are three top-level object classes: Accounts **302**, Users **306** and Machines **304**. These three top-level object classes are collectively referred to as Entities. Individual Entities have a unique entityid, specified when the Entity is created. Each User **306** is associated with exactly one Account. Each Machine **304** is associated with exactly one Account. As illustrated by the 1 to N relationship an account **302** may have a plurality of users **306**. As illustrated by the 1 to M relationship, an account **302** may have a plurality of machines **304**. For example, a household CATV account **302** may include several family members as users **306**, and have more than one CATV converter **304**.

[0117] All Entities may have primitive Properties. Properties are typed; the current set of types is {string; integer; boolean; binary}. Besides the primitive Properties, Entities may have Collection Properties associated with them. Collections are structured objects—i.e., have primitive Properties of their own. Collections may have many Instances for a given Entity; each Instance has a unique instanceid, which can be used to access that Instance.

[0118] As shown in **FIG. 5**, the account entity is associated with one or more attributes **510**, services **516** and collections **508**. An account entity **502** stores the properties of the account and billing information.

[0119] Similarly, **FIG. 6** illustrates the machine entity **604** being associated with one or more attributes **610**, services **616** and collections **608**. The machine entity **604** represents an interactive device.

[0120] Similarly, the user entity **706** is associated with one or more attributes **710**, services **716** and collections **708**. A user entity **706** represents the user of an interactive device or application. In addition, the user entity **706** is associative with one or more cookies **718**. As shown in **FIG. 4**, cookies have special Collection Properties, with particular semantics, such as name **406**, path **404**, domain **402**, value **410**, expiration **412** and security level **414**.

[0121] Besides being associated with particular Entities, Collection Properties may have values at the global (System) level. Customers may define additional Properties for Accounts, Users, Machines, or Collection Properties, and may define additional Collection Properties. Entities have Server Groups, which specify where their data is located. Users and Machines associated with an Account have the same Server Group as the account.

DIGITAL IDENTITY OBJECT MODEL

[0122] **FIG. 8** is a hierarchical representation of the Digital identity object model. Various types of information are represented in the digital identity server to support the needs of administrators, applications, set-top boxes, and other users. The information is structured according to a particular model (schema), which reflects the world of Accounts, Machines, and Users. Information that is handled by the digital identity server can be accessed and manipulated in a variety of ways, including through CORBA, and from the settop box.

ENTITIES AND RELATIONSHIPS

[0123] The primary objects in the system are Entities **802**, which correspond to Accounts **806**, Machines **808**, and Users **810**. There are three subclasses of Entities to represent the three cases listed above:

[0124] Accounts: Represents a billing account. An Account may have multiple Machine and multiple User entities associated with it.

[0125] Machine: Represents a single set-top box. Each Machine object must always have an associated Account.

[0126] User: Represents a User on a set-top box. Each User object must always have an associated Account.

[0127] Standard UML notation is used to represent both the digital identity object model and the associated CORBA IDL. The boxes in **FIG. 8** represent classes (CORBA interfaces) of which there are 7 in the system. Each box in **FIG. 8** is divided into three regions; the top region shows the class name; the middle region shows the attributes that are defined in the base schema; the bottom region shows the operations (methods) that are defined in the CORBA IDL.

[0128] The lines among the boxes indicate relationships. The ones with arrowheads are generalizations; the others are associations, with the multiplicity indicated by the numbers at either end.

DIGITAL IDENTITY SERVER GROUPS

[0129] To improve scalability, network operators can create digital identity server groups. A deployment can have one or more digital identity server groups **902, 904**, depending on how the network operator configures the system. Each server group **902, 904** has its own configuration settings and its own database. The use of server groups is convenient when managing a large number of subscribers.

[0130] Besides a database for each server group, there is a single, shared digital identity system database **906** that can be accessed by every digital identity server. The system database **906** contains basic information for all subscribers, while each server group database **902, 904** only contains information about the subscribers in that particular group.

DIGITAL IDENTITY ADAPTERS

[0131] Digital identity adapters **908, 910** communicate with the digital identity servers **902, 904** across an API **912** that provides a single point of access to all data in all digital identity servers. This provides the following benefits:

[0132] Different TV Navigator clients (such as TV Navigator Standard and Compact clients) can all access the same digital identity server or server groups.

[0133] Developers can create custom provisioning applications that use the services of the digital identity CORBA adapter. These applications can also interface to external billing, customer service, and subscriber management systems, to interoperate with legacy systems.

[0134] A simple graphical user interface is provided at the digital identity console **914**, which can access and modify persistent data stored in the digital identity servers. The digital identity console **914** is implemented as a CORBA client.

[0135] Through the provisioning plugin architecture, digital identity provides access to external back-end data stores, such as LDAP servers enabling system operators to access legacy data.

[0136] The digital identity architecture supports the development of new client adapters, as needed for emerging protocols.

[0137] Below are several examples of sample code for provisioning an account making requests and providing responses.

```
SAMPLE CODE FOR PROVISIONING AN ACCOUNT
Transaction.open( );
Account johnsAccount = Account.create("LondonServerGroup");
johnsAccount.setAttribute("sysFirstName", "John");
johnsAccount.setAttribute("sysLastName", "Smith");
Machine setTopBox1 = Machine.create(johnsAccount,
        "macAddr1");
Machine setTopBox2 = Machine.create(johnsAccount,
        "macAddr2");
User john = User.create(johnsAccount);
john.setAttribute("sysName", "John");
john.setAttribute("sysPassword", "1234");
setTopBox1.addUser(john);
setTobBox2.addUser(john);
Transaction.commit( );
Example XML Request
<XMSG>
<AUTH CLASS ="USER" TYPE = "password" ID = "Ryan" TOKEN = "a secret">
    <REQUEST OP = "GET">
        <OBJ N = "/user" ID ="Ryan"/>
    </REQUEST>
</AUTH>
</XMSG> Example XML Response
<XMSG>
<RESPONSE OP = "GET" STATUS = "OK">
    <OBJ N = "/user" ID = "Ryan"/>
        <AT N = "sysFirstName">Ryan</AT>
        <AT N = "sysLastName">King</AT>
    </OBJ></RESPONSE>
</XMSG>
```

What is claimed is:

1. In a system having an electronic communication device operated by a user, said system further including a remote database and a server on a distributed computing network, a method comprising:

entering identification information into said electronic communication device;

transmitting said identification information to said server;

retrieving personalized information from said database;

forwarding said personalized information to said electronic communication device,

wherein said electronic communication device is a CATV settop box; and

wherein said personalized information is related to said user's CATV subscription account, whereby said CATV settop box is personalized to said user.

* * * * *