Likewise, if the TCP connection fails 250 to retrieve a frame 235 then an attempt to establish a HTTP connection 225 is made. Alternatively, more than one attempt to establish a TCP connection will be made. Further attempts to establish a connection are made with progressively narrower transports until the transport mode with a narrowest acceptable bandwidth has been reached. For example, a HTTP connection. Attempts to establish this transport mode will be continue until a connection is established, 240/255. At this point it is assumed that the server is down or the network has failed. When the server comes up or the network improves, video will begin streaming once the connection has been established. In an alternative embodiment, the transport mode with the narrowest acceptable bandwidth will be attempted a specified number of times.

Once a successful video connection has been established and an audio format has been selected, an audio connection will be made and display the data will begin 260.

If at any point a connection fails 265, a connection will be attempted with the next bandwidth, for example from UDP to TCP. While in the alternative transport mode, the mode with a lower bandwidth, periodic attempts to establish a connection with the original transport will be made. No user intervention is needed during operation of the method.

Alternatively, if the connection is maintained, the datagram frame counter is incremented 280. A request is then made for the next packet based on the frame counter value from the server. The steaming video data will continue until the client terminates the connection with the server.

If displaying a stream on-demand, the client is given the ability to fast forward, rewind, and jump to arbitrary parts of the stream. A clock is also displayed indicating the current location in the stream. The client initially sends a query to the server requesting the length of the stream in frames. Proceeding with each frame, the server sends the frame number, allowing the client's clock to remain accurate regardless of the speed of the stream. When the user wishes to fast forward, rewind, or jump to a specific location in the stream, it sends a request with the desired target frame number to the server. The server responds by seeking to the requested frame number and streaming from there. All other functions of rate adaption and protocol selection operate as normal.

An advantage to the present invention, related to the client's ability to move on the stream, is error resilience. That is, since each video frame is independent, if an error occurs in a frame the invention can either attempt to fix the error or drop the frame. When a frame is dropped, the invention requests the next frame and preserves the continuity of the video and/or audio display. This is especially advantageous in devices having limited processing power, and therefore, lacking the resources to fix errors.

Having described the role of the client above, the method will described in reference to various types of capture. Other types of capture may also be implemented in the present invention.

The server may capture a thread 305 from a specified source, e.g., a local capture card. Example of a local capture card include the SunVideoPlus Osprey and the SunVideo (sun) capture board. In this instance, the server makes use of native Solaris threads to achieve rate-adaptive connections to the clients. A thread is a placeholder information associated with a single use of a program that can handle multiple concurrent users. From the program's point-of-view, a thread is the information needed to serve one individual user or a particular service request. If multiple users are using the program or concurrent requests from other programs occur, a thread is created and maintained for each of them. The thread allows a program to know which user is being served as the program alternately gets re-entered on behalf of different users. One way thread information is kept is by storing it in a special data area and putting the address of that data area in a register. The OS always saves the contents of the register when the program is interrupted and restores it when it gives the program control again.

On startup, the server creates two threads which capture video 300 and audio 400 from the specified source. The server places the captured data 310 into a shared area of memory 315 which is accessible to all other threads within the server. Other threads are created which accept incoming requests for each transport type. These incoming requests for each client are handled as part of a non-blocking loop. Each transport type is handled differently.

For example, for UDP video 325 connections, the non-blocking loop accepts and services connections. Since UDP is "connectionless," there is no need to maintain a persistent connection to the client and each client connection is really a request for a single frame. The thread receives a single byte datagram from a client 340, immediately retrieving 355 and sending 370 back the frame currently stored in the shared memory segment 315. A datagram is a self-contained, independent entity of data carrying sufficient information to be routed from the source to the destination computer without reliance on earlier exchanges between this source and destination computer and the transporting network. This process continues indefinitely.

Since sending a UDP datagram is a non-blocking operation, the server need only have one thread. Thus, the amount of CPU time and memory required from UDP is drastically less than that needed for other connections.

Another example is TCP connections 330. The server waits for a new connection to be established with a client 345. A non-blocking loop accepts incoming requests. The client corresponding to the requests are arranged in a connection pool list. Since TCP is a connected protocol, an independent thread is then created to service 360 each client. The service threads act like the UDP thread, awaiting 375 a single byte (request) from the client, the requested frame is retrieved 385 by the server from the shared memory 315. The retrieved frame is then sent 390 to the client.

The above two examples are rate adaptive solutions. By allowing the clients to control the flow of video frames rather than simply pushing each frame to each client as it is captured, data bottlenecks typically associated with streaming media over the Internet are eliminated.

As an illustration, two desktop PCs are connected to a server. The first by 100Mbps Ethernet, the second by a 28.8Kbps dialup Internet connection. The 100Mbps machine will receive video at the rate it is captured, up to 30fps. The rate of speed of the video is due in great part to the available bandwidth, the 100Mbps sends out frame requests to the server fast enough to allow for a 30 fps stream.

However, the 28.8Kbps machine requires more time to receive each frame and therefore sends out frame requests less frequently. Thus the rate of speed is about 3 to 4fps, and both machines will be at the same place in the stream. The illustration method of the present invention accomplishes this through client side requests for current frames. These requests may or may not be for the next sequential frame. Comparing the machines, the 100Mbps machine will show a higher quality video, while the 28.8Kbps machine will appear to be skipping frames in order to maintain a real time stream.

This approach has several advantages over conventional streaming methods. Current methods require re-encoding the video at several frame rates in order to support users with different bandwidth availability. The user is also required to select a rate beforehand that is appropriate for their connection. In contrast, the illustrative method according to the present invention needs only one rate of capture and encoding. A user no longer needs to know anything about their network bandwidth. The user will connect to the stream at a rate which adapts itself to the environment. Additionally, degraded network conditions will not create a bottleneck and will not cause the stream to fall behind real time, rather the frame rate will decrease and more frames will be skipped. When conditions improve, so will the frame rate. Further, this approach will take advantage of high-bandwidth consumer access devices, such as xDSL and 38GHz wireless connections without any changes or updating the server of client.

Other transport modes are supported by the invention, for example, HTTP 335. HTTP is a one-way protocol and thus cannot perform true rate adaption. It is included for clients that do not have UDP, TCP, or another transport available to them. This may be due to a network firewall, or a packet filter for example. Further, some Internet Service Providers (ISP) may not support connections on certain ports, or not support UDP at all. HTTP connections are achieved through the use of an external program which is activated by a web server 500, e.g. common gateway interface (CGI). A main server creates one thread to accept and service connections from this external program via local UNIX domain sockets and UDP 505. Although UDP is used here, it is only for local redirection of frames from the main server to the CGI. The server waits for a CGI request 350. The CGI requests frames at a steady rate from the main server 510. As the main server retrieves frames 365 from the shared memory 315, the frames are re-broadcast to the client 380 back through the web server 520 via HTTP. To avoid bottlenecks, from the audio push 525, this transport mode transmits 380/515 at a default rate of 1fps in order to remain real time regardless of available bandwidth.

Another capture can be from a local file which is looped. This method obtains video and audio data from a single file that has been pre-encoded from a capture board and stored on the server. It operates exactly as above and uses the exact same thread structure. Data is read in from the specified file and placed into the shared memory segments at the rate it was encoded. When the end of the file is reached, it re-starts from the beginning.

Still another capture can be had from a local file on-demand. This method deals with more than one source of audio/video data being sent to all clients. Each client needs its own unique copy of the server which then acts as described above using the requested file. This is accomplished by creating another layer of threads which create entire server sets of threads for each client. While this creates a high amount of overhead, the data is being delivered in an compressed and encoded state, therefore it is not necessary to make deliveries in real time. In addition, the server returns the number of frames in the stream to the client as well as the frame number of each frame sent, therefore, the client's clock remains accurate. The server also receive requests from the client indicating that it wishes to jump forward of backward to a specific frame, the server seeks to this location in the file and continues streaming.

Yet another capture is from a remote IP address, according to this method the server receives data from another server on a remote machine. The remote machine may be located anywhere on the network that is accessible from the local machine's network. The local server acts as a single client to the remote server, and pulls a stream from the remote server in the same rate-adaptive fashion as the Java client. The local server then places the data into its shared memory buffers, and re-broadcasts it as if it were being captured locally. All other functions of the local server operate as normal.

Because the local server is acting as a regular client, it does not matter how the remote server is captures data, it can be from any of the methods described above. It can even be capturing its data from another remote server, allowing for server chains to be created that, for example, re-broadcast a source feed on different networks.

Like video 300, audio data 400 may be transported by the methods described above: UDP 440, TCP 445, and HTTP 450. However, it is pushed data. As new audio data is captured 410, it is immediately sent out to all clients 430. This allows for a steady audio stream which will inherently be in sync with the video played in real time according to the invention. The server also attempts to adapt the streamed audio to the clients available bandwidth. This is accomplished by the server making available multiple types of audio simultaneously, for example, raw uncompressed 410, Global System for Mobile communication (GSM) encoded audio 415, and G.728 encoded audio 420 (G728 is specified in ITU-T recommendation G.728, "Coding of speech at 16Kbit/s using low-delay code excited linear prediction"). Uncompressed audio 410 occupies 64k of bandwidth, G.728 420 occupies 16Kbps of bandwidth, and GSM 415 occupies 13Kbps of bandwidth. The client times the amount of time between two frames 265, for example, the second and third frames, and based on this time selects the appropriate available audio format 270. The server creates a thread 435 to accept incoming audio connections via each of the available transport modes: UDP 440, TCP 445, and HTTP 450. The server waits for a new connection 455 a, b, c to be established by a client. Upon connection, the client sends preferred audio format information to the server 460 a, b, c. The server then creates a service thread 275/465 a, b, c, for the client which delivers audio in the requested format. These service threads wait for a signal from the corresponding audio encoding thread 470a, b, c, that more audio has become available. The server retrieves audio data from a selected buffer 475a, b, c, then the selected buffer is sent to the client 285/480a, b, c, and immediately wait for another signal 470a, b, c. HTTP connections will automatically default to the lowest bandwidth signal, since a degraded connection is assumed.

Advantageously, the system and method according to an illustrative embodiment of the invention functions in low bit rate environments, at about 9.6 Kbps, and suffers no degradation even during a transmission over a 14.4 Kbps network. Theoretically there is no upper bound as the system utilizes an adaptive bandwidth method. That is, the system scales the data stream to the available bandwidth so that as bandwidth increases the stream rate will increase accordingly. The flow of data from the server to the client is intelligently managed so as to completely eliminate network bottlenecks traditionally associated with streaming media. This is especially beneficial in wireless environments where available bandwidth may be limited or inconsistent. The end user is allowed to receive multimedia data without fore knowledge of the available bandwidth while maintaining a real-time stream. This also frees the content provider from having to provide multiple media streams to accommodate differing connection speeds. In order to control the drain of the server's network capacity, the system and method provides for a cap. The cap is utilized by the server to restrict usage of the stream to maintain the integrity of the server's network.

Having described preferred embodiments of a system and method for multimedia streaming, it is noted that modifications and variations can be made by persons skilled in the art in light of the above teachings. It is therefore to be understood that changes may be made in the particular embodiments of the invention disclosed which are within the scope and spirit of the invention as outlined by the appended claims. Having thus described the invention with the details and particularity required by the patent laws, what is claimed and desired protected by Letters Patent is set forth in the appended claims.

WHAT IS CLAIMED IS:

1. A method for streaming video data over a network in real time, comprising:

initializing a transport mode for the video data;

sending from a client to a server a data request for a single frame of video data;

retrieving the single frame from a memory at the server; and

sending the video data to the client.

2. The method for streaming video in Claim 1, wherein the steps of sending a data request for a single frame from the client to a server, retrieving the video data from a shared memory, and sending the retrieved video data to the client, are repeated for each video frame.

3. The method for streaming video as in claim 2, wherein said each video frame is processed for display independently from processing of another video frame.

4. The method for streaming video in Claim 1, wherein the step of initializing further comprises:

determining a list of available transport modes for the client;

determine incompatibilities between the available transport modes and software;

choosing a transport mode from the list; and

initializing parameters of the transport mode at the client for client control of video streaming.

5. The method for streaming video in Claim 4, wherein the step of initializing is performed by a client application which is capable of running in different operating systems.

6. The method for streaming video in Claim 4, wherein the step of initializing is performed by a client embedded in a web page.

7. The method for streaming video in Claim 6, wherein the client embedded in the web page is chosen from one of a common gateway interface and an active server page.

8. The method for streaming video in Claim 1, wherein the transport mode is chosen from the group consisting of a UDP, a TCP, and a HTTP.

9. The method for streaming video in Claim 1, wherein storing video further comprises:

capturing a thread from a specified source; and

storing the captured thread in the server's shared area of memory.

10. A storage medium having a stored program which is executable by a processor for causing the processor to perform method steps for streaming video communication, the method steps comprising:

requesting a packet representing a single datagram from a server over a communication network;

receiving a requested packet;

processing and displaying said requested packet;

incrementing a datagram frame counter;

requesting a next packet based on the frame counter value from the server; and

asynchronously processing and displaying said next packet when received.

11. The method of claim 10, wherein communications between said processor and said server is by Wireless Application Protocol (WAP).

12. The method of claim 10, wherein said server is accessed by said processor via HTTP.

13. The method of claim 10, wherein said packet representing a datagram is JPEG encoded.

14. The method of claim 10, wherein said step of asynchronously processing and displaying is independent of data from said step of processing and displaying said requested packet.

15. An apparatus for communicating streaming video data between a plurality of users and a server connected by a communication network, comprising:

a stored program executable by a processor in said server for causing the server to: receive requests for individual datagrams from the plurality of users; and

forwarding individual datagrams in response to each request to the user making the request at a rate based on available bandwidth of the user making the request.

16. The apparatus according to claim 15, wherein said plurality of users are wireless mobile devices.

17. The apparatus according to claim 15, wherein said server is accessible via HTTP.

18. The apparatus according to claim 15, wherein the datagrams are JPEG encoded.

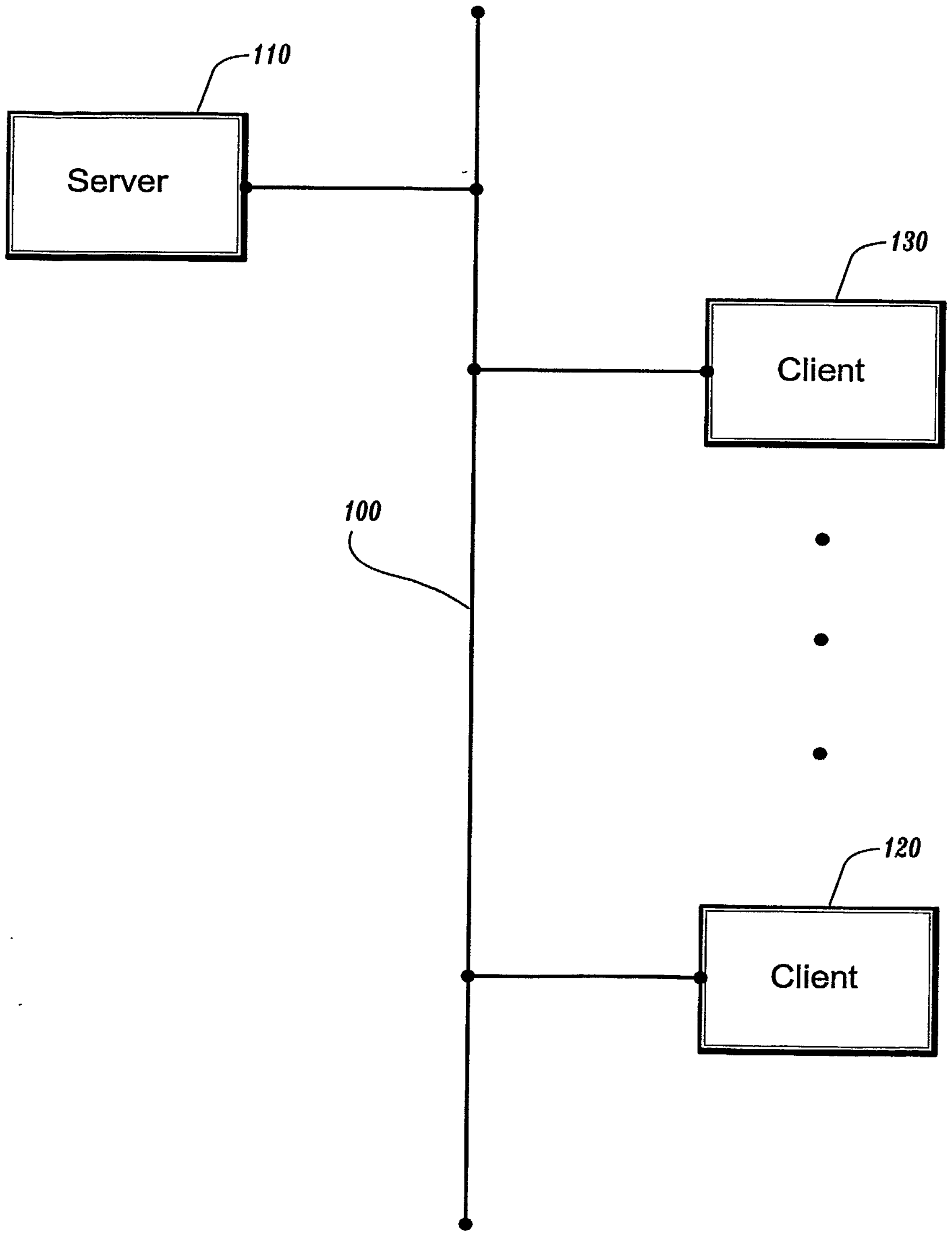


FIG. 1

Java Audio/Video Client

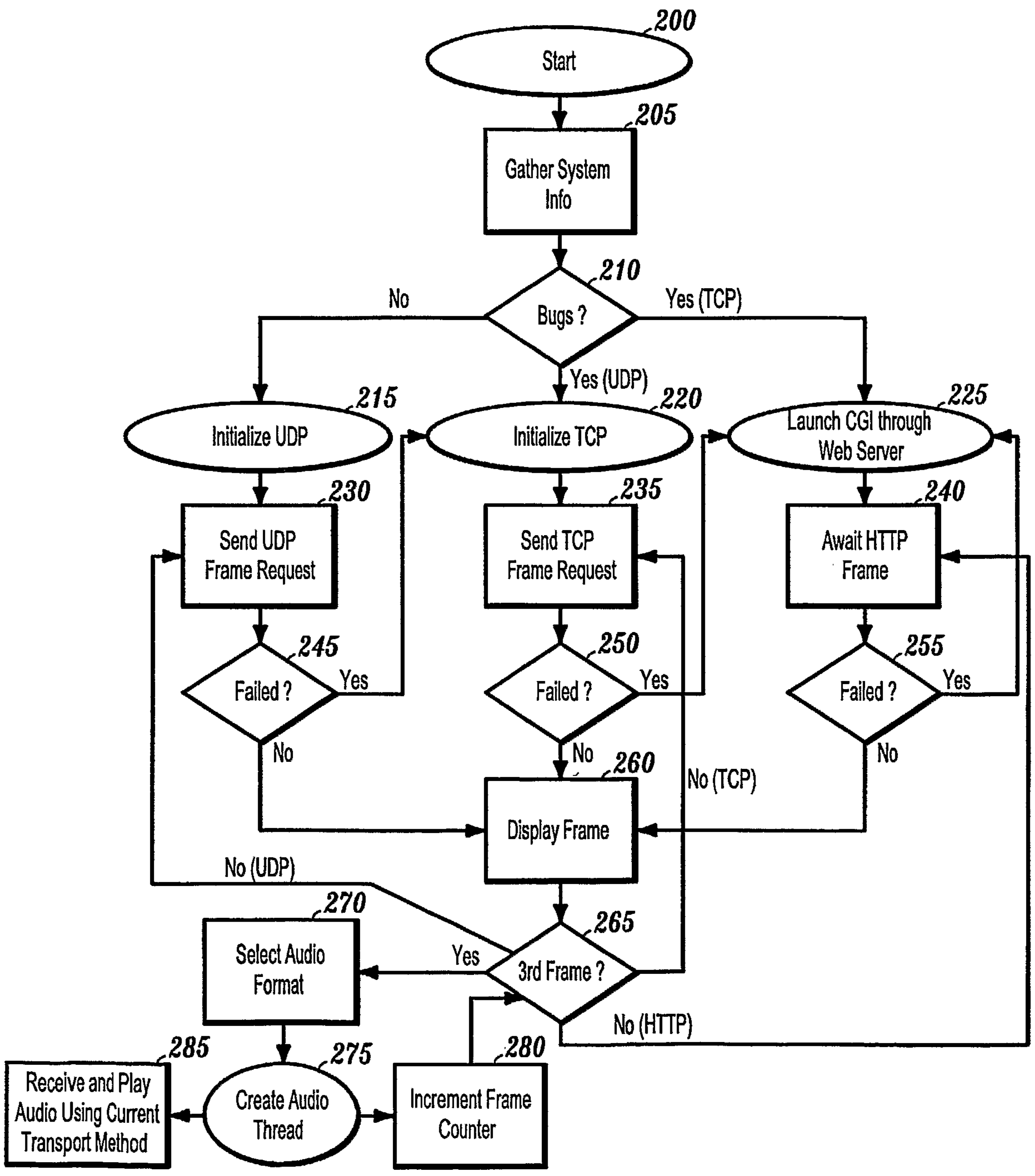


FIG. 2

SUBSTITUTE SHEET (RULE 26)

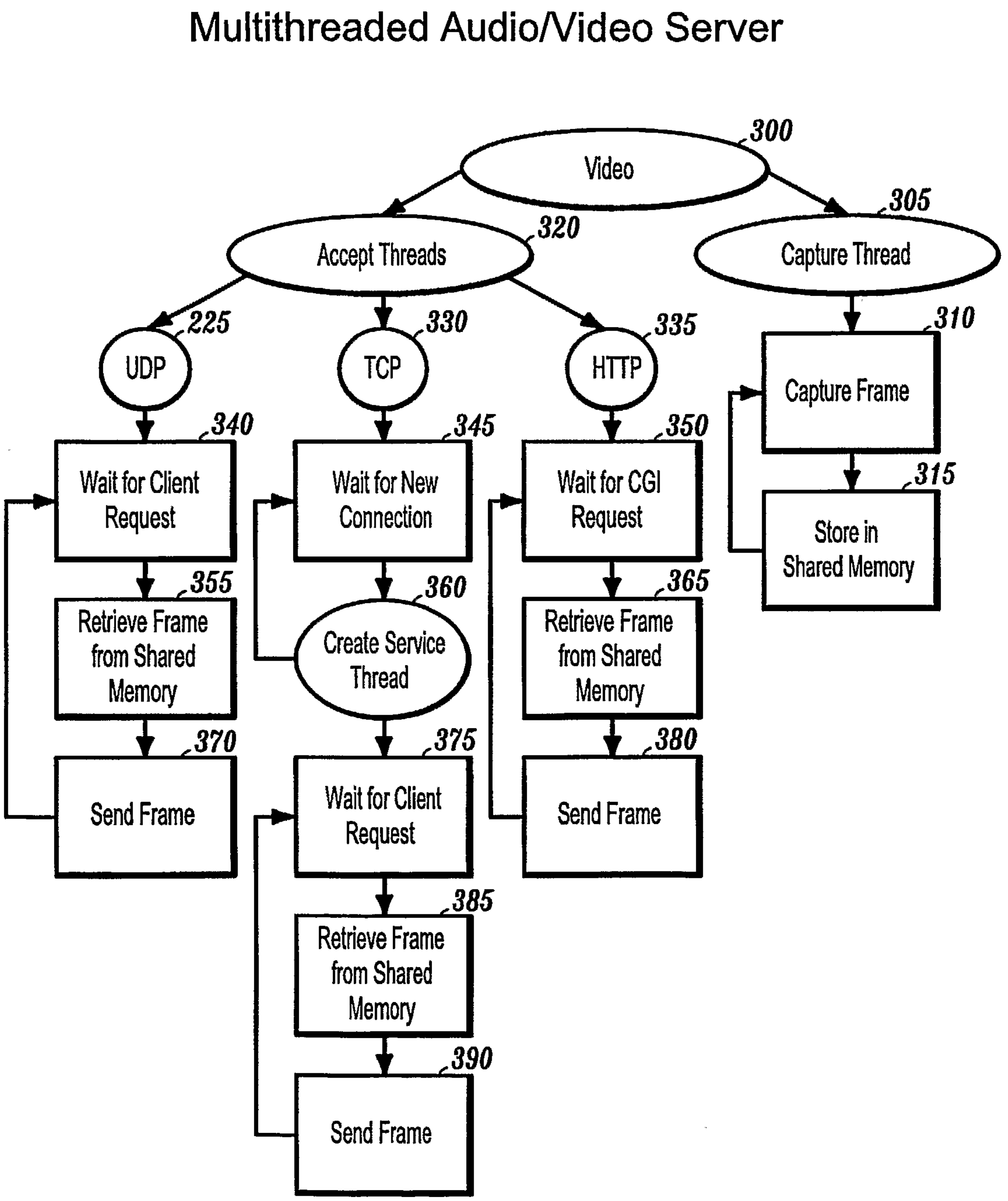

Multithreaded Audio/Video Server
Video
300
Accept Threads
320
Capture Thread
305
UDP
225
TCP
330
HTTP
335
Capture Frame
310
Store in Shared Memory
315
Wait for Client Request
340
Wait for New Connection
345
Wait for CGI Request
350
Retrieve Frame from Shared Memory
355
Create Service Thread
360
Retrieve Frame from Shared Memory
365
Send Frame
370
Wait for Client Request
375
Send Frame
380
Retrieve Frame from Shared Memory
385
Send Frame
390

FIG. 3

Multithreaded Audio/Video Server

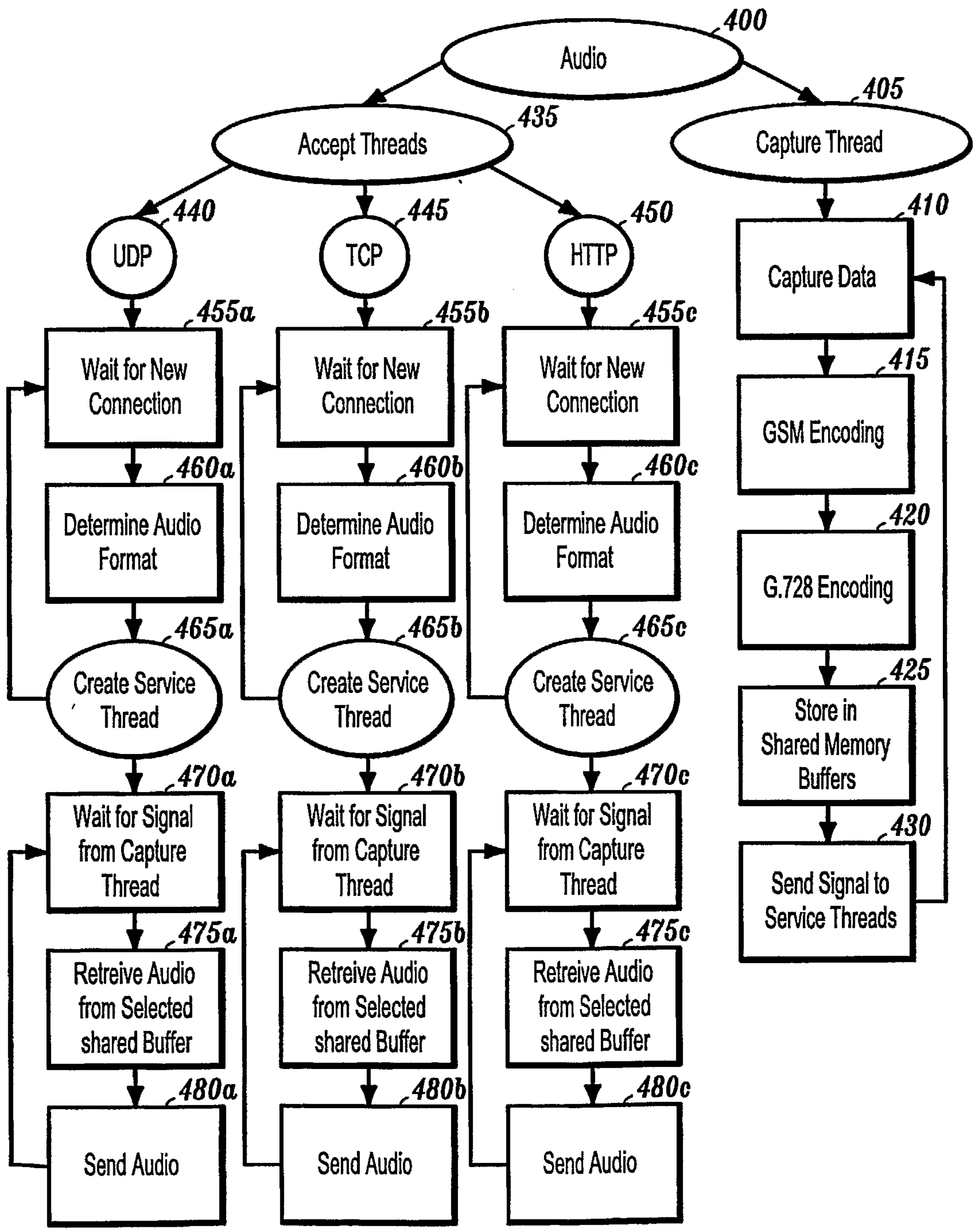


FIG. 4

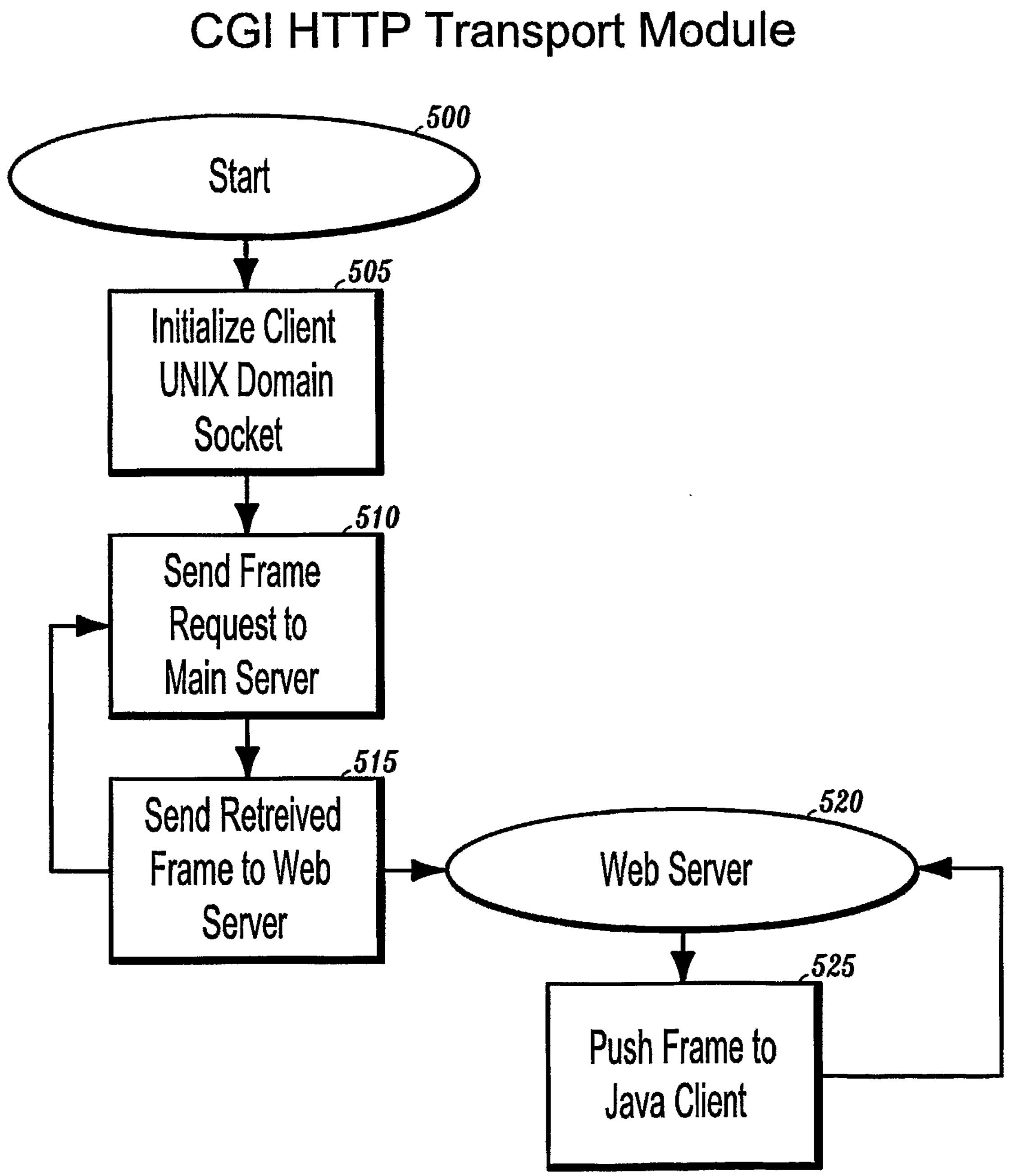

CGI HTTP Transport Module
500
Start
505
Initialize Client UNIX Domain Socket
510
Send Frame Request to Main Server
515
Send Retreived Frame to Web Server
520
Web Server
525
Push Frame to Java Client

FIG. 5

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 01/40452

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H04N7/24

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 H04N H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, WPI Data, PAJ, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 97 22201 A (XIE DONG ;CAMPBELL ROY H (US); CHEN ZHIGANG (US); TAN SEE MONG (US) 19 June 1997 (1997-06-19) abstract page 19, line 10 - line 22 page 21, line 8 - line 10 page 24, line 6 - line 10 page 24, line 16 -page 25, line 5 page 26, line 6 - line 16 ---	1-3,8, 15-18
X	US 5 929 850 A (HAASS JON C ET AL) 27 July 1999 (1999-07-27) abstract column 3, line 24 - line 43 column 6, line 9 - line 17 column 12, line 47 -column 13, line 30; figure 10 --- -/--	1-3

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
2 October 2001	16/10/2001

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	Beaudet, J-P

Form PCT/ISA/210 (second sheet) (July 1992)

International Application No
PCT/US 01/40452

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
A	US 5 999 979 A (RAVI HEMANTH SRINIVAS ET AL) 7 December 1999 (1999-12-07) the whole document ---	1-18
A	ERLANDSON C ET AL: "WAP - THE WIRELESS APPLICATION PROTOCOL" ERICSSON REVIEW, ERICSSON. STOCKHOLM, SE, no. 4, 1998, pages 150-153, XP000792053 ISSN: 0014-0171 the whole document ---	1-18
A	KALVA H ET AL: "IMPLEMENTING MULTIPLEXING, STREAMING, AND SERVER INTERACTION FOR MPEG-4" IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, IEEE INC. NEW YORK, US, vol. 9, no. 8, December 1999 (1999-12), pages 1299-1312, XP000873453 ISSN: 1051-8215 paragraphs ‘0002!,‘0005! -----	1-18

INTERNATIONAL SEARCH REPORT
Information on patent family members

International Application No
PCT/US 01/40452

Patent document cited in search report		Publication date	Patent family member(s)			Publication date
WO 9722201	A	19-06-1997	EP	0867003	A2	30-09-1998
			JP	2000515692	T	21-11-2000
			WO	9722201	A2	19-06-1997
US 5929850	A	27-07-1999	AU	3648197	A	21-01-1998
			EP	0909512	A1	21-04-1999
			JP	2001508602	T	26-06-2001
			WO	9800975	A1	08-01-1998
US 5999979	A	07-12-1999	EP	0956686	A1	17-11-1999
			JP	2000509592	T	25-07-2000
			WO	9834385	A1	06-08-1998
			EP	0956702	A1	17-11-1999
			WO	9834405	A1	06-08-1998
			US	6014706	A	11-01-2000
			US	6230172	B1	08-05-2001

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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
25 July 2002 (25.07.2002)

PCT

(10) International Publication Number
WO 02/057943 A1

(51) International Patent Classification⁷: G06F 15/16

(21) International Application Number: PCT/US02/01840

(22) International Filing Date: 18 January 2002 (18.01.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/263,058 18 January 2001 (18.01.2001) US

(71) Applicant *(for all designated States except US)*: **YAHOO! INC.** [US/US]; 701 First Avenue, Sunnyvale, CA 94089 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **JUSTIN, Paul, Madison** [US/US]; 1504 Concord Drive, Richardson, TX 75081 (US). **ANTHONY, Joseph, Rodiger** [US/US]; 5004 Lake Vista Drive, The Colony, TX 75056 (US). **SYED, Noman, Kazmi** [PA/US]; 12480 Abrams Road, #2601, Dallas, TX 75243 (US). **WILLIAM, David, Turner** [US/US]; 6326 Bryan Parkway, Dalls, TX 75214 (US). **STEVEN, Alexander, Roberts** [US/US]; 5406 Kingston Drive, Richardson, TX 75082-4067 (US). **GHOUSUD-DIN, Syed** [US/US]; 504 Sheffield Drive, Richardson, TX 75081 (US). **VINAY, Polavarapu** [IN/US]; 6913 Valleyview Lane- Apt. #230, Irvin, TX 75039 (US). **BHARGAVARAM, Bulleyya, Gade** [IN/US]; 1213 Meadow Creek Drive, Apt. #J, Irving, TX 75038 (US).

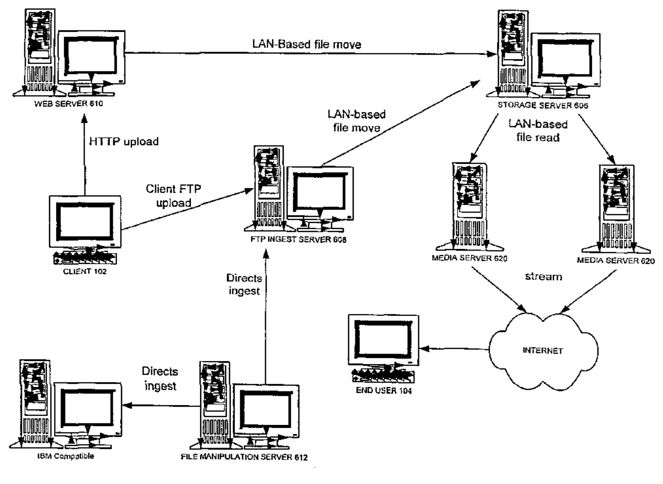
(74) Agents: **DECARLO, James, J.** et al.; Stroock & Stroock & Lavan LLP, 180 Maiden Lane, New York, NY 10038 (US).

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

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent

[Continued on next page]

(54) Title: METHOD AND SYSTEM FOR MENEGING DIGITAL CONTENT, INCLUDING STREAMING MEDIA



(57) **Abstract:** Method and system for uploading, managing and delivering digital content, including streaming media. The system according to one embodiment allows receives digital content from the client (102), assigns a stream identifier (ID) to the content and stores the content. The client is given a playlist uniform resource locator (URL) for publishing on its web site (610), the URL including the stream ID. Activation of the URL by an end user (104) causes the stream to be served to the end user, without the client receiving or providing an indication of the specifics of where the content was stored. An embodiment of the present invention provides a system and method that permit clients to actively manage their content, including defining logical folders and subfolders containing item(s) of the content and defining logical stream groups, containing items of the content.

WO 02/057943 A1

(BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

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

METHOD AND SYSTEM FOR MANAGING DIGITAL CONTENT, INCLUDING STREAMING MEDIA

CROSS REFERENCE TO RELATED APPLICATIONS

[001] This application is a continuation-in-part of, and claims the benefit of, United States Provisional Application No. 60/263,058, filed on January 18, 2001, for METHOD AND SYSTEM FOR MANAGING STREAMING MEDIA, hereby incorporated by reference.

BACKGROUND OF INVENTION

1. Field of the Invention

[002] The present invention relates generally to a method and system for providing digital content via the Internet and, more particularly, to a method and system for allowing clients to up-load, manage, and deliver streaming media content via the Internet.

2. Description of Related Art

[003] With the advent of the Internet and the World Wide Web, an industry has developed around the delivery of digital content, such as streaming media content. By way of example, streaming media may be used for any of a number of purposes, including entertainment, distance learning and corporate purposes. Entertainment companies stream movies and sporting events, distance learning companies stream educational content, and corporations stream training materials.

[004] Although some streaming media content providers may have relatively few items of content to provide, some content providers have hundreds, even thousands of content files. Storing and streaming this number of content files can be costly. Furthermore, streaming content requires a level of technical expertise often not found in companies focusing on creating content. Consequently, content providers have turned to content management service providers to store and stream content on behalf of the content providers.

[005] As more content providers turn to service providers for their streaming technology needs, service providers must manage more client accounts and greater numbers of content files. Furthermore, to maintain existing content provider clients and attract new clients, service providers must provide a content management system that not only is capable of organizing large numbers of content files, but also is easy for the content providers to use. Accordingly, there exists a need for a content management system that allows clients to easily up-load, manage, and deliver streaming media content via the Internet.

3. Summary of the Invention

[006] The present invention solves the foregoing and other needs. In certain embodiments, a method and system is provided for allowing client content providers to upload, manage and deliver streaming media and other digital content. The system according to one embodiment allows receives digital content from the client, assigns a stream identifier (ID) to the content and stores the content. The client is given a playlist uniform resource locator (URL) for publishing on its web site, the URL including the stream ID. Activation of the URL by an end user causes the stream to be served to the end user, without the client receiving or providing an indication of the specifics of where the content was stored.

[007] Another embodiment of the present invention permits clients to actively manage their content, including defining logical folders and subfolders containing item(s) of content; defining logical stream groups, containing items of content.

[008] A system according to one embodiment of the present invention couples one or more media servers to a storage server. While the storage server has stored a copy of the content, the media servers coupled thereto do not. In operation, the media servers perform a read of the contents stored at the storage server when requested by an end user.

BRIEF DESCRIPTION OF THE DRAWINGS

[009] The following drawing figures, which form a part of this application, are illustrative of embodiments of the present invention and are not meant to limit the scope of the invention in any manner, which scope shall be based on the claims appended hereto.

[0010] Figure 1 is a schematic illustrating the system architecture of one embodiment of the present invention;

[0011] Figure 2 is a schematic illustrating the database of one embodiment of the present invention;

[0012] Figures 3a-j are web pages of the Content Management client-side web site according to one embodiment of the present invention;

[0013] Figures 4a - b are flowcharts illustrating processes of up-loading content files into the content management system according to one embodiment of the present invention;

[0014] Figure 5 is a flowchart illustrating the process of creating a playlist according to one embodiment of the present invention;

[0015] Figure 6 is a schematic illustrating the system architecture of an alternate embodiment of the present invention;

[0016] Figure 7 is a schematic illustrating a workflow of a content management system according to the alternate embodiment of Figure 6; and

[0017] Figures 8a and 8b represent a schematic illustrating the database of the embodiment of Figure 6.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

[0018] Certain preferred embodiments of the present invention will now be discussed with reference to the aforementioned figures, wherein like reference numerals refer to like components. Turning first to the schematic of Figure 1, a system 100 according to one embodiment of the present invention is shown. In general, the system 100 allow clients 102 to up-load streaming media content to the system 100, manage such content, and make the content available to end-users 104 via a web site on the Internet.

[0019] As will become apparent from the following discussion, each of the clients 102 and end-users 104 has a processor, such as a personal computer (PC), web enabled cellular telephone or personal digital assistant (PDA) and the like, coupled to the Internet by any one of a number of known manners. Furthermore, each client 102 preferably includes an Internet browser, such as that offered by Microsoft Corporation under the tradename INTERNET EXPLORER or that offered by Netscape Corp. under the tradename NETSCAPE NAVIGATOR, as well as file transfer protocol (FTP) client software for interacting with various components of the system 100. Additionally, each end-user processor preferably includes an Internet browser, such as INTERNET EXPLORER or NETSCAPE NAVIGATOR, and a media player, such as that offered by Microsoft Corporation under the tradename WINDOWS MEDIA PLAYER or that offered by Real Networks Corp. under the tradename REALPLAYER. It is to be understood that although the present embodiment is described in terms of Windows Media content and Real Media content, it is within the scope of the present invention to utilize any media format heretofore or hereafter known. Furthermore, it is to be understood that although the present embodiment is described in the context of streaming media, the present invention is applicable to digital content other than streaming media.

[0020] As discussed in greater detail below, the client 102 up-loads streaming media content to the repository or storage server (and associated storage) 106 either directly, via an HTTP up-load, or via an FTP ingest server (and associated storage) 108. Once the client 102 has up-loaded its streaming media content, the client may manage its content via web pages on a content management web site provided by a client-side web server 110. As discussed in greater detail with reference to Figure 3a-j, the web site serves as an interface through which the client 102 may log into its account and select any of the following exemplary functions: up-load new content; search the client's existing up-loaded content; create and edit playlists; browse the client's contents stored on the FTP server 108; and perform a batch up-load of content from the FTP server 108 to the repository server 106. Additionally, the web site serves as an interface for the client 102 to perform various administrative functions, such as setting and changing any of the client-defined account and file information described below. In alternate embodiments, clients 102 may manage their accounts programmatically, via scripts running on the web server 110 or other system service described below.

[0021] The system further includes a file management server 112, administrative staff terminals 114, and a script processor 116. In general, the file management server 112 controls the activities of the various other components to which it is coupled and the movement of content within the system. The administrative terminals 114, in turn, are coupled to the file management server 112 to allow the administrative staff of the content management service provider maintaining the content management system 100 to control and monitor the system 100. The script processor 116 works in conjunction with the file management server 112 to control operation of the system 100 by running various software scripts and/or components performing most of the functionality described herein. As described in greater detail below, the script processor 116 includes various software modules, including a task scheduler and program scripts and objects for batch up-loads of content files and for recovery of deleted content files. It is to be understood that such task scheduling software may be obtained by any of a number of third-party vendors. Similarly, based upon the description of the program scripts and objects herein, the program scripts and objects may be written in any programming language heretofore or hereafter known, such as PERL, Visual Basic, JavaScript, C++, and the like.

[0022] The content management system 100 also includes a Content Management (CM) Database 118. As described in greater detail below with reference to Figure 2, the CM Database 118 includes numerous relational databases or tables in a database. In general, the CM Database 118 preferably includes account information, which identifies each client's account, stream information, which identifies and describes each item of streaming content within a client's account, playlist information, which identifies and describes each client's playlist, batch information, which defines the status of each client's batch up-load requests, and storage location information, which tracks the storage of content on the repository 106 and streaming media 120 servers.

[0023] Under direction of the file management server 112, the up-loaded content is eventually transferred from the repository server 106 to one or more streaming media server (and associate storage) 120, where it is available for streaming via the Internet. As described in greater detail below, the current embodiment utilizes a playlist server 122 for dynamically generating a playlist metafile (such as ASX, RAM, SMIL, and RPM files, to name a few) to be provided to the end-user's windows media player. In addition to the archived content located on the streaming media server storage 120, the present environment provides for streaming of live content acquired via encoders 124 coupled to the streaming media servers 120.

[0024] As illustrated in Figure 1, while the clients 102 and end-users 104 are coupled to each other and to the content management system 100 via the Internet, the service provider's components are coupled to each other via a local area network (LAN). More specifically, the repository 106, FTP server 108, client web server 110, file management 112, administrative terminals 114, script processor 116, CM Database 118, streaming media server 120, and playlist server 122 are all in communication via a secure LAN. In alternate embodiments, components of the system are coupled differently, for example, each coupled directly to the Internet, coupled via a wide area network (WAN) and the like.

[0025] In general, the division of functionality among the web server 110, file management server 112, playlist server 122 and script processor 116 described herein is exemplary, as the functions may be segregated and grouped differently. For example, it is to be understood that although the present embodiment utilizes client web servers 110 that are separate from the playlist server 122, in alternate embodiments the functionality of the playlist server 122 described herein may be implemented in software residing on the web server 110, thereby obviating the need for a separate playlist server.

Content Management Database 118

[0026] The Content Management (CM) Database 118 will now be described in greater detail with reference to Figure 2, and continuing reference to Figure 1. The CM Database 118 includes several logically discrete tables of information, each of which is described below. As will be appreciated by one skilled in the art, the following logical arrangement of information in tables is exemplary, and other arrangements are within the scope of the present invention; for example; those with fewer or more tables and/or fewer or more fields. It is also to be understood that although the CM Database 118 is described as a central database, it is within the scope of the present invention to distribute the contents of the CM Database 118 throughout the system.

[0027] The CM Database 118 includes a CM Accounts Table 210. In general, the CM Accounts Table 210 includes records for each client account, as identified by a CM account ID. Each such record includes account identifying information, such as account name, account directory, username, password, contact information (such as contact name, e-mail address, and telephone number), default title and author (which are preferences used to identify items of content in the event no other title or author information is provided by the client 102), parent account ID (in the event the current account is related as a child to another account), an indication of whether or not the client has an FTP account, and maximum allowable amount size of content (e.g., in Kilo-bytes). It is within the scope of the present invention to implement security features to maintain the integrity of the content, such as those features and methods described in applicant's co-pending International Application No. PCT/US01/46726, filed November 5, 2001, for SYSTEM AND METHOD FOR CONTROLLING ACCESS TO DIGITAL CONTENT, INCLUDING STREAMING MEDIA, and International Serial No. PCT/US01/18324, filed June 6, 2001, for SYSTEM AND METHOD FOR CONTROLLING ACCESS TO DIGITAL CONTENT, INCLUDING STREAMING MEDIA, which are hereby incorporated by reference. As such, the table includes security related information, including default preferences indicating whether or not the content is secure (allows security), and if so, a cryptographic key used for accessing such content (default sec key) and a time period for which the key will be valid (default sec. interval).

[0028] The CM Accounts Table 210 also includes a field for a "site id," which uniquely identifies a site consisting of a group of one or more FTP servers 108 and those repository servers 106 and streaming media servers 120 to which the content on the FTP servers 108 can be up-loaded. Two representative sites are illustrated in Figure 1, although the system 100 could have any number of sites. Although not required in the present invention, each client account is associated with a site located in the client's geography or the geography where the endorsers 104 likely to access the client's content are located so to speed delivery of the streaming media content. Furthermore, it is to be understood that logically grouping the FTP server 108, repository servers 106 and streaming media servers 120 into sites is not required. As noted below, much of the account identifying information of the present embodiment is provided by the client 102 during the registration process.

[0029] The CM Database 118 further includes a series of tables containing content – identifying information. More specifically, the CM Assets Table 212 includes records, each of which identifies an item of streaming content by a stream ID. Furthermore, each record includes the CM account ID, creation date of the item of content, description of the content, an identification of whether the content is audio and or video, the platform to which the content relates, the date on which the content was last modified, any code necessary for viewing of the content, the length and size of the content, the expiration date (if any) of the content, the individual that up-loaded the content and the date on which the content was processed into the system 100. It is to be understood that each record in the CM assets Table 212 correlates an item of content, as identified by the stream ID, to a particular client account, by reference to the CM account ID.

[0030] The Streams2 Table 214 also includes records identifying each item of content, as identified by the stream ID. Fields contained in the Streams2 Table 214 include the stream type, such as windows media player or real media player, title of the content, author of the content, status of the content, copyright notice for the content, URL prefix, bite rate of the content, file name of the content (as provided by the client 102), stream format, such as windows media player or real media player, ad tag (which specified where an advertisement is to be inserted), start time (if the Stream is a portion of the file), duration, and expiration date.

[0031] As illustrated in Figure 2, the CM Assets Table 212 has three additional tables related thereto: the CM Stream Keywords Table 216, CM Assets Locations Table 218 and CM Archive Volume Table 220. The CM Stream Keywords Table 216 includes keywords, as provided by the client 102, for each item of content as identified by the stream ID.

[0032] The CM Assets Locations Table 218, which is also related to the CM Assets Table 212, includes records identified by location ID. The location ID is a unique identifier for each portion of storage pertaining an item of content. As such, location IDs may refer to either a repository server 106 or a streaming media server 120. A given content file may be stored at multiple location ids, for example one on a media server 120 and one on a repository server 106. More specifically, each record in the CM Asset Locations Table 218 includes a CM volume ID, for identifying the portion of the server on which the associated content resides; stream ID, for identifying the item of content stored at the location ID; and status of the item of content. The status of the file corresponding to each location is determined based on the following table.

Status	**Meaning**	**Applicable to file located on:**
Available	File is present on a media server and is available for streaming	Streaming server
Back up	File has been copies to a streaming media server	Repository server
Copying	File is being copied to a repository server or media server	Repository server; media server
Deleting	File is being deleted	Repository server; media server
Failed	File up-load or copy has failed	Repository server; media server
Moving	File is being moved from the current server to another server of the same type	Repository server; media server
New	File has recently been up-loaded and not copied	Repository server

[0033] The CM Archive Volume Table 220 is also related to the CM Asset Table 212 and, more particularly, to the CM Asset Locations Table 218. CM Archive Volume Table 220 includes records for each CM volume ID. Each record includes: volume type, which indicates whether the volume ID pertains to a repository server 106, streaming media server 120, FTP server 108, script processing server 116 or web server 110; archive server name, indicating the LAN name of the particular repository server 106 or streaming media server 120, as the case may be, containing the volume ID; the server's DNS address; the platform for the content, such as Window's Media or Real Media; the status of the content associated with the volume id; the maximum storage capacity for the server continuing the volume id; and the side id of the server containing the volume id.

[0034] Also associated with each client account identified in the CM Accounts Table 210 is client playlist information. Such playlist information is contained within the CM Playlists Table 222, Content Group (CG) Table 224 and CG Streams Table 226. In general, the CM Playlist Table 222 includes records identifying each client playlist, as identified by a content group ID. Each record further includes the CM account ID and the sort order, which indicates the ordering of the items of streaming content associated with the particular playlist.

[0035] Content Group Table 224 also includes records for each playlist, and identified by the playlist or content group (CG) ID. In general, each record contains the playlist details, including the CG ad tag, the type of playlist, the CG description, the CG format, the CG label, the meta-url for the playlist, CG notes, and an indication as to whether or not the content group is shared.

[0036] The playlist entries are identified in a CG Stream Table 226. More specifically, the CG Stream Table 226 includes records identifying each item of content as part of a content group and further includes an indication of the order of the particular item of content within the content group. As such, each record in the table includes the content group ID, stream ID, and sort order.

[0037] The CM Database 118 also includes a table that indicates its status of client-requested batch up-loads. More specifically, the CM Batch Jobs Table 228 includes records identifying the status of each requested batch job, as identified by a CM batch job ID. Each record in the CM Batch Job Table 228 includes the CM account ID, identifying the account to which the batch job relates, the file name of the batch file (filename), the time which the client 102 submitted the batch request (submit time), the time at which the process was started (process start time), the time at which the batch process ended (process end time), and an indication as to whether or not the batch file was removed from the client's account (file removed). Because each record in the CM Batch Jobs Table 228 includes the CM account ID, each such record is related to a record in the CM Accounts Table 210.

[0038] The CM Database also includes a Stream-Servers Table 230. The records in the Stream-Servers Table 230 correlate each file, as identified by its stream ID, with the hostname (or domain name server (DNS) address) of the streaming server 120 on which it resides. While each stream ID will only have one record, a particular file will have as many records as it has copies residing on different streaming servers 120.

[0039] The Servers2 Table 232 includes a record for each streaming server 120, as identified by its server address, setting forth various metrics for the server 120, such as the weight or relative number of streams being served to end users 104.

[0040] The CM Sites Table 234 contains a record for each site, as identified by its site id. More specifically, each record identifies the name of the site, the FTP server uniform resource locator (URL) associated with the site and the processor URL associated with the site. In the present embodiment, each site only includes one FTP server 108.

[0041] As illustrated in Figure 2, the CM Database also includes tables relating to the deletion and recovery of files, identifying each client's service level and categorizing the clients' accounts.

[0042] The Account Servicelevels Table 236 stores a description of each client's service level. As illustrated, this table 236 is keyed to the Account Table 210 by virtue of the account service level ID (acct srvlevel id) field. Each service level generally corresponds to a different set of options, functions and/or capabilities offered to the client, for example, the amount of storage allowed, number of uploads, the number of sites provided, and the like.

[0043] The Categories Table 238 stores records identifying the category to which a particular account relates. More specifically, each record corresponds to a particular category, as identified by category ID. In general, a category is a grouping of accounts, for example, of all accounts related to a particular parent account. Thus, each record identifies the parent ID, name, and description. As illustrated, records in the Categories Table 238 are keyed to accounts in the Account Table 210 by the category ID field.

[0044] The CM database 118 also tracks deleted accounts and deleted content. To this end, the Deleted Accounts Table 240 keeps a record for each account, identified by account ID, that is removed from the system 100. By keeping such records, the system administrator is able to provide accurate record keeping and report generation.

[0045] The Deferred Delete Table 242, Deleted Assets Table 244 and the Recovery Jobs Table 246 are used in the deletion of content. More specifically, a Stream may be deleted either upon request of the client 102 or based on the content expiring. As noted above, the Streams2 Table 214 includes an expiration date field for each Stream ID. A script running on the system periodically scans the Streams2 Table 214 for records having an expiration date equal to or greater than (i.e., after) the then current time. For each record containing such an expiration date, the systems proceeds to delete the content. The process of deletion of the present embodiment includes changing the status field in the Asset Locations Table 218 to expired and, in the Streams-Servers Table 230 changing the hostname for the stream being deleted to another system server that hosts a default message, indicating to the end user 104 that the requested content is not available. The system also replaces the stream's actual filename, as stored in the Streams2 Table 214, with the filename of such default message.

[0046] Upon deletion of a stream, a record is also created in the Deleted Assets Table 244, specifying various stream information, as well as the expiration date and deletion date of the content.

[0047] In the event the stream to be deleted is currently being provided to an end-user 104, the system will be unable to delete the content. In such an instance, the system creates a record in the Deferred Delete Table 242, specifying the server name, file name and share information for the content to be deleted. As illustrated, for each unique combination of server name, share information and file name, the Deferred we defer delete Table 242 also stores the site ID, stream ID, the date on which the delete request was submitted to the system, the date of the last attempted deletion, and the number of failed deletion attempts. A script running on the system periodically searches the Deferred Delete Table 242 for entry and, upon the finding a populated record, proceeds to attempt to delete the content specified by the entry. Upon successful deletion, an entry is created in a Deleted Assets Table 244 and a status is changed in the Asset Locations Table 218.

Registration Process

[0048] Having described the components of the present embodiment, the operation thereof will now be described. As an initial matter, each client 102 must register with the service provider. In general, such registration includes the client 102 providing the service provider with account identifying information, a subset of which includes client identifying information. More specifically, the client provides the account identifying and client identifying information contained within the CM Accounts Table 210. The client may also select the designed service level. In one embodiment of the present invention, the client 102 provides such information via a secure web page generated by the client web server 110. The client web server 110 receives the information and by executing a common gateway interface (CGI) script, writes the information via the LAN to the CM Database 118. In an alternate embodiment, the client 102 manually provides the information to the service provider, who, in turn, manually enters the information into the CM Database 118 via the administration terminals 114. In either embodiment, once the service provider receives the account identifying and client identifying information, a CM account ID is assigned and the corresponding record in the CM Accounts Table 210 is populated.

Client-Side Content Management Web Site

[0049] As noted above, the client web server 110 provides a content management web site that serves as an interface between the client 104 and the content management system. While the present embodiment utilizes such interactive web site for allowing the client 102 to interface with the system and to manage the client's content, it is to be understood that other interface devices may be used, including voice recognition devices, text-based systems, or by passing variables on a query string and running a script, or any other interface means. Furthermore, although the present embodiment passes data between the web server 110 and the script processor 116 in XML format, it is to be understood that other formats may be utilized. The content management web site of the present embodiment will now be discussed with reference to Figures 3a - j.

[0050] Once a client 102 logs into the system by submitting its username and password, the client is presented with the option to perform content management functions or administrative functions. As shown in Figure 3a, when the client 102 selects administrative functions from a high level menu 302, the web page presents the client 102 with the ability to set account preferences, including default title, default author, default security key, and default security interval. As with the other web pages discussed herein, entry of information via the client 102 is received by the client web server 110 and placed in the appropriate fields in the CM database 118. With the account preferences web page of Figure 3a, the default preferences are stored in the CM Accounts Table 210. In alternate embodiments, the web page also displays account settings such as maximum allowable storage (max content Kb), actual storaged used (e.g., sum of Kb size fields in Assets Table 212), and any other information stored in the database 118.

[0051] As shown in Figure 3b, when the client 102 selects the manage content option in the high level menu, the web site presents the client with six options: "Upload New Content", "Browse FTP Content", "Batch Uploads", "Playlists", "Search" and "Recycle". When the "Upload New Content" option is selected, the system presents the client 102 with the web page shown in Figure 3b. As discussed in greater detail below, the client 102 is able to identify a content file residing on its local machine, and upload the file to a repository server 106. In general, the client 102 specifies the file name and file details in an upload content form 306. These file details are then stored in the CM Assets Table 212 and the Streams2 Table 214. Notably, the account preferences set by the client 102 on the web page of Figure 3a are provided as default entries in the upload content form 306.

[0052] When the client 102 selects the "browse FTP content" option, the system presents the client with the web page shown in Figure 3c. As the name indicates, this option provides the client 102 with a listing of all content files previously uploaded by the client 102 to the client's FTP account on the FTP ingest server 108. The system first identifies the FTP ingest server 108 containing the client's FTP account. A call is made to the particular FTP ingest server 108, which in turn executes a script to read all file directories and files within the client's account. The FTP server 108 returns a listing of such file directories and files in XML format to the web server 110. The web server 110, in turn, converts the XML information into HTML, which is displayed to the client 102. As shown in Figure 3c, the client 102 is presented with the name of its FTP account 308, as well as a listing of the client's file directories 310 and the client's content files 312.

[0053] The browse FTP content web page allows the client 102 to either delete a content file or cause a content file to be ingested from the FTP server 108 to a repository server 106. Such operations will be discussed in greater detail below.

[0054] The "batch uploads" option is shown in Figure 3d. As shown therein, the batch upload web page of the current embodiment is divided into two sections: one illustrating the batch files which have been uploaded to the client's FTP account, but not processed; and one identifying the client's batch files for which processing has begun 316. By way of example, batch file "1-100.bul" has been uploaded but not processed, while batch file "BillDefault.txt" was previously uploaded and processed.

[0055] It should also be noted that the batch upload web page may display any file detail stored in the CM database 118 corresponding to each batch file. More specifically, for each batch file listed as having been processed, information may be extracted from the submit time, process start time and process end time fields of the CM Batch Jobs Table 228.

[0056] The listing of available batch files 314 is generated by the system browsing the client's FTP content and extracting all pure text files. The listing of processed batch files is created by the system by searching the CM Batch Jobs Table 228 for all records identified by the client's CM account ID. The client 102 may select one of the available batch files for either deletion or processing.

[0057] The web page displayed when the client 102 selects the "playlists" option is illustrated in Figure 3e. In general, this web page provides the client 102 with the ability to edit playlists. To this end, the web page presents the client 102 with a list of current playlists 318. The system generates the list of playlists by searching the CM Playlists Table 222 for all entries corresponding to the client's CM account ID, thereby resulting in a list of playlist or content group (CG) IDs, each of which identifies a different playlist for the client 102. For each CG ID, the system retrieves the CG description and format in the Content Group Table

224. The information is transferred in XML format and displayed in the playlist list 318 in HTML format.

[0058] The system also provides the user with a search form 320, whereby the client 102 can enter playlist details, such as playlist/CG ID, CG description and CG format, in which the system uses in searching the Content Group Table 224 in order to return a list of matching playlists.

[0059] The client 102 may also select a current playlist to manage. As shown in Figure 3f, when the client 102 chooses to manage a playlist, the system retrieves from the CM database 318 the CG id 322, various playlist details 324 as stored in the Content Group Table 224, and a list of streams or content files making up the identified playlist 326 as retrieved from the Content Group Streams Table 226 based on the CG ID. From the list of streams, the client 102 may select any number for deletion from the playlist, in which case the system removes the stream's id from the corresponding record in the Content Group Streams Table 226.

[0060] Notably, the web page also provides the client 102 with the uniform resource locator (URL) for the playlist 328. In general, the playlist URL 328 is incorporated into the client's web page as a link, and when activated by an end user 104, causes the playlist identified by the embedded playlist/CG ID to be played. The process of generating the playlist URL 328 and playing the streams associated with the playlist is described in greater detail with reference to Figures 5.

[0061] The web page displayed when the client 102 selects the "search" option is shown in Figure 3g. The system provides the client 102 with a search form 330 that enables the client 102 to enter the file details, including title, author, keywords, stream id, creation date, and duration, as search criteria. The web server 110 passes the file details entered by the client 102 to the script processor 116, which uses the values to search the CM Assets

Table 212, CM Stream Keywords Table 216 and the Streams2 Table 214. The script processor 116, in return, returns various file details of the client's content files meeting the search criteria. Such results are shown in Figure 3h. The search form 330 also includes an option to display the results in XML format (as received from the script processor 116) and to search not only the client's current account, but also all child accounts (as indicated by the parent id field in the CM Accounts Table 210).

[0062] As an alternative to using the search form 330, the client 102 may simply browse all content in its content management account by clicking the "Browse" button. Once the client 102 clicks the browse button, the web server calls a script on the script processor 116 that retrieves all files associated with the client's username and password (and, thus, account id).

[0063] Exemplary search results are illustrated in Figure 3h. Each content file may be identified by any of the file details stored in the CM Database 118. In the present embodiment, each file is identified by the stream ID, title, stream type, stream description, creation date, file size, and status, all of which are stored in the CM Assets Table 212 and Streams2 Table 214.

[0064] Furthermore, the web page displaying the search and browse results (Figure 3h) provides the client 102 the option to select one or more files and either "delete" the selected files or "create a playlist" by adding the selected files to a new or existing playlist. In an alternate embodiment, clients 102 are also provided with the option to view the streams URL playlist URL 328. In the event the "create a playlist" button is activated, the "create a playlist" window shown in Figure 3i is displayed. The window displays a list of the client's existing playlists 332 and the files contained within each playlist 334 (as provided in the "playlist" web page), and the window provides the client 102 a form 336 to enter playlist details, including description and format, for a playlist to be created using the files selected on the web page of Figure 3h.

[0065] It should be understood that the foregoing client web pages represent an exemplary client interface and that other interfaces are within the scope of the present invention. For example, clients may access their account via a script, such as a virtual basic script, on an active server page (ASP), and pass the relevant variable in a query string rather than entering them on a web page.

[0066] Thus, in certain embodiments clients 102 are able to manage playlists via an ASP page running on one of the system servers (e.g., the web server 210 or a separate application server, not shown) by passing certain variables on a query string. In one such embodiment, the client 102 log into the system by paring its username and password and preferably as part of the same query string, passes any of a number of parameters depending upon the desired function.

[0067] For example, the client may pass a parameter specifying what action is to be performed on the playlist. Such action may include: adding or creating a new playlist; editing the playlist (e.g., editing the playlist parameters or streams comprising the playlist); removing or deleting the playlist; listing playlist in the account (e.g., by name or ID); listing the playlists and the contents of each playlist and the like. The playlist ID parameter is specified to identify the playlist on which the system is to act.

[0068] By way of example, playlist parameter that can be specified (and thus edited in the CM Database 118 is the name of the playlist. The playlist name can also be used to identify the playlist which the system is to act. Another parameter that can be specified, and thus edited) is the playlist format (e.g., RAM for RealMedia content and ASX for Windows Media content).

[0069] The client can also pass parameters specifying the action to be taken, if any, on individual streams comprising the specified playlist. Such actions may include, for example: adding a stream to the playlist; removing a stream from the playlist; removing all streams from the playlist; one or more stream IDs, to identifying the steam(s) when adding or removing them.

[0070] Preferably, under a client uses such an ASP to manage playlists, the server returns a message, such as an XML message, indicating whether the requested action was successfully performed and, if so, any information generated, much as the playlist ID of a newly created playlist. In the event the requested action was not successfully performed, the server responds with a error message identifying the error.

Uploading Content

[0071] As an initial matter, the system must identify the server to which a client's content should be uploaded. The site id is used to identify to which server a particular client's content should be uploaded, copied or moved. Specifically, once a client 102 logs into the system by providing its username and password, the system can identify the client's record in the CM Accounts Table 210, which provides the client's site id. Based on the site id, the system searches the CM Archive Volume Table 220 and identifies all records having the client's site id. Each of these records identifies a different server associated with the client's site id. If the system needs to access the FTP server 108 for the client (either to read the contents of or transfer a file to or from the client's account), the system identifies those CM Archive Volume Table records having the site id and a volume type corresponding to "FTP". Similarly, the system can identify repository servers 104 and streaming servers 120 associated with the client's site id by identifying those CM Archive Volume Table records having a type corresponding to "Repository" and "Streaming/Archive", respectively.

[0072] As described with reference to Figures 4a-b, the system of the present embodiment allows the client 102 to upload content to the repository servers 106 by any of several different processes. The first processes (described with reference to Figures 4a and b) transfer content files from the client's FTP account on the FTP server 108 to a repository server 106. As such, the first step in these processes is the client 102 uploading content to the FTP server 108 as noted above. The last manner is from the client's machine to a repository server 106 via HTTP upload.

[0073] As an initial step in the HTTP upload process of Figure 4b, the client 102 logs onto a web page provided by the client web server 110 by entering the client's user name and password. Having logged onto the system, the client then selects the "Upload New Content" option presented on the CM web page and identifies the content file locally stored on the client's machine that the client 102 desires to upload. Step 480. As illustrated in Figure 3b, the client 102 has the option of identifying the file by file name or by clicking the "Browse" button, which opens a window listing the client's locally stored files. Additionally, the client 102 specifies at least required file details for inclusion in the CM Database 118. Step 484. The client web server 110 proceeds to transfer the file from the client's machine onto the repository server 106 via an HTTP upload. Step 488. The system assigns a stream id to the file and creates a record in each of the CM Assets Table 212, Streams2 Table 214 and CM Stream Keywords Table 216, and the system assigns a location id and creates a record in the CM Assets Location Table 218. Step 492. The client web server 110 proceeds to write the file details, as provided by the client 102, into the appropriate fields in the CM Database 118. Finally, the system returns the stream id to the client 102, which is displayed on the CM web page with a confirmation message. Step 496. In the event any of the foregoing steps fails, the system notifies the client 102 via a message on the CM web page.

[0074] Clients 102 may also programmatically upload content from their FTP Account on the FTP server 108 to a repository server 106 by executing a script. It is to be understood that although the following embodiment utilizes a Virtual Basic ("VBS") script on an active server page (ASP) of the web server 110, the functionality described may be implemented using any programming language such as Java Script.

[0075] As an initial step in the programmatic upload and ingest process, the client 102 uploads one or more files to the FTP ingest server 108 utilizing FTP client software residing on the client's machine. Once the files have been uploaded, the client 102 may use the ASP and pass certain variables on a query string to cause a file to be ingested or to perform another content management function.

[0076] As part of the ingest request of the present embodiment, the client 102 must enter its user-name and password, as well as certain file details. Required file details include the name of the file in the client's FTP directory ("Filename"), the bit rate of the file ("Bitrate"), the title of the file ("Title"), and the author of the file ("Author").

[0077] The following optional file details may also be provided by the client 102 depending on the desired action: the copyright notice to be added to the content file ("Copyright"), a description of the file ("Description"), key words associated with the file to be used in searches ("Keywords"), an indication as to whether or not the file is secure ("IsSecure"), an alphanumeric security string for use in accessing the file if it is identified as being secure ("SecKey"), the security interval, the description of an existing playlist in the client's account ("PlaylistDesc"), an indication whether to add the file to the playlist identified by the aforementioned description or to replace the existing files associated with the aforementioned playlist with the file being ingested ("PlaylistAction"), Playlist ID (CGID) and an indication whether to automatically place the ingested file on a suitable streaming media server 120 or to only place the ingested file onto a repository server 106, where it will stay until the administrative staff manually causes the file to be placed on a streaming media server 120 ("AutoArchive"). In short, any stored file detail may be specified. The process of adding and replacing files in a playlist is discussed in greater detail below.

[0078] It is to be understood that the aforementioned file details are merely exemplary and that other file details may be considered required or optional in alternate embodiments of the present invention. Furthermore, it is to be understood such file details may be manually entered by the client 102 when submitting the ingest request, or, in an embodiment of the system wherein the client 102 enters the file details upon uploading the content files to the FTP server 108, the system may automatically retrieve the file details from the CM Database 118 based on the client's selection of the file to be ingested.

[0079] In general, the query string or command line identifies the active server page script, the location of the script, and the variables being passed to the script, including the user name, password, required file details, and any optional variables. The following is one example of such a command line:

```
http://contentmgmt.broadcast.com/vbs_script_ingest.asp?UserName=Foo&Password=Bar&
FileName=testfile.rm&BitRate=5600&Title=Test File&Author=John Doe
```

[0080] The foregoing exemplary command line passes the real media file "testfile.rm," which has a bit rate of 5600 kilobytes, is entitled "test file" and is authored by "John Doe." Furthermore, the client 102 is identified by the user name "Foo" and password "Bar." Such variables are passed to the script "Vbs_script_ingest.asp," located at "content mgmt.broadcast.com."

[0081] The following exemplary command line logs in user "Foo" using the password "Bar," uploading the file "Testfile," giving the file a Title, Author, Copyright, Description, and Keywords.

```
http://contentmgmt.broadcast.com/vbs_script_ingest.asp?username=Foo&Password=Bar&Title=Te
File&Author=Yahoo! Broadcast&CopyRight=(c)2000 All Rights Reserved&Description= Testing
```

[0082] When the script is called, the system receives the file details and causes the identified file to be moved from the FTP ingest server 108 to a repository server 106. Based on whether the file was successfully uploaded, the system provides to the client 102 one of two XML strings. Such XML string includes a success code field (scode), a success code description (scode_Description) and, if the upload was successful, the stream ID.

[0083] Thus, if the file was not successfully uploaded, the following XML string is returned to the client 102, wherein an scode of "1" indicates a failure and the value of the scode_description indicates the reason for the failure (e.g., an invalid username).

```
<?xml version="1.0" ?>
<response scode="1" scode_description="invalid username" />
```

[0084] In the event that the file was successfully uploaded, the system returns the following XML string wherein the scode and scode_description indicate the successful upload

```
<?xml version="1.0" ?>
<response scode="0" scode_description="Success" stream_id="999999" />
```

[0085] The system then renames the file and updates the appropriate fields in the CM database 118. More specifically, with the Stream ID assigned to the file, a record is created in each of the CM Assets Table 212, Streams2 Table 214 and CM Stream Keywords Table 216. Additionally, a location id is assigned and a record is created in the CM Assets Locations Table 218.

[0086] As will now be described with reference to Figure 4a, and continuing reference to Figure 3d, the client 102 may also perform two types of batch uploads, thereby transferring files from the client's FTP account to a repository server 106. The two types of batch uploads will be referred to as "default" and "standard" uploading. In general, default batch uploading applies default file details contained in the batch file to an entire sub-directory of content files. On the other hand, the standard batch upload process applies file details contained within the batch file on a file-by-file basis.

[0087] As an initial step, the client uploads the media files and batch file to the FTP ingest server 108. Step 446. In the case of a default batch upload, the batch file must be placed in the root directory of the client's FTP account. In the case of the standard upload process, the batch file, along with the media files, are left in the root folder of the client's FTP account.

[0088] The format of the batch file also depends upon the type of batch upload. The default batch upload file takes on the following format wherein the file is first identified as being for use in a "DEFAULT" batch upload and the relevant file details are separated by double colons as follows:

```
DEFAULT::DIRECTORY::BITRATE::TITLE::AUTHOR::COPYRIGHT::
DESCRIPTION::KEYWORDS::ISSECURE::SECURITYKEY:: SECURITY INTERVAL
```

[0089] The default batch file also includes an indication of the FTP DIRECTORY where the content files to which the default file details are to be applied are located. In the event the client 102 does not wish to supply an optional file detail, the location for such file detail is simply left blank. Furthermore, by placing the word "FILENAME" in parenthesis in either the Title or Description field in the batch file, the system will automatically include the file's file name in such field. By way of example, the following default batch file will apply to all files in the client's FTP directory named "288_Filesdirectory," which contains filing have a bitrate of 2880 kilobytes, entitled "Upload," including the file name, authored by J. Doe, a copyright notice reading "Copyright©2001," no description and no keywords, an indication that the file is secure, having the security key "mysecuritykey," and having a security interval of 122 minutes.

```
DEFAULT::288_FILES::2880::Upload(FILENAME)::J.DOE::Copyright©2001::::::Y::MY
SECURITYKEY::122
```

[0090] The standard batch upload batch file includes the same file details, however, does not include the "DEFAULT" identifier or the "DIRECTORY" identifier. Instead, each line in the standard batch file begins with the file name to which the file detail set forth in that line are to be applied. As such, the following is the format of the standard batch file:

```
FILENAME::BITRATE::TITLE::AUTHOR::COPYRIGHT::DESCRIPTION::KEYWORDS::I
SSECURE::SECURITYKEY::SECURITYINTERVAL
```

[0091] Once the media files and the batch file is uploaded to the system, the client logs into the system via the CM website and selects the "Batch Upload" option, as shown in Figure 3d. Step 450. As part of the process of selecting the Batch Upload option, the client 102 is presented with a list of both processes and unprocessed batch files. (See Figure 3d). The client 102 selects an unprocessed batch file for processing.

[0092] Upon selecting a batch file for processing, the system executes a script that causes an entry to be created in the CM Batch Jobs Table 228. Step 454.

[0093] At some later point, according to a predefined interval, a task scheduler running on the script processor 116 causes a batch script to execute. Step 458. As described below, the batch script proceeds to read and process the unprocessed batch jobs identified in the CM Batch Jobs Table 228. In an alternate embodiment, processing of the batch files is not automated and instead is manually initiated by the administrative staff via the file management server 112.

[0094] Once the batch process is started, the system proceeds to read the first record in the CM Batch Jobs Table 228. Step 462. Specifically, the system reads the record to determine whether the start time field equals a null value or whether some actual start time. Step 466. If the start time field has an actual start time, then the batch process job has already been completed and the system proceeds to read the next record in the CM Batch Jobs Table 228. Step 462.

[0095] In the event that the start time value of the current record equals the null value, the system executes the batch job. Accordingly, the system causes the first file associated with the batch file to be moved to a repository server 106 and assigns the file a stream id. Step 470.

[0096] The system proceeds to rename the file that has been moved and updates the CM database 118 to reflect the associated file details provided by the batch file. Step 474. More specifically, the system parses the batch file based on the batch file format discussed above, inserting the file details into the CM Assets Table 212, Streams2 Table 214 and the CM Stream Keywords Table 216. The system also creates a record in the CM Asset Locations Table 218. Once the file details have been entered into the appropriate records in the CM database 118, the system determines whether or not all content files associated with the batch file have been moved. Step 476. In other words, the system determines whether the end of the batch job has been reached. If the end of the batch job has been reached, then the system updates the record in the CM Batch Jobs Table 228 by entering the process end time. Step 478. If the end of the batch job has not been reached, then the system proceeds to move the next file associated with the batch file, assign a stream id, and update the CM database 118 accordingly. Steps 470, 474. Once the system cycles through the CM Batch Jobs Table 228, the processing of the batch jobs ceases until the scheduler begins the batch process again.

Selecting The Streaming Media Server on Which to Place a Client's Content

[0097] In the present embodiment, the system places each client's content on the streaming media server 120 that has the most free disk or storage space. More specifically, the script processor 116 (or other server in alternate embodiments) reads the client's record in the CM Accounts Table 210, identifying the client's site id. Using the client's site id, the system searches the CM Archive Volume Table 220 and locates all records having a volume type indicating streaming/archive. From this group of records, the system identifies only those having a platform (e.g., Windows Media or Real Media) corresponding to the format of the content file being uploaded. The resulting records are all potential streaming media serves 120 on which the content could be uploaded.

[0098] For each of the servers identified by the remaining records (each record and volume id corresponds to one server), the system proceeds to determine the available storage capacity by subtracting the used capacity from the maximum capacity. Using the records identifying all potential streaming media servers 120, the system notes the maximum storage capacity of each such media server 120, as set forth in the "max kb" field of the CM Archive Volume Table 220. The system then calculates how much of that capacity is actually being used by summing the size of all files stored on each such media server 120. More specifically, for each volume id (server), the system identifies all of the records in the CM Asset Locations Table 218 having the same volume id and identifies the stream id's in each of those records. The result is a list of all files, as identified by stream id's, residing on the server 120 corresponding to the given volume id. The system reads and sums the files sizes by accessing the CM Assets Table 212. For each volume id, the system subtracts the total storage used by the files residing on the server having that volume id from the maximum storage capacity of the server, thereby calculating the actual available storage.

[0099] The content file being uploaded is placed on the streaming media server 120 (i.e., volume id) having the most available storage capacity. The system uses the selected volume id to create a new record in the CM Assets Locations Table 218 for the archived content file.

Creation of Playlist

[00100] As will now be described with reference to Figure 5, clients 102 may also programmatically create playlists in the CM Playlists Database. The client 102 logs into his/her account via the web server 110 and selects the "Playlist" link. Steps 510 and 515. The client 102 is presented with a form into which the client 102 enters playlist details, such as description and file format, and selects and orders stream files for inclusion in the playlist. Step 520. Once the client 102 has finished entering the playlist details, the system creates a new playlist record in the CM Playlists Database and assigns a CG ID to the playlist. This CG ID is returned to the client 102 for inclusion in the client's web site 102. Step 525. The system also creates corresponding records in both the CG Database and the CG Streams Database based upon the playlist details. Steps 530 and 535. Next, the system retrieves the stream ID for the next stream file in the playlist and enters the stream ID in a record in the CG Streams Database. Step 550. The system then queries whether the end of the playlist was reached. Step 555. If yes, then the system has completed creating records for the

playlist. However, if more streams are found, the system loops back to Step 550 to create additional records.

[00101] In this way, the playlist details are stored on the CM Database 118 so as to be accessible by the playlist server 122 to dynamically generate playlist redirector files, as described below.

Dynamic Playlist Redirector Files Creation

[00102] Referring back to FIG. 1, the media servers 120 are connected to the world wide web, or some other global communications network, through the LAN. In this respect, streaming content is made available to end users 104 through the world wide web.

[00103] Upon completion of the scheduling and production phase of the web-cast event, a uniform resource locator (URL) or link is returned to the client 102 to be embedded in the client's web page. An end user 104 desiring to listen to or view the web-cast on their computer or other device can click on the URL. A playlist or CG ID (12345) is preferably embedded within the URL, as shown below:

```
<A href="http://playlistserver.company.com/makeplaylist.asp?id=12345">
```

In the illustrative URL shown above, the link points to the playlist server 122 that will execute the "makeplaylist.asp" program and dynamically generate a playlist redirector file. One skilled in the art will recognize that although the "makeplaylist" application has an Active Server Page (or ASP) extension, it is not necessary to use ASP technologies. Rather, any programming or scripting language or technology could be used to provide the desired functionality, for example, .dll.. It is preferred, however, that the program run on the server side so as to alleviate any processing bottlenecks on the end user side.

[00104] The "makeplaylist" program functions to cause the playlist server 122 to make a call to the CM database 118 to retrieve the data necessary to dynamically generate the playlist redirector file. The playlist server 122 first accesses the CG Streams Table 226 and

locates the record pertaining to the CG ID. As shown in FIG. 2, the CG Streams Table 226 contains the individual stream IDs and their respective sort orders. The stream IDs and the sort order are returned to the "makeplaylist" program which then makes a call to the Streams2 Tables 214 in the CM database 118 that contain data associated with the individual streams. Namely, the Streams2 Table 214 includes a URL prefix and a stream filename. The individual streams identifiers are also used by the "makeplaylist" program to call to the Stream-Servers Table 230 in the CM database 118. The Stream-Servers Table 230 includes the location or (DNS) hostname of the particular media server 120 containing the stream file associated with a particular stream identifier. Using the URL prefix, the hostname, and the stream filename, the "makeplaylist" dynamically generates a URL for each stream file. An example of such a URL is listed below, wherein the URL prefix is "mms://", the hostname is "mediaserver.company.com" and the stream file name is "filename.asf.":

```
<"mms://mediaserver.company.com/filename.asf">
```

[00105] Using the individual stream URLs, the "makeplaylist" program then dynamically generates a redirector file to be passed to the media player stored on the end user's computer 104. An example of a redirector file for use with Windows Media Technologies (WMT) is shown below:

```
<ASX>
  <ENTRY>
    <REF HREF="mms://mediaserver.company.com/stream1.asf">
    <REF HREF="mms://mediaserver.company.com/stream2.asf">
    <REF HREF="mms://mediaserver.company.com/stream3.asf">
  </ENTRY>
</ASX>
```

[00106] One skilled in the art will recognize, of course, that different media technologies utilize different formats of redirector files and, therefore, the term "redirector

file" is not limited to the ASX-type file shown above. Lastly, the end user's media player pulls each identified stream file from the media server 120 identified in the redirector file in the order in which it appears in the file.

[00107] It is to be understood that other system architectures are within the scope of the present invention. For example, certain embodiments include sites each having its servers on a separate LAN coupled to the administrative components (e.g., playlist server, file management server, script processor, database) via a secure SMTP connection. Furthermore, certain embodiments do not, as in the embodiment of Figure 1, include repository servers that maintain backed-up copies of files. One such embodiment that does not include repository servers for storing backups will now be described with reference to the schematic of Figure 6, the workflow diagram of Figure 7 and the CM Database schematic of Figure 8.

[00108] As illustrated in Figure 6, the architecture of the present embodiment essentially mirrors that of the embodiment of Figure 1. Indeed, the operation of the components illustrated therein is based on, and in many respects identical to the components described above with reference to Figure 1. However, rather than including repository servers for maintaining a backup of content stored on the streaming media servers, the present embodiment includes storage servers 606 that maintain a copy of each item of content to the exclusion of maintaining a duplicate copy on a streaming media server 620. In short, other than an external backup, such as on a tape backup system or mirror or redundant disk system, preferably the only copy of each item of content for a given site is on the storage servers 606. Accordingly, as described in greater detail below, when providing access to the content, the streaming media servers 620 performs a LAN-based read of the appropriate content on the storage servers 606.

[00109] It should be appreciated that coupling multiple media servers 620 to a single storage server 606 having the only copy (vis-à-vis those media servers 620) provides several

benefits. For example, the content management service provide saves money because less storage is used. Additionally, because multiple media servers 620 may access and, thus, stream the same item of content, clients 102 will encounter fewer instances of end users 104 not being able to receive the content because a media server 620 is already providing the maximum number of streams. While so utilizing the storage servers 606 and media servers 620 is preferable, it is within the scope of the present invention to have some sites employ an architecture wherein a repository or backup server and the media server each has a copy of the content, and have other sites employ a architecture where media servers do not have a copy of the content.

[00110] Unlike the embodiment of Figure 1, the present embodiment also includes site-specific file manipulation servers 607. As described in greater detail below, each such file manipulation server 607 causes files to be moved from each site-specific FTP ingest server 608 to the appropriate storage server 606.

[00111] It is also within the scope of the invention to have one or more sites in the present embodiment on separate networks in communication with the administrative servers and processors via SMTP or other connection, which themselves may be on a separate network.

[00112] Having described the architecture of the present embodiment, its general operation will now be described with reference to the workflow diagram of Figure 7. It is to be understood, however, that aside from the differences noted or otherwise to be understood based on the following description, the components of the present embodiment operate as like named components of the embodiment of Figure 1. As illustrated therein, clients 102 are able to upload content either via an HTTP upload to the web server 610 or via an FTP upload to the client's primary site, as identified in the client's account record in the CM database 618. The web server 610, in turn, performs a LAN-based file move to the appropriate storage

server 606. In alternate embodiments, the read is via a direct, point to point connection. Under direction of the file management server 612, each site-specific file manipulation server 607 causes the content previously uploaded to the FTP ingest server 608 to be moved via the LAN to the appropriate storage server 606.

[00113] With the content stored on the appropriate storage server 606, each streaming media server 620 receiving a request for content will read the file from the appropriate storage server 606 (in the present embodiment, via a LAN based file read) to obtain the content and ultimately provide it to the end user 104 requesting it.

[00114] Because the functionality of the present embodiment, in large part, parallels that of the embodiment of Figure 1, the CM database 618 illustrated in Figure 8, largely parallels the CM database 118 of Figure 2, with similarly named tables and fields providing similar functionality and storing similar information unless stated or understood otherwise. However, the present CM database 618 includes variations to accommodate the use of storage servers 606, as well as to provide additional functionality. Such variations and additional functionality will now be described with reference to the database schematic of Figure 8 and continuing reference to the schematic of Figure 6.

Multiple Media Servers Per Storage Server

[00115] As noted above, an item of content stored on a storage server 608 is accessed by each media server 620 that provides an end user 104 access to such content. Furthermore, in the present embodiment, multiple media server 620 may access a given item of content on a storage server 608. Consequently, the CM database 618 tracks which one or more media servers 620 are coupled to, and may access, a given storage server 608. To accommodate this potential one to many relationship, the CM database 618 includes the archive volume-media server ("AVMS") Table 846 and the Media Servers Table 848. Each record in the AVMS Table 846 associates a volume ID with a media server, as identified by media server (MS) ID,

that is permitted to access content stored at the volume ID. Each MS ID, in turn, corresponds to a record in the Media Servers Table 848, which provides media server identifying information, as indicated. The "status" column stores one of the following values: CE (Check Enabled); MD (Manually Disabled); and ME (Manually Enabled).

[00116] As with the prior embodiment, each volume ID is essentially defined in a record in the Archive Volume Table 820. The volume type field in the archive volume table 820 specifies to which type of server (FTP ingest server file manipulation server, storage server, or web server) the volume ID corresponds. The server category field identifies the server as being used for business services or public services or any other category useful for reporting and maintenance purposes. The platform field is used to identify the media platform corresponding to the volume type or to indicate that the platform is not applicable or to indicate all platforms (e.g., a character such as "B").

[00117] The "archive server" field (in Archive Volume Table 820), and the "server name" field (in the Media Servers Table) store the computer's WINS/NETBIOS name for LAN-based file access (for example, PUB01), so that shared resources (such as directories, etc.) on the server can be accessed via that name (for example, \\PUB01\PRoot1).

Account Categories

[00118] The present embodiment has various other features reflected in the CM database 618. For example, as illustrated in the Accounts Table 810, each account is associated with an account category, as identified by account category ID (acct cat ID). Each account category ID corresponds to an entry in the Account Categories Table 850, which sets forth the name of the category. In general, an account category refers to a type of account, such as Internet radio or television or corporate account. The content management service provider may use such information for reporting and, because accounts can be segregated into broad categories, for simplifying the searching of accounts.

Time Zones

[00119] The Accounts Table 810 also includes a time zone ID field (TZ ID), which identifies a client-selectable time zone. Based on the client's selection, and the corresponding time zone ID, all information presented to the client 102 is based on the selected time zone. More specifically, the date and time field stored within the data base 618 are stored in a Greenwich meantime, however prior to being displayed to each client 102, the system converts the time into that of the client-selected time zone. Similarly, dates and times entered by each client 102 is converted to Greenwich Mean Time prior to being stored in the database 618. In the present embodiment, each time zone ID corresponds to a record in a time zone table (not shown), identifying the time zone and providing the relative difference between the Greenwich Mean Time and that time zone. Thus, by way of example, the expiration date, deletion date, creation date and any other time related information is presented in the Client's selected time zone frame.

Default Expiration Interval

[00120] The Accounts Table 810 also includes a default expiration interval field (default exp interval). In the event a client 102 fails to specify an expiration date for a

particular file being uploaded, the system automatically adds the value stored (e.g., minutes hours, days, weeks, and the like) in the default expiration interval field to the date and time that the file is uploaded, thereby arriving at the default expiration date for the file. Such default expiration interval may be set by the client 102 during registration, or at any other point in the process.

Accessors

[00121] The current embodiment also includes tables that specify who may access and alter account information. Specifically, the Account Accessors Table 852 and the Accessors Table 854 specify, for each account ID, one or more individuals (accessors) that may access the account. As illustrated, there may be several records in the account accessors table 852 for each account ID, thereby specifying multiple accessors for each account. Such accessors may be individuals external to the content management system and internal to the client 102, or administrative personnel (internal to the service provider).

Stream Categories

[00122] Unlike the embodiment of Figure 2, the Streams2 Table 814 includes a field specifying each stream's category (category ID). Stream categories are essentially any type or division of content deemed helpful to the content management service provider or the clients 102. Furthermore, categories may be a broad or as focused as desired. In the present embodiment, stream categories are hierarchical in nature, with only the most detailed category being assignable to a stream. For example, the system may provide a first level category of "sports," with a second level of multiple subcategories, such as "baseball," "basketball," "football," and the like. Each subcategory, in turn, includes more detailed third level of subcategories, for example, under basketball, could be "NBA," "WBA" and "College." Only these most detailed categories are assignable in the present embodiment.

[00123] Preferably, each client 102 specifies a default stream category to be applied to uploaded streams that is stored in the Account Table 810 (in the default stream cat ID field). When new content is uploaded via the client web site or programmatically, the client 102 has the option of using the default category or supplying a new category. In either case, the system creates a new record in the Streams2 Table 814 and populates the category ID field accordingly.

[00124] Each category ID corresponds to an entry in the Categories Table 838, which specifies the name of the category, the parent category of the subject category, the type of category (i.e., whether the category is assignable or not) and the level of the category in the category hierarchy. This system retrieves this information and constructs a visual display of the hierarchy on the client-side web page and permits each client to select the desired category. The Categories Table 838 is also keyed to the default category ID in the Account Table 810.

Replication Sites

[00125] The present embodiment also allows each client 102 to have multiple sites – a primary site and one or more replication sites – associated with its account. In general, when uploading content via the client website, the client 102 may specify any one or more of the replication sites to which the content will be uploaded (in addition to the primary site). The client's primary site is identified by the site ID stored in the client's record in the Account Table 810. The replication sites are identified in the Account Sites Table 856. More specifically, the Account Sites Table 856 includes records that associate multiple site IDs to the client's account ID. Of course, there is no requirement that a particular account have any replication sites associated therewith.

[00126] In operation, when a client 102 uploads content, the content gets passed to the storage server 606 associated with the client's primary site. As with the previous embodiment

the stream ID assigned to the content gets returned to the client 102. Thus, in the context of the process of Figure 4b, an additional step 486 would be performed, namely, that if the client 102 is authorized to replicate its content to any site other than their primary site, the client optionally selects one or more replication sites to which the file being uploaded should be replicated.

[00127] In the background, when the client 102 uploads the content and identifies a replication site to which the content should be stored, an entry is made in a replication queue with the information identifying the content and the storage server 606 of the replication site. Periodically, a script checks the replication queue for entries and, for each entry in the queue, causes the appropriate content to be stored in the specified storage server 606 corresponding to the replication site. Thus, in the context of the process of Figure 4b, step 498 would be added to account for the system replicating the files to replication sites if one or more replication sites were chosen by the client while specifying file details. It should be noted that for each storage server 606 containing a copy of the content, there is a record in the Assets Location Table 818, each corresponding to the same stream ID and file name. This stream ID, universal to all copies of the file at the primary and replication sites, simplifies management of the copies, as there is only one entry in the Assets Table 812.

[00128] The ability to replicate content and the lack of a back-up copy of each item of content on a repository server, as in the embodiment of Figure 2, results in certain changes in the Asset Locations Table 818 status field. More specifically, the status is never "backing-up" and, instead, may indicate that the content is: available; being deleted; has expired; failed to be properly stored; being moved; new; being trashed; or being replicated, as described above. Notably, in the present embodiment, there is no need to copy a content file from one storage server 606 to another within the same site because multiple media servers 620 can access the content.

Recycle Bin

[00129] The status of being "trashed" is used in conjunction with a figurative "recycle bin," like that describe above with reference to Figure 3j. When browsing or searching content, a client 102 may select one or more items of content for deletion, or placement in the recycle bin. When selected for deletion, the system changes the value of the status field for the corresponding record in the Asset Locations Table 818 to "trashed." The system also creates a record in the Deleted Assets Table 844 and enters as the deletion date a date (and time) some predetermined period in the future (e.g., one day, one week). A script running on the script processor or other server periodically searches (e.g., every three hours; every day; and the like) for records in the Asset Locations Table 818 having a status of "trashed." For each such record, the system locates the corresponding record in the Deleted Assets Table 844 and reads the deletion date. If the deletion date equals or is greater than (i.e., after the then current date (and time), then the system proceeds to delete the content. Until such time as the content is actually deleted, the client 102 may restore the content from the recycle bin via a web page identifying the contents (or, in alternate embodiments, programmatically). When the client 102 selects an item of content for restoration, the system changes the status in the Asset Locations Table 818 from "trashed" back to "available" and deletes the record in the Deleted Assets Table 844.

Stream Groups

[00130] The present embodiment also allows clients 102 to logically group Streams into stream groups for ease of presentation to end users 104. More specifically, each logical stream group includes content files having the same susceptive content, but different media formats and/or bit rates. When a client 102 uploads content, the client web page presents the client 102 with the option of associating the uploaded content with an existing stream group or with a newly created stream group.

[00131] Stream groups are reflected in the Streams2 Groups Table 858, the Streams2 Group Table 860 and the Streams2 Group Streams Table 862. More specifically, the Streams2 Groups Table 858 contains a record for each Stream Group, as identified by Stream Group ID, associated with a particular account. The Streams2 Group Table 860 provides Stream information corresponding to each Stream Group, as identified by Stream Group ID. Such Stream information includes, for example, the Stream Group name, and the title, author, description, associated key words and type of Stream Group. Lastly, the Streams2 Group Streams Table 862 identifies each Stream comprising a given Stream Group. Additionally, the Table 862 identifies the corresponding bit rate for the particular stream. By characterizing streams into groups, clients are able to easily track a particular piece of content and provide to their end-users 104 the stream having the appropriate media format and bit rate.

[00132] The client-side web site also preferably presents a web page for managing stream groups that parallels the page provided for managing playlists. For example, a stream group may be selected, identifying information, such as name, title, description, keywords and the like maybe edited, and the content of the stream group may be modified by adding or removing content (by adding or deleting, respectively, a record in the Stream2 Group Streams Table 862).

[00133] Alternatively, as with all content management functions, the stream groups may be edited via the client-side web site as described by passing variables in a query string to a script. In one such embodiment, the script requires values for three parameters: username-password; and stream group action, which may indicate: adding a stream to an existing group; creating a new group; editing a group; removing a stream from a group; listing the stream groups; or listing the groups and the contents thereof. Depending upon the action specified, the following optional parameters must be specified: name; title;

description; author; keywords; miscellaneous, metadata of the group; whether promotable; stream group ID (used to specify a group to be edited or removed or to edit streams within a group); and stream action. Stream action, in turn, may indicate: adding a stream; removing a stream; removing all streams; stream ID(s) (specified when adding or removing a particular stream).

[00134] The Stream2 Details Table 870 is related stream groups. Each record in the Stream2 Details Table 870 corresponds to a stream, as specified by stream ID. The information stored for the streams includes: (1) miscellaneous metadata, such as an extension of any other field that was truncated due to length of field restrictions; (2) if the stream supports multi bit rates, a stream group is created containing all bit rate versions of the stream; the stream group ID is stored in the multi bit rate stream group ID field (mbr sg id); and (3) and indication of whether or not the stream supports multiple bit rates (is multi bit rate).

[00135] Collectively, the information is used to identify those streams that support multiple bit rates and to identify the particular stream group (out of potentially many to which the stream belongs) corresponding to the stream's multi bit stream group.

Stream Folders

[00136] The present embodiment also allows clients 104 to organize content into logical folders. The Streams Folders Table 864 and Folder Content Table 866 permit this. The Streams Folders Table 864 identifies folders, by folder ID, associated with each account ID. Each record also specifies the folder name and the ID of the parent folder, as folders may be hierarchical. The Folder Content Table 866 identifies the content within each folder, as identified by folder ID. More specifically, the present embodiment allows streams, stream groups, and playlists to be included within a folder. Consequently, the folder content Table 866 is designed to specify any of the foregoing types of content. To this end, the folder

content Table 866 includes a content type field, which specifies to which of the foregoing types of content the record pertains. Each record also includes a content ID field which specifies the ID of the content to which the record pertains. Thus, if the content type specifies playlists, the content ID field will include the appropriate CG ID; if the content file specifies Stream Group, the content ID field specifies the Stream Group ID; if the content type specifies Stream, then the content ID field includes the Stream ID.

[00137] Preferably, the client web page includes a separate page that provides the client 102 the opportunity to both create folders and edit existing folders. To create a folder, the client 102 simply needs to provide the folder name and any parent folder to which the new folder relates and to select content comprising the folder. In essence, for each new folder created, a new record is created in a Streams Folders Table 864, and for each content added to the folder, a new record is created in a Folder Content Table 866. When editing an existing folder, the client web page preferably presents to the client 102 a listing of folders, as identified in the Streams Folders Table 864, and upon the client's selecting a folder, the system retrieves the details corresponding to the content ID stored in a Folder Content Table 866 and displays those details to the clients 102.

Promotability

[00138] The present embodiment also provides the clients 102 with the option of marking content as promotable. By marking content promotable, the client 102 essentially authorizes the content management service provider to select the content and place it on any web site operated by the service provider. For example, if a client 102 provides news clippings, it may mark the content promotable, thereby authorizing the service provider to provide a link to the content on the service provider's news page. As illustrated in the CM database 618, Stream Groups may be identified as promotable in the Streams2 Groups Table 858 and folders may be identified as promotable in the Streams Folders Table 864. Clients

102 may indicate that a particular content is promotable when up-loading the content, or creating or editing a folder and stream group. In alternate embodiments, individual streams may be specified as promotable or not by including a promotable field in either of the Assets Table 812 or Streams2 Table 814.

[00139] The Database 618 also includes an FTP Info Table 868 for storing the site ID for each FTP Ingest Server 608, as identified by hostname. The Table 868 also identifies the username and password needed to access each FTP Ingest Server 608.

Playlist Resolution

[00140] Because the present embodiment does not store streams on the media servers 620 and because clients 102 can have replication sites, the processes of creating the playlist URL and creating the redirector file is different than described with regard to the embodiment of Figure 1. The differences will now be described with reference to Figures 6 and 8.

[00141] In the event a client 102 has one or more replication sites and the content to be presented is stored at multiple sites, the client preferably selects the content and the site from which the client desires it to be streamed. For example, a client may have a Japanese language stream; if stored on a site in Japan and a site in the United States, the client may desire it be streamed from the site in Japan. Consequently, the playlist URL of the present embodiment includes and indication of not only the content (e.g., stream ID, CG ID), but also an indication of the location (e.g., site ID) from where the content should be streamed. An exemplary playlist URL, in which the playlist server 622 is "playlist.company.com", the program to dynamically generate the redirector file is "makeplaylist.dll", the CG ID is "12345" and the site ID is "16", is as follows:

<href="http://playlist.company.com/makeplaylist.dll?CG_ID=12345&SITE_ID=16">

[00142] The playlist URL is constructed generally as described above. However, when the playlist is created, the system preferably provides the client 102 with a series of

URLs, each corresponding to a different site at which the content is stored, and specifies the site name and time zone, as retrieved from the Sites Table 834. Alternatively, the client manually enters the site ID.

[00143] As with the previous embodiment, the makeplaylist component dynamically generates a redirector file based on the CG ID and, in the present embodiment, the site ID. More specifically, the component causes the playlist server 622 to access the Content Group Streams Table 826 and identify the streams within the identified playlist and their sort order. For each such stream, the server accesses the Streams2 Table 814 and retrieves the URL prefix and filename. As described in greater detail below, because the content is stored on the storage servers 606, and not the media servers 620, the filename refers each stream's filename as stored on the storage server 606.

[00144] The playlist server 622 also accesses the Stream-Servers Table 830 to identify the hostname of the server on which each stream resides. It should be noted that because the client has a replication site with the stream, there will be multiple records in the Stream-Servers Table 830 for each stream ID. Consequently, the playlist server 622 must determine which record corresponds to the site identified in the playlist URL. The server does this by accessing the Servers2 Table 832 and identifying the record having both one of the hostnames previously identified from the Stream-Servers Table 830 and the site ID from the URL. The hostname from this record is used in creating the redirector file. The system does this for each stream in the playlist.

[00145] In certain embodiment, a site will include multiple streaming media servers 622 capable of streaming a given content file. In other words, multiple media servers 622 can be coupled to a single storage server 606. In such an instance, the system must determine which of the media servers 620 to use. This many-to-one relationship will be reflected by multiple records in the Servers2 Table 832 having the appropriate site ID. In one

embodiment, the decision is random or pseudorandom. However, in the present embodiment, the decision is based on load balancing and the weight field in the Servers2 Table 832.

[00146] The weight field is a preset indication of the percentage of streams to be served from a given media server 620, as compared to other media servers 620 capable of streaming the same content. For example, if two media server 620 were coupled to a given storage server 606 and could stream a file, and one could serve twice the streams of the other, then the weight of the higher capacity server would be approximately 66 and the weight of the lower capacity server would be approximately 33. When determining from which server to stream, the system would analyze the ratio of current streams being served of the two servers and utilize the server streaming a percentage of the total streams less than its weight. In other words, the system would seek to maintain the ratio of streams served equal to the ratio of the weights.

[00147] Once the media server hostname is determined, the component on the playlist server 620 obtains the filepath prefix from the record in the Stream-Servers Table 830 corresponding to the relevant stream id and hostname. As will be described in greater detail, the filepath prefix serves as a mounting point for the media server 620 so that the media server 620 can access the storage server 606 to read the requested stream. This filepath prefix is appended to the beginning of the filename and placed in the redirector file.

[00148] An exemplary redirector file, in which the playlist comprises a single stream having a stream ID of "2000" and a filename of "/test/10/2000.asf", the media server 620 hostname is "mediaserver.company.com", and the mounting point is "proot1/PubShare01", is as follows:

```
<ASX>
   <ENTRY>
      <REF
HREF="mms://mediaserver.company.com/proot1/PubShare01/test/10/2000.asf">
   </ENTRY>
</ASX>
```

[00149] It should be noted that in the present embodiment, the mount point is the same as the root share and share information for the server volume ID. Therefore, the media server 620 may read the correct contents of the storage server 606. Placing the filepath prefix in the redirector file is useful in situations where files stored on a storage server 606 are not allowed to be stored in one single share, but multiple shares are preferably created for the sake of efficiency of the backup and recovery processes (e.g., mirroring a portion of all shares). In this scenario, the hard drive(s) storage space on a storage server 606 is required to be stored under one of those shares.

[00150] By way of illustrative example, the Archive Volume Table 820 currently has the following record for a share created on the PUB01 storage server 606. The record in the Archive Volume Table 820 for that filer/share combination is as follows: cm_vol_id=142; volume_type=S; archive_server=\\PUB01; share=/PubShare01; server_addr=pub01.broadcast.com; platform=B; max_content_kb=131072000; status=CE; site_id=16; local_path=NULL; checking_status= [not used]; server_category=PUB; root_share=/proot1; attach_type=N; total_kb_used=90455873.

[00151] This would cause a record to be created in the Stream –Servers Table 830 having a filepath prefix of /proot1/PubShare01.

[00152] Let's say that the file path stored in the filename column in streams2 table was '/test/10/2000.asf'. Assuming that the stream ID 2000 was a Windows Media stream that is accessed with the mms protocol, then playlist will construct the final URL as follows:

mms://mediaserver.company.com/proot1/pubshare01/test/10/2000.asf

[00153] In this scenario, for this URL to work correctly, the streaming server must have a mount point with the same name as the first component of the URL after the server hostname. In the above case example, the server name is mediaserver.company.com, and the first component of the URL after that hostname is "proot1." So, a mount point having the name proot1 is created on the streaming server such that it points to the following LAN file storage path: \\PUB01\proot1

[00154] A root share is optional, and is used only to reduce the number of mount points needed to be created on a streaming server in a situation whereby the storage device or hard drive on a storage server 606 has more than one share on it (for such reasons as efficiency of the backup processes).

[00155] A storage share on a storage server, however, is required because the systems will store a file in that share (further organized in account-specific directory, such as 'test' in the above example, and a hashed directory such as '10' in the above example). In the above example, the storage server 606 \\PUB01 has the following shares: proot1 and pubshare01.

[00156] Those skilled in the art will recognize that the method and system of the present invention has many applications, may be implemented in many manners and, as such, is not to be limited by the foregoing exemplary embodiments and examples. In this regard, any number of the features of the different embodiments described herein may be combined into one single embodiment. Moreover, the scope of the present invention covers conventionally known and future developed variations and modifications to the system components described herein, as would be understood by those skilled in the art.

CLAIMS

1. A method of providing access to digital content, the method comprising:

receiving digital content from a client;

storing the digital content on a server having a hostname, the digital content having a filename;

assigning a unique identifier to the digital content;

providing the client with the link containing the unique identifier;

receiving a request for the content, the request based on activation of the link;

determining the hostname and filename based on the unique identifier; and

creating a redirector file, the redirector file including the hostname and filename.

2. The method of claim 1, wherein the unique identifier is a stream identifier.

3. The method of claim 1, wherein the unique identifier is the name of the digital content.

4. The method of claim 1, wherein the digital content is a streaming media file.

5. The method of claim 1, wherein the digital content is a playlist comprising multiple streaming media files.

6. The method of claim 1 wherein the link specifies a program for determining the hostname and filename based on the unique identifier.

7. A method of providing an end user access to digital content, the method comprising:

causing digital content to be stored on a server having a hostname, the digital content having a filename when stored;

receiving a unique identifier to the digital content;

publishing a link for activation by the end user, the link including the unique identifier of the digital content, wherein activation of the link causes resolution of the unique identifier into the hostname and filename and causes the digital content to be provided to the end user based on the hostname and filename.

8. The method of claim 7, wherein the unique identifier is a stream identifier.

9. The method of claim 7, wherein the unique identifier is the name of the digital content.

10. The method of claim 7, wherein the digital content is a streaming media file.

11. The method of claim 7, wherein the digital content is a playlist comprising multiple streaming media files.

12. A system for managing digital content and provide digital content to end users, the system comprising:

one or more first servers configured to receive digital content;

one or more storage servers configured to store digital content received by the first servers;

a plurality of media servers coupled to at least one of the storage servers, the media servers configured to receive a request to experience an item of digital content and, in response to the request, read the item of digital content stored on the at least one storage server.

13. The system of claim 12, wherein the first servers include a web server.

14. The system of claim 12, wherein the first servers include an FTP ingest server.

15. The system of claim 12, wherein the first servers are the storage servers.

16. The system of claim 12 wherein the request is from an end user and the media servers are further configured to communicate the item of content to the end user.

17. The system of claim 12 further comprising a playlist server configured to resolve the request and select one of the media servers to read the item of digital content.

18. The system of claim 17 wherein each media server has a mounting point associated therewith, the mounting point referring to the at least one storage server.

19. A method of managing digital content of a client, the method comprising:

associating the client with a primary site;

associating the client with one or more replication sites;

providing the client an option to upload digital content to both the primary site and any one or more of the replication sites;

receiving a request to upload an item of digital content, the request specifying one or more replication sites; and

based on the request, uploading the item of digital content to the primary site and the specified replication sites.

20. The method of claim 19 further comprising associating the item of content with a universal identifier identifying the item of content as uploaded on the primary site and the specified replication sites.

21. The method of claim 19, wherein the associating the client with replication sites is performed in response to the client specifying replication sites.

22. The method of claim 19 further comprising:

receiving a request to provide the uploaded content; and

providing the uploaded content from the primary site or the one or more specified replication sites based on client input.

23. A method of managing digital content of a client, the method comprising:

receiving a selection of a time zone from the client;

associating the client with the time zone;

storing time related information for the client based on a standard time;

converting the stored time related information based on a standard time to time related information based on the time zone associated with the client; and

providing to the client the time related information based on the time zone.

24. The method of claim 23 wherein the universal time is Greenwich Mean Time.

25. The method of claim 23 wherein the converting includes adding or subtracting a number of hours.

26. The method of claim 23 wherein the time related information is a date and time.

27. The method of claim 23 further comprising:

receiving from the client time related information based on the time zone;

converting the time related information based on the time zone into time related information based on the standard time; and

storing the time related information based on the standard time.

28. A method of providing digital content, the method comprising:

generating a playlist uniform resource locator (URL) identifying digital content and a one of a plurality of sites having a copy of the digital content;

identifying a hostname associated with the digital content and the one site, the hostname identifying a media server associated with the one site; and

providing the digital content from the media server.

29. The method of claim 28 wherein generating the playlist URL includes receiving from a client providing the digital content a selection of the one site from the plurality of sites.

30. The method of claim 28 wherein the one site includes a storage server and multiple media servers and wherein the digital content is stored on the storage server, the identifying hostname includes selecting the media server from the multiple media servers.

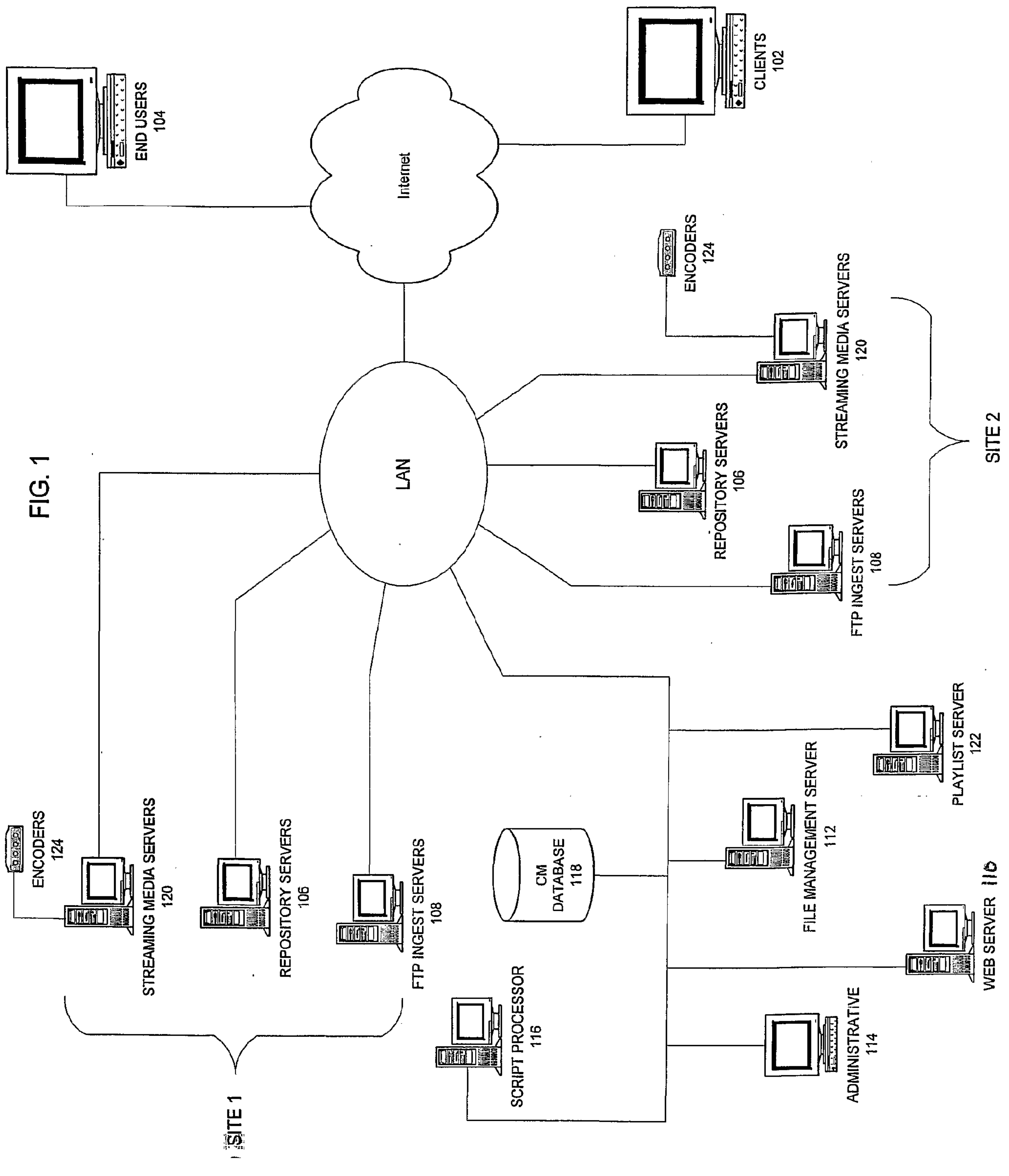
31. The method of claim 28 wherein selecting one of the multiple media servers is random.

32. The method of claim 28 wherein selecting one of the multiple media servers is based on relative load on the multiple media servers.

33. The method of claim 28 wherein identifying the hostname includes selecting the media server from multiple media servers.

34. The method of claim 28 wherein the digital content is a playlist comprising multiple streams and wherein the identifying the hostname is performed by each of the multiple streams.

FIG. 1


END USERS 104
Internet
CLIENTS 102
ENCODERS 124
STREAMING MEDIA SERVERS 120
REPOSITORY SERVERS 106
FTP INGEST SERVERS 108
SITE 2
LAN
ENCODERS 124
STREAMING MEDIA SERVERS 120
REPOSITORY SERVERS 106
FTP INGEST SERVERS 108
SITE 1
CM DATABASE 118
SCRIPT PROCESSOR 116
PLAYLIST SERVER 122
FILE MANAGEMENT SERVER 112
WEB SERVER 116
ADMINISTRATIVE 114

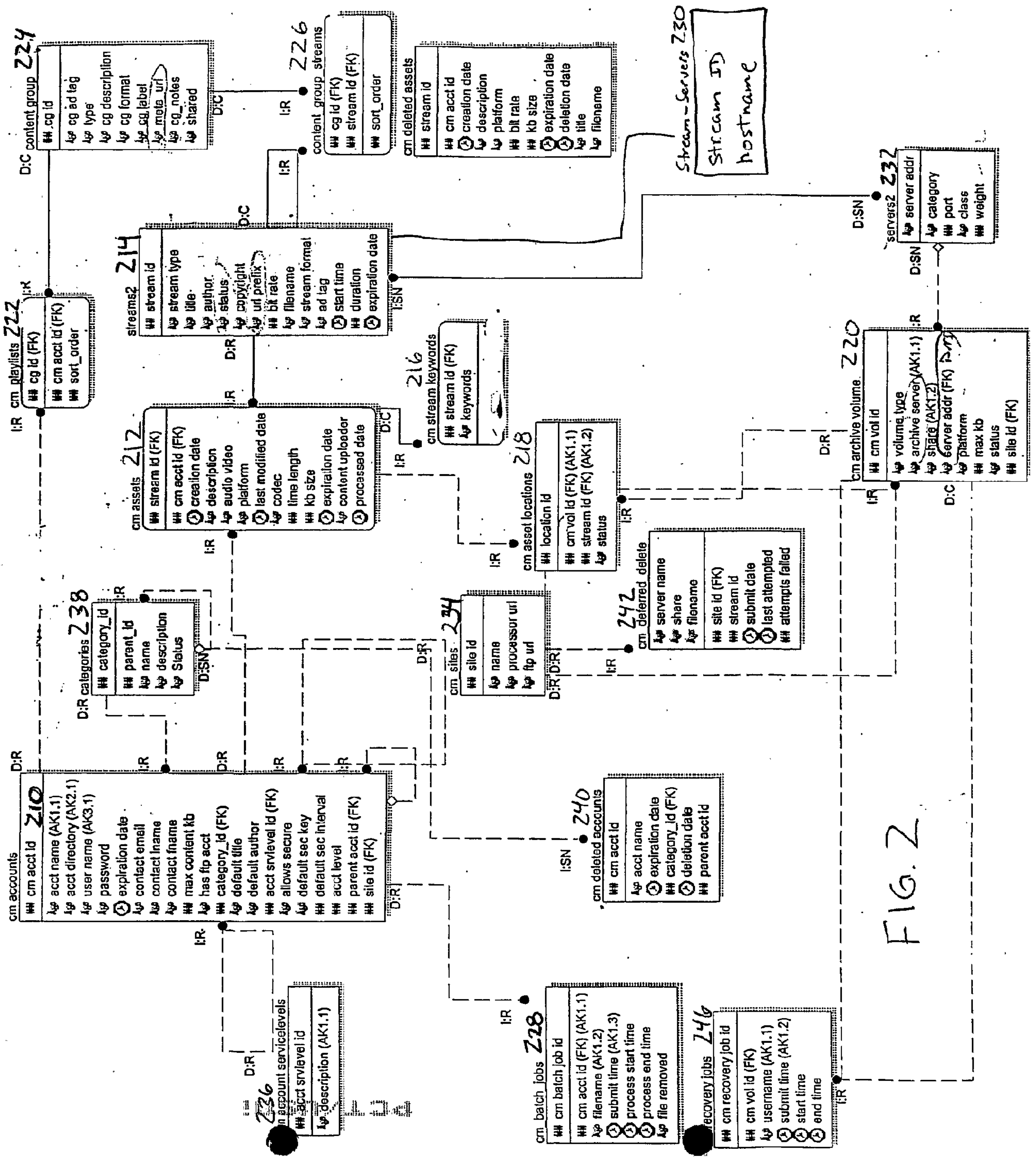
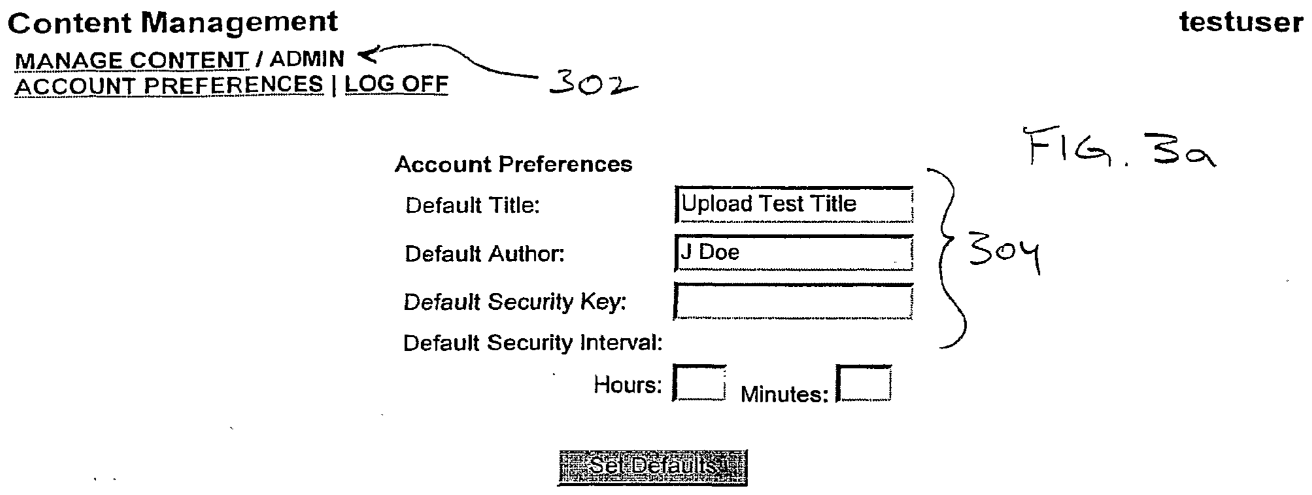

content group 224
cg id
cg ad tag
type
cg description
cg format
cg label
meta url
cg notes
shared
content group streams 226
cg id (FK)
stream id (FK)
sort order
cm deleted assets
stream id
cm acct id
creation date
description
platform
bit rate
kb size
expiration date
deletion date
title
filename
Stream-Servers 230
stream ID
hostname
servers2 232
server addr
category
port
class
weight
streams2 214
stream id
stream type
title
author
status
copyright
url prefix
bit rate
filename
stream format
ad tag
start time
duration
expiration date
cm playlists 222
cg id (FK)
cm acct id (FK)
sort order
cm stream keywords 216
stream id (FK)
keywords
cm assets 212
stream id (FK)
cm acct id (FK)
creation date
description
audio video
platform
last modified date
codec
time length
kb size
expiration date
content uploader
processed date
cm asset locations 218
location id
cm vol id (FK) (AK1.1)
stream id (FK) (AK1.2)
status
cm archive volume 220
cm vol id
volume type
archive server (AK1.1)
share (AK1.2)
server addr (FK)
platform
max kb
status
site id (FK)
categories 238
category id
parent id
name
description
Status
cm sites 234
site id
name
processor url
ftp url
cm deferred delete 242
server name
share
filename
site id (FK)
stream id
submit date
last attempted
attempts failed
cm accounts 210
cm acct id
acct name (AK1.1)
acct directory (AK2.1)
user name (AK3.1)
password
expiration date
contact email
contact lname
contact fname
max content kb
has ftp acct
category id (FK)
default title
default author
acct srvlevel id (FK)
allows secure
default sec key
default sec interval
acct level
parent acct id (FK)
site id (FK)
cm deleted accounts 240
cm acct id
acct name
expiration date
category id (FK)
deletion date
parent acct id
cm account servicelevels 236
acct srvlevel id
description (AK1.1)
cm batch jobs 228
cm batch job id
cm acct id (FK) (AK1.1)
filename (AK1.2)
submit time (AK1.3)
process start time
process end time
file removed
recovery jobs 246
cm recovery job id
cm vol id (FK)
username (AK1.1)
submit time (AK1.2)
start time
end time
D:C
I:R
D:R
I:SN
D:SN

FIG. 2



Content Management testuser

MANAGE CONTENT / ADMIN
ACCOUNT PREFERENCES | LOG OFF

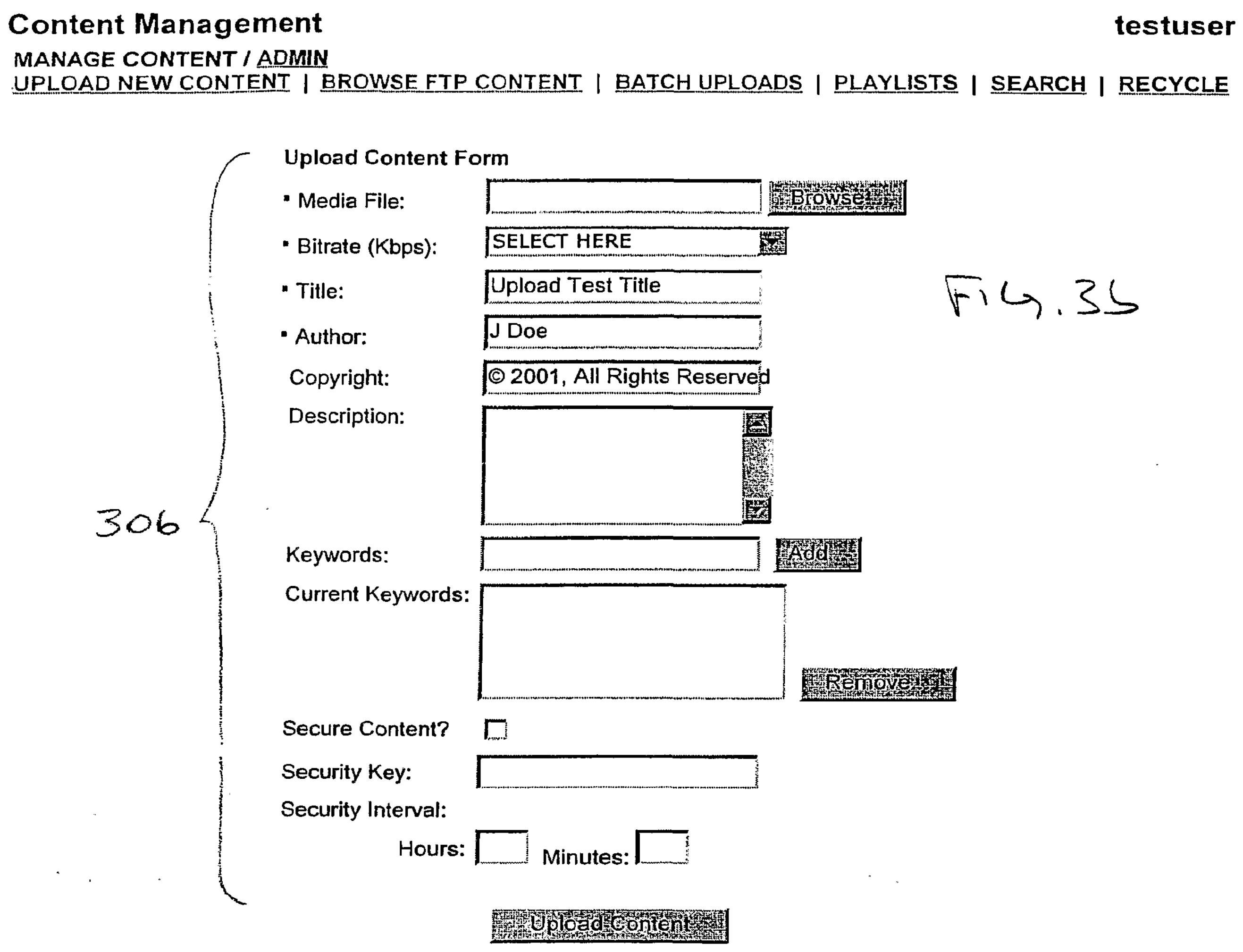
302

FIG. 3a

Account Preferences

Default Title:	Upload Test Title
Default Author:	J Doe
Default Security Key:	
Default Security Interval:	
Hours:	Minutes:

304



Content Management **testuser**

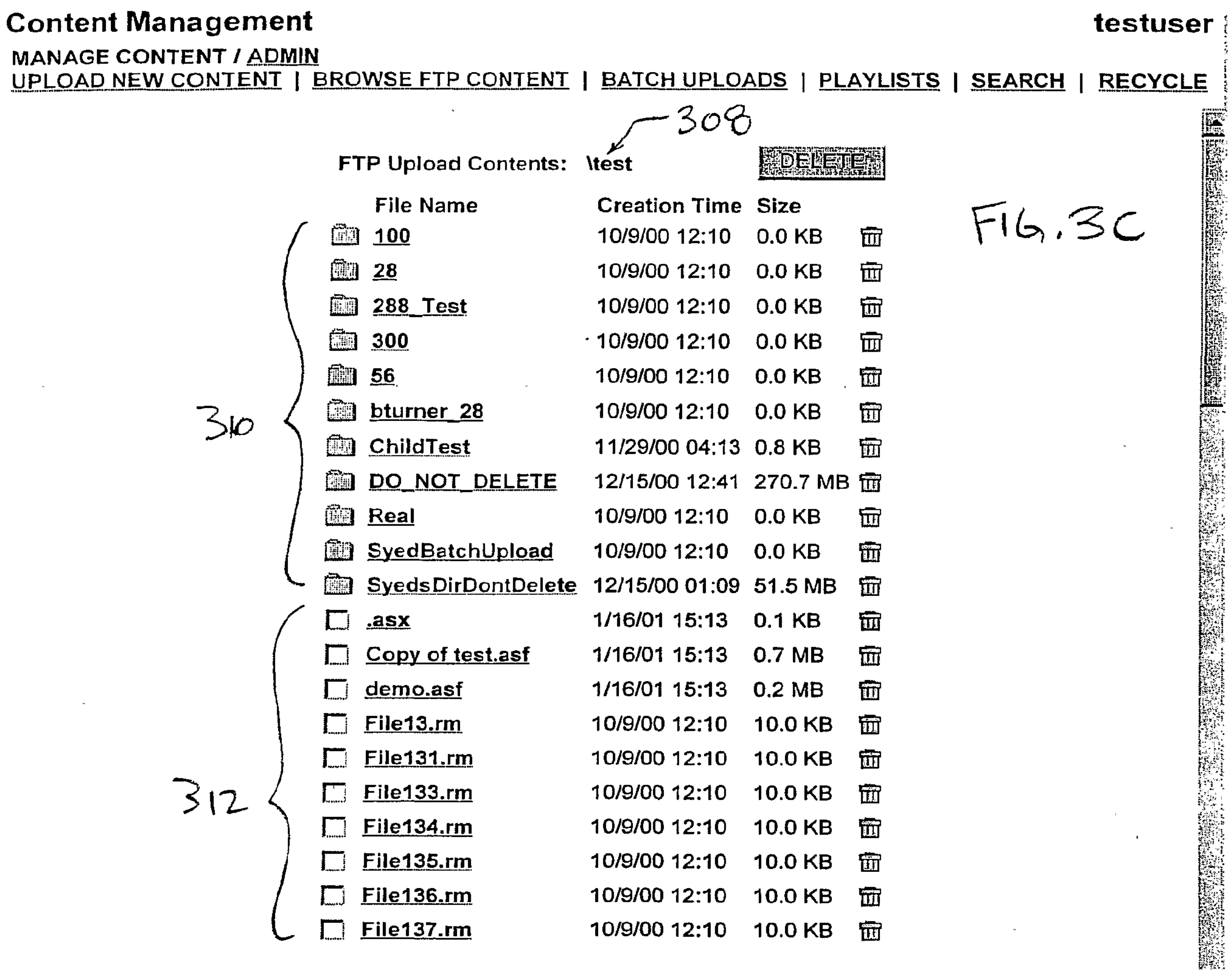
MANAGE CONTENT / ADMIN

UPLOAD NEW CONTENT | BROWSE FTP CONTENT | BATCH UPLOADS | PLAYLISTS | SEARCH | RECYCLE

Upload Content Form

• Media File:	
• Bitrate (Kbps):	SELECT HERE
• Title:	Upload Test Title
• Author:	J Doe
Copyright:	© 2001, All Rights Reserved
Description:	
Keywords:	
Current Keywords:	
Secure Content?	☐
Security Key:	
Security Interval:	Hours: Minutes:

FIG. 3b



Content Management testuser

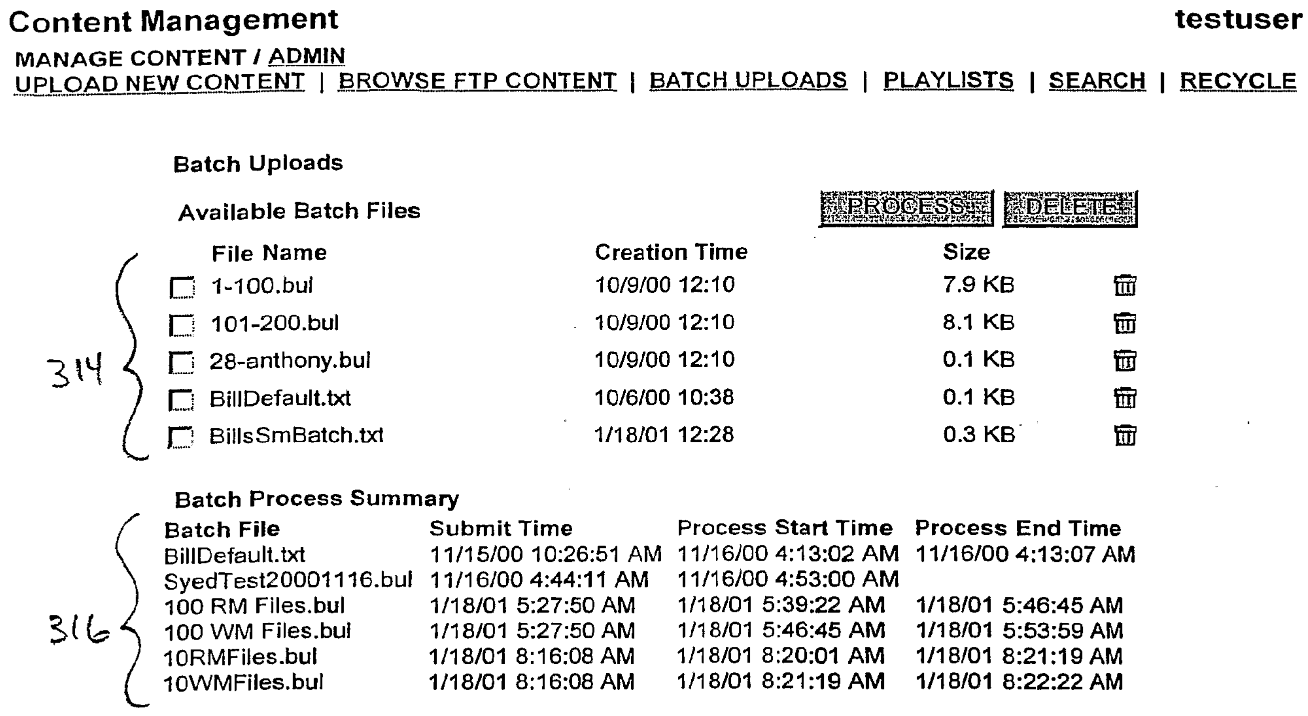
MANAGE CONTENT / ADMIN

UPLOAD NEW CONTENT | BROWSE FTP CONTENT | BATCH UPLOADS | PLAYLISTS | SEARCH | RECYCLE

FTP Upload Contents: \test DELETE

File Name	Creation Time	Size
100	10/9/00 12:10	0.0 KB
28	10/9/00 12:10	0.0 KB
288_Test	10/9/00 12:10	0.0 KB
300	10/9/00 12:10	0.0 KB
56	10/9/00 12:10	0.0 KB
bturner_28	10/9/00 12:10	0.0 KB
ChildTest	11/29/00 04:13	0.8 KB
DO_NOT_DELETE	12/15/00 12:41	270.7 MB
Real	10/9/00 12:10	0.0 KB
SyedBatchUpload	10/9/00 12:10	0.0 KB
SyedsDirDontDelete	12/15/00 01:09	51.5 MB
.asx	1/16/01 15:13	0.1 KB
Copy of test.asf	1/16/01 15:13	0.7 MB
demo.asf	1/16/01 15:13	0.2 MB
File13.rm	10/9/00 12:10	10.0 KB
File131.rm	10/9/00 12:10	10.0 KB
File133.rm	10/9/00 12:10	10.0 KB
File134.rm	10/9/00 12:10	10.0 KB
File135.rm	10/9/00 12:10	10.0 KB
File136.rm	10/9/00 12:10	10.0 KB
File137.rm	10/9/00 12:10	10.0 KB

FIG. 3C



Content Management testuser

MANAGE CONTENT / ADMIN

UPLOAD NEW CONTENT | BROWSE FTP CONTENT | BATCH UPLOADS | PLAYLISTS | SEARCH | RECYCLE

Batch Uploads

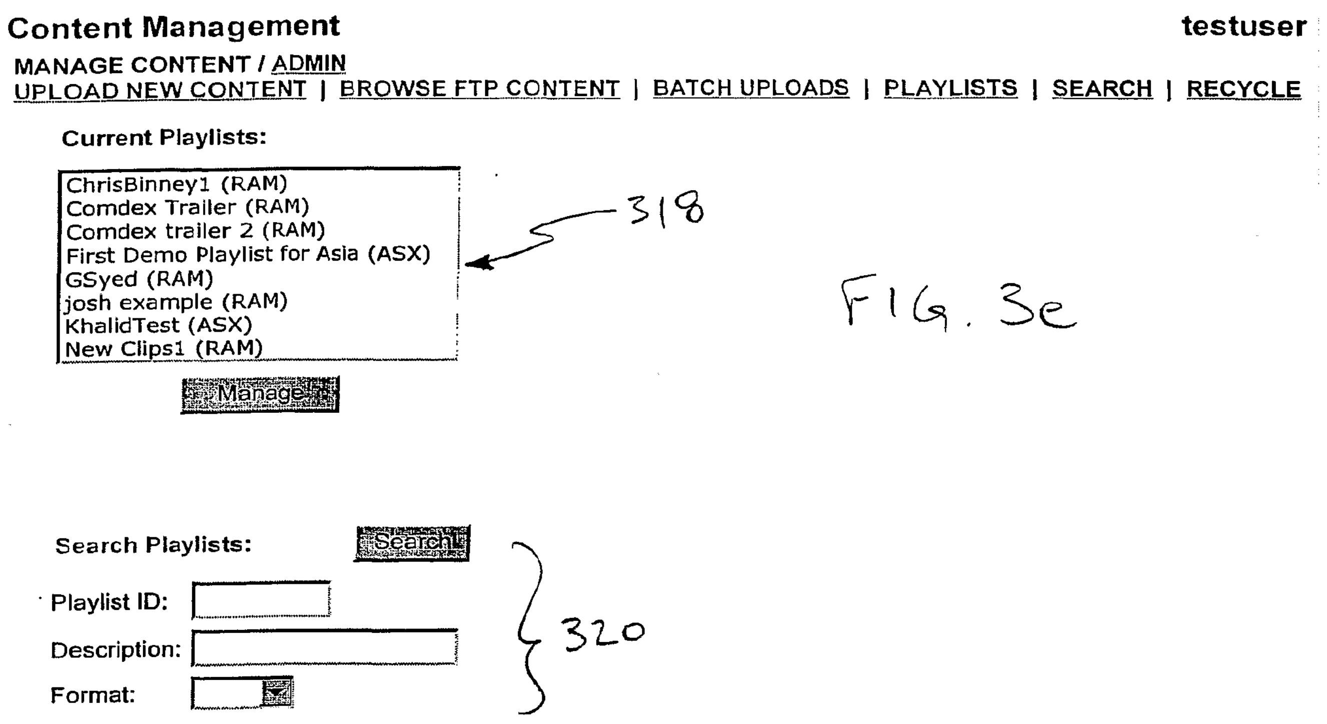
Available Batch Files

File Name	Creation Time	Size
☐ 1-100.bul	10/9/00 12:10	7.9 KB
☐ 101-200.bul	10/9/00 12:10	8.1 KB
☐ 28-anthony.bul	10/9/00 12:10	0.1 KB
☐ BillDefault.txt	10/6/00 10:38	0.1 KB
☐ BillsSmBatch.txt	1/18/01 12:28	0.3 KB

Batch Process Summary

Batch File	Submit Time	Process Start Time	Process End Time
BillDefault.txt	11/15/00 10:26:51 AM	11/16/00 4:13:02 AM	11/16/00 4:13:07 AM
SyedTest20001116.bul	11/16/00 4:44:11 AM	11/16/00 4:53:00 AM	
100 RM Files.bul	1/18/01 5:27:50 AM	1/18/01 5:39:22 AM	1/18/01 5:46:45 AM
100 WM Files.bul	1/18/01 5:27:50 AM	1/18/01 5:46:45 AM	1/18/01 5:53:59 AM
10RMFiles.bul	1/18/01 8:16:08 AM	1/18/01 8:20:01 AM	1/18/01 8:21:19 AM
10WMFiles.bul	1/18/01 8:16:08 AM	1/18/01 8:21:19 AM	1/18/01 8:22:22 AM

FIG. 3d


Content Management testuser

MANAGE CONTENT / ADMIN

UPLOAD NEW CONTENT | BROWSE FTP CONTENT | BATCH UPLOADS | PLAYLISTS | SEARCH | RECYCLE

Current Playlists:

ChrisBinney1 (RAM)
Comdex Trailer (RAM)
Comdex trailer 2 (RAM)
First Demo Playlist for Asia (ASX)
GSyed (RAM)
josh example (RAM)
KhalidTest (ASX)
New Clips1 (RAM)

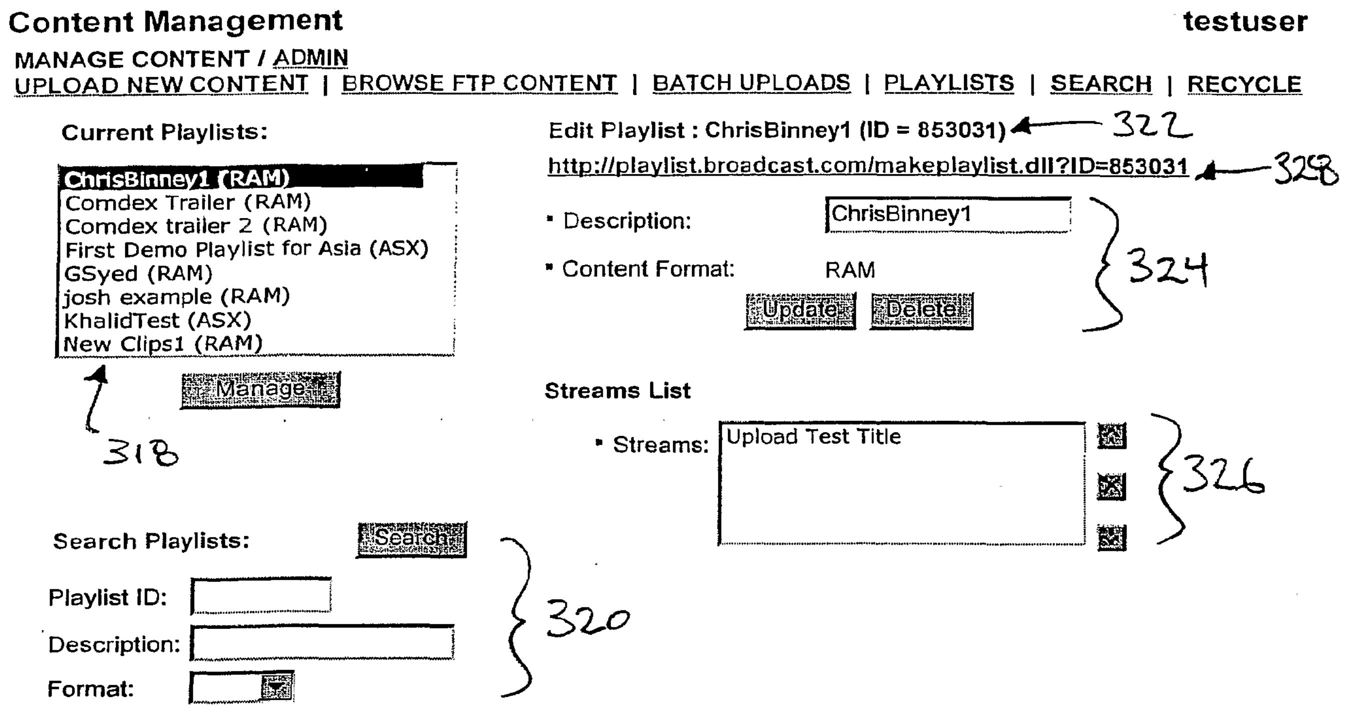
Search Playlists:

Playlist ID:

Description:

Format:

FIG. 3e



Content Management testuser

MANAGE CONTENT / ADMIN

UPLOAD NEW CONTENT | BROWSE FTP CONTENT | BATCH UPLOADS | PLAYLISTS | SEARCH | RECYCLE

Current Playlists:

- ChrisBinney1 (RAM)
- Comdex Trailer (RAM)
- Comdex trailer 2 (RAM)
- First Demo Playlist for Asia (ASX)
- GSyed (RAM)
- josh example (RAM)
- KhalidTest (ASX)
- New Clips1 (RAM)

Edit Playlist : ChrisBinney1 (ID = 853031)

http://playlist.broadcast.com/makeplaylist.dll?ID=853031

- Description: ChrisBinney1
- Content Format: RAM

Streams List

- Streams: Upload Test Title

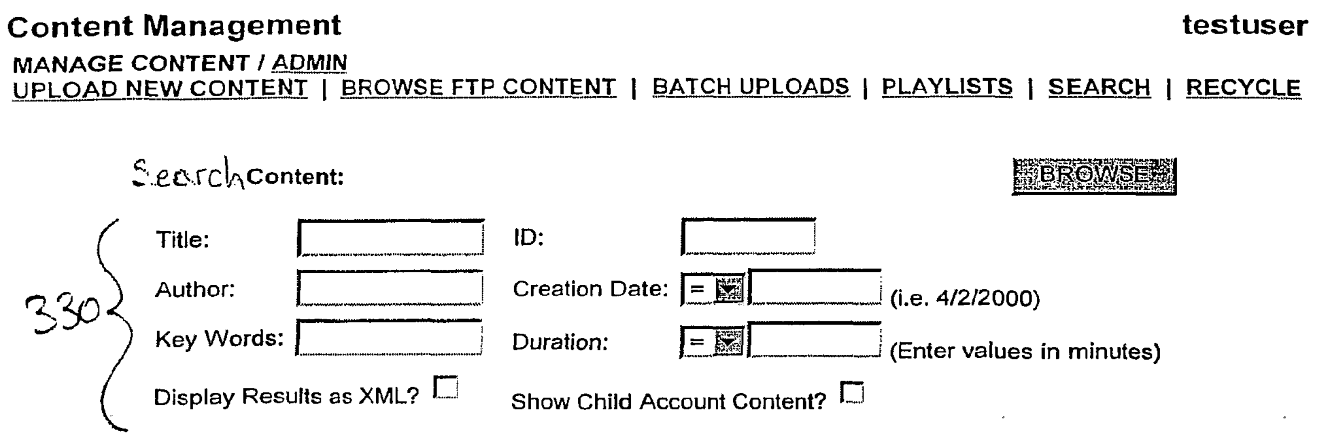
Search Playlists:

Playlist ID:

Description:

Format:

FIG. 3f


Content Management testuser

MANAGE CONTENT / ADMIN

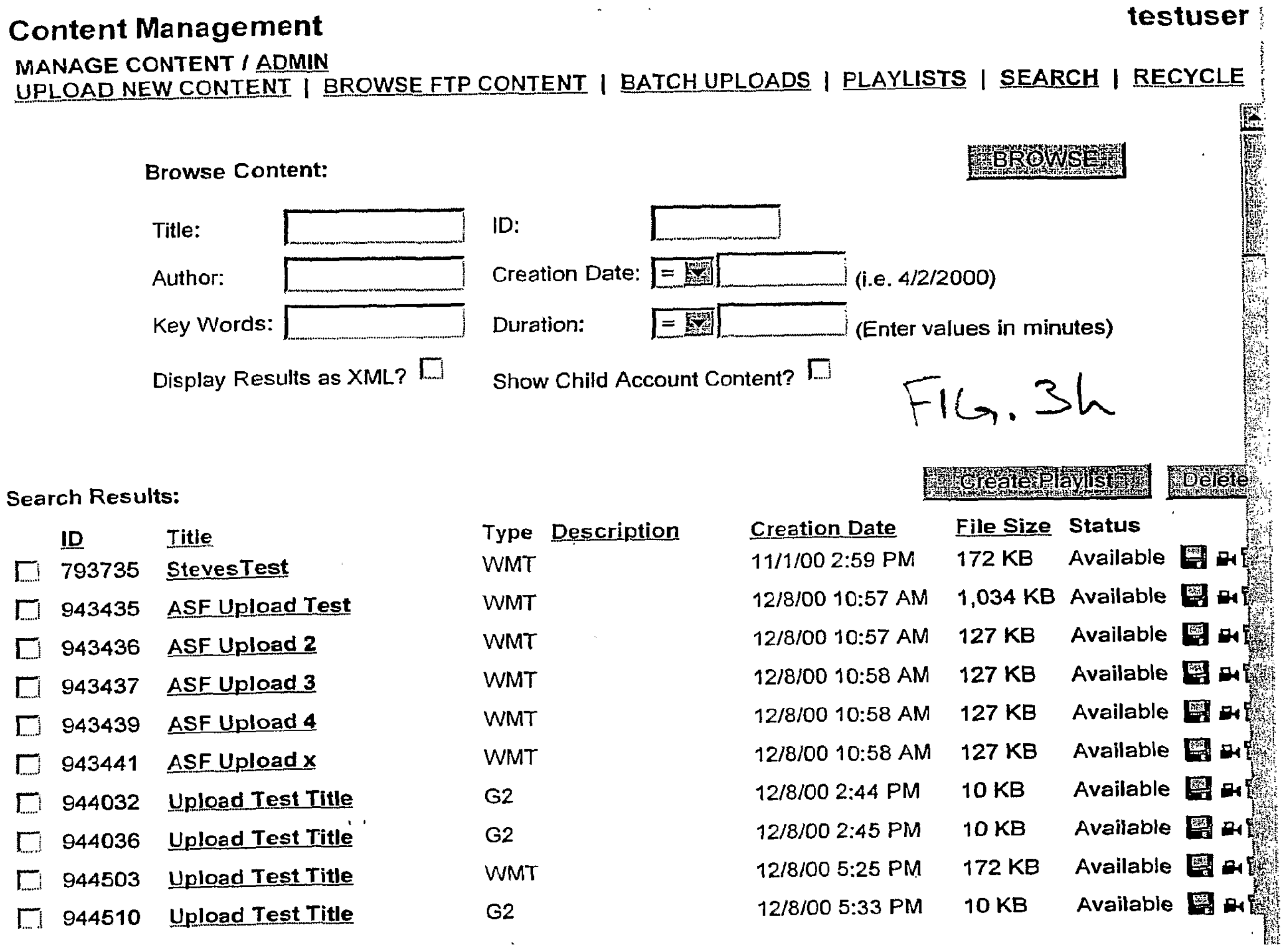
UPLOAD NEW CONTENT | BROWSE FTP CONTENT | BATCH UPLOADS | PLAYLISTS | SEARCH | RECYCLE

Search Content:

330

Title:		ID:	
Author:		Creation Date:	= (i.e. 4/2/2000)
Key Words:		Duration:	= (Enter values in minutes)
Display Results as XML?	☐	Show Child Account Content?	☐

FIG. 3g



Content Management

testuser

MANAGE CONTENT / ADMIN

UPLOAD NEW CONTENT | BROWSE FTP CONTENT | BATCH UPLOADS | PLAYLISTS | SEARCH | RECYCLE

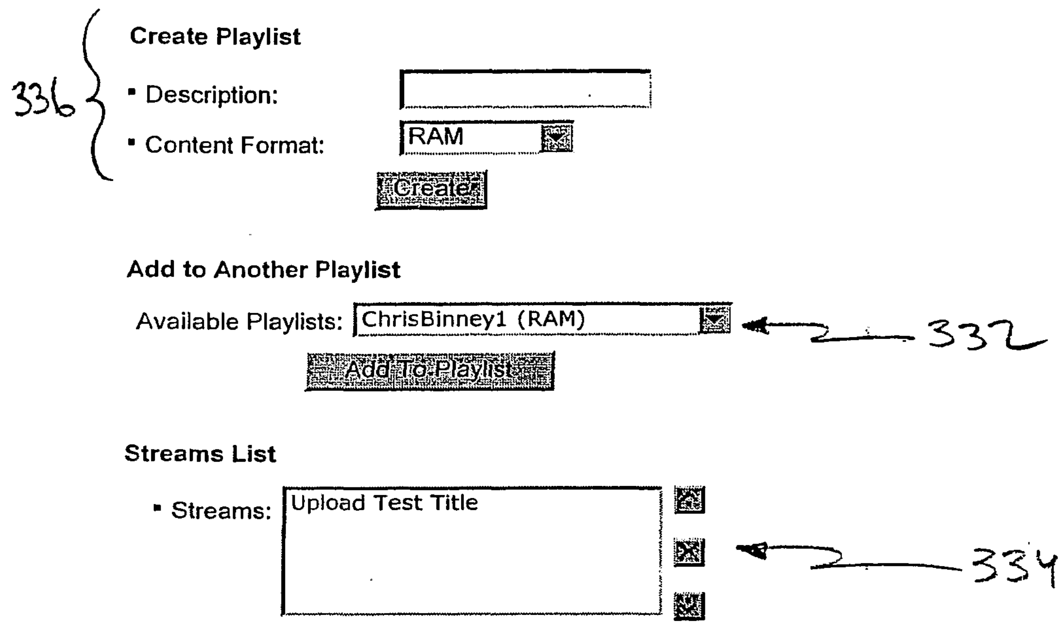
Browse Content:

Title:		ID:	
Author:		Creation Date:	= (i.e. 4/2/2000)
Key Words:		Duration:	= (Enter values in minutes)
Display Results as XML? ☐		Show Child Account Content? ☐	

FIG. 3h

Search Results:

	ID	Title	Type	Description	Creation Date	File Size	Status
☐	793735	StevesTest	WMT		11/1/00 2:59 PM	172 KB	Available
☐	943435	ASF Upload Test	WMT		12/8/00 10:57 AM	1,034 KB	Available
☐	943436	ASF Upload 2	WMT		12/8/00 10:57 AM	127 KB	Available
☐	943437	ASF Upload 3	WMT		12/8/00 10:58 AM	127 KB	Available
☐	943439	ASF Upload 4	WMT		12/8/00 10:58 AM	127 KB	Available
☐	943441	ASF Upload x	WMT		12/8/00 10:58 AM	127 KB	Available
☐	944032	Upload Test Title	G2		12/8/00 2:44 PM	10 KB	Available
☐	944036	Upload Test Title	G2		12/8/00 2:45 PM	10 KB	Available
☐	944503	Upload Test Title	WMT		12/8/00 5:25 PM	172 KB	Available
☐	944510	Upload Test Title	G2		12/8/00 5:33 PM	10 KB	Available



Create Playlist

- Description:
- Content Format: RAM

Create

Add to Another Playlist

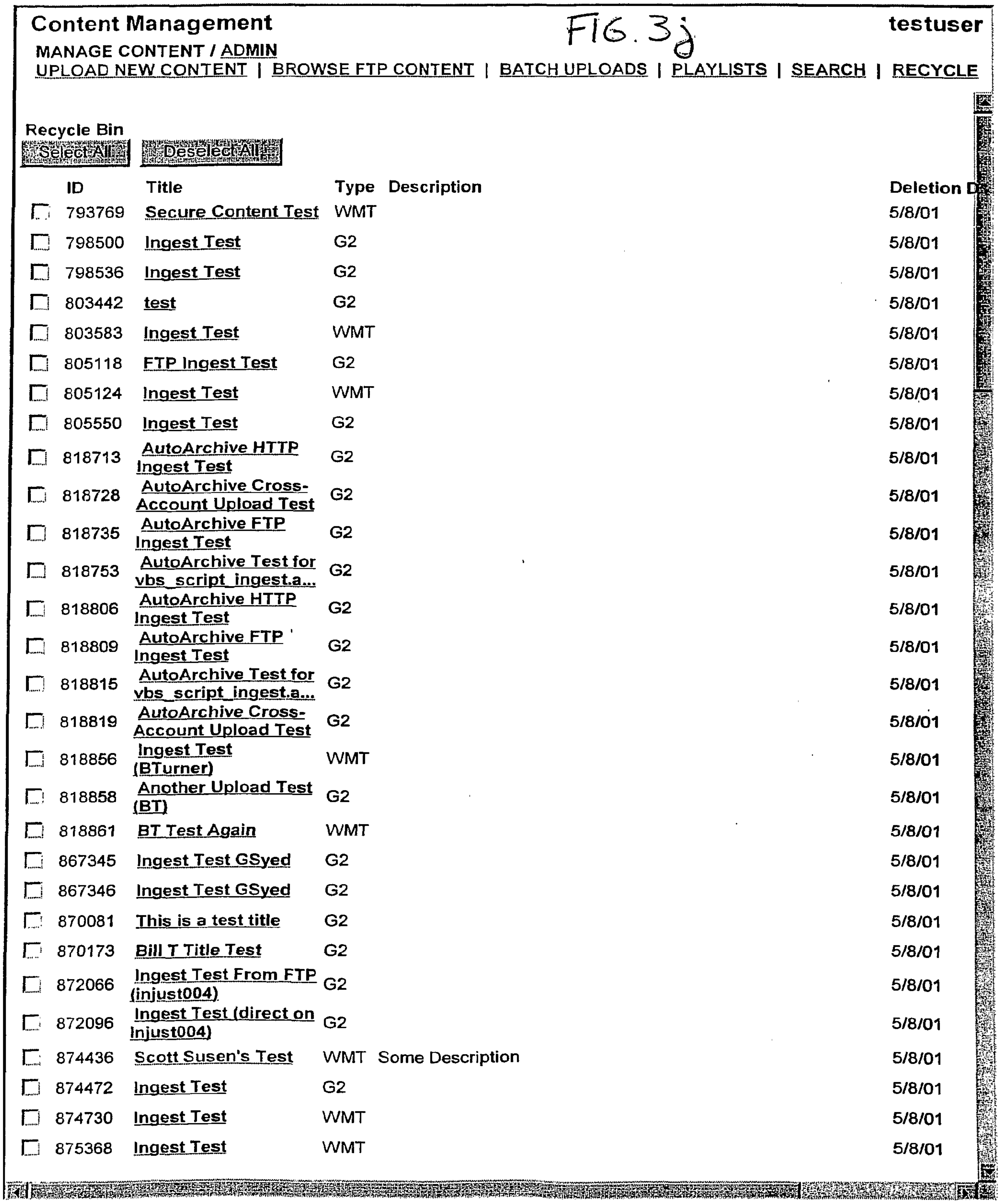
Available Playlists: ChrisBinney1 (RAM)

Add To Playlist

Streams List

- Streams: Upload Test Title

FIG. 3j


Content Management testuser

MANAGE CONTENT / ADMIN

UPLOAD NEW CONTENT | BROWSE FTP CONTENT | BATCH UPLOADS | PLAYLISTS | SEARCH | RECYCLE

Recycle Bin

Select All | Deselect All

	ID	Title	Type	Description	Deletion D
☐	793769	Secure Content Test	WMT		5/8/01
☐	798500	Ingest Test	G2		5/8/01
☐	798536	Ingest Test	G2		5/8/01
☐	803442	test	G2		5/8/01
☐	803583	Ingest Test	WMT		5/8/01
☐	805118	FTP Ingest Test	G2		5/8/01
☐	805124	Ingest Test	WMT		5/8/01
☐	805550	Ingest Test	G2		5/8/01
☐	818713	AutoArchive HTTP Ingest Test	G2		5/8/01
☐	818728	AutoArchive Cross-Account Upload Test	G2		5/8/01
☐	818735	AutoArchive FTP Ingest Test	G2		5/8/01
☐	818753	AutoArchive Test for vbs_script_ingest.a...	G2		5/8/01
☐	818806	AutoArchive HTTP Ingest Test	G2		5/8/01
☐	818809	AutoArchive FTP Ingest Test	G2		5/8/01
☐	818815	AutoArchive Test for vbs_script_ingest.a...	G2		5/8/01
☐	818819	AutoArchive Cross-Account Upload Test	G2		5/8/01
☐	818856	Ingest Test (BTurner)	WMT		5/8/01
☐	818858	Another Upload Test (BT)	G2		5/8/01
☐	818861	BT Test Again	WMT		5/8/01
☐	867345	Ingest Test GSyed	G2		5/8/01
☐	867346	Ingest Test GSyed	G2		5/8/01
☐	870081	This is a test title	G2		5/8/01
☐	870173	Bill T Title Test	G2		5/8/01
☐	872066	Ingest Test From FTP (injust004)	G2		5/8/01
☐	872096	Ingest Test (direct on Injust004)	G2		5/8/01
☐	874436	Scott Susen's Test	WMT	Some Description	5/8/01
☐	874472	Ingest Test	G2		5/8/01
☐	874730	Ingest Test	WMT		5/8/01
☐	875368	Ingest Test	WMT		5/8/01

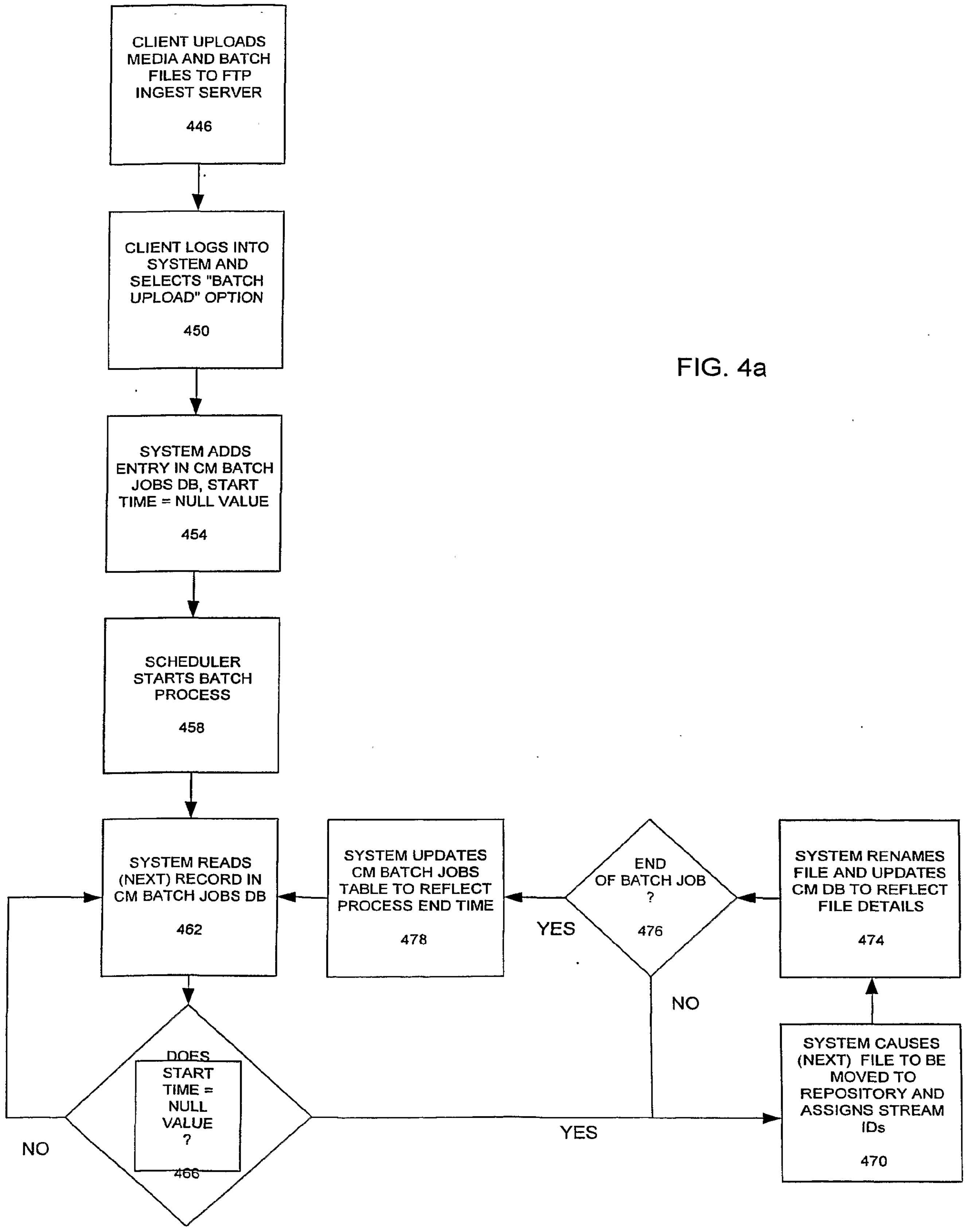

CLIENT UPLOADS MEDIA AND BATCH FILES TO FTP INGEST SERVER
446
CLIENT LOGS INTO SYSTEM AND SELECTS "BATCH UPLOAD" OPTION
450
SYSTEM ADDS ENTRY IN CM BATCH JOBS DB, START TIME = NULL VALUE
454
SCHEDULER STARTS BATCH PROCESS
458
SYSTEM READS (NEXT) RECORD IN CM BATCH JOBS DB
462
DOES START TIME = NULL VALUE ?
466
NO
YES
SYSTEM CAUSES (NEXT) FILE TO BE MOVED TO REPOSITORY AND ASSIGNS STREAM IDs
470
SYSTEM RENAMES FILE AND UPDATES CM DB TO REFLECT FILE DETAILS
474
END OF BATCH JOB ?
476
NO
YES
SYSTEM UPDATES CM BATCH JOBS TABLE TO REFLECT PROCESS END TIME
478

FIG. 4a

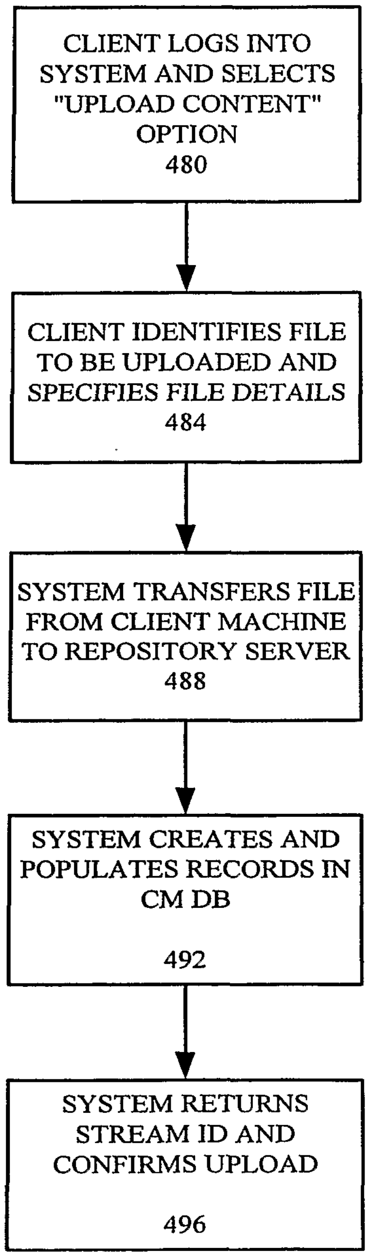

CLIENT LOGS INTO SYSTEM AND SELECTS "UPLOAD CONTENT" OPTION
480
CLIENT IDENTIFIES FILE TO BE UPLOADED AND SPECIFIES FILE DETAILS
484
SYSTEM TRANSFERS FILE FROM CLIENT MACHINE TO REPOSITORY SERVER
488
SYSTEM CREATES AND POPULATES RECORDS IN CM DB
492
SYSTEM RETURNS STREAM ID AND CONFIRMS UPLOAD
496

FIG. 4b

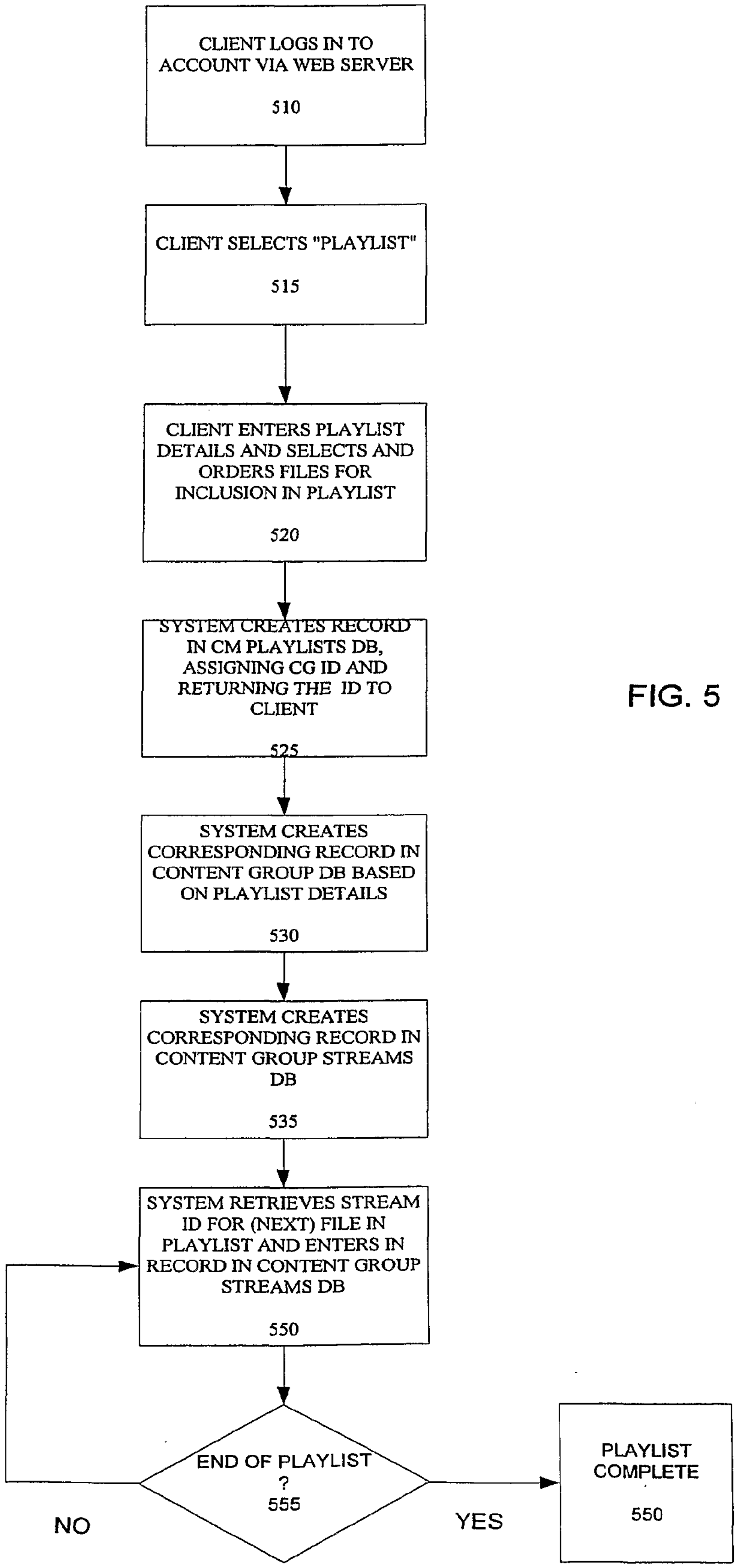

CLIENT LOGS IN TO ACCOUNT VIA WEB SERVER
510
CLIENT SELECTS "PLAYLIST"
515
CLIENT ENTERS PLAYLIST DETAILS AND SELECTS AND ORDERS FILES FOR INCLUSION IN PLAYLIST
520
SYSTEM CREATES RECORD IN CM PLAYLISTS DB, ASSIGNING CG ID AND RETURNING THE ID TO CLIENT
525
SYSTEM CREATES CORRESPONDING RECORD IN CONTENT GROUP DB BASED ON PLAYLIST DETAILS
530
SYSTEM CREATES CORRESPONDING RECORD IN CONTENT GROUP STREAMS DB
535
SYSTEM RETRIEVES STREAM ID FOR (NEXT) FILE IN PLAYLIST AND ENTERS IN RECORD IN CONTENT GROUP STREAMS DB
550
END OF PLAYLIST ?
555
NO
YES
PLAYLIST COMPLETE
550

FIG. 5

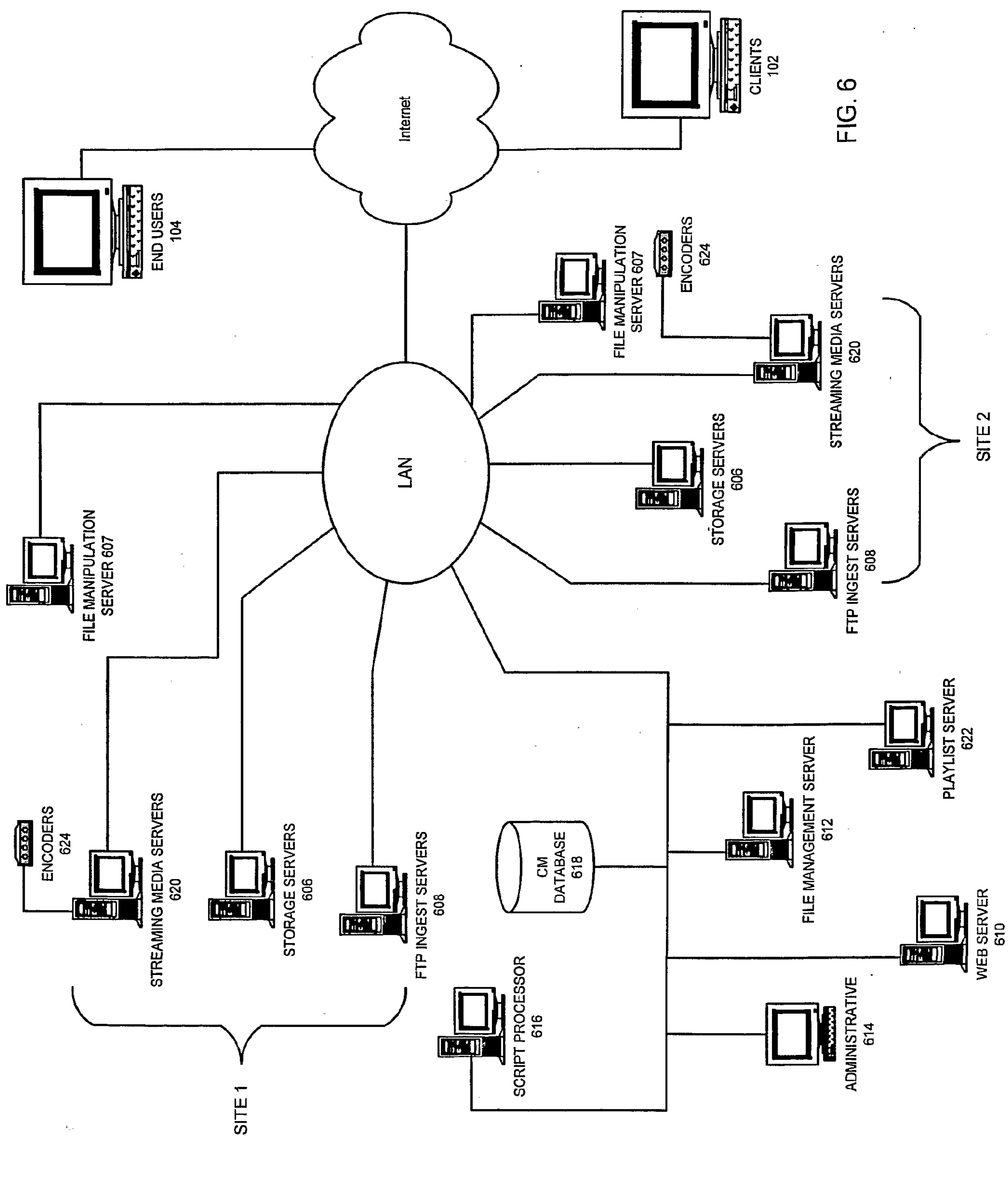

Internet
CLIENTS 102
END USERS 104
LAN
FILE MANIPULATION SERVER 607
ENCODERS 624
STREAMING MEDIA SERVERS 620
STORAGE SERVERS 606
FTP INGEST SERVERS 608
SITE 2
FILE MANIPULATION SERVER 607
ENCODERS 624
STREAMING MEDIA SERVERS 620
STORAGE SERVERS 606
FTP INGEST SERVERS 608
SITE 1
CM DATABASE 618
PLAYLIST SERVER 622
FILE MANAGEMENT SERVER 612
WEB SERVER 610
ADMINISTRATIVE 614
SCRIPT PROCESSOR 616

FIG. 6

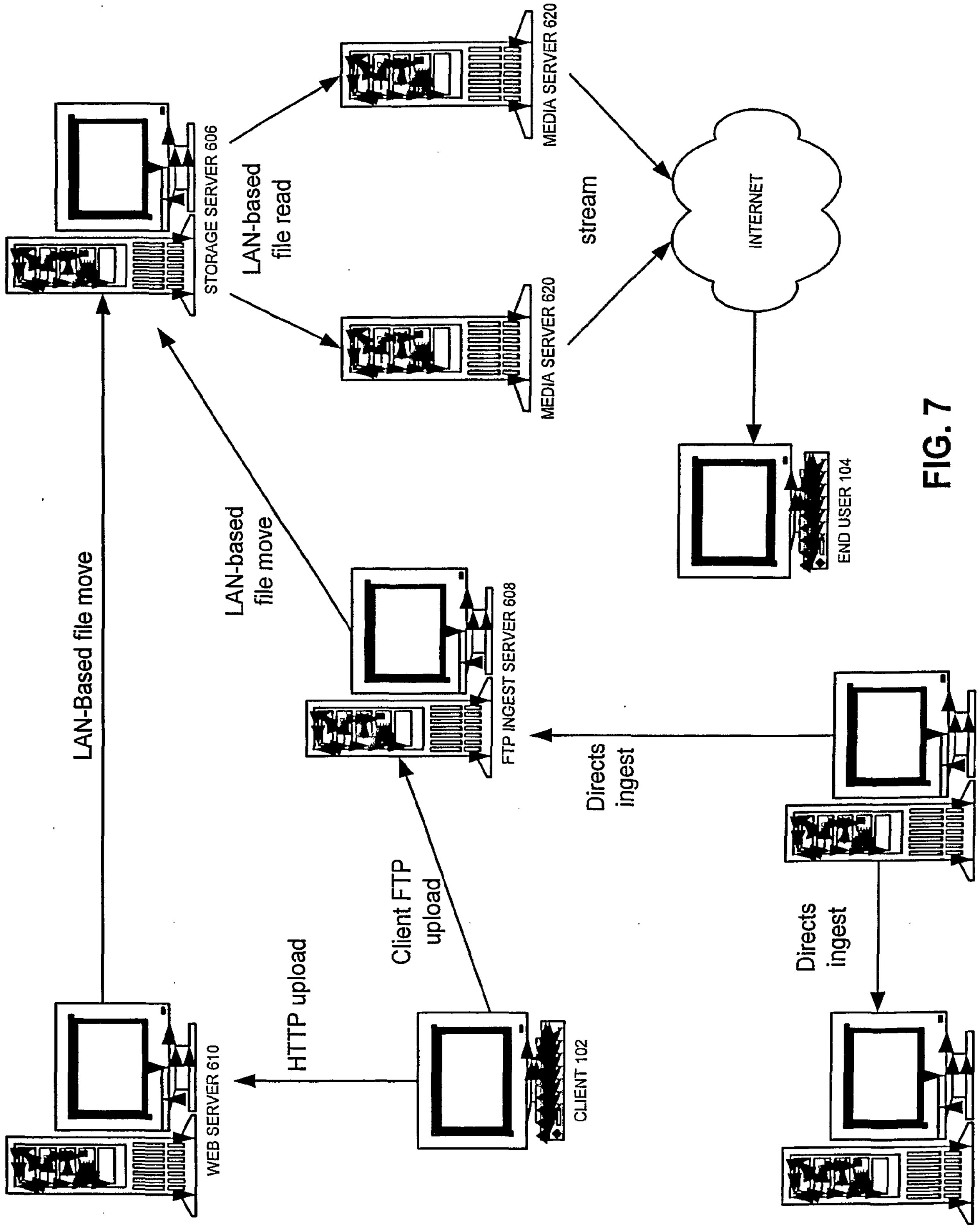
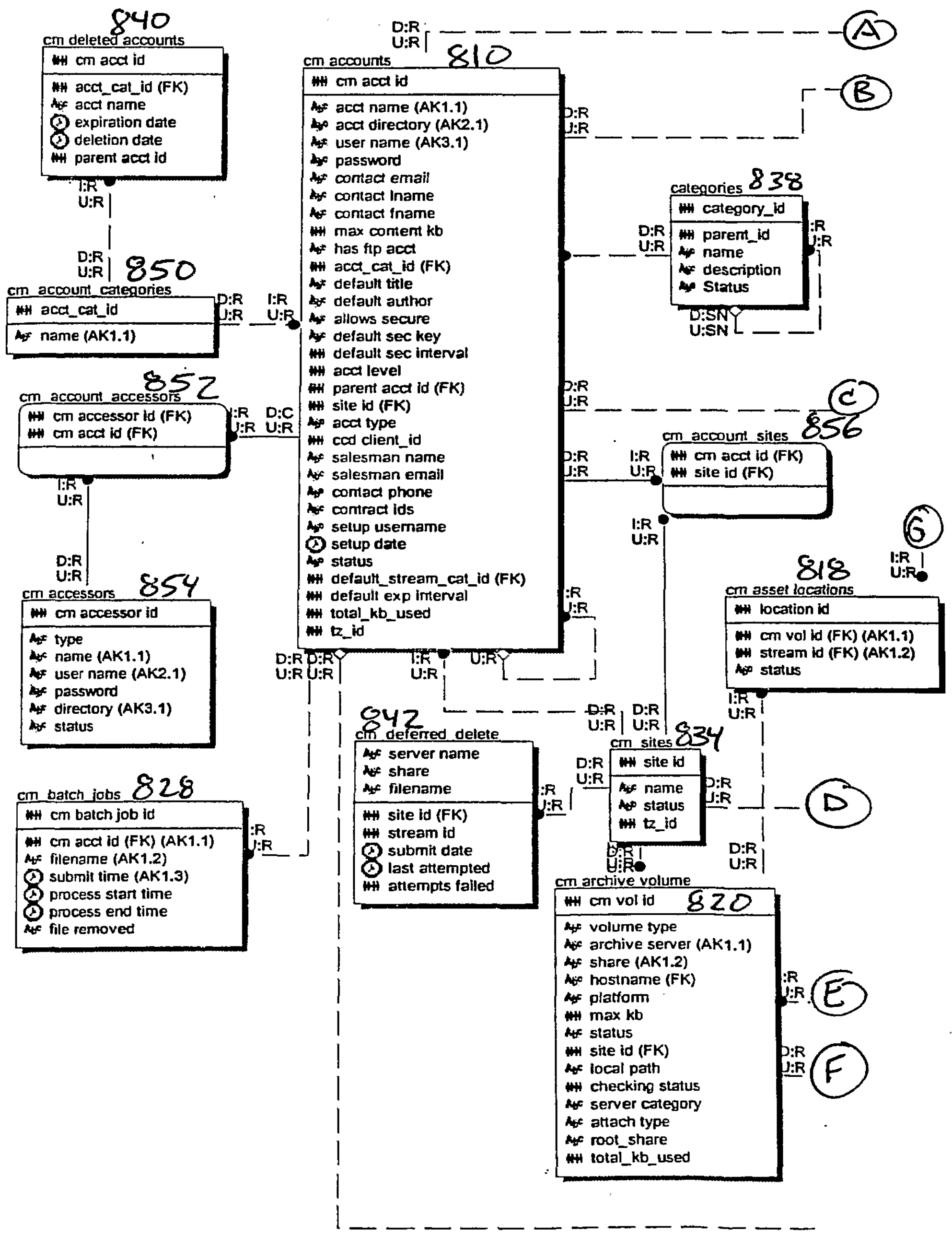

WEB SERVER 610
LAN-Based file move
STORAGE SERVER 606
LAN-based file read
MEDIA SERVER 620
MEDIA SERVER 620
stream
INTERNET
CLIENT 102
HTTP upload
Client FTP upload
FTP INGEST SERVER 608
LAN-based file move
END USER 104
IBM Compatible
Directs ingest
FILE MANIPULATION SERVER 612
Directs ingest

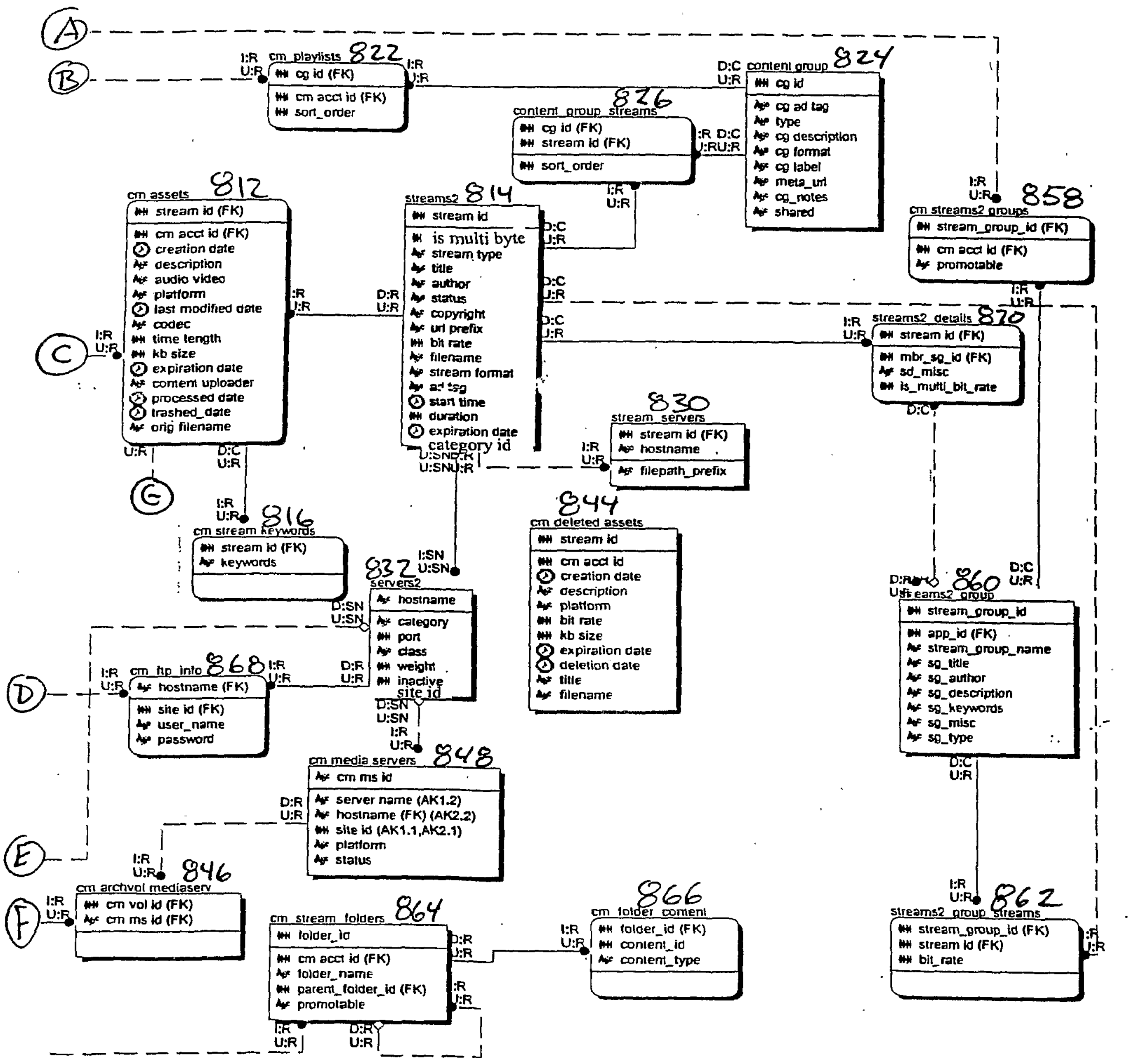
FIG. 7

FIG 8a



IPR2022-01227
EXHIBIT 1003 - PAGE 02129

FIG 8b


cm_playlists 822
cg id (FK)
cm acct id (FK)
sort_order
content group 824
cg id
cg ad tag
type
cg description
cg format
cg label
meta_url
cg_notes
shared
content_group_streams 826
cg id (FK)
stream id (FK)
sort_order
cm assets 812
stream id (FK)
cm acct id (FK)
creation date
description
audio video
platform
last modified date
codec
time length
kb size
expiration date
content uploader
processed date
trashed_date
orig filename
streams2 814
stream id
is multi byte
stream type
title
author
status
copyright
url prefix
bit rate
filename
stream format
ad tag
start time
duration
expiration date
category id
cm streams2 groups 858
stream_group_id (FK)
cm acct id (FK)
promotable
streams2 details 870
stream id (FK)
mbr_sg_id (FK)
sd_misc
is_multi_bit_rate
stream servers 830
stream id (FK)
hostname
filepath_prefix
cm stream keywords 816
stream id (FK)
keywords
cm deleted assets 844
stream id
cm acct id
creation date
description
platform
bit rate
kb size
expiration date
deletion date
title
filename
servers2 832
hostname
category
port
class
weight
inactive
site id
cm ftp info 868
hostname (FK)
site id (FK)
user_name
password
streams2 group 860
stream_group_id
app_id (FK)
stream_group_name
sg_title
sg_author
sg_description
sg_keywords
sg_misc
sg_type
cm media servers 848
cm ms id
server name (AK1.2)
hostname (FK) (AK2.2)
site id (AK1.1,AK2.1)
platform
status
cm archvol mediaserv 846
cm vol id (FK)
cm ms id (FK)
cm stream folders 864
folder_id
cm acct id (FK)
folder_name
parent_folder_id (FK)
promotable
cm folder content 866
folder_id (FK)
content_id
content_type
streams2 group streams 862
stream_group_id (FK)
stream id (FK)
bit_rate
A
B
C
D
E
F
G
I:R
U:R
D:C
D:R
I:SN
U:SN
D:SN

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US02/01840

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7) :G06F 15/16
US CL : 709/203, 231
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED
Minimum documentation searched (classification system followed by classification symbols)
U.S. : 709/203, 231

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EAST, IEEE Online
search terms: streaming, media, client, server, uploading, web site, internet

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	CHO et al. "Multimedia Service Interworking over Heterogeneous Networking Environments" IEEE 1999 pages 63-69,	1-34
X,E	US 2002/0010759 A1 (HITSON et al.) 24 January 2002 Figures 6-8	1-34
X,P	US 6,248, 946 B1 (DWEK) 19 June 2001, figures 1, 3a-3b	1-34
X,P	US 6,226,672 A (DEMARTIN et al.) 01 May 2001	1, 7, 12, 19, 28

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 22 MARCH 2002	Date of mailing of the international search report 10 MAY 2002
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer [illegible] LARRY DONAGHUE [illegible] Telephone No. (703) 305-9675

Form PCT/ISA/210 (second sheet) (July 1998)★

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

CORRECTED VERSION

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
25 July 2002 (25.07.2002)

PCT

(10) International Publication Number
WO 02/057943 A1

(51) International Patent Classification⁷: G06F 15/16

(21) International Application Number: PCT/US02/01840

(22) International Filing Date: 18 January 2002 (18.01.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/263,058 18 January 2001 (18.01.2001) US

(71) Applicant *(for all designated States except US)*: **YAHOO! INC.** [US/US]; 701 First Avenue, Sunnyvale, CA 94089 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **JUSTIN, Paul, Madison** [US/US]; 1504 Concord Drive, Richardson, TX 75081 (US). **ANTHONY, Joseph, Rodiger** [US/US]; 5004 Lake Vista Drive, The Colony, TX 75056 (US). **SYED, Noman, Kazmi** [PA/US]; 12480 Abrams Road, #2601, Dallas, TX 75243 (US). **WILLIAM, David, Turner** [US/US]; 6326 Bryan Parkway, Dalls, TX 75214 (US). **STEVEN, Alexander, Roberts** [US/US]; 5406 Kingston Drive, Richardson, TX 75082-4067 (US). **GHOUSUDDIN, Syed** [US/US]; 504 Sheffield Drive, Richardson, TX 75081 (US). **VINAY, Polavarapu** [IN/US]; 6913 Valleyview Lane- Apt. #230, Irvin, TX 75039 (US). **BHARGAVARAM, Bulleyya, Gade** [IN/US]; 1213 Meadow Creek Drive, Apt. #J, Irving, TX 75038 (US).

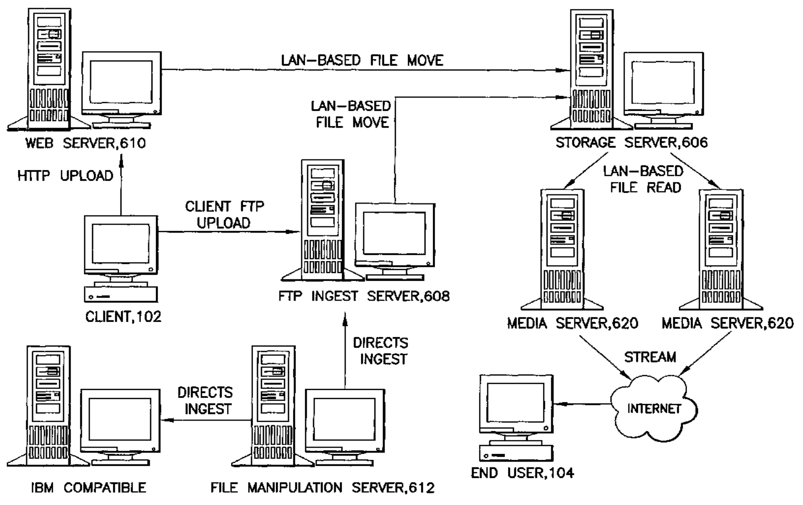
(74) Agents: **DECARLO, James, J.** et al.; Stroock & Stroock & Lavan LLP, 180 Maiden Lane, New York, NY 10038 (US).

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

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW),

[Continued on next page]

(54) Title: METHOD AND SYSTEM FOR MANAGING DIGITAL CONTENT, INCLUDING STREAMING MEDIA



(57) Abstract: Method and system for uploading, managing and delivering digital content, including streaming media. The system according to one embodiment allows receives digital content from the client (102), assigns a stream identifier (ID) to the content and stores the content. The client is given a playlist uniform resource locator (URL) for publishing on its web site (610), the URL including the stream ID. Activation of the URL by an end user (104) causes the stream to be served to the end user, without the client receiving or providing an indication of the specifics of where the content was stored. An embodiment of the present invention provides a system and method that permit clients to actively manage their content, including defining logical folders and subfolders containing item(s) of the content and defining logical stream groups, containing items of the content.

WO 02/057943 A1

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, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

(48) Date of publication of this corrected version:
6 March 2003

(15) Information about Corrections:
see PCT Gazette No. 10/2003 of 6 March 2003, Section II
Previous Correction:
see PCT Gazette No. 04/2003 of 23 January 2003, Section II

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

METHOD AND SYSTEM FOR MANAGING DIGITAL CONTENT, INCLUDING STREAMING MEDIA

CROSS REFERENCE TO RELATED APPLICATIONS

[001] This application is a continuation-in-part of, and claims the benefit of, United States Provisional Application No. 60/263,058, filed on January 18, 2001, for METHOD AND SYSTEM FOR MANAGING STREAMING MEDIA, hereby incorporated by reference.

BACKGROUND OF INVENTION

1. Field of the Invention

[002] The present invention relates generally to a method and system for providing digital content via the Internet and, more particularly, to a method and system for allowing clients to up-load, manage, and deliver streaming media content via the Internet.

2. Description of Related Art

[003] With the advent of the Internet and the World Wide Web, an industry has developed around the delivery of digital content, such as streaming media content. By way of example, streaming media may be used for any of a number of purposes, including entertainment, distance learning and corporate purposes. Entertainment companies stream movies and sporting events, distance learning companies stream educational content, and corporations stream training materials.

[004] Although some streaming media content providers may have relatively few items of content to provide, some content providers have hundreds, even thousands of content files. Storing and streaming this number of content files can be costly. Furthermore, streaming content requires a level of technical expertise often not found in companies focusing on creating content. Consequently, content providers have turned to content management service providers to store and stream content on behalf of the content providers.

[005] As more content providers turn to service providers for their streaming technology needs, service providers must manage more client accounts and greater numbers of content files. Furthermore, to maintain existing content provider clients and attract new clients, service providers must provide a content management system that not only is capable of organizing large numbers of content files, but also is easy for the content providers to use. Accordingly, there exists a need for a content management system that allows clients to easily up-load, manage, and deliver streaming media content via the Internet.

3. Summary of the Invention

[006] The present invention solves the foregoing and other needs. In certain embodiments, a method and system is provided for allowing client content providers to upload, manage and deliver streaming media and other digital content. The system according to one embodiment allows receives digital content from the client, assigns a stream identifier (ID) to the content and stores the content. The client is given a playlist uniform resource locator (URL) for publishing on its web site, the URL including the stream ID. Activation of the URL by an end user causes the stream to be served to the end user, without the client receiving or providing an indication of the specifics of where the content was stored.

[007] Another embodiment of the present invention permits clients to actively manage their content, including defining logical folders and subfolders containing item(s) of content; defining logical stream groups, containing items of content.

[008] A system according to one embodiment of the present invention couples one or more media servers to a storage server. While the storage server has stored a copy of the content, the media servers coupled thereto do not. In operation, the media servers perform a read of the contents stored at the storage server when requested by an end user.

BRIEF DESCRIPTION OF THE DRAWINGS

[009] The following drawing figures, which form a part of this application, are illustrative of embodiments of the present invention and are not meant to limit the scope of the invention in any manner, which scope shall be based on the claims appended hereto.

[0010] Figure 1 is a schematic illustrating the system architecture of one embodiment of the present invention;

[0011] Figure 2 is a schematic illustrating the database of one embodiment of the present invention;

[0012] Figures 3a-j are web pages of the Content Management client-side web site according to one embodiment of the present invention;

[0013] Figures 4a - b are flowcharts illustrating processes of up-loading content files into the content management system according to one embodiment of the present invention;

[0014] Figure 5 is a flowchart illustrating the process of creating a playlist according to one embodiment of the present invention;

[0015] Figure 6 is a schematic illustrating the system architecture of an alternate embodiment of the present invention;

[0016] Figure 7 is a schematic illustrating a workflow of a content management system according to the alternate embodiment of Figure 6; and

[0017] Figures 8a and 8b represent a schematic illustrating the database of the embodiment of Figure 6.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

[0018] Certain preferred embodiments of the present invention will now be discussed with reference to the aforementioned figures, wherein like reference numerals refer to like components. Turning first to the schematic of Figure 1, a system 100 according to one embodiment of the present invention is shown. In general, the system 100 allow clients 102 to up-load streaming media content to the system 100, manage such content, and make the content available to end-users 104 via a web site on the Internet.

[0019] As will become apparent from the following discussion, each of the clients 102 and end-users 104 has a processor, such as a personal computer (PC), web enabled cellular telephone or personal digital assistant (PDA) and the like, coupled to the Internet by any one of a number of known manners. Furthermore, each client 102 preferably includes an Internet browser, such as that offered by Microsoft Corporation under the tradename INTERNET EXPLORER or that offered by Netscape Corp. under the tradename NETSCAPE NAVIGATOR, as well as file transfer protocol (FTP) client software for interacting with various components of the system 100. Additionally, each end-user processor preferably includes an Internet browser, such as INTERNET EXPLORER or NETSCAPE NAVIGATOR, and a media player, such as that offered by Microsoft Corporation under the tradename WINDOWS MEDIA PLAYER or that offered by Real Networks Corp. under the tradename REALPLAYER. It is to be understood that although the present embodiment is described in terms of Windows Media content and Real Media content, it is within the scope of the present invention to utilize any media format heretofore or hereafter known. Furthermore, it is to be understood that although the present embodiment is described in the context of streaming media, the present invention is applicable to digital content other than streaming media.

[0020] As discussed in greater detail below, the client 102 up-loads streaming media content to the repository or storage server (and associated storage) 106 either directly, via an HTTP up-load, or via an FTP ingest server (and associated storage) 108. Once the client 102 has up-loaded its streaming media content, the client may manage its content via web pages on a content management web site provided by a client-side web server 110. As discussed in greater detail with reference to Figure 3a-j, the web site serves as an interface through which the client 102 may log into its account and select any of the following exemplary functions: up-load new content; search the client's existing up-loaded content; create and edit playlists; browse the client's contents stored on the FTP server 108; and perform a batch up-load of content from the FTP server 108 to the repository server 106. Additionally, the web site serves as an interface for the client 102 to perform various administrative functions, such as setting and changing any of the client-defined account and file information described below. In alternate embodiments, clients 102 may manage their accounts programmatically, via scripts running on the web server 110 or other system service described below.

[0021] The system further includes a file management server 112, administrative staff terminals 114, and a script processor 116. In general, the file management server 112 controls the activities of the various other components to which it is coupled and the movement of content within the system. The administrative terminals 114, in turn, are coupled to the file management server 112 to allow the administrative staff of the content management service provider maintaining the content management system 100 to control and monitor the system 100. The script processor 116 works in conjunction with the file management server 112 to control operation of the system 100 by running various software scripts and/or components performing most of the functionality described herein. As described in greater detail below, the script processor 116 includes various software modules, including a task scheduler and program scripts and objects for batch up-loads of content files and for recovery of deleted content files. It is to be understood that such task scheduling software may be obtained by any of a number of third-party vendors. Similarly, based upon the description of the program scripts and objects herein, the program scripts and objects may be written in any programming language heretofore or hereafter known, such as PERL, Visual Basic, JavaScript, C++, and the like.

[0022] The content management system 100 also includes a Content Management (CM) Database 118. As described in greater detail below with reference to Figure 2, the CM Database 118 includes numerous relational databases or tables in a database. In general, the CM Database 118 preferably includes account information, which identifies each client's account, stream information, which identifies and describes each item of streaming content within a client's account, playlist information, which identifies and describes each client's playlist, batch information, which defines the status of each client's batch up-load requests, and-storage location information, which tracks the storage of content on the repository 106 and streaming media 120 servers.

[0023] Under direction of the file management server 112, the up-loaded content is eventually transferred from the repository server 106 to one or more streaming media server (and associate storage) 120, where it is available for streaming via the Internet. As described in greater detail below, the current embodiment utilizes a playlist server 122 for dynamically generating a playlist metafile (such as ASX, RAM, SMIL, and RPM files, to name a few) to be provided to the end-user's windows media player. In addition to the archived content located on the streaming media server storage 120, the present environment provides for streaming of live content acquired via encoders 124 coupled to the streaming media servers 120.

[0024] As illustrated in Figure 1, while the clients 102 and end-users 104 are coupled to each other and to the content management system 100 via the Internet, the service provider's components are coupled to each other via a local area network (LAN). More specifically, the repository 106, FTP server 108, client web server 110, file management 112, administrative terminals 114, script processor 116, CM Database 118, streaming media server 120, and playlist server 122 are all in communication via a secure LAN. In alternate embodiments, components of the system are coupled differently, for example, each coupled directly to the Internet, coupled via a wide area network (WAN) and the like.

[0025] In general, the division of functionality among the web server 110, file management server 112, playlist server 122 and script processor 116 described herein is exemplary, as the functions may be segregated and grouped differently. For example, it is to be understood that although the present embodiment utilizes client web servers 110 that are separate from the playlist server 122, in alternate embodiments the functionality of the playlist server 122 described herein may be implemented in software residing on the web server 110, thereby obviating the need for a separate playlist server.

Content Management Database 118

[0026] The Content Management (CM) Database 118 will now be described in greater detail with reference to Figure 2, and continuing reference to Figure 1. The CM Database 118 includes several logically discrete tables of information, each of which is described below. As will be appreciated by one skilled in the art, the following logical arrangement of information in tables is exemplary, and other arrangements are within the scope of the present invention; for example; those with fewer or more tables and/or fewer or more fields. It is also to be understood that although the CM Database 118 is described as a central database, it is within the scope of the present invention to distribute the contents of the CM Database 118 throughout the system.

[0027] The CM Database 118 includes a CM Accounts Table 210. In general, the CM Accounts Table 210 includes records for each client account, as identified by a CM account ID. Each such record includes account identifying information, such as account name, account directory, username, password, contact information (such as contact name, e-mail address, and telephone number), default title and author (which are preferences used to identify items of content in the event no other title or author information is provided by the client 102), parent account ID (in the event the current account is related as a child to another account), an indication of whether or not the client has an FTP account, and maximum allowable amount size of content (e.g., in Kilo-bytes). It is within the scope of the present invention to implement security features to maintain the integrity of the content, such as those features and methods described in applicant's co-pending International Application No. PCT/US01/46726, filed November 5, 2001, for SYSTEM AND METHOD FOR CONTROLLING ACCESS TO DIGITAL CONTENT, INCLUDING STREAMING MEDIA, and International Serial No. PCT/US01/18324, filed June 6, 2001, for SYSTEM AND METHOD FOR CONTROLLING ACCESS TO DIGITAL CONTENT, INCLUDING STREAMING MEDIA, which are hereby incorporated by reference. As such, the table includes security related information, including default preferences indicating whether or not the content is secure (allows security), and if so, a cryptographic key used for accessing such content (default sec key) and a time period for which the key will be valid (default sec. interval).

[0028] The CM Accounts Table 210 also includes a field for a "site id," which uniquely identifies a site consisting of a group of one or more FTP servers 108 and those repository servers 106 and streaming media servers 120 to which the content on the FTP servers 108 can be up-loaded. Two representative sites are illustrated in Figure 1, although the system 100 could have any number of sites. Although not required in the present invention, each client account is associated with a site located in the client's geography or the geography where the endorsers 104 likely to access the client's content are located so to speed delivery of the streaming media content. Furthermore, it is to be understood that logically grouping the FTP server 108, repository servers 106 and streaming media servers 120 into sites is not required. As noted below, much of the account identifying information of the present embodiment is provided by the client 102 during the registration process.

[0029] The CM Database 118 further includes a series of tables containing content – identifying information. More specifically, the CM Assets Table 212 includes records, each of which identifies an item of streaming content by a stream ID. Furthermore, each record includes the CM account ID, creation date of the item of content, description of the content, an identification of whether the content is audio and or video, the platform to which the content relates, the date on which the content was last modified, any code necessary for viewing of the content, the length and size of the content, the expiration date (if any) of the content, the individual that up-loaded the content and the date on which the content was processed into the system 100. It is to be understood that each record in the CM assets Table 212 correlates an item of content, as identified by the stream ID, to a particular client account, by reference to the CM account ID.

[0030] The Streams2 Table 214 also includes records identifying each item of content, as identified by the stream ID. Fields contained in the Streams2 Table 214 include the stream type, such as windows media player or real media player, title of the content, author of the content, status of the content, copyright notice for the content, URL prefix, bite rate of the content, file name of the content (as provided by the client 102), stream format, such as windows media player or real media player, ad tag (which specified where an advertisement is to be inserted), start time (if the Stream is a portion of the file), duration, and expiration date.

[0031] As illustrated in Figure 2, the CM Assets Table 212 has three additional tables related thereto: the CM Stream Keywords Table 216, CM Assets Locations Table 218 and CM Archive Volume Table 220. The CM Stream Keywords Table 216 includes keywords, as provided by the client 102, for each item of content as identified by the stream ID.

[0032] The CM Assets Locations Table 218, which is also related to the CM Assets Table 212, includes records identified by location ID. The location ID is a unique identifier for each portion of storage pertaining an item of content. As such, location IDs may refer to either a repository server 106 or a streaming media server 120. A given content file may be stored at multiple location ids, for example one on a media server 120 and one on a repository server 106. More specifically, each record in the CM Asset Locations Table 218 includes a CM volume ID, for identifying the portion of the server on which the associated content resides; stream ID, for identifying the item of content stored at the location ID; and status of the item of content. The status of the file corresponding to each location is determined based on the following table:

Status	**Meaning**	**Applicable to file located on:**
Available	File is present on a media server and is available for streaming	Streaming server
Back up	File has been copies to a streaming media server	Repository server
Copying	File is being copied to a repository server or media server	Repository server; media server
Deleting	File is being deleted	Repository server; media server
Failed	File up-load or copy has failed	Repository server; media server
Moving	File is being moved from the current server to another server of the same type	Repository server; media server
New	File has recently been up-loaded and not copied	Repository server

[0033] The CM Archive Volume Table 220 is also related to the CM Asset Table 212 and, more particularly, to the CM Asset Locations Table 218. CM Archive Volume Table 220 includes records for each CM volume ID. Each record includes: volume type, which indicates whether the volume ID pertains to a repository server 106, streaming media server 120, FTP server 108, script processing server 116 or web server 110; archive server name, indicating the LAN name of the particular repository server 106 or streaming media server 120, as the case may be, containing the volume ID; the server's DNS address; the platform for the content, such as Window's Media or Real Media; the status of the content associated with the volume id; the maximum storage capacity for the server continuing the volume id; and the side id of the server containing the volume id.

[0034] Also associated with each client account identified in the CM Accounts Table 210 is client playlist information. Such playlist information is contained within the CM Playlists Table 222, Content Group (CG) Table 224 and CG Streams Table 226. In general, the CM Playlist Table 222 includes records identifying each client playlist, as identified by a content group ID. Each record further includes the CM account ID and the sort order, which indicates the ordering of the items of streaming content associated with the particular playlist.

[0035] Content Group Table 224 also includes records for each playlist, and identified by the playlist or content group (CG) ID. In general, each record contains the playlist details, including the CG ad tag, the type of playlist, the CG description, the CG format, the CG label, the meta-url for the playlist, CG notes, and an indication as to whether or not the content group is shared.

[0036] The playlist entries are identified in a CG Stream Table 226. More specifically, the CG Stream Table 226 includes records identifying each item of content as part of a content group and further includes an indication of the order of the particular item of content within the content group. As such, each record in the table includes the content group ID, stream ID, and sort order.

[0037] The CM Database 118 also includes a table that indicates its status of client-requested batch up-loads. More specifically, the CM Batch Jobs Table 228 includes records identifying the status of each requested batch job, as identified by a CM batch job ID. Each record in the CM Batch Job Table 228 includes the CM account ID, identifying the account to which the batch job relates, the file name of the batch file (filename), the time which the client 102 submitted the batch request (submit time), the time at which the process was started (process start time), the time at which the batch process ended (process end time), and an indication as to whether or not the batch file was removed from the client's account (file removed). Because each record in the CM Batch Jobs Table 228 includes the CM account ID, each such record is related to a record in the CM Accounts Table 210.

[0038] The CM Database also includes a Stream-Servers Table 230. The records in the Stream-Servers-Table 230 correlate each file, as identified by its stream ID, with the hostname (or domain name server (DNS) address) of the streaming server 120 on which it resides. While each stream ID will only have one record, a particular file will have as many records as it has copies residing on different streaming servers 120.

[0039] The Servers2 Table 232 includes a record for each streaming server 120, as identified by its server address, setting forth various metrics for the server 120, such as the weight or relative number of streams being served to end users 104.

[0040] The CM Sites Table 234 contains a record for each site, as identified by its site id. More specifically, each record identifies the name of the site, the FTP server uniform resource locator (URL) associated with the site and the processor URL associated with the site. In the present embodiment, each site only includes one FTP server 108.

[0041] As illustrated in Figure 2, the CM Database also includes tables relating to the deletion and recovery of files, identifying each client's service level and categorizing the clients' accounts.

[0042] The Account Servicelevels Table 236 stores a description of each client's service level. As illustrated, this table 236 is keyed to the Account Table 210 by virtue of the account service level ID (acct srvlevel id) field. Each service level generally corresponds to a different set of options, functions and/or capabilities offered to the client, for example, the amount of storage allowed, number of uploads, the number of sites provided, and the like.

[0043] The Categories Table 238 stores records identifying the category to which a particular account relates. More specifically, each record corresponds to a particular category, as identified by category ID. In general, a category is a grouping of accounts, for example, of all accounts related to a particular parent account. Thus, each record identifies the parent ID, name, and description. As illustrated, records in the Categories Table 238 are keyed to accounts in the Account Table 210 by the category ID field.

[0044] The CM database 118 also tracks deleted accounts and deleted content. To this end, the Deleted Accounts Table 240 keeps a record for each account, identified by account ID, that is removed from the system 100. By keeping such records, the system administrator is able to provide accurate record keeping and report generation.

[0045] The Deferred Delete Table 242, Deleted Assets Table 244 and the Recovery Jobs Table 246 are used in the deletion of content. More specifically, a Stream may be deleted either upon request of the client 102 or based on the content expiring. As noted above, the Streams2 Table 214 includes an expiration date field for each Stream ID. A script running on the system periodically scans the Streams2 Table 214 for records having an expiration date equal to or greater than (i.e., after) the then current time. For each record containing such an expiration date, the systems proceeds to delete the content. The process of deletion of the present embodiment includes changing the status field in the Asset Locations Table 218 to expired and, in the Streams-Servers Table 230 changing the hostname for the stream being deleted to another system server that hosts a default message, indicating to the end user 104 that the requested content is not available. The system also replaces the stream's actual filename, as stored in the Streams2 Table 214, with the filename of such default message.

[0046] Upon deletion of a stream, a record is also created in the Deleted Assets Table 244, specifying various stream information, as well as the expiration date and deletion date of the content.

[0047] In the event the stream to be deleted is currently being provided to an end-user 104, the system will be unable to delete the content. In such an instance, the system creates a record in the Deferred Delete Table 242, specifying the server name, file name and share information for the content to be deleted. As illustrated, for each unique combination of server name, share information and file name, the Deferred we defer delete Table 242 also stores the site ID, stream ID, the date on which the delete request was submitted to the system, the date of the last attempted deletion, and the number of failed deletion attempts. A script running on the system periodically searches the Deferred Delete Table 242 for entry and, upon the finding a populated record, proceeds to attempt to delete the content specified by the entry. Upon successful deletion, an entry is created in a Deleted Assets Table 244 and a status is changed in the Asset Locations Table 218.

Registration Process

[0048] Having described the components of the present embodiment, the operation thereof will now be described. As an initial matter, each client 102 must register with the service provider. In general, such registration includes the client 102 providing the service provider with account identifying information, a subset of which includes client identifying information. More specifically, the client provides the account identifying and client identifying information contained within the CM Accounts Table 210. The client may also select the designed service level. In one embodiment of the present invention, the client 102 provides such information via a secure web page generated by the client web server 110. The client web server 110 receives the information and by executing a common gateway interface (CGI) script, writes the information via the LAN to the CM Database 118. In an alternate embodiment, the client 102 manually provides the information to the service provider, who, in turn, manually enters the information into the CM Database 118 via the administration terminals 114. In either embodiment, once the service provider receives the account identifying and client identifying information, a CM account ID is assigned and the corresponding record in the CM Accounts Table 210 is populated.

<u>Client-Side Content Management Web Site</u>

[0049] As noted above, the client web server 110 provides a content management web site that serves as an interface between the client 104 and the content management system. While the present embodiment utilizes such interactive web site for allowing the client 102 to interface with the system and to manage the client's content, it is to be understood that other interface devices may be used, including voice recognition devices, text-based systems, or by passing variables on a query string and running a script, or any other interface means. Furthermore, although the present embodiment passes data between the web server 110 and the script processor 116 in XML format, it is to be understood that other formats may be utilized. The content management web site of the present embodiment will now be discussed with reference to Figures 3a - j.

[0050] Once a client 102 logs into the system by submitting its username and password, the client is presented with the option to perform content management functions or administrative functions. As shown in Figure 3a, when the client 102 selects administrative functions from a high level menu 302, the web page presents the client 102 with the ability to set account preferences, including default title, default author, default security key, and default security interval. As with the other web pages discussed herein, entry of information via the client 102 is received by the client web server 110 and placed in the appropriate fields in the CM database 118. With the account preferences web page of Figure 3a, the default preferences are stored in the CM Accounts Table 210. In alternate embodiments, the web page also displays account settings such as maximum allowable storage (max content Kb), actual storaged used (e.g., sum of Kb size fields in Assets Table 212), and any other information stored in the database 118.

[0051] As shown in Figure 3b, when the client 102 selects the manage content option in the high level menu, the web site presents the client with six options: "Upload New Content", "Browse FTP Content", "Batch Uploads", "Playlists", "Search" and "Recycle". When the "Upload New Content" option is selected, the system presents the client 102 with the web page shown in Figure 3b. As discussed in greater detail below, the client 102 is able to identify a content file residing on its local machine, and upload the file to a repository server 106. In general, the client 102 specifies the file name and file details in an upload content form 306. These file details are then stored in the CM Assets Table 212 and the Streams2 Table 214. Notably, the account preferences set by the client 102 on the web page of Figure 3a are provided as default entries in the upload content form 306.

[0052] When the client 102 selects the "browse FTP content" option, the system presents the client with the web page shown in Figure 3c. As the name indicates, this option provides the client 102 with a listing of all content files previously uploaded by the client 102 to the client's FTP account on the FTP ingest server 108. The system first identifies the FTP ingest server 108 containing the client's FTP account. A call is made to the particular FTP ingest server 108, which in turn executes a script to read all file directories and files within the client's account. The FTP server 108 returns a listing of such file directories and files in XML format to the web server 110. The web server 110, in turn, converts the XML information into HTML, which is displayed to the client 102. As shown in Figure 3c, the client 102 is presented with the name of its FTP account 308, as well as a listing of the client's file directories 310 and the client's content files 312.

[0053] The browse FTP content web page allows the client 102 to either delete a content file or cause a content file to be ingested from the FTP server 108 to a repository server 106. Such operations will be discussed in greater detail below.

[0054] The "batch uploads" option is shown in Figure 3d. As shown therein, the batch upload web page of the current embodiment is divided into two sections: one illustrating the batch files which have been uploaded to the client's FTP account, but not processed; and one identifying the client's batch files for which processing has begun 316. By way of example, batch file "1-100.bul" has been uploaded but not processed, while batch file "BillDefault.txt" was previously uploaded and processed.

[0055] It should also be noted that the batch upload web page may display any file detail stored in the CM database 118 corresponding to each batch file. More specifically, for each batch file listed as having been processed, information may be extracted from the submit time, process start time and process end time fields of the CM Batch Jobs Table 228.

[0056] The listing of available batch files 314 is generated by the system browsing the client's FTP content and extracting all pure text files. The listing of processed batch files is created by the system by searching the CM Batch Jobs Table 228 for all records identified by the client's CM account ID. The client 102 may select one of the available batch files for either deletion or processing.

[0057] The web page displayed when the client 102 selects the "playlists" option is illustrated in Figure 3e. In general, this web page provides the client 102 with the ability to edit playlists. To this end, the web page presents the client 102 with a list of current playlists 318. The system generates the list of playlists by searching the CM Playlists Table 222 for all entries corresponding to the client's CM account ID, thereby resulting in a list of playlist or content group (CG) IDs, each of which identifies a different playlist for the client 102. For each CG ID, the system retrieves the CG description and format in the Content Group Table

224. The information is transferred in XML format and displayed in the playlist list 318 in HTML format.

[0058] The system also provides the user with a search form 320, whereby the client 102 can enter playlist details, such as playlist/CG ID, CG description and CG format, in which the system uses in searching the Content Group Table 224 in order to return a list of matching playlists.

[0059] The client 102 may also select a current playlist to manage. As shown in Figure 3f, when the client 102 chooses to manage a playlist, the system retrieves from the CM database 318 the CG id 322, various playlist details 324 as stored in the Content Group Table 224, and a list of streams or content files making up the identified playlist 326 as retrieved from the Content Group Streams Table 226 based on the CG ID. From the list of streams, the client 102 may select any number for deletion from the playlist, in which case the system removes the stream's id from the corresponding record in the Content Group Streams Table 226.

[0060] Notably, the web page also provides the client 102 with the uniform resource locator (URL) for the playlist 328. In general, the playlist URL 328 is incorporated into the client's web page as a link, and when activated by an end user 104, causes the playlist identified by the embedded playlist/CG ID to be played. The process of generating the playlist URL 328 and playing the streams associated with the playlist is described in greater detail with reference to Figures 5.

[0061] The web page displayed when the client 102 selects the "search" option is shown in Figure 3g. The system provides the client 102 with a search form 330 that enables the client 102 to enter the file details, including title, author, keywords, stream id, creation date, and duration, as search criteria. The web server 110 passes the file details entered by the client 102 to the script processor 116, which uses the values to search the CM Assets

Table 212, CM Stream Keywords Table 216 and the Streams2 Table 214. The script processor 116, in return, returns various file details of the client's content files meeting the search criteria. Such results are shown in Figure 3h. The search form 330 also includes an option to display the results in XML format (as received from the script processor 116) and to search not only the client's current account, but also all child accounts (as indicated by the parent id field in the CM Accounts Table 210).

[0062] As an alternative to using the search form 330, the client 102 may simply browse all content in its content management account by clicking the "Browse" button. Once the client 102 clicks the browse button, the web server calls a script on the script processor 116 that retrieves all files associated with the client's username and password (and, thus, account id).

[0063] Exemplary search results are illustrated in Figure 3h. Each content file may be identified by any of the file details stored in the CM Database 118. In the present embodiment, each file is identified by the stream ID, title, stream type, stream description, creation date, file size, and status, all of which are stored in the CM Assets Table 212 and Streams2 Table 214.

[0064] Furthermore, the web page displaying the search and browse results (Figure 3h) provides the client 102 the option to select one or more files and either "delete" the selected files or "create a playlist" by adding the selected files to a new or existing playlist. In an alternate embodiment, clients 102 are also provided with the option to view the streams URL playlist URL 328. In the event the "create a playlist" button is activated, the "create a playlist" window shown in Figure 3i is displayed. The window displays a list of the client's existing playlists 332 and the files contained within each playlist 334 (as provided in the "playlist" web page), and the window provides the client 102 a form 336 to enter playlist details, including description and format, for a playlist to be created using the files selected on the web page of Figure 3h.

[0065] It should be understood that the foregoing client web pages represent an exemplary client interface and that other interfaces are within the scope of the present invention. For example, clients may access their account via a script, such as a virtual basic script, on an active server page (ASP), and pass the relevant variable in a query string rather than entering them on a web page.

[0066] Thus, in certain embodiments clients 102 are able to manage playlists via an ASP page running on one of the system servers (e.g., the web server 210 or a separate application server, not shown) by passing certain variables on a query string. In one such embodiment, the client 102 log into the system by paring its username and password and preferably as part of the same query string, passes any of a number of parameters depending upon the desired function.

[0067] For example, the client may pass a parameter specifying what action is to be performed on the playlist. Such action may include: adding or creating a new playlist; editing the playlist (e.g., editing the playlist parameters or streams comprising the playlist); removing or deleting the playlist; listing playlist in the account (e.g., by name or ID); listing the playlists and the contents of each playlist and the like. The playlist ID parameter is specified to identify the playlist on which the system is to act.

[0068] By way of example, playlist parameter that can be specified (and thus edited in the CM Database 118 is the name of the playlist. The playlist name can also be used to identify the playlist which the system is to act. Another parameter that can be specified, and thus edited) is the playlist format (e.g., RAM for RealMedia content and ASX for Windows Media content).

[0069] The client can also pass parameters specifying the action to be taken, if any, on individual streams comprising the specified playlist. Such actions may include, for example: adding a stream to the playlist; removing a stream from the playlist; removing all streams from the playlist; one or more stream IDs, to identifying the steam(s) when adding or removing them.

[0070] Preferably, under a client uses such an ASP to manage playlists, the server returns a message, such as an XML message, indicating whether the requested action was successfully performed and, if so, any information generated, much as the playlist ID of a newly created playlist. In the event the requested action was not successfully performed, the server responds with a error message identifying the error.

Uploading Content

[0071] As an initial matter, the system must identify the server to which a client's content should be uploaded. The site id is used to identify to which server a particular client's content should be uploaded, copied or moved. Specifically, once a client 102 logs into the system by providing its username and password, the system can identify the client's record in the CM Accounts Table 210, which provides the client's site id. Based on the site id, the system searches the CM Archive Volume Table 220 and identifies all records having the client's site id. Each of these records identifies a different server associated with the client's site id. If the system needs to access the FTP server 108 for the client (either to read the contents of or transfer a file to or from the client's account), the system identifies those CM Archive Volume Table records having the site id and a volume type corresponding to "FTP". Similarly, the system can identify repository servers 104 and streaming servers 120 associated with the client's site id by identifying those CM Archive Volume Table records having a type corresponding to "Repository" and "Streaming/Archive", respectively.

[0072] As described with reference to Figures 4a-b, the system of the present embodiment allows the client 102 to upload content to the repository servers 106 by any of several different processes. The first processes (described with reference to Figures 4a and b) transfer content files from the client's FTP account on the FTP server 108 to a repository server 106. As such, the first step in these processes is the client 102 uploading content to the FTP server 108 as noted above. The last manner is from the client's machine to a repository server 106 via HTTP upload.

[0073] As an initial step in the HTTP upload process of Figure 4b, the client 102 logs onto a web page provided by the client web server 110 by entering the client's user name and password. Having logged onto the system, the client then selects the "Upload New Content" option presented on the CM web page and identifies the content file locally stored on the client's machine that the client 102 desires to upload. Step 480. As illustrated in Figure 3b, the client 102 has the option of identifying the file by file name or by clicking the "Browse" button, which opens a window listing the client's locally stored files. Additionally, the client 102 specifies at least required file details for inclusion in the CM Database 118. Step 484. The client web server 110 proceeds to transfer the file from the client's machine onto the repository server 106 via an HTTP upload. Step 488. The system assigns a stream id to the file and creates a record in each of the CM Assets Table 212, Streams2 Table 214 and CM Stream Keywords Table 216, and the system assigns a location id and creates a record in the CM Assets Location Table 218. Step 492. The client web server 110 proceeds to write the file details, as provided by the client 102, into the appropriate fields in the CM Database 118. Finally, the system returns the stream id to the client 102, which is displayed on the CM web page with a confirmation message. Step 496. In the event any of the foregoing steps fails, the system notifies the client 102 via a message on the CM web page.

[0074] Clients 102 may also programmatically upload content from their FTP Account on the FTP server 108 to a repository server 106 by executing a script. It is to be understood that although the following embodiment utilizes a Virtual Basic ("VBS") script on an active server page (ASP) of the web server 110, the functionality described may be implemented using any programming language such as Java Script.

[0075] As an initial step in the programmatic upload and ingest process, the client 102 uploads one or more files to the FTP ingest server 108 utilizing FTP client software residing on the client's machine. Once the files have been uploaded, the client 102 may use the ASP and pass certain variables on a query string to cause a file to be ingested or to perform another content management function.

[0076] As part of the ingest request of the present embodiment, the client 102 must enter its user-name and password, as well as certain file details. Required file details include the name of the file in the client's FTP directory ("Filename"), the bit rate of the file ("Bitrate"), the title of the file ("Title"), and the author of the file ("Author").

[0077] The following optional file details may also be provided by the client 102 depending on the desired action: the copyright notice to be added to the content file ("Copyright"), a description of the file ("Description"), key words associated with the file to be used in searches ("Keywords"), an indication as to whether or not the file is secure ("IsSecure"), an alphanumeric security string for use in accessing the file if it is identified as being secure ("SecKey"), the security interval, the description of an existing playlist in the client's account ("PlaylistDesc"), an indication whether to add the file to the playlist identified by the aforementioned description or to replace the existing files associated with the aforementioned playlist with the file being ingested ("PlaylistAction"), Playlist ID (CGID) and an indication whether to automatically place the ingested file on a suitable streaming media server 120 or to only place the ingested file onto a repository server 106, where it will stay until the administrative staff manually causes the file to be placed on a streaming media server 120 ("AutoArchive"). In short, any stored file detail may be specified. The process of adding and replacing files in a playlist is discussed in greater detail below.

[0078] It is to be understood that the aforementioned file details are merely exemplary and that other file details may be considered required or optional in alternate embodiments of the present invention. Furthermore, it is to be understood such file details may be manually entered by the client 102 when submitting the ingest request, or, in an embodiment of the system wherein the client 102 enters the file details upon uploading the content files to the FTP server 108, the system may automatically retrieve the file details from the CM Database 118 based on the client's selection of the file to be ingested.

[0079] In general, the query string or command line identifies the active server page script, the location of the script, and the variables being passed to the script, including the user name, password, required file details, and any optional variables. The following is one example of such a command line:

```
http://contentmgmt.broadcast.com/vbs_script_ingest.asp?UserName=Foo&Password=Bar&
FileName=testfile.rm&BitRate=5600&Title=Test File&Author=John Doe
```

[0080] The foregoing exemplary command line passes the real media file "testfile.rm," which has a bit rate of 5600 kilobytes, is entitled "test file" and is authored by "John Doe." Furthermore, the client 102 is identified by the user name "Foo" and password "Bar." Such variables are passed to the script "Vbs_script_ingest.asp," located at "content mgmt.broadcast.com."

[0081] The following exemplary command line logs in user "Foo" using the password "Bar," uploading the file "Testfile," giving the file a Title, Author, Copyright, Description, and Keywords.

```
http://contentmgmt.broadcast.com/vbs_script_ingest.asp?username=Foo&Password=Bar&Title=Tes
File&Author=Yahoo! Broadcast&CopyRight=(c)2000 All Rights Reserved&Description= Testing T
```

[0082] When the script is called, the system receives the file details and causes the identified file to be moved from the FTP ingest server 108 to a repository server 106. Based on whether the file was successfully uploaded, the system provides to the client 102 one of two XML strings. Such XML string includes a success code field (scode), a success code description (scode_Description) and, if the upload was successful, the stream ID.

[0083] Thus, if the file was not successfully uploaded, the following XML string is returned to the client 102, wherein an scode of "1" indicates a failure and the value of the scode_description indicates the reason for the failure (e.g., an invalid username).

```
<?xml version="1.0" ?>
<response scode="1" scode_description="invalid username" />
```

[0084] In the event that the file was successfully uploaded, the system returns the following XML string wherein the scode and scode_description indicate the successful upload

```
<?xml version="1.0" ?>
<response scode="0" scode_description="Success" stream_id="999999" />
```

[0085] The system then renames the file and updates the appropriate fields in the CM database 118. More specifically, with the Stream ID assigned to the file, a record is created in each of the CM Assets Table 212, Streams2 Table 214 and CM Stream Keywords Table 216. Additionally, a location id is assigned and a record is created in the CM Assets Locations Table 218.

[0086] As will now be described with reference to Figure 4a, and continuing reference to Figure 3d, the client 102 may also perform two types of batch uploads, thereby transferring files from the client's FTP account to a repository server 106. The two types of batch uploads will be referred to as "default" and "standard" uploading. In general, default batch uploading applies default file details contained in the batch file to an entire sub-directory of content files. On the other hand, the standard batch upload process applies file details contained within the batch file on a file-by-file basis.

[0087] As an initial step, the client uploads the media files and batch file to the FTP ingest server 108. Step 446. In the case of a default batch upload, the batch file must be placed in the root directory of the client's FTP account. In the case of the standard upload process, the batch file, along with the media files, are left in the root folder of the client's FTP account.

[0088] The format of the batch file also depends upon the type of batch upload. The default batch upload file takes on the following format wherein the file is first identified as being for use in a "DEFAULT" batch upload and the relevant file details are separated by double colons as follows:

```
DEFAULT::DIRECTORY::BITRATE::TITLE::AUTHOR::COPYRIGHT::
DESCRIPTION::KEYWORDS::ISSECURE::SECURITYKEY:: SECURITY INTERVAL
```

[0089] The default batch file also includes an indication of the FTP DIRECTORY where the content files to which the default file details are to be applied are located. In the event the client 102 does not wish to supply an optional file detail, the location for such file detail is simply left blank. Furthermore, by placing the word "FILENAME" in parenthesis in either the Title or Description field in the batch file, the system will automatically include the file's file name in such field. By way of example, the following default batch file will apply to all files in the client's FTP directory named "288_Filesdirectory," which contains filing have a bitrate of 2880 kilobytes, entitled "Upload," including the file name, authored by J. Doe, a copyright notice reading "Copyright©2001," no description and no keywords, an indication that the file is secure, having the security key "mysecuritykey," and having a security interval of 122 minutes.

```
DEFAULT::288_FILES::2880::Upload(FILENAME)::J.DOE::Copyright©2001::::::Y::MY
SECURITYKEY::122
```

[0090] The standard batch upload batch file includes the same file details, however, does not include the "DEFAULT" identifier or the "DIRECTORY" identifier. Instead, each line in the standard batch file begins with the file name to which the file detail set forth in that line are to be applied. As such, the following is the format of the standard batch file:

```
FILENAME::BITRATE::TITLE::AUTHOR::COPYRIGHT::DESCRIPTION::KEYWORDS::I
SSECURE::SECURITYKEY::SECURITYINTERVAL
```

[0091] Once the media files and the batch file is uploaded to the system, the client logs into the system via the CM website and selects the "Batch Upload" option, as shown in Figure 3d. Step 450. As part of the process of selecting the Batch Upload option, the client 102 is presented with a list of both processes and unprocessed batch files. (See Figure 3d). The client 102 selects an unprocessed batch file for processing.

[0092] Upon selecting a batch file for processing, the system executes a script that causes an entry to be created in the CM Batch Jobs Table 228. Step 454.

[0093] At some later point, according to a predefined interval, a task scheduler running on the script processor 116 causes a batch script to execute. Step 458. As described below, the batch script proceeds to read and process the unprocessed batch jobs identified in the CM Batch Jobs Table 228. In an alternate embodiment, processing of the batch files is not automated and instead is manually initiated by the administrative staff via the file management server 112.

[0094] Once the batch process is started, the system proceeds to read the first record in the CM Batch Jobs Table 228. Step 462. Specifically, the system reads the record to determine whether the start time field equals a null value or whether some actual start time. Step 466. If the start time field has an actual start time, then the batch process job has already been completed and the system proceeds to read the next record in the CM Batch Jobs Table 228. Step 462.

[0095] In the event that the start time value of the current record equals the null value, the system executes the batch job. Accordingly, the system causes the first file associated with the batch file to be moved to a repository server 106 and assigns the file a stream id. Step 470.

[0096] The system proceeds to rename the file that has been moved and updates the CM database 118 to reflect the associated file details provided by the batch file. Step 474. More specifically, the system parses the batch file based on the batch file format discussed above, inserting the file details into the CM Assets Table 212, Streams2 Table 214 and the CM Stream Keywords Table 216. The system also creates a record in the CM Asset Locations Table 218. Once the file details have been entered into the appropriate records in the CM database 118, the system determines whether or not all content files associated with the batch file have been moved. Step 476. In other words, the system determines whether the end of the batch job has been reached. If the end of the batch job has been reached, then the system updates the record in the CM Batch Jobs Table 228 by entering the process end time. Step 478. If the end of the batch job has not been reached, then the system proceeds to move the next file associated with the batch file, assign a stream id, and update the CM database 118 accordingly. Steps 470, 474. Once the system cycles through the CM Batch Jobs Table 228, the processing of the batch jobs ceases until the scheduler begins the batch process again.

Selecting The Streaming Media Server on Which to Place a Client's Content

[0097] In the present embodiment, the system places each client's content on the streaming media server 120 that has the most free disk or storage space. More specifically, the script processor 116 (or other server in alternate embodiments) reads the client's record in the CM Accounts Table 210, identifying the client's site id. Using the client's site id, the system searches the CM Archive Volume Table 220 and locates all records having a volume type indicating streaming/archive. From this group of records, the system identifies only those having a platform (e.g., Windows Media or Real Media) corresponding to the format of the content file being uploaded. The resulting records are all potential streaming media serves 120 on which the content could be uploaded.

[0098] For each of the servers identified by the remaining records (each record and volume id corresponds to one server), the system proceeds to determine the available storage capacity by subtracting the used capacity from the maximum capacity. Using the records identifying all potential streaming media servers 120, the system notes the maximum storage capacity of each such media server 120, as set forth in the "max kb" field of the CM Archive Volume Table 220. The system then calculates how much of that capacity is actually being used by summing the size of all files stored on each such media server 120. More specifically, for each volume id (server), the system identifies all of the records in the CM Asset Locations Table 218 having the same volume id and identifies the stream id's in each of those records. The result is a list of all files, as identified by stream id's, residing on the server 120 corresponding to the given volume id. The system reads and sums the files sizes by accessing the CM Assets Table 212. For each volume id, the system subtracts the total storage used by the files residing on the server having that volume id from the maximum storage capacity of the server, thereby calculating the actual available storage.

[0099] The content file being uploaded is placed on the streaming media server 120 (i.e., volume id) having the most available storage capacity. The system uses the selected volume id to create a new record in the CM Assets Locations Table 218 for the archived content file.

Creation of Playlist

[00100] As will now be described with reference to Figure 5, clients 102 may also programmatically create playlists in the CM Playlists Database. The client 102 logs into his/her account via the web server 110 and selects the "Playlist" link. Steps 510 and 515. The client 102 is presented with a form into which the client 102 enters playlist details, such as description and file format, and selects and orders stream files for inclusion in the playlist. Step 520. Once the client 102 has finished entering the playlist details, the system creates a new playlist record in the CM Playlists Database and assigns a CG ID to the playlist. This CG ID is returned to the client 102 for inclusion in the client's web site 102. Step 525. The system also creates corresponding records in both the CG Database and the CG Streams Database based upon the playlist details. Steps 530 and 535. Next, the system retrieves the stream ID for the next stream file in the playlist and enters the stream ID in a record in the CG Streams Database. Step 550. The system then queries whether the end of the playlist was reached. Step 555. If yes, then the system has completed creating records for the

playlist. However, if more streams are found, the system loops back to Step 550 to create additional records.

[00101] In this way, the playlist details are stored on the CM Database 118 so as to be accessible by the playlist server 122 to dynamically generate playlist redirector files, as described below.

Dynamic Playlist Redirector Files Creation

[00102] Referring back to FIG. 1, the media servers 120 are connected to the world wide web, or some other global communications network, through the LAN. In this respect, streaming content is made available to end users 104 through the world wide web.

[00103] Upon completion of the scheduling and production phase of the web-cast event, a uniform resource locator (URL) or link is returned to the client 102 to be embedded in the client's web page. An end user 104 desiring to listen to or view the web-cast on their computer or other device can click on the URL. A playlist or CG ID (12345) is preferably embedded within the URL, as shown below:

```
<A href="http://playlistserver.company.com/makeplaylist.asp?id=12345">
```

In the illustrative URL shown above, the link points to the playlist server 122 that will execute the "makeplaylist.asp" program and dynamically generate a playlist redirector file. One skilled in the art will recognize that although the "makeplaylist" application has an Active Server Page (or ASP) extension, it is not necessary to use ASP technologies. Rather, any programming or scripting language or technology could be used to provide the desired functionality, for example, .dll.. It is preferred, however, that the program run on the server side so as to alleviate any processing bottlenecks on the end user side.

[00104] The "makeplaylist" program functions to cause the playlist server 122 to make a call to the CM database 118 to retrieve the data necessary to dynamically generate the playlist redirector file. The playlist server 122 first accesses the CG Streams Table 226 and

locates the record pertaining to the CG ID. As shown in FIG. 2, the CG Streams Table 226 contains the individual stream IDs and their respective sort orders. The stream IDs and the sort order are returned to the "makeplaylist" program which then makes a call to the Streams2 Tables 214 in the CM database 118 that contain data associated with the individual streams. Namely, the Streams2 Table 214 includes a URL prefix and a stream filename. The individual streams identifiers are also used by the "makeplaylist" program to call to the Stream-Servers Table 230 in the CM database 118. The Stream-Servers Table 230 includes the location or (DNS) hostname of the particular media server 120 containing the stream file associated with a particular stream identifier. Using the URL prefix, the hostname, and the stream filename, the "makeplaylist" dynamically generates a URL for each stream file. An example of such a URL is listed below, wherein the URL prefix is “mms://”, the hostname is "mediaserver.company.com" and the stream file name is "filename.asf.":

```
<"mms://mediaserver.company.com/filename.asf">
```

[00105] Using the individual stream URLs, the "makeplaylist" program then dynamically generates a redirector file to be passed to the media player stored on the end user's computer 104. An example of a redirector file for use with Windows Media Technologies (WMT) is shown below:

```
<ASX>
   <ENTRY>
      <REF HREF="mms://mediaserver.company.com/stream1.asf">
      <REF HREF="mms://mediaserver.company.com/stream2.asf">
      <REF HREF="mms://mediaserver.company.com/stream3.asf">
   </ENTRY>
</ASX>
```

[00106] One skilled in the art will recognize, of course, that different media technologies utilize different formats of redirector files and, therefore, the term "redirector

file" is not limited to the ASX-type file shown above. Lastly, the end user's media player pulls each identified stream file from the media server 120 identified in the redirector file in the order in which it appears in the file.

[00107] It is to be understood that other system architectures are within the scope of the present invention. For example, certain embodiments include sites each having its servers on a separate LAN coupled to the administrative components (e.g., playlist server, file management server, script processor, database) via a secure SMTP connection. Furthermore, certain embodiments do not, as in the embodiment of Figure 1, include repository servers that maintain backed-up copies of files. One such embodiment that does not include repository servers for storing backups will now be described with reference to the schematic of Figure 6, the workflow diagram of Figure 7 and the CM Database schematic of Figure 8.

[00108] As illustrated in Figure 6, the architecture of the present embodiment essentially mirrors that of the embodiment of Figure 1. Indeed, the operation of the components illustrated therein is based on, and in many respects identical to the components described above with reference to Figure 1. However, rather than including repository servers for maintaining a backup of content stored on the streaming media servers, the present embodiment includes storage servers 606 that maintain a copy of each item of content to the exclusion of maintaining a duplicate copy on a streaming media server 620. In short, other than an external backup, such as on a tape backup system or mirror or redundant disk system, preferably the only copy of each item of content for a given site is on the storage servers 606. Accordingly, as described in greater detail below, when providing access to the content, the streaming media servers 620 performs a LAN-based read of the appropriate content on the storage servers 606.

[00109] It should be appreciated that coupling multiple media servers 620 to a single storage server 606 having the only copy (vis-à-vis those media servers 620) provides several

benefits. For example, the content management service provide saves money because less storage is used. Additionally, because multiple media servers 620 may access and, thus, stream the same item of content, clients 102 will encounter fewer instances of end users 104 not being able to receive the content because a media server 620 is already providing the maximum number of streams. While so utilizing the storage servers 606 and media servers 620 is preferable, it is within the scope of the present invention to have some sites employ an architecture wherein a repository or backup server and the media server each has a copy of the content, and have other sites employ a architecture where media servers do not have a copy of the content.

[00110] Unlike the embodiment of Figure 1, the present embodiment also includes site-specific file manipulation servers 607. As described in greater detail below, each such file manipulation server 607 causes files to be moved from each site-specific FTP ingest server 608 to the appropriate storage server 606.

[00111] It is also within the scope of the invention to have one or more sites in the present embodiment on separate networks in communication with the administrative servers and processors via SMTP or other connection, which themselves may be on a separate network.

[00112] Having described the architecture of the present embodiment, its general operation will now be described with reference to the workflow diagram of Figure 7. It is to be understood, however, that aside from the differences noted or otherwise to be understood based on the following description, the components of the present embodiment operate as like named components of the embodiment of Figure 1. As illustrated therein, clients 102 are able to upload content either via an HTTP upload to the web server 610 or via an FTP upload to the client's primary site, as identified in the client's account record in the CM database 618. The web server 610, in turn, performs a LAN-based file move to the appropriate storage

server 606. In alternate embodiments, the read is via a direct, point to point connection. Under direction of the file management server 612, each site-specific file manipulation server 607 causes the content previously uploaded to the FTP ingest server 608 to be moved via the LAN to the appropriate storage server 606.

[00113] With the content stored on the appropriate storage server 606, each streaming media server 620 receiving a request for content will read the file from the appropriate storage server 606 (in the present embodiment, via a LAN based file read) to obtain the content and ultimately provide it to the end user 104 requesting it.

[00114] Because the functionality of the present embodiment, in large part, parallels that of the embodiment of Figure 1, the CM database 618 illustrated in Figure 8, largely parallels the CM database 118 of Figure 2, with similarly named tables and fields providing similar functionality and storing similar information unless stated or understood otherwise. However, the present CM database 618 includes variations to accommodate the use of storage servers 606, as well as to provide additional functionality. Such variations and additional functionality will now be described with reference to the database schematic of Figure 8 and continuing reference to the schematic of Figure 6.

Multiple Media Servers Per Storage Server

[00115] As noted above, an item of content stored on a storage server 608 is accessed by each media server 620 that provides an end user 104 access to such content. Furthermore, in the present embodiment, multiple media server 620 may access a given item of content on a storage server 608. Consequently, the CM database 618 tracks which one or more media servers 620 are coupled to, and may access, a given storage server 608. To accommodate this potential one to many relationship, the CM database 618 includes the archive volume-media server ("AVMS") Table 846 and the Media Servers Table 848. Each record in the AVMS Table 846 associates a volume ID with a media server, as identified by media server (MS) ID,

that is permitted to access content stored at the volume ID. Each MS ID, in turn, corresponds to a record in the Media Servers Table 848, which provides media server identifying information, as indicated. The "status" column stores one of the following values: CE (Check Enabled); MD (Manually Disabled); and ME (Manually Enabled).

[00116] As with the prior embodiment, each volume ID is essentially defined in a record in the Archive Volume Table 820. The volume type field in the archive volume table 820 specifies to which type of server (FTP ingest server file manipulation server, storage server, or web server) the volume ID corresponds. The server category field identifies the server as being used for business services or public services or any other category useful for reporting and maintenance purposes. The platform field is used to identify the media platform corresponding to the volume type or to indicate that the platform is not applicable or to indicate all platforms (e.g., a character such as "B").

[00117] The "archive server" field (in Archive Volume Table 820), and the "server name" field (in the Media Servers Table) store the computer's WINS/NETBIOS name for LAN-based file access (for example, PUB01), so that shared resources (such as directories, etc.) on the server can be accessed via that name (for example, \\PUB01\PRoot1).

Account Categories

[00118] The present embodiment has various other features reflected in the CM database 618. For example, as illustrated in the Accounts Table 810, each account is associated with an account category, as identified by account category ID (acct cat ID). Each account category ID corresponds to an entry in the Account Categories Table 850, which sets forth the name of the category. In general, an account category refers to a type of account, such as Internet radio or television or corporate account. The content management service provider may use such information for reporting and, because accounts can be segregated into broad categories, for simplifying the searching of accounts.

Time Zones

[00119] The Accounts Table 810 also includes a time zone ID field (TZ ID), which identifies a client-selectable time zone. Based on the client's selection, and the corresponding time zone ID, all information presented to the client 102 is based on the selected time zone. More specifically, the date and time field stored within the data base 618 are stored in a Greenwich meantime, however prior to being displayed to each client 102, the system converts the time into that of the client-selected time zone. Similarly, dates and times entered by each client 102 is converted to Greenwich Mean Time prior to being stored in the database 618. In the present embodiment, each time zone ID corresponds to a record in a time zone table (not shown), identifying the time zone and providing the relative difference between the Greenwich Mean Time and that time zone. Thus, by way of example, the expiration date, deletion date, creation date and any other time related information is presented in the Client's selected time zone frame.

Default Expiration Interval

[00120] The Accounts Table 810 also includes a default expiration interval field (default exp interval). In the event a client 102 fails to specify an expiration date for a

particular file being uploaded, the system automatically adds the value stored (e.g., minutes hours, days, weeks, and the like) in the default expiration interval field to the date and time that the file is uploaded, thereby arriving at the default expiration date for the file. Such default expiration interval may be set by the client 102 during registration, or at any other point in the process.

Accessors

[00121] The current embodiment also includes tables that specify who may access and alter account information. Specifically, the Account-Accessors Table 852 and the Accessors Table 854 specify, for each account ID, one or more individuals (accessors) that may access the account. As illustrated, there may be several records in the account accessors table 852 for each account ID, thereby specifying multiple accessors for each account. Such accessors may be individuals external to the content management system and internal to the client 102, or administrative personnel (internal to the service provider).

Stream Categories

[00122] Unlike the embodiment of Figure 2, the Streams2 Table 814 includes a field specifying each stream's category (category ID). Stream categories are essentially any type or division of content deemed helpful to the content management service provider or the clients 102. Furthermore, categories may be a broad or as focused as desired. In the present embodiment, stream categories are hierarchical in nature, with only the most detailed category being assignable to a stream. For example, the system may provide a first level category of "sports," with a second level of multiple subcategories, such as "baseball," "basketball," "football," and the like. Each subcategory, in turn, includes more detailed third level of subcategories, for example, under basketball, could be "NBA," "WBA" and "College." Only these most detailed categories are assignable in the present embodiment.

[00123] Preferably, each client 102 specifies a default stream category to be applied to uploaded streams that is stored in the Account Table 810 (in the default stream cat ID field). When new content is uploaded via the client web site or programmatically, the client 102 has the option of using the default category or supplying a new category. In either case, the system creates a new record in the Streams2 Table 814 and populates the category ID field accordingly.

[00124] Each category ID corresponds to an entry in the Categories Table 838, which specifies the name of the category, the parent category of the subject category, the type of category (i.e., whether the category is assignable or not) and the level of the category in the category hierarchy. This system retrieves this information and constructs a visual display of the hierarchy on the client-side web page and permits each client to select the desired category. The Categories Table 838 is also keyed to the default category ID in the Account Table 810.

Replication Sites

[00125] The present embodiment also allows each client 102 to have multiple sites – a primary site and one or more replication sites – associated with its account. In general, when uploading content via the client website, the client 102 may specify any one or more of the replication sites to which the content will be uploaded (in addition to the primary site). The client's primary site is identified by the site ID stored in the client's record in the Account Table 810. The replication sites are identified in the Account Sites Table 856. More specifically, the Account Sites Table 856 includes records that associate multiple site IDs to the client's account ID. Of course, there is no requirement that a particular account have any replication sites associated therewith.

[00126] In operation, when a client 102 uploads content, the content gets passed to the storage server 606 associated with the client's primary site. As with the previous embodiment

the stream ID assigned to the content gets returned to the client 102. Thus, in the context of the process of Figure 4b, an additional step 486 would be performed, namely, that if the client 102 is authorized to replicate its content to any site other than their primary site, the client optionally selects one or more replication sites to which the file being uploaded should be replicated.

[00127] In the background, when the client 102 uploads the content and identifies a replication site to which the content should be stored, an entry is made in a replication queue with the information identifying the content and the storage server 606 of the replication site. Periodically, a script checks the replication queue for entries and, for each entry in the queue, causes the appropriate content to be stored in the specified storage server 606 corresponding to the replication site. Thus, in the context of the process of Figure 4b, step 498 would be added to account for the system replicating the files to replication sites if one or more replication sites were chosen by the client while specifying file details. It should be noted that for each storage server 606 containing a copy of the content, there is a record in the Assets Location Table 818, each corresponding to the same stream ID and file name. This stream ID, universal to all copies of the file at the primary and replication sites, simplifies management of the copies, as there is only one entry in the Assets Table 812.

[00128] The ability to replicate content and the lack of a back-up copy of each item of content on a repository server, as in the embodiment of Figure 2, results in certain changes in the Asset Locations Table 818 status field. More specifically, the status is never "backing-up" and, instead, may indicate that the content is: available; being deleted; has expired; failed to be properly stored; being moved; new; being trashed; or being replicated, as described above. Notably, in the present embodiment, there is no need to copy a content file from one storage server 606 to another within the same site because multiple media servers 620 can access the content.

Recycle Bin

[00129] The status of being "trashed" is used in conjunction with a figurative "recycle bin," like that describe above with reference to Figure 3j. When browsing or searching content, a client 102 may select one or more items of content for deletion, or placement in the recycle bin. When selected for deletion, the system changes the value of the status field for the corresponding record in the Asset Locations Table 818 to "trashed." The system also creates a record in the Deleted Assets Table 844 and enters as the deletion date a date (and time) some predetermined period in the future (e.g., one day, one week). A script running on the script processor or other server periodically searches (e.g., every three hours; every day; and the like) for records in the Asset Locations Table 818 having a status of "trashed." For each such record, the system locates the corresponding record in the Deleted Assets Table 844 and reads the deletion date. If the deletion date equals or is greater than (i.e., after the then current date (and time), then the system proceeds to delete the content. Until such time as the content is actually deleted, the client 102 may restore the content from the recycle bin via a web page identifying the contents (or, in alternate embodiments, programmatically). When the client 102 selects an item of content for restoration, the system changes the status in the Asset Locations Table 818 from "trashed" back to "available" and deletes the record in the Deleted Assets Table 844.

Stream Groups

[00130] The present embodiment also allows clients 102 to logically group Streams into stream groups for ease of presentation to end users 104. More specifically, each logical stream group includes content files having the same susceptive content, but different media formats and/or bit rates. When a client 102 uploads content, the client web page presents the client 102 with the option of associating the uploaded content with an existing stream group or with a newly created stream group.

[00131] Stream groups are reflected in the Streams2 Groups Table 858, the Streams2 Group Table 860 and the Streams2 Group Streams Table 862. More specifically, the Streams2 Groups Table 858 contains a record for each Stream Group, as identified by Stream Group ID, associated with a particular account. The Streams2 Group Table 860 provides Stream information corresponding to each Stream Group, as identified by Stream Group ID. Such Stream information includes, for example, the Stream Group name, and the title, author, description, associated key words and type of Stream Group. Lastly, the Streams2 Group Streams Table 862 identifies each Stream comprising a given Stream Group. Additionally, the Table 862 identifies the corresponding bit rate for the particular stream. By characterizing streams into groups, clients are able to easily track a particular piece of content and provide to their end-users 104 the stream having the appropriate media format and bit rate.

[00132] The client-side web site also preferably presents a web page for managing stream groups that parallels the page provided for managing playlists. For example, a stream group may be selected, identifying information, such as name, title, description, keywords and the like maybe edited, and the content of the stream group may be modified by adding or removing content (by adding or deleting, respectively, a record in the Stream2 Group Streams Table 862).

[00133] Alternatively, as with all content management functions, the stream groups may be edited via the client-side web site as described by passing variables in a query string to a script. In one such embodiment, the script requires values for three parameters: username-password; and stream group action, which may indicate: adding a stream to an existing group; creating a new group; editing a group; removing a stream from a group; listing the stream groups; or listing the groups and the contents thereof. Depending upon the action specified, the following optional parameters must be specified: name; title;

description; author; keywords; miscellaneous, metadata of the group; whether promotable; stream group ID (used to specify a group to be edited or removed or to edit streams within a group); and stream action. Stream action, in turn, may indicate: adding a stream; removing a stream; removing all streams; stream ID(s) (specified when adding or removing a particular stream).

[00134] The Stream2 Details Table 870 is related stream groups. Each record in the Stream2 Details Table 870 corresponds to a stream, as specified by stream ID. The information stored for the streams includes: (1) miscellaneous metadata, such as an extension of any other field that was truncated due to length of field restrictions; (2) if the stream supports multi bit rates, a stream group is created containing all bit rate versions of the stream; the stream group ID is stored in the multi bit rate stream group ID field (mbr sg id); and (3) and indication of whether or not the stream supports multiple bit rates (is multi bit rate).

[00135] Collectively, the information is used to identify those streams that support multiple bit rates and to identify the particular stream group (out of potentially many to which the stream belongs) corresponding to the stream's multi bit stream group.

Stream Folders

[00136] The present embodiment also allows clients 104 to organize content into logical folders. The Streams Folders Table 864 and Folder Content Table 866 permit this. The Streams Folders Table 864 identifies folders, by folder ID, associated with each account ID. Each record also specifies the folder name and the ID of the parent folder, as folders may be hierarchical. The Folder Content Table 866 identifies the content within each folder, as identified by folder ID. More specifically, the present embodiment allows streams, stream groups, and playlists to be included within a folder. Consequently, the folder content Table 866 is designed to specify any of the foregoing types of content. To this end, the folder

content Table 866 includes a content type field, which specifies to which of the foregoing types of content the record pertains. Each record also includes a content ID field which specifies the ID of the content to which the record pertains. Thus, if the content type specifies playlists, the content ID field will include the appropriate CG ID; if the content file specifies Stream Group, the content ID field specifies the Stream Group ID; if the content type specifies Stream, then the content ID field includes the Stream ID.

[00137] Preferably, the client web page includes a separate page that provides the client 102 the opportunity to both create folders and edit existing folders. To create a folder, the client 102 simply needs to provide the folder name and any parent folder to which the new folder relates and to select content comprising the folder. In essence, for each new folder created, a new record is created in a Streams Folders Table 864, and for each content added to the folder, a new record is created in a Folder Content Table 866. When editing an existing folder, the client web page preferably presents to the client 102 a listing of folders, as identified in the Streams Folders Table 864, and upon the client's selecting a folder, the system retrieves the details corresponding to the content ID stored in a Folder Content Table 866 and displays those details to the clients 102.

Promotability

[00138] The present embodiment also provides the clients 102 with the option of marking content as promotable. By marking content promotable, the client 102 essentially authorizes the content management service provider to select the content and place it on any web site operated by the service provider. For example, if a client 102 provides news clippings, it may mark the content promotable, thereby authorizing the service provider to provide a link to the content on the service provider's news page. As illustrated in the CM database 618, Stream Groups may be identified as promotable in the Streams2 Groups Table 858 and folders may be identified as promotable in the Streams Folders Table 864. Clients

102 may indicate that a particular content is promotable when up-loading the content, or creating or editing a folder and stream group. In alternate embodiments, individual streams may be specified as promotable or not by including a promotable field in either of the Assets Table 812 or Streams2 Table 814.

[00139] The Database 618 also includes an FTP Info Table 868 for storing the site ID for each FTP Ingest Server 608, as identified by hostname. The Table 868 also identifies the username and password needed to access each FTP Ingest Server 608.

Playlist Resolution

[00140] Because the present embodiment does not store streams on the media servers 620 and because clients 102 can have replication sites, the processes of creating the playlist URL and creating the redirector file is different than described with regard to the embodiment of Figure 1. The differences will now be described with reference to Figures 6 and 8.

[00141] In the event a client 102 has one or more replication sites and the content to be presented is stored at multiple sites, the client preferably selects the content and the site from which the client desires it to be streamed. For example, a client may have a Japanese language stream; if stored on a site in Japan and a site in the United States, the client may desire it be streamed from the site in Japan. Consequently, the playlist URL of the present embodiment includes and indication of not only the content (e.g., stream ID, CG ID), but also an indication of the location (e.g., site ID) from where the content should be streamed. An exemplary playlist URL, in which the playlist server 622 is "playlist.company.com", the program to dynamically generate the redirector file is "makeplaylist.dll", the CG ID is "12345" and the site ID is "16", is as follows:

<href="http://playlist.company.com/makeplaylist.dll?CG_ID=12345&SITE_ID=16">

[00142] The playlist URL is constructed generally as described above. However, when the playlist is created, the system preferably provides the client 102 with a series of

URLs, each corresponding to a different site at which the content is stored, and specifies the site name and time zone, as retrieved from the Sites Table 834. Alternatively, the client manually enters the site ID.

[00143] As with the previous embodiment, the makeplaylist component dynamically generates a redirector file based on the CG ID and, in the present embodiment, the site ID. More specifically, the component causes the playlist server 622 to access the Content Group Streams Table 826 and identify the streams within the identified playlist and their sort order. For each such stream, the server accesses the Streams2 Table 814 and retrieves the URL prefix and filename. As described in greater detail below, because the content is stored on the storage servers 606, and not the media servers 620, the filename refers each stream's filename as stored on the storage server 606.

[00144] The playlist server 622 also accesses the Stream-Servers Table 830 to identify the hostname of the server on which each stream resides. It should be noted that because the client has a replication site with the stream, there will be multiple records in the Stream-Servers Table 830 for each stream ID. Consequently, the playlist server 622 must determine which record corresponds to the site identified in the playlist URL. The server does this by accessing the Servers2 Table 832 and identifying the record having both one of the hostnames previously identified from the Stream-Servers Table 830 and the site ID from the URL. The hostname from this record is used in creating the redirector file. The system does this for each stream in the playlist.

[00145] In certain embodiment, a site will include multiple streaming media servers 622 capable of streaming a given content file. In other words, multiple media servers 622 can be coupled to a single storage server 606. In such an instance, the system must determine which of the media servers 620 to use. This many-to-one relationship will be reflected by multiple records in the Servers2 Table 832 having the appropriate site ID. In one

embodiment, the decision is random or pseudorandom. However, in the present embodiment, the decision is based on load balancing and the weight field in the Servers2 Table 832.

[00146] The weight field is a preset indication of the percentage of streams to be served from a given media server 620, as compared to other media servers 620 capable of streaming the same content. For example, if two media server 620 were coupled to a given storage server 606 and could stream a file, and one could serve twice the streams of the other, then the weight of the higher capacity server would be approximately 66 and the weight of the lower capacity server would be approximately 33. When determining from which server to stream, the system would analyze the ratio of current streams being served of the two servers and utilize the server streaming a percentage of the total streams less than its weight. In other words, the system would seek to maintain the ratio of streams served equal to the ratio of the weights.

[00147] Once the media server hostname is determined, the component on the playlist server 620 obtains the filepath prefix from the record in the Stream-Servers Table 830 corresponding to the relevant stream id and hostname. As will be described in greater detail, the filepath prefix serves as a mounting point for the media server 620 so that the media server 620 can access the storage server 606 to read the requested stream. This filepath prefix is appended to the beginning of the filename and placed in the redirector file.

[00148] An exemplary redirector file, in which the playlist comprises a single stream having a stream ID of "2000" and a filename of "/test/10/2000.asf", the media server 620 hostname is "mediaserver.company.com", and the mounting point is "proot1/PubShare01", is as follows:

```
<ASX>
   <ENTRY>
      <REF
HREF="mms://mediaserver.company.com/proot1/PubShare01/test/10/2000.asf">
   </ENTRY>
</ASX>
```

[00149] It should be noted that in the present embodiment, the mount point is the same as the root share and share information for the server volume ID. Therefore, the media server 620 may read the correct contents of the storage server 606. Placing the filepath prefix in the redirector file is useful in situations where files stored on a storage server 606 are not allowed to be stored in one single share, but multiple shares are preferably created for the sake of efficiency of the backup and recovery processes (e.g., mirroring a portion of all shares). In this scenario, the hard drive(s) storage space on a storage server 606 is required to be stored under one of those shares.

[00150] By way of illustrative example, the Archive Volume Table 820 currently has the following record for a share created on the PUB01 storage server 606. The record in the Archive Volume Table 820 for that filer/share combination is as follows: cm_vol_id=142; volume_type=S; archive_server=\\PUB01; share=/PubShare01; server_addr=pub01.broadcast.com; platform=B; max_content_kb=131072000; status=CE; site_id=16; local_path=NULL; checking_status= [not used]; server_category=PUB; root_share=/proot1; attach_type=N; total_kb_used=90455873.

[00151] This would cause a record to be created in the Stream –Servers Table 830 having a filepath prefix of /proot1/PubShare01.

[00152] Let's say that the file path stored in the filename column in streams2 table was '/test/10/2000.asf'. Assuming that the stream ID 2000 was a Windows Media stream that is accessed with the mms protocol, then playlist will construct the final URL as follows:

mms://mediaserver.company.com/proot1/pubshare01/test/10/2000.asf

[00153] In this scenario, for this URL to work correctly, the streaming server must have a mount point with the same name as the first component of the URL after the server hostname. In the above case example, the server name is mediaserver.company.com, and the first component of the URL after that hostname is "proot1." So, a mount point having the name proot1 is created on the streaming server such that it points to the following LAN file storage path: \\PUB01\proot1

[00154] A root share is optional, and is used only to reduce the number of mount points needed to be created on a streaming server in a situation whereby the storage device or hard drive on a storage server 606 has more than one share on it (for such reasons as efficiency of the backup processes).

[00155] A storage share on a storage server, however, is required because the systems will store a file in that share (further organized in account-specific directory, such as 'test' in the above example, and a hashed directory such as '10' in the above example). In the above example, the storage server 606 \\PUB01 has the following shares: proot1 and pubshare01.

[00156] Those skilled in the art will recognize that the method and system of the present invention has many applications, may be implemented in many manners and, as such, is not to be limited by the foregoing exemplary embodiments and examples. In this regard, any number of the features of the different embodiments described herein may be combined into one single embodiment. Moreover, the scope of the present invention covers conventionally known and future developed variations and modifications to the system components described herein, as would be understood by those skilled in the art.

CLAIMS

1. A method of providing access to digital content, the method comprising:

receiving digital content from a client;

storing the digital content on a server having a hostname, the digital content having a filename;

assigning a unique identifier to the digital content;

providing the client with the link containing the unique identifier;

receiving a request for the content, the request based on activation of the link;

determining the hostname and filename based on the unique identifier; and

creating a redirector file, the redirector file including the hostname and filename.

2. The method of claim 1, wherein the unique identifier is a stream identifier.

3. The method of claim 1, wherein the unique identifier is the name of the digital content.

4. The method of claim 1, wherein the digital content is a streaming media file.

5. The method of claim 1, wherein the digital content is a playlist comprising multiple streaming media files.

6. The method of claim 1 wherein the link specifies a program for determining the hostname and filename based on the unique identifier.

7. A method of providing an end user access to digital content, the method comprising:

causing digital content to be stored on a server having a hostname, the digital content having a filename when stored;

receiving a unique identifier to the digital content;

publishing a link for activation by the end user, the link including the unique identifier of the digital content, wherein activation of the link causes resolution of the unique identifier into the hostname and filename and causes the digital content to be provided to the end user based on the hostname and filename.

8. The method of claim 7, wherein the unique identifier is a stream identifier.

9. The method of claim 7, wherein the unique identifier is the name of the digital content.

10. The method of claim 7, wherein the digital content is a streaming media file.

11. The method of claim 7, wherein the digital content is a playlist comprising multiple streaming media files.

12. A system for managing digital content and provide digital content to end users, the system comprising:

one or more first servers configured to receive digital content;

one or more storage servers configured to store digital content received by the first servers;

a plurality of media servers coupled to at least one of the storage servers, the media servers configured to receive a request to experience an item of digital content and, in response to the request, read the item of digital content stored on the at least one storage server.

13. The system of claim 12, wherein the first servers include a web server.

14. The system of claim 12, wherein the first servers include an FTP ingest server.

15. The system of claim 12, wherein the first servers are the storage servers.

16. The system of claim 12 wherein the request is from an end user and the media servers are further configured to communicate the item of content to the end user.

17. The system of claim 12 further comprising a playlist server configured to resolve the request and select one of the media servers to read the item of digital content.

18. The system of claim 17 wherein each media server has a mounting point associated therewith, the mounting point referring to the at least one storage server.

19. A method of managing digital content of a client, the method comprising:

associating the client with a primary site;

associating the client with one or more replication sites;

providing the client an option to upload digital content to both the primary site and any one or more of the replication sites;

receiving a request to upload an item of digital content, the request specifying one or more replication sites; and

based on the request, uploading the item of digital content to the primary site and the specified replication sites.

20. The method of claim 19 further comprising associating the item of content with a universal identifier identifying the item of content as uploaded on the primary site and the specified replication sites.

21. The method of claim 19, wherein the associating the client with replication sites is performed in response to the client specifying replication sites.

22. The method of claim 19 further comprising:

receiving a request to provide the uploaded content; and

providing the uploaded content from the primary site or the one or more specified replication sites based on client input.

23. A method of managing digital content of a client, the method comprising:

receiving a selection of a time zone from the client;

associating the client with the time zone;

storing time related information for the client based on a standard time;

converting the stored time related information based on a standard time to time related information based on the time zone associated with the client; and

providing to the client the time related information based on the time zone.

24. The method of claim 23 wherein the universal time is Greenwich Mean Time.

25. The method of claim 23 wherein the converting includes adding or subtracting a number of hours.

26. The method of claim 23 wherein the time related information is a date and time.

27. The method of claim 23 further comprising:

receiving from the client time related information based on the time zone;

converting the time related information based on the time zone into time related information based on the standard time; and

storing the time related information based on the standard time.

28. A method of providing digital content, the method comprising:

generating a playlist uniform resource locator (URL) identifying digital content and a one of a plurality of sites having a copy of the digital content;

identifying a hostname associated with the digital content and the one site, the hostname identifying a media server associated with the one site; and

providing the digital content from the media server.

29. The method of claim 28 wherein generating the playlist URL includes receiving from a client providing the digital content a selection of the one site from the plurality of sites.

30. The method of claim 28 wherein the one site includes a storage server and multiple media servers and wherein the digital content is stored on the storage server, the identifying hostname includes selecting the media server from the multiple media servers.

31. The method of claim 28 wherein selecting one of the multiple media servers is random.

32. The method of claim 28 wherein selecting one of the multiple media servers is based on relative load on the multiple media servers.

33. The method of claim 28 wherein identifying the hostname includes selecting the media server from multiple media servers.

34. The method of claim 28 wherein the digital content is a playlist comprising multiple streams and wherein the identifying the hostname is performed by each of the multiple streams.

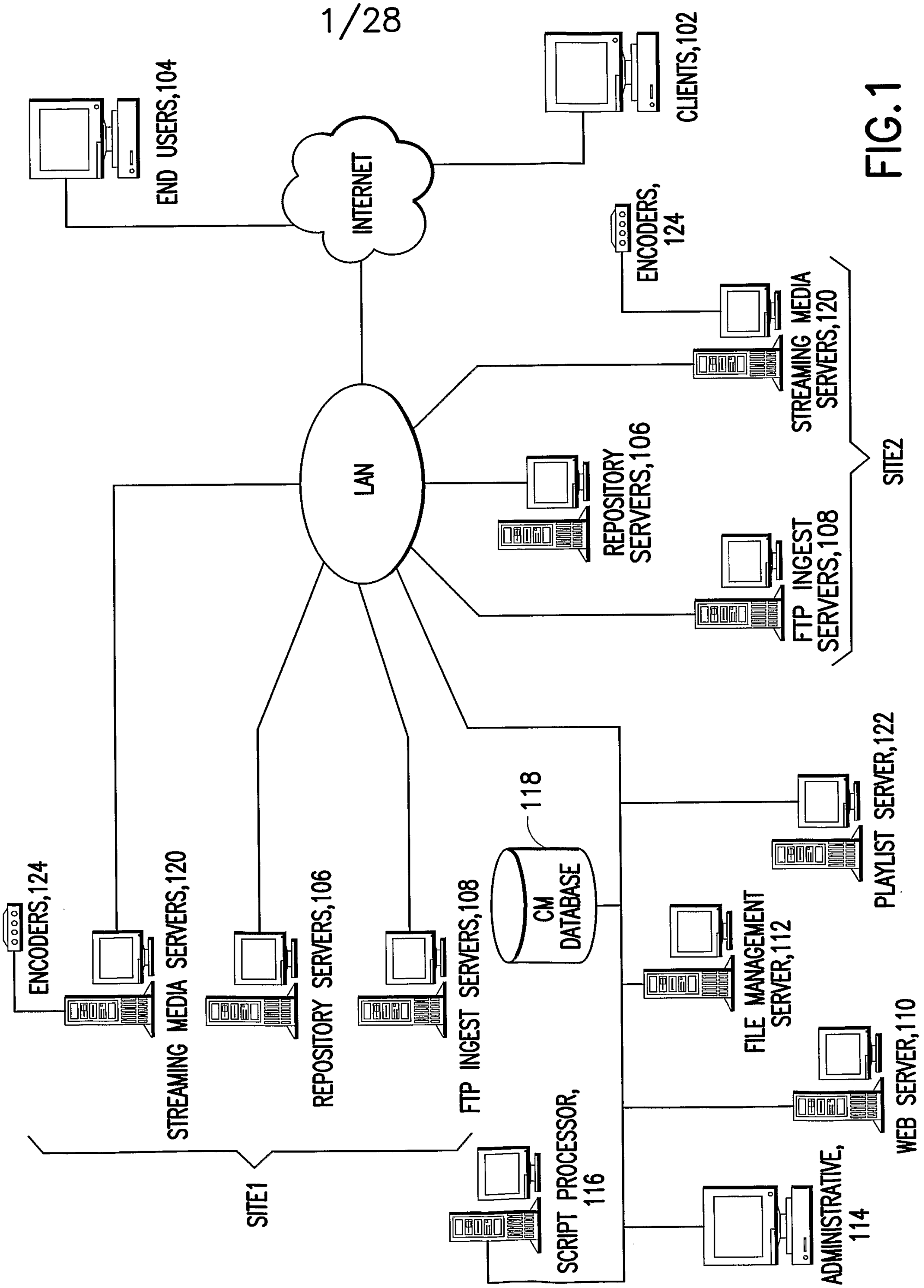
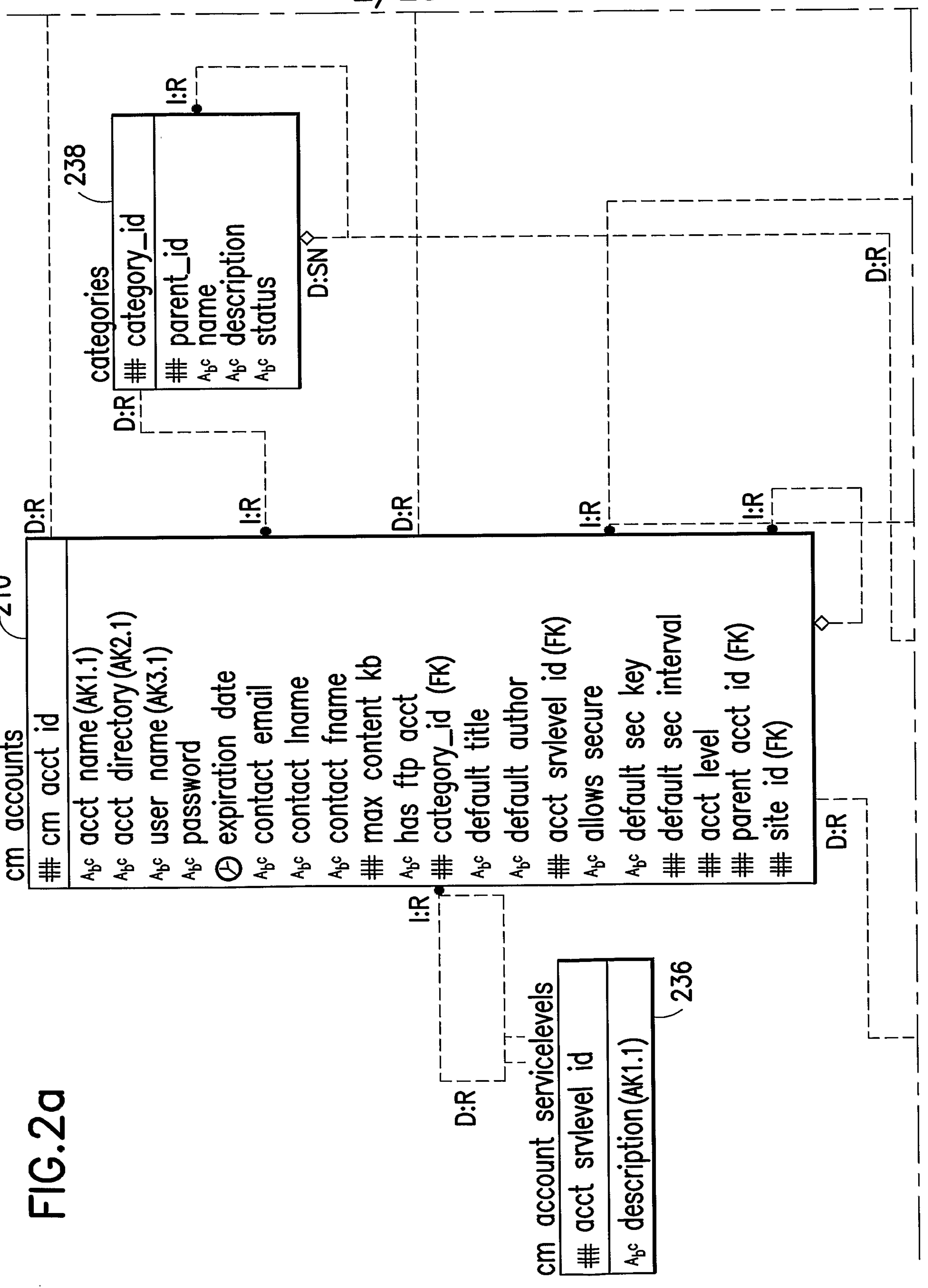

END USERS,104
CLIENTS,102
INTERNET
ENCODERS, 124
LAN
STREAMING MEDIA SERVERS,120
REPOSITORY SERVERS,106
FTP INGEST SERVERS,108
SITE2
ENCODERS,124
STREAMING MEDIA SERVERS,120
REPOSITORY SERVERS,106
FTP INGEST SERVERS,108
SITE1
118
CM DATABASE
PLAYLIST SERVER,122
FILE MANAGEMENT SERVER,112
WEB SERVER,110
SCRIPT PROCESSOR, 116
ADMINISTRATIVE, 114

FIG.1



categories
238
category_id
parent_id
name
description
status
I:R
D:SN
D:R
cm accounts
210
cm acct id
acct name (AK1.1)
acct directory (AK2.1)
user name (AK3.1)
password
expiration date
contact email
contact lname
contact fname
max content kb
has ftp acct
category_id (FK)
default title
default author
acct srvlevel id (FK)
allows secure
default sec key
default sec interval
acct level
parent acct id (FK)
site id (FK)
D:R
I:R
D:R
I:R
I:R
D:R
D:R
I:R
D:R
cm account servicelevels
236
acct srvlevel id
description (AK1.1)

FIG.2a

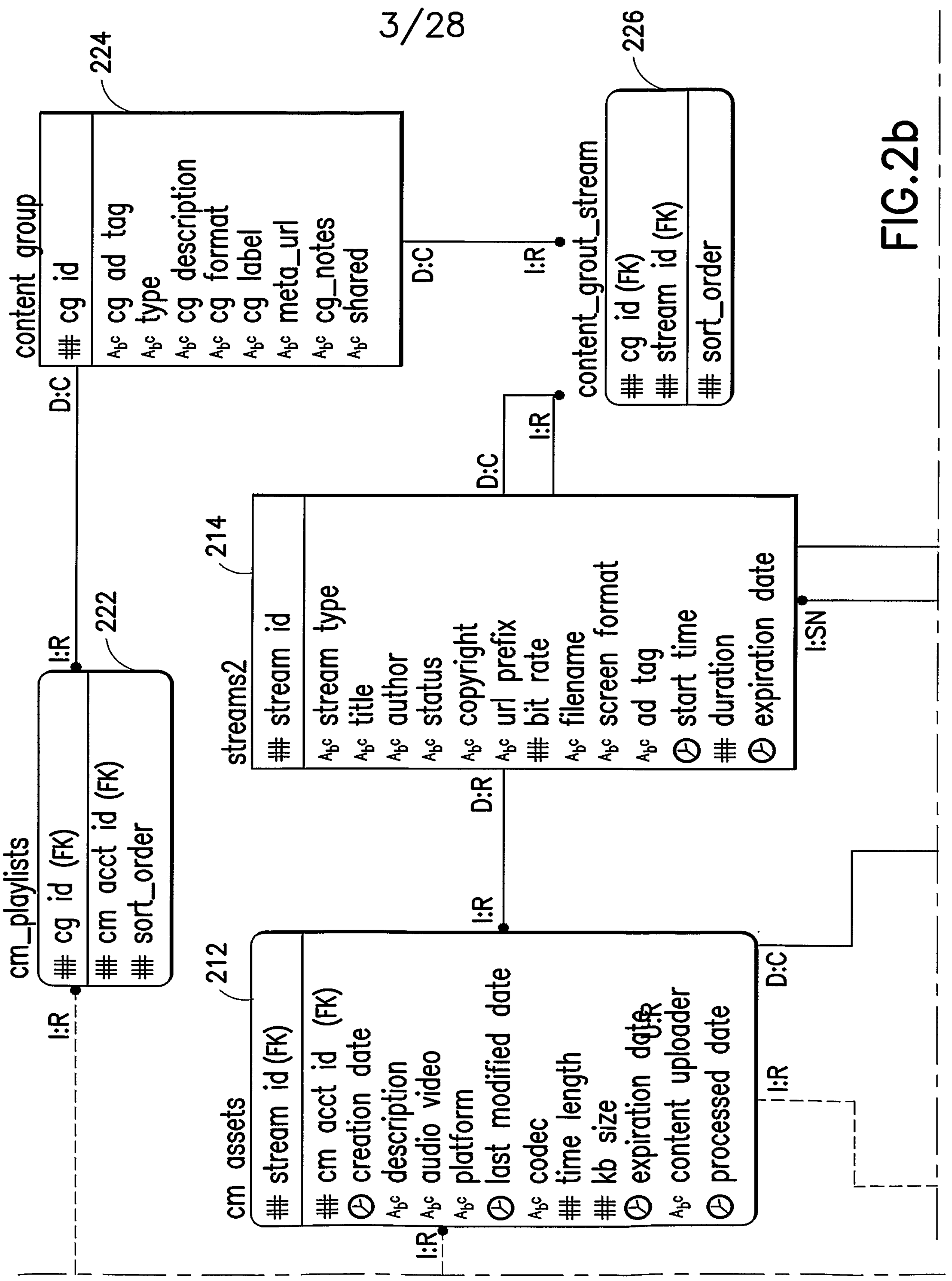

224
content_group
# cg id
Abc cg ad tag
Abc type
Abc cg description
Abc cg format
Abc cg label
Abc meta_url
Abc cg_notes
Abc shared
D:C
I:R
226
content_grout_stream
# cg id (FK)
# stream id (FK)
# sort_order
D:C
I:R
222
cm_playlists
# cg id (FK)
# cm acct id (FK)
# sort_order
I:R
I:R
D:C
214
streams2
# stream id
Abc stream type
Abc title
Abc author
Abc status
Abc copyright
Abc url prefix
# bit rate
Abc filename
Abc screen format
Abc ad tag
start time
# duration
expiration date
I:SN
D:R
I:R
212
cm assets
# stream id (FK)
# cm acct id (FK)
creation date
Abc description
Abc audio video
Abc platform
last modified date
Abc codec
# time length
# kb size
expiration date
U:R
Abc content uploader
processed date
D:C
I:R
I:R

FIG.2b

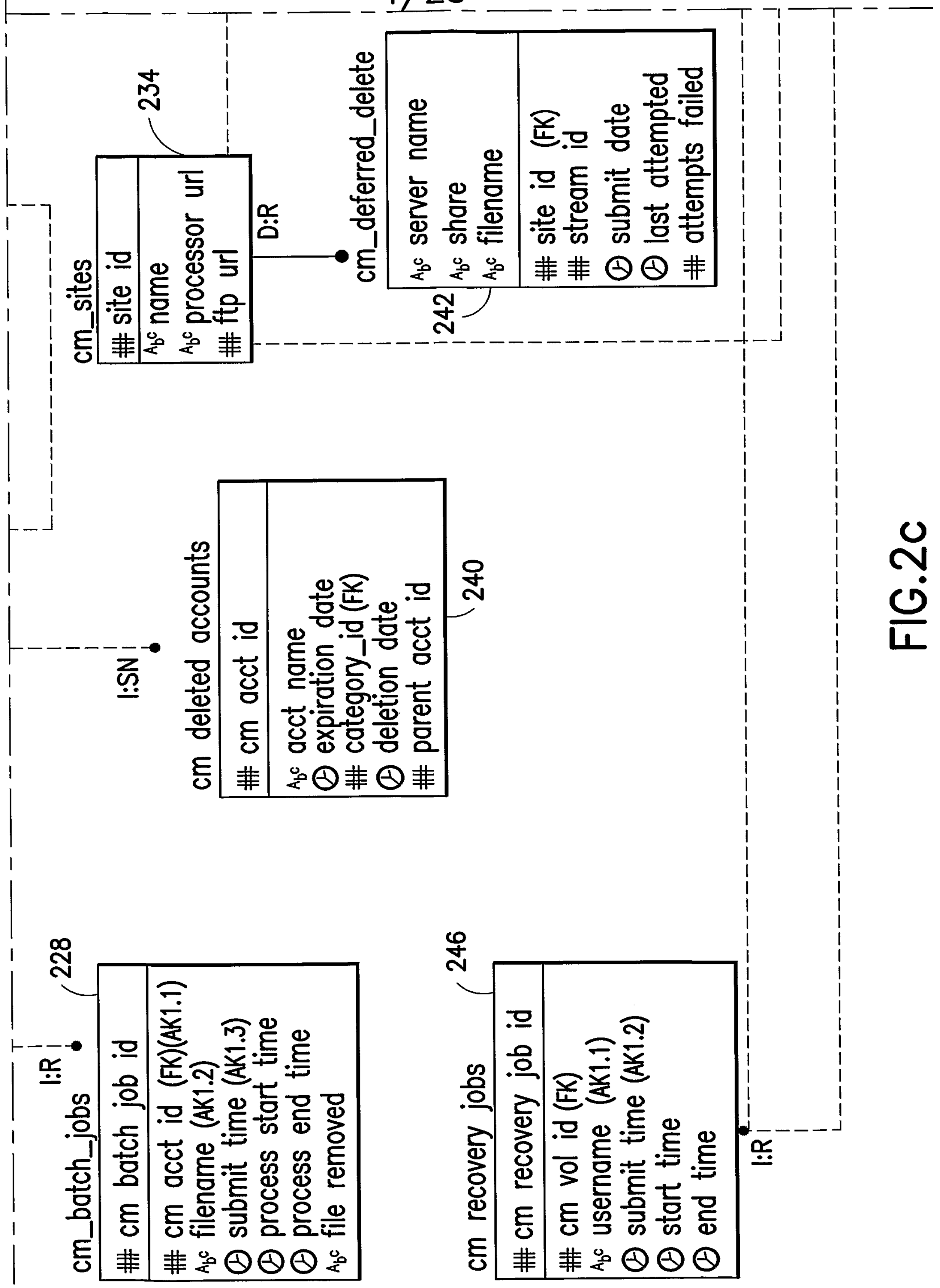

cm_sites
234
# site id
Abc name
Abc processor url
# ftp url
D:R
cm_deferred_delete
Abc server name
Abc share
Abc filename
# site id (FK)
# stream id
submit date
last attempted
# attempts failed
242
cm deleted accounts
# cm acct id
Abc acct name
expiration date
# category_id (FK)
deletion date
# parent acct id
240
I:SN
cm_batch_jobs
228
I:R
# cm batch job id
# cm acct id (FK)(AK1.1)
Abc filename (AK1.2)
submit time (AK1.3)
process start time
process end time
Abc file removed
cm recovery jobs
246
# cm recovery job id
# cm vol id (FK)
Abc username (AK1.1)
submit time (AK1.2)
start time
end time
I:R

FIG.2c

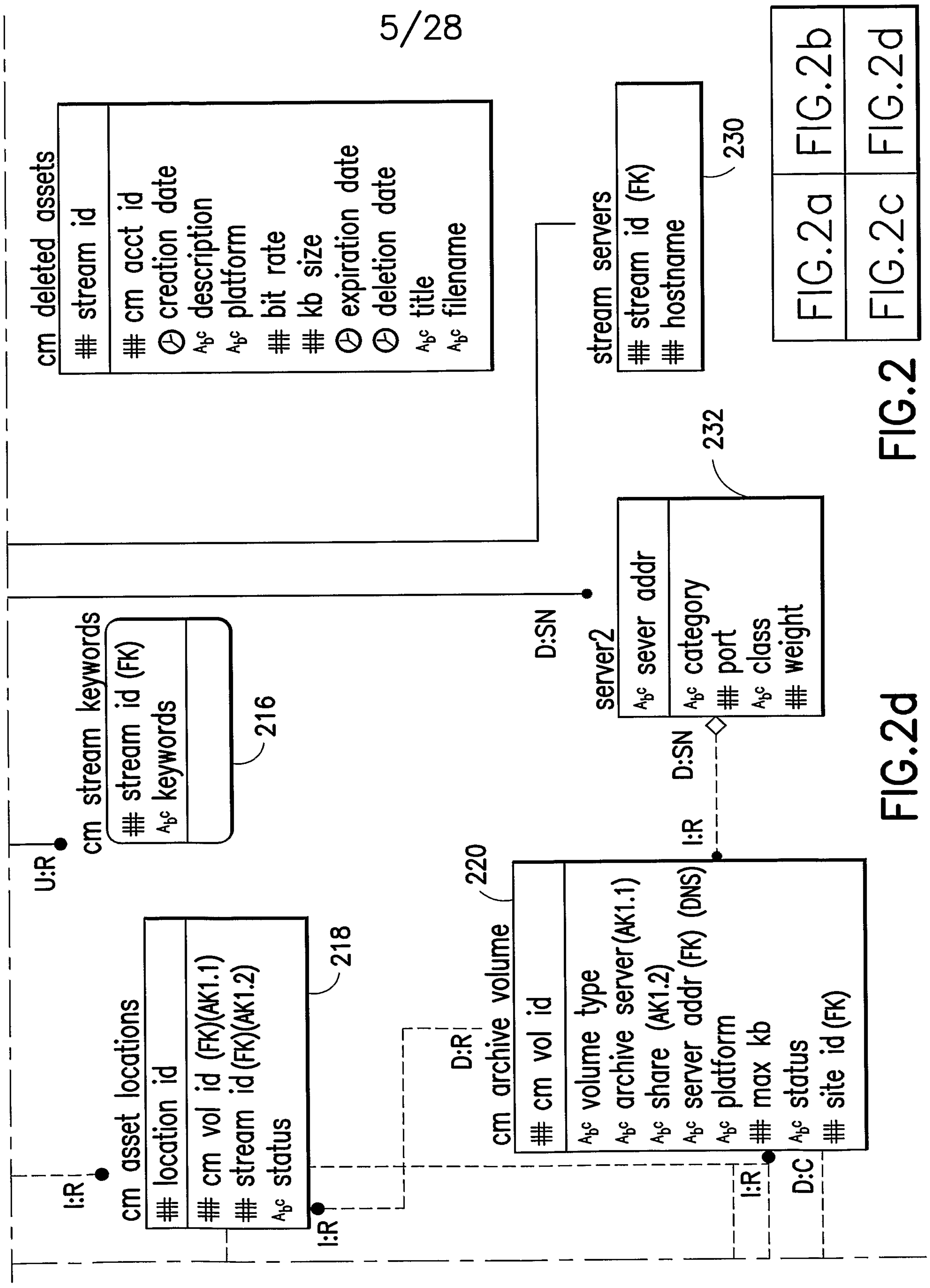
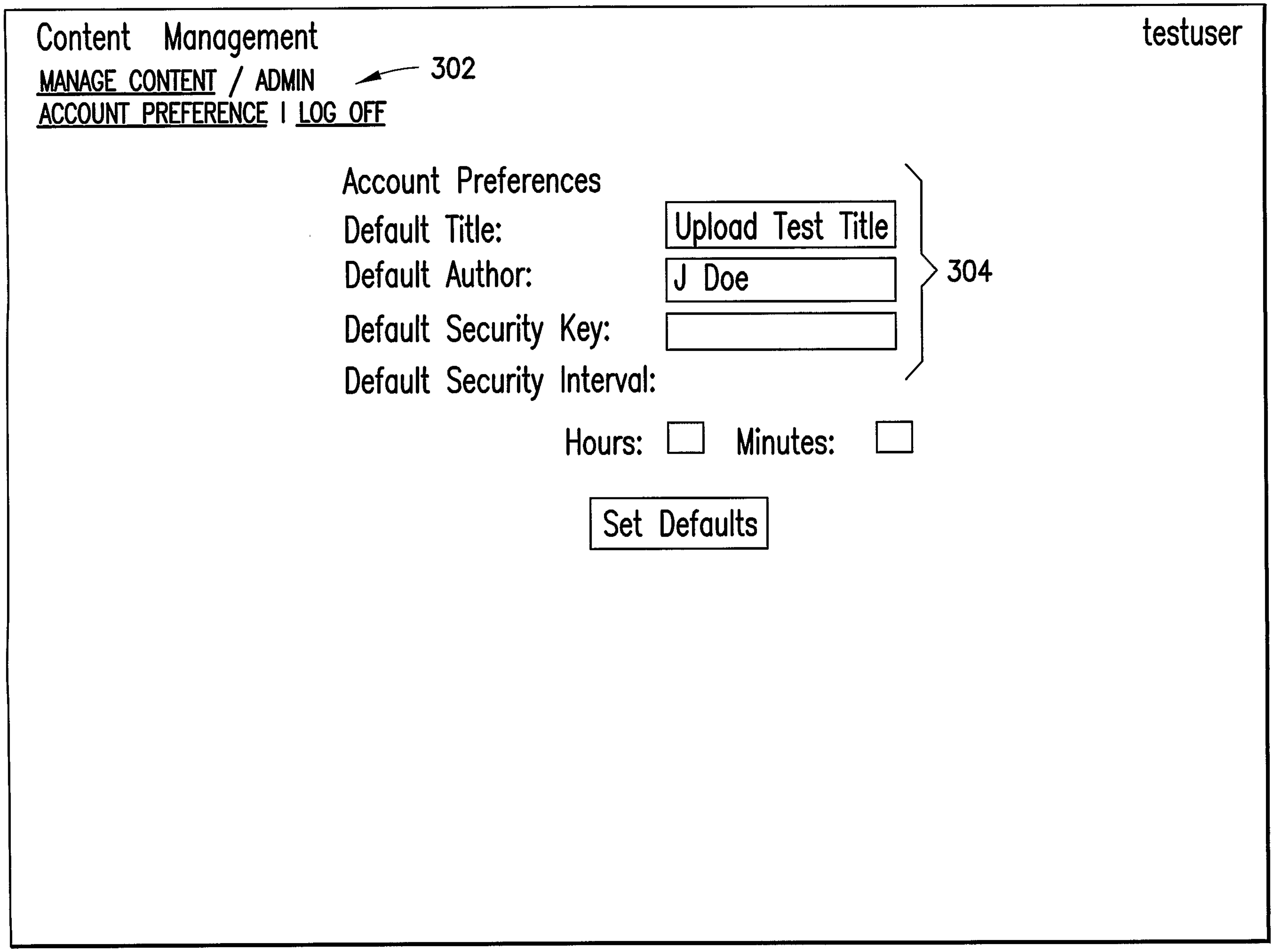

cm deleted assets
# stream id
# cm acct id
creation date
Abc description
Abc platform
# bit rate
# kb size
expiration date
deletion date
Abc title
Abc filename
stream servers
# stream id (FK)
# hostname
230
FIG.2a
FIG.2b
FIG.2c
FIG.2d
232
server2
Abc sever addr
Abc category
# port
Abc class
# weight
D:SN
D:SN
I:R
cm stream keywords
# stream id (FK)
Abc keywords
216
U:R
cm asset locations
# location id
# cm vol id (FK)(AK1.1)
# stream id (FK)(AK1.2)
Abc status
218
I:R
I:R
D:R
220
cm archive volume
# cm vol id
Abc volume type
Abc archive server (AK1.1)
Abc share (AK1.2)
Abc server addr (FK) (DNS)
Abc platform
# max kb
Abc status
# site id (FK)
I:R
D:C

FIG.2d

FIG.2


Content Management

testuser

MANAGE CONTENT / ADMIN

ACCOUNT PREFERENCE | LOG OFF

Account Preferences

Default Title: Upload Test Title

Default Author: J Doe

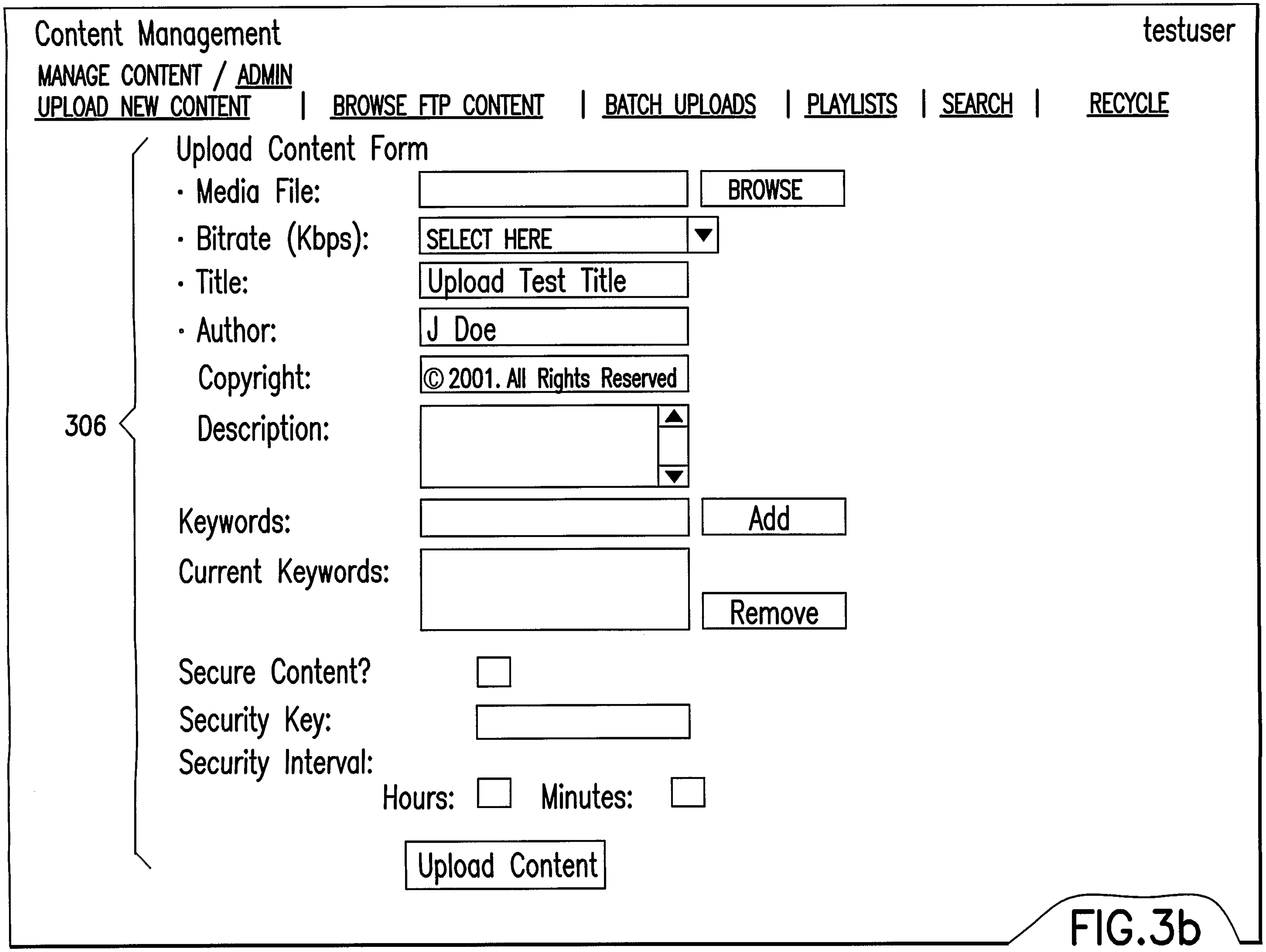
Default Security Key:

Default Security Interval:

Hours: Minutes:

Set Defaults

FIG.3a



Content Management testuser

MANAGE CONTENT / ADMIN

UPLOAD NEW CONTENT | BROWSE FTP CONTENT | BATCH UPLOADS | PLAYLISTS | SEARCH | RECYCLE

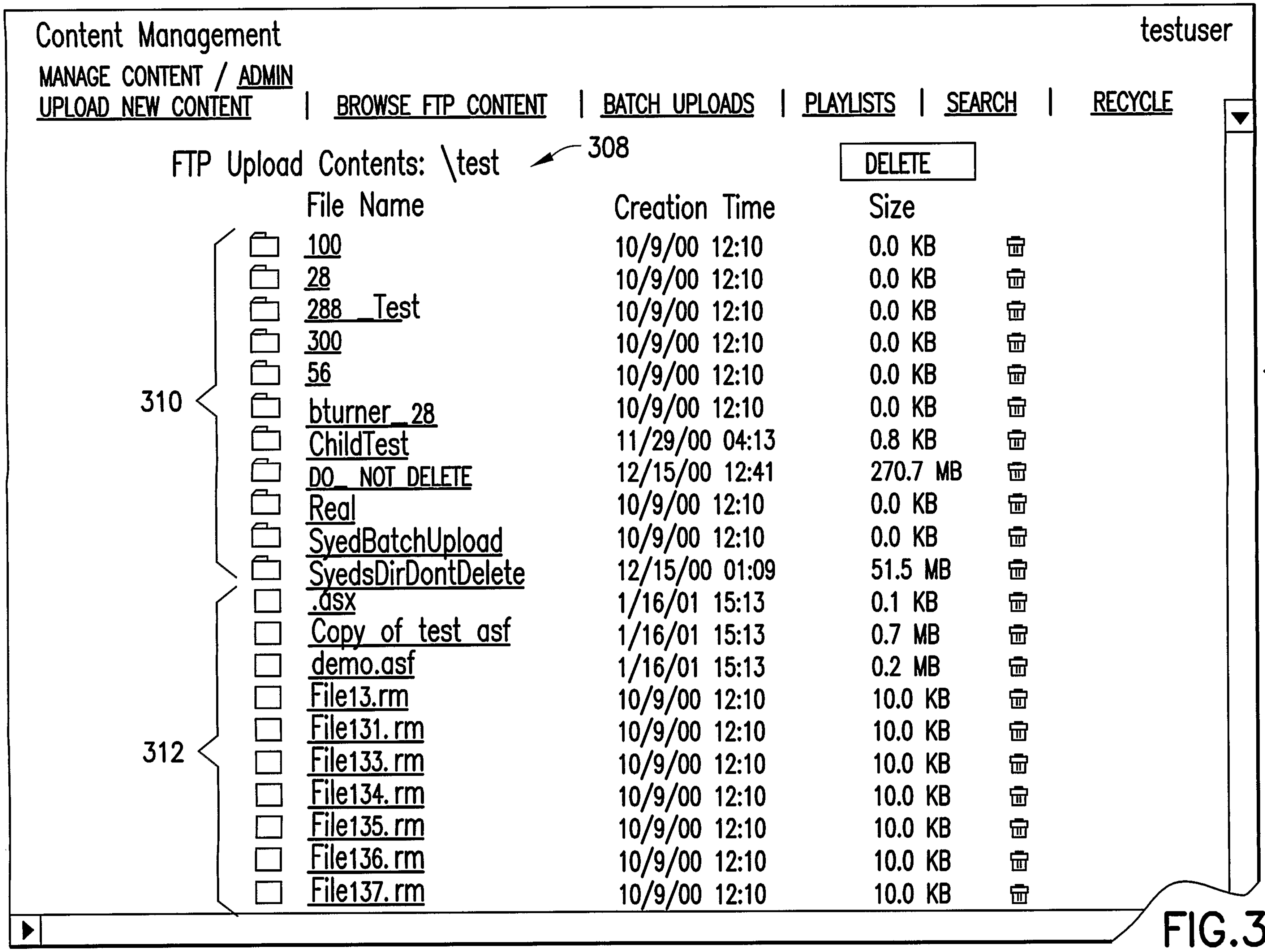
Upload Content Form

· Media File:		BROWSE
· Bitrate (Kbps):	SELECT HERE ▼	
· Title:	Upload Test Title	
· Author:	J Doe	
Copyright:	© 2001. All Rights Reserved	
Description:		
Keywords:		Add
Current Keywords:		Remove
Secure Content?	☐	
Security Key:		
Security Interval:	Hours: ☐ Minutes: ☐	

Upload Content

306

FIG.3b



Content Management testuser

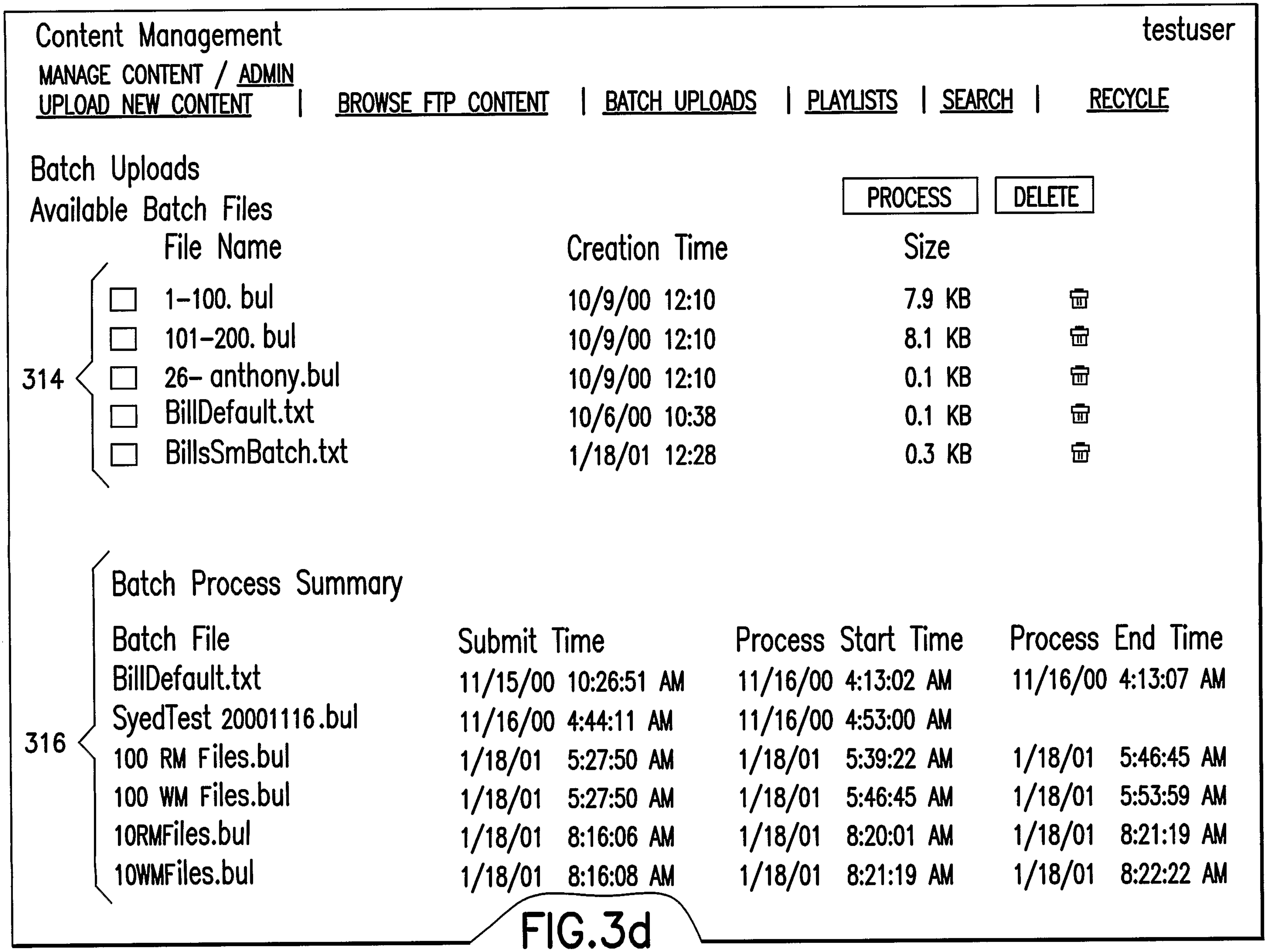
MANAGE CONTENT / ADMIN

UPLOAD NEW CONTENT | BROWSE FTP CONTENT | BATCH UPLOADS | PLAYLISTS | SEARCH | RECYCLE

FTP Upload Contents: \test

	File Name	Creation Time	Size	
🗀	100	10/9/00 12:10	0.0 KB	🗑
🗀	28	10/9/00 12:10	0.0 KB	🗑
🗀	288 _Test	10/9/00 12:10	0.0 KB	🗑
🗀	300	10/9/00 12:10	0.0 KB	🗑
🗀	56	10/9/00 12:10	0.0 KB	🗑
🗀	bturner_28	10/9/00 12:10	0.0 KB	🗑
🗀	ChildTest	11/29/00 04:13	0.8 KB	🗑
🗀	DO_ NOT DELETE	12/15/00 12:41	270.7 MB	🗑
🗀	Real	10/9/00 12:10	0.0 KB	🗑
🗀	SyedBatchUpload	10/9/00 12:10	0.0 KB	🗑
🗀	SyedsDirDontDelete	12/15/00 01:09	51.5 MB	🗑
☐	.asx	1/16/01 15:13	0.1 KB	🗑
☐	Copy of test asf	1/16/01 15:13	0.7 MB	🗑
☐	demo.asf	1/16/01 15:13	0.2 MB	🗑
☐	File13.rm	10/9/00 12:10	10.0 KB	🗑
☐	File131.rm	10/9/00 12:10	10.0 KB	🗑
☐	File133.rm	10/9/00 12:10	10.0 KB	🗑
☐	File134.rm	10/9/00 12:10	10.0 KB	🗑
☐	File135.rm	10/9/00 12:10	10.0 KB	🗑
☐	File136.rm	10/9/00 12:10	10.0 KB	🗑
☐	File137.rm	10/9/00 12:10	10.0 KB	🗑

FIG.3c


Content Management
testuser
MANAGE CONTENT / ADMIN
UPLOAD NEW CONTENT | BROWSE FTP CONTENT | BATCH UPLOADS | PLAYLISTS | SEARCH | RECYCLE
PROCESS
DELETE
314
316

Batch Uploads

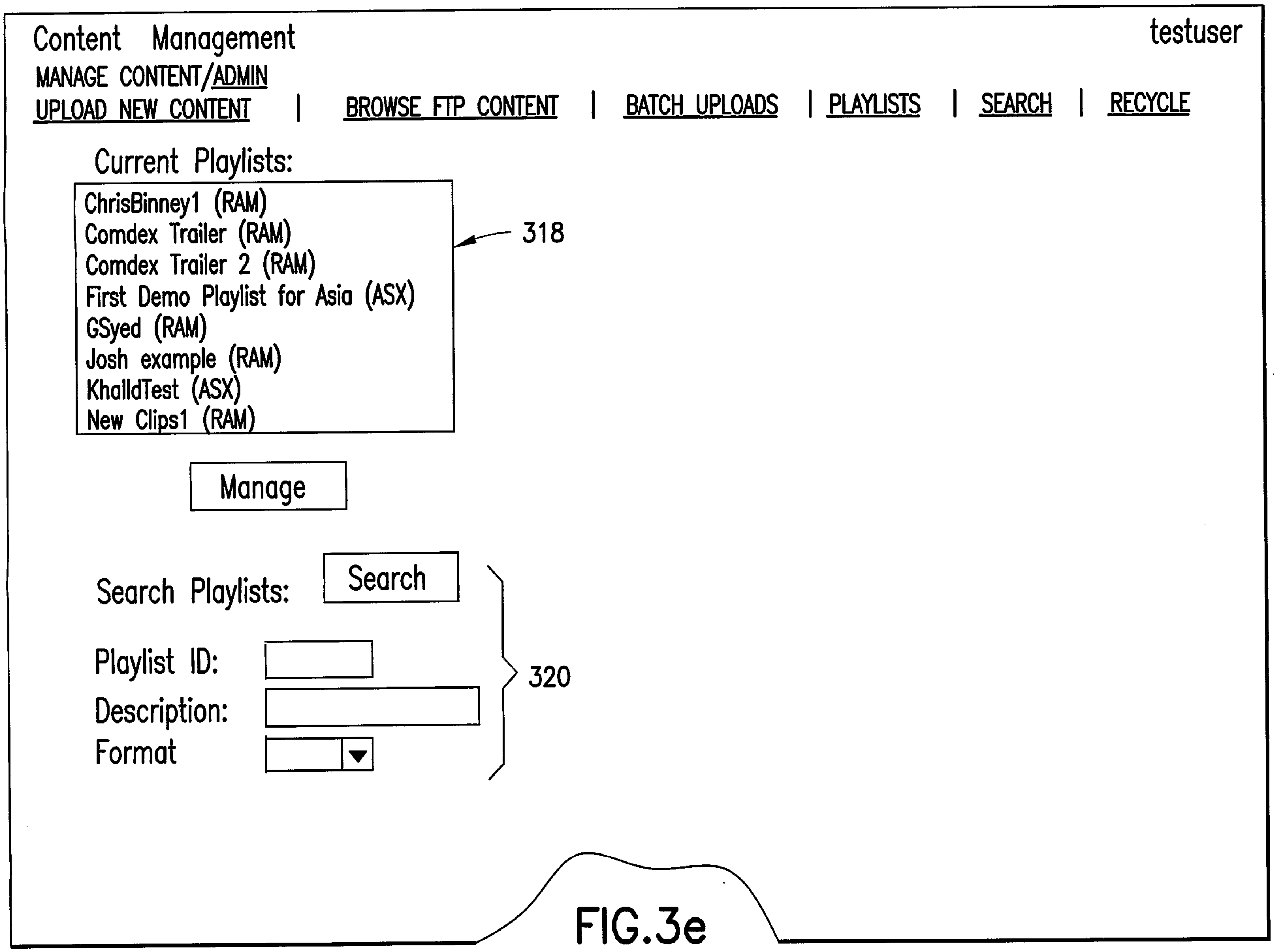
Available Batch Files

	File Name	Creation Time	Size	
☐	1–100. bul	10/9/00 12:10	7.9 KB	🗑
☐	101–200. bul	10/9/00 12:10	8.1 KB	🗑
☐	26– anthony.bul	10/9/00 12:10	0.1 KB	🗑
☐	BillDefault.txt	10/6/00 10:38	0.1 KB	🗑
☐	BillsSmBatch.txt	1/18/01 12:28	0.3 KB	🗑

Batch Process Summary

Batch File	Submit Time	Process Start Time	Process End Time
BillDefault.txt	11/15/00 10:26:51 AM	11/16/00 4:13:02 AM	11/16/00 4:13:07 AM
SyedTest 20001116.bul	11/16/00 4:44:11 AM	11/16/00 4:53:00 AM	
100 RM Files.bul	1/18/01 5:27:50 AM	1/18/01 5:39:22 AM	1/18/01 5:46:45 AM
100 WM Files.bul	1/18/01 5:27:50 AM	1/18/01 5:46:45 AM	1/18/01 5:53:59 AM
10RMFiles.bul	1/18/01 8:16:06 AM	1/18/01 8:20:01 AM	1/18/01 8:21:19 AM
10WMFiles.bul	1/18/01 8:16:08 AM	1/18/01 8:21:19 AM	1/18/01 8:22:22 AM

FIG.3d



Content Management testuser

MANAGE CONTENT/ADMIN

UPLOAD NEW CONTENT | BROWSE FTP CONTENT | BATCH UPLOADS | PLAYLISTS | SEARCH | RECYCLE

Current Playlists:

ChrisBinney1 (RAM)
Comdex Trailer (RAM)
Comdex Trailer 2 (RAM)
First Demo Playlist for Asia (ASX)
GSyed (RAM)
Josh example (RAM)
KhalldTest (ASX)
New Clips1 (RAM)

Manage

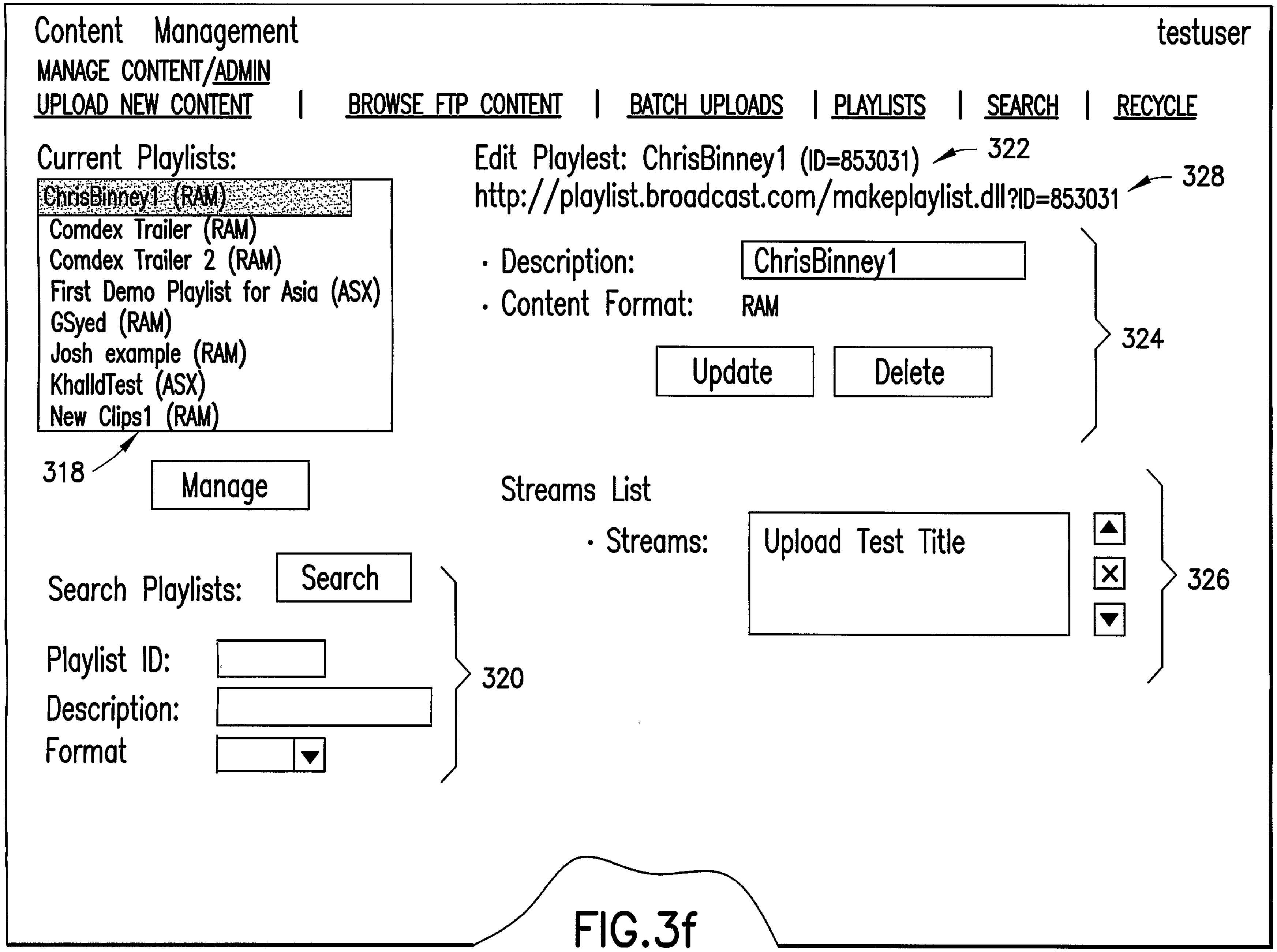
Search Playlists: Search

Playlist ID:

Description:

Format

FIG.3e



Content Management testuser

MANAGE CONTENT/ADMIN

UPLOAD NEW CONTENT | BROWSE FTP CONTENT | BATCH UPLOADS | PLAYLISTS | SEARCH | RECYCLE

Current Playlists:

ChrisBinney1 (RAM)
Comdex Trailer (RAM)
Comdex Trailer 2 (RAM)
First Demo Playlist for Asia (ASX)
GSyed (RAM)
Josh example (RAM)
KhalldTest (ASX)
New Clips1 (RAM)

Edit Playlest: ChrisBinney1 (ID=853031)

http://playlist.broadcast.com/makeplaylist.dll?ID=853031

- Description: ChrisBinney1
- Content Format: RAM

Streams List

- Streams: Upload Test Title

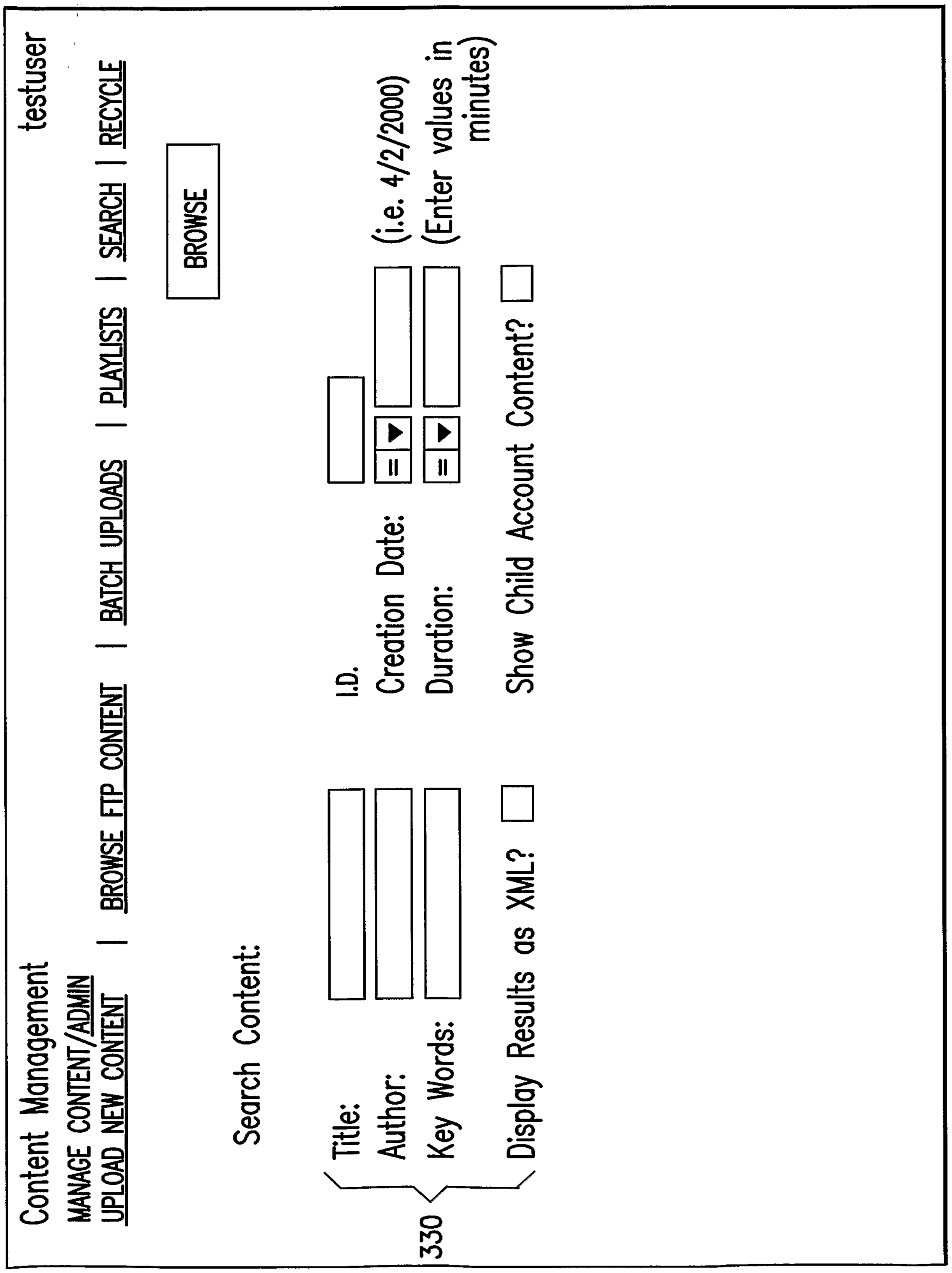
Search Playlists:

Playlist ID:

Description:

Format

FIG.3f



Content Management testuser

MANAGE CONTENT/ADMIN

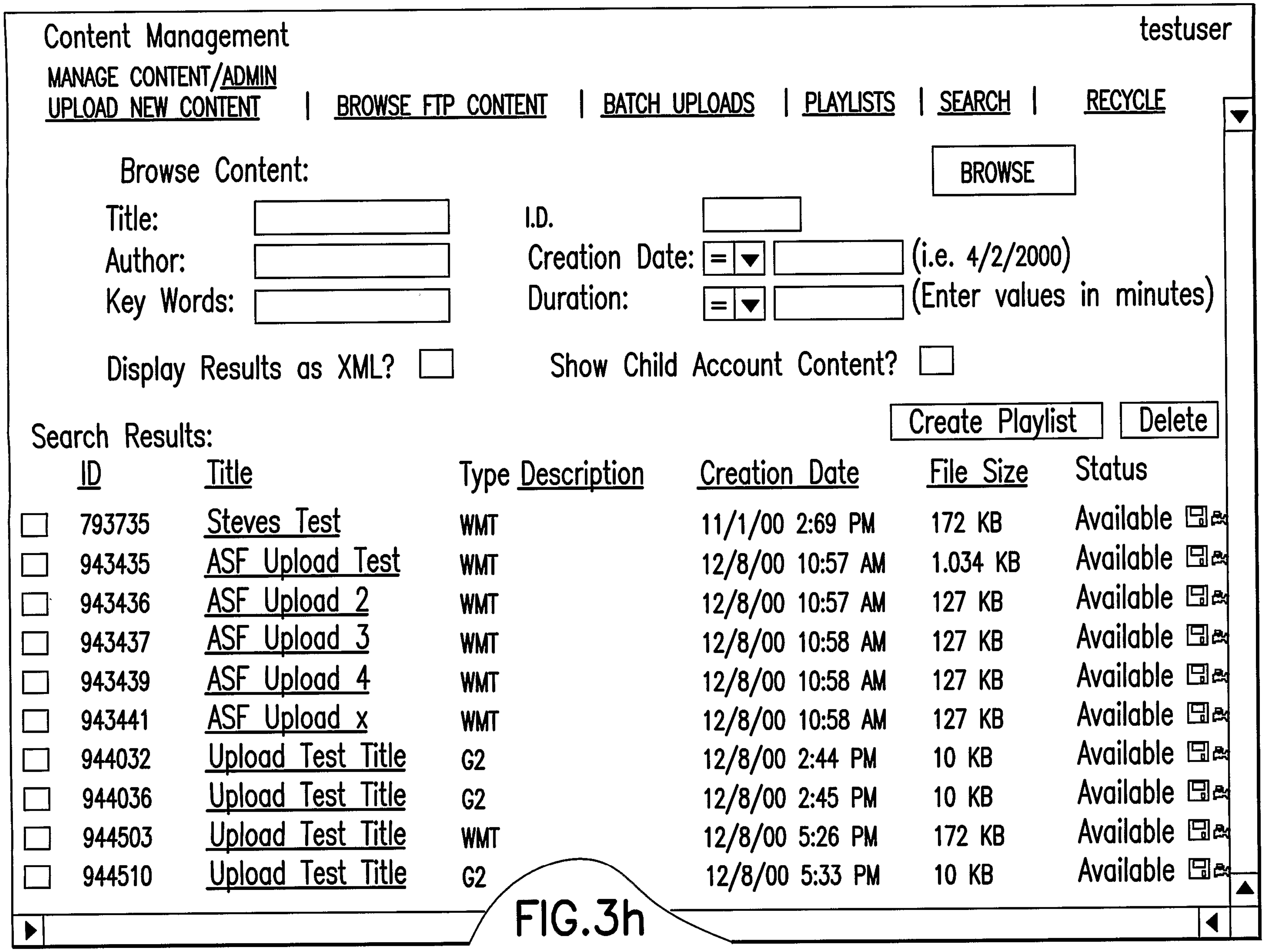
UPLOAD NEW CONTENT | BROWSE FTP CONTENT | BATCH UPLOADS | PLAYLISTS | SEARCH | RECYCLE

BROWSE

Search Content:

Title:		I.D.	
Author:		Creation Date:	= ▶ (i.e. 4/2/2000)
Key Words:		Duration:	= ▶ (Enter values in minutes)
Display Results as XML?	☐	Show Child Account Content?	☐

FIG.3g



Content Management testuser

MANAGE CONTENT/ADMIN
UPLOAD NEW CONTENT | BROWSE FTP CONTENT | BATCH UPLOADS | PLAYLISTS | SEARCH | RECYCLE
BROWSE

Browse Content:

Title:		I.D.	
Author:		Creation Date: =	(i.e. 4/2/2000)
Key Words:		Duration: =	(Enter values in minutes)

Display Results as XML? ☐ Show Child Account Content? ☐

Search Results:

	ID	Title	Type	Description	Creation Date	File Size	Status
☐	793735	Steves Test	WMT		11/1/00 2:69 PM	172 KB	Available
☐	943435	ASF Upload Test	WMT		12/8/00 10:57 AM	1.034 KB	Available
☐	943436	ASF Upload 2	WMT		12/8/00 10:57 AM	127 KB	Available
☐	943437	ASF Upload 3	WMT		12/8/00 10:58 AM	127 KB	Available
☐	943439	ASF Upload 4	WMT		12/8/00 10:58 AM	127 KB	Available
☐	943441	ASF Upload x	WMT		12/8/00 10:58 AM	127 KB	Available
☐	944032	Upload Test Title	G2		12/8/00 2:44 PM	10 KB	Available
☐	944036	Upload Test Title	G2		12/8/00 2:45 PM	10 KB	Available
☐	944503	Upload Test Title	WMT		12/8/00 5:26 PM	172 KB	Available
☐	944510	Upload Test Title	G2		12/8/00 5:33 PM	10 KB	Available

FIG.3h

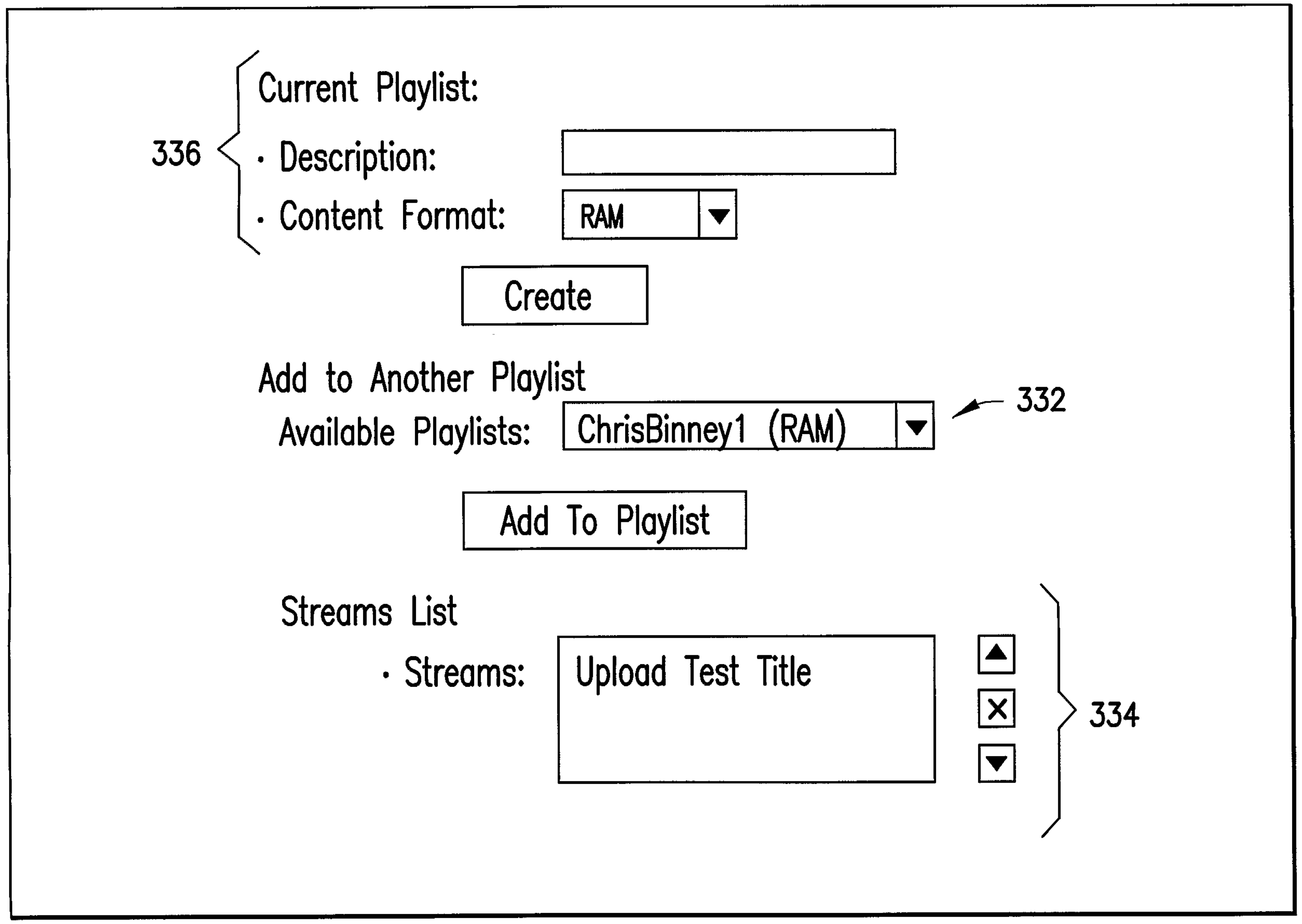
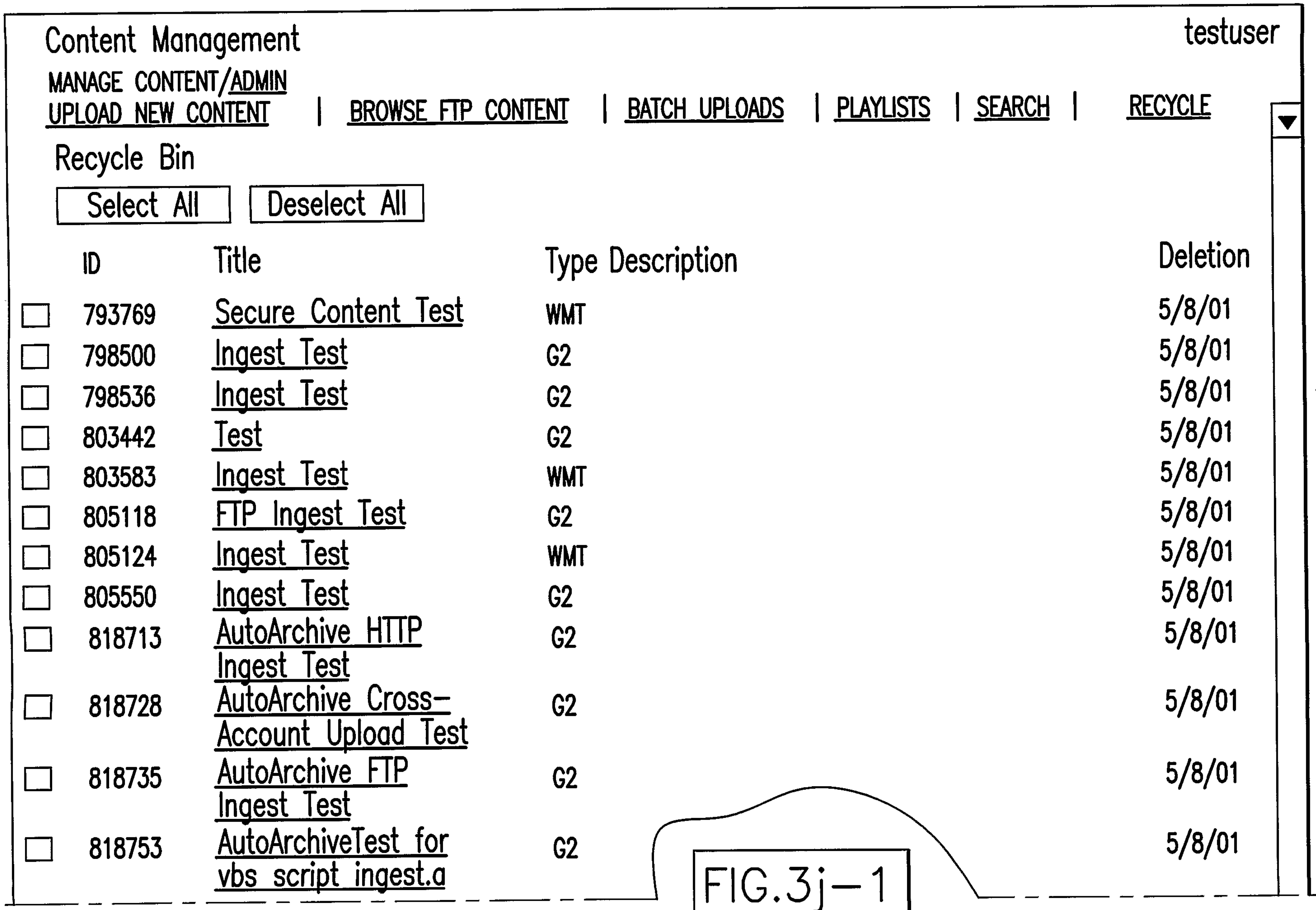

Current Playlist:
336
· Description:
· Content Format:
RAM
Create
Add to Another Playlist
Available Playlists:
ChrisBinney1 (RAM)
332
Add To Playlist
Streams List
· Streams:
Upload Test Title
334

FIG.3i


Content Management testuser

MANAGE CONTENT/ADMIN
UPLOAD NEW CONTENT | BROWSE FTP CONTENT | BATCH UPLOADS | PLAYLISTS | SEARCH | RECYCLE

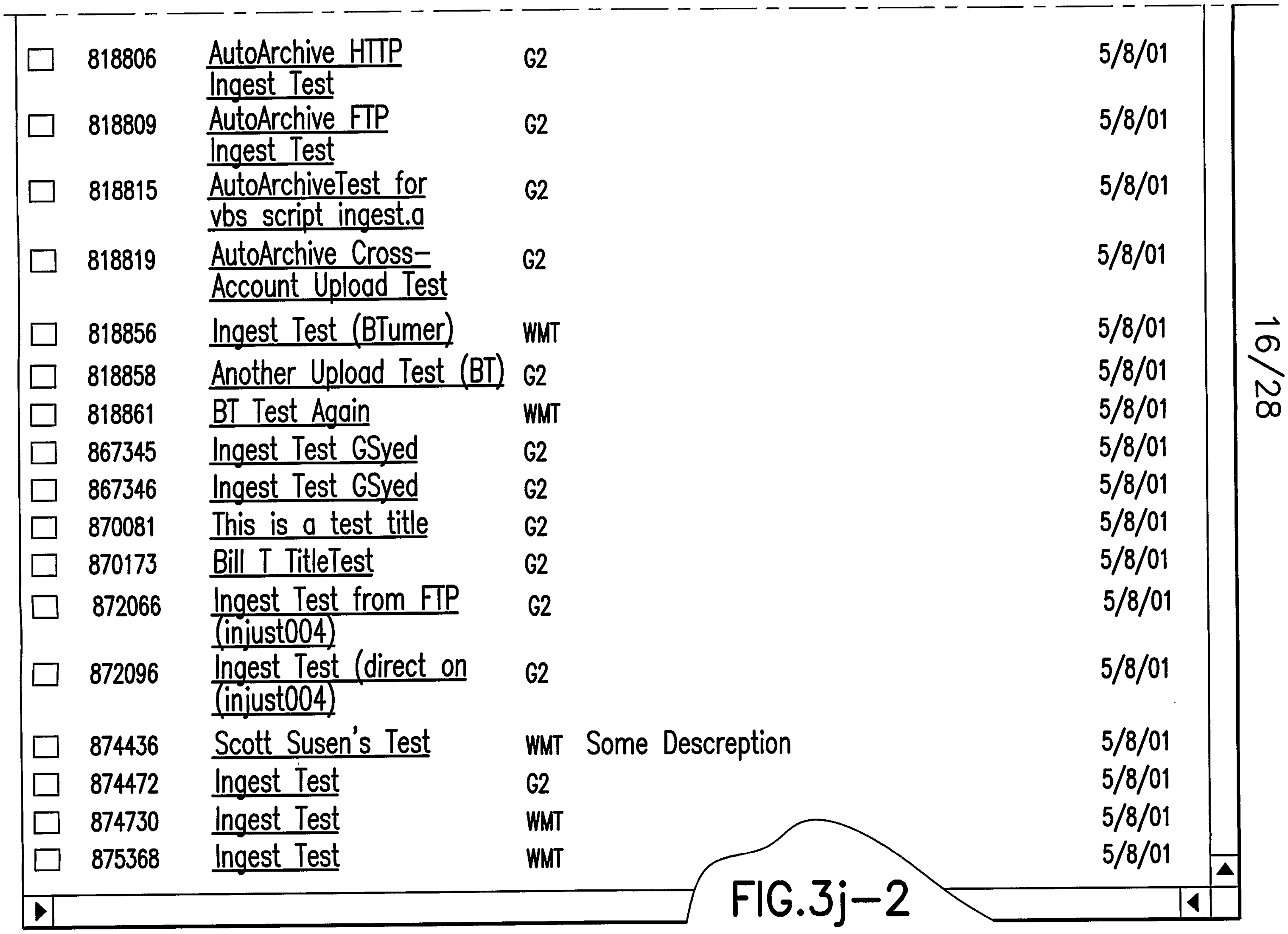
Recycle Bin

Select All Deselect All

	ID	Title	Type	Description	Deletion
☐	793769	Secure Content Test	WMT		5/8/01
☐	798500	Ingest Test	G2		5/8/01
☐	798536	Ingest Test	G2		5/8/01
☐	803442	Test	G2		5/8/01
☐	803583	Ingest Test	WMT		5/8/01
☐	805118	FTP Ingest Test	G2		5/8/01
☐	805124	Ingest Test	WMT		5/8/01
☐	805550	Ingest Test	G2		5/8/01
☐	818713	AutoArchive HTTP Ingest Test	G2		5/8/01
☐	818728	AutoArchive Cross-Account Upload Test	G2		5/8/01
☐	818735	AutoArchive FTP Ingest Test	G2		5/8/01
☐	818753	AutoArchiveTest for vbs script ingest.a	G2		5/8/01

FIG.3j FIG.3j–1 FIG.3j–2

FIG.3j–1



☐	818806	AutoArchive HTTP Ingest Test	G2		5/8/01
☐	818809	AutoArchive FTP Ingest Test	G2		5/8/01
☐	818815	AutoArchiveTest for vbs script ingest.a	G2		5/8/01
☐	818819	AutoArchive Cross-Account Upload Test	G2		5/8/01
☐	818856	Ingest Test (BTumer)	WMT		5/8/01
☐	818858	Another Upload Test (BT)	G2		5/8/01
☐	818861	BT Test Again	WMT		5/8/01
☐	867345	Ingest Test GSyed	G2		5/8/01
☐	867346	Ingest Test GSyed	G2		5/8/01
☐	870081	This is a test title	G2		5/8/01
☐	870173	Bill T TitleTest	G2		5/8/01
☐	872066	Ingest Test from FTP (injust004)	G2		5/8/01
☐	872096	Ingest Test (direct on (injust004)	G2		5/8/01
☐	874436	Scott Susen's Test	WMT	Some Descreption	5/8/01
☐	874472	Ingest Test	G2		5/8/01
☐	874730	Ingest Test	WMT		5/8/01
☐	875368	Ingest Test	WMT		5/8/01

FIG.3j–2

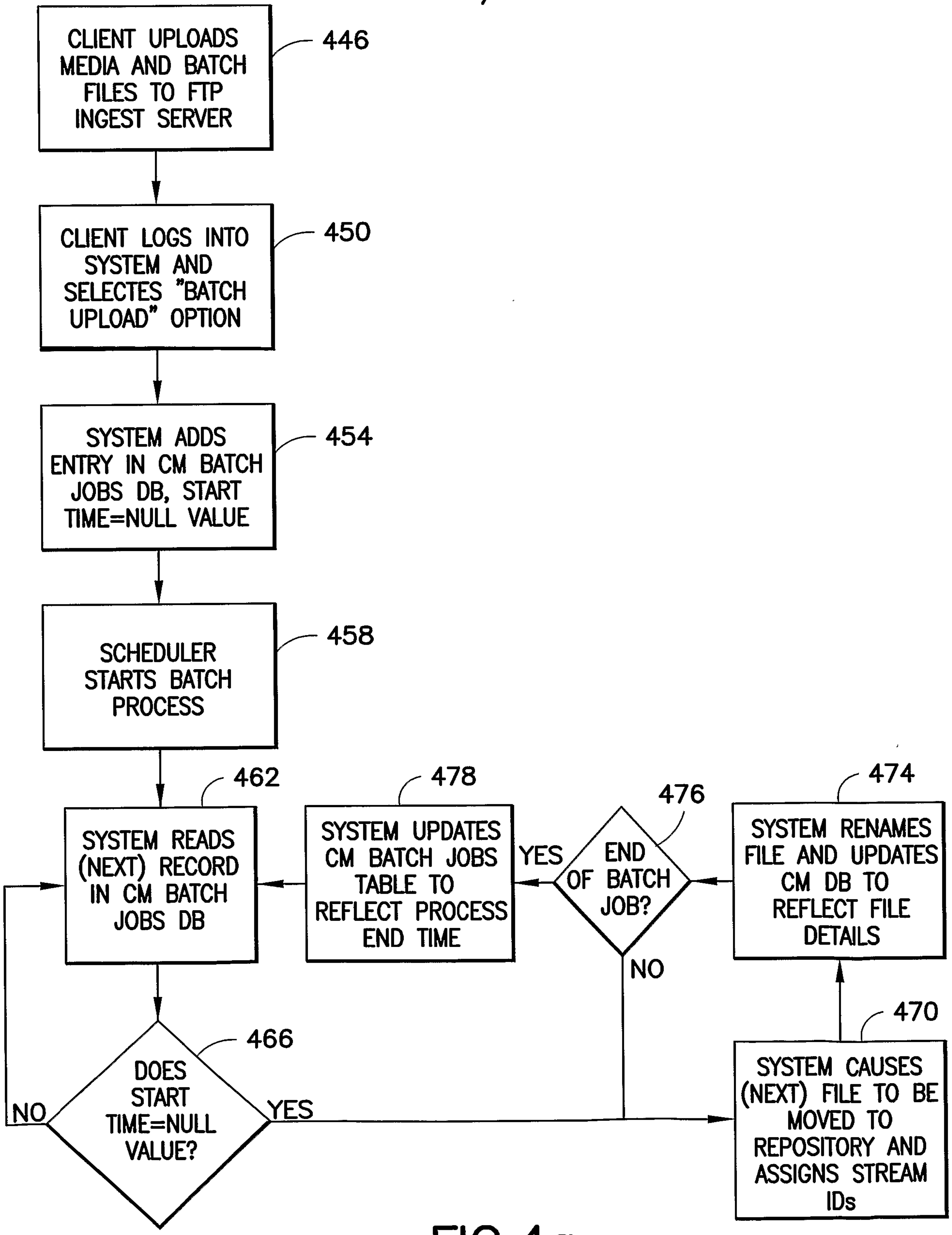

CLIENT UPLOADS MEDIA AND BATCH FILES TO FTP INGEST SERVER
446
CLIENT LOGS INTO SYSTEM AND SELECTES "BATCH UPLOAD" OPTION
450
SYSTEM ADDS ENTRY IN CM BATCH JOBS DB, START TIME=NULL VALUE
454
SCHEDULER STARTS BATCH PROCESS
458
SYSTEM READS (NEXT) RECORD IN CM BATCH JOBS DB
462
SYSTEM UPDATES CM BATCH JOBS TABLE TO REFLECT PROCESS END TIME
478
END OF BATCH JOB?
476
YES
NO
SYSTEM RENAMES FILE AND UPDATES CM DB TO REFLECT FILE DETAILS
474
DOES START TIME=NULL VALUE?
466
NO
YES
SYSTEM CAUSES (NEXT) FILE TO BE MOVED TO REPOSITORY AND ASSIGNS STREAM IDs
470

FIG.4a

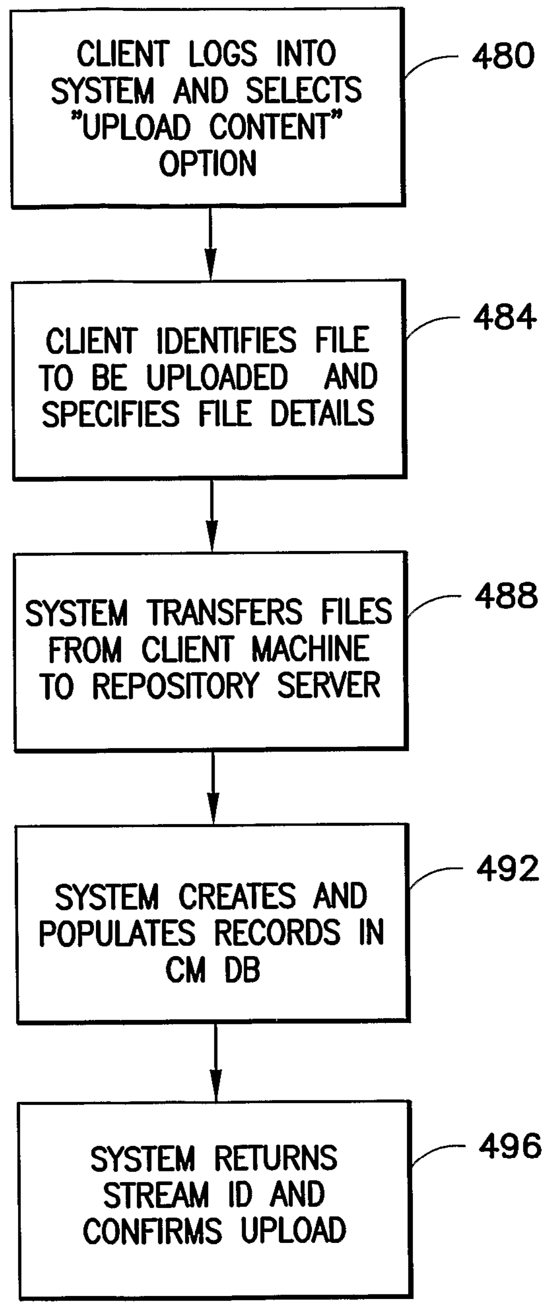

CLIENT LOGS INTO SYSTEM AND SELECTS "UPLOAD CONTENT" OPTION
480
CLIENT IDENTIFIES FILE TO BE UPLOADED AND SPECIFIES FILE DETAILS
484
SYSTEM TRANSFERS FILES FROM CLIENT MACHINE TO REPOSITORY SERVER
488
SYSTEM CREATES AND POPULATES RECORDS IN CM DB
492
SYSTEM RETURNS STREAM ID AND CONFIRMS UPLOAD
496

FIG.4b

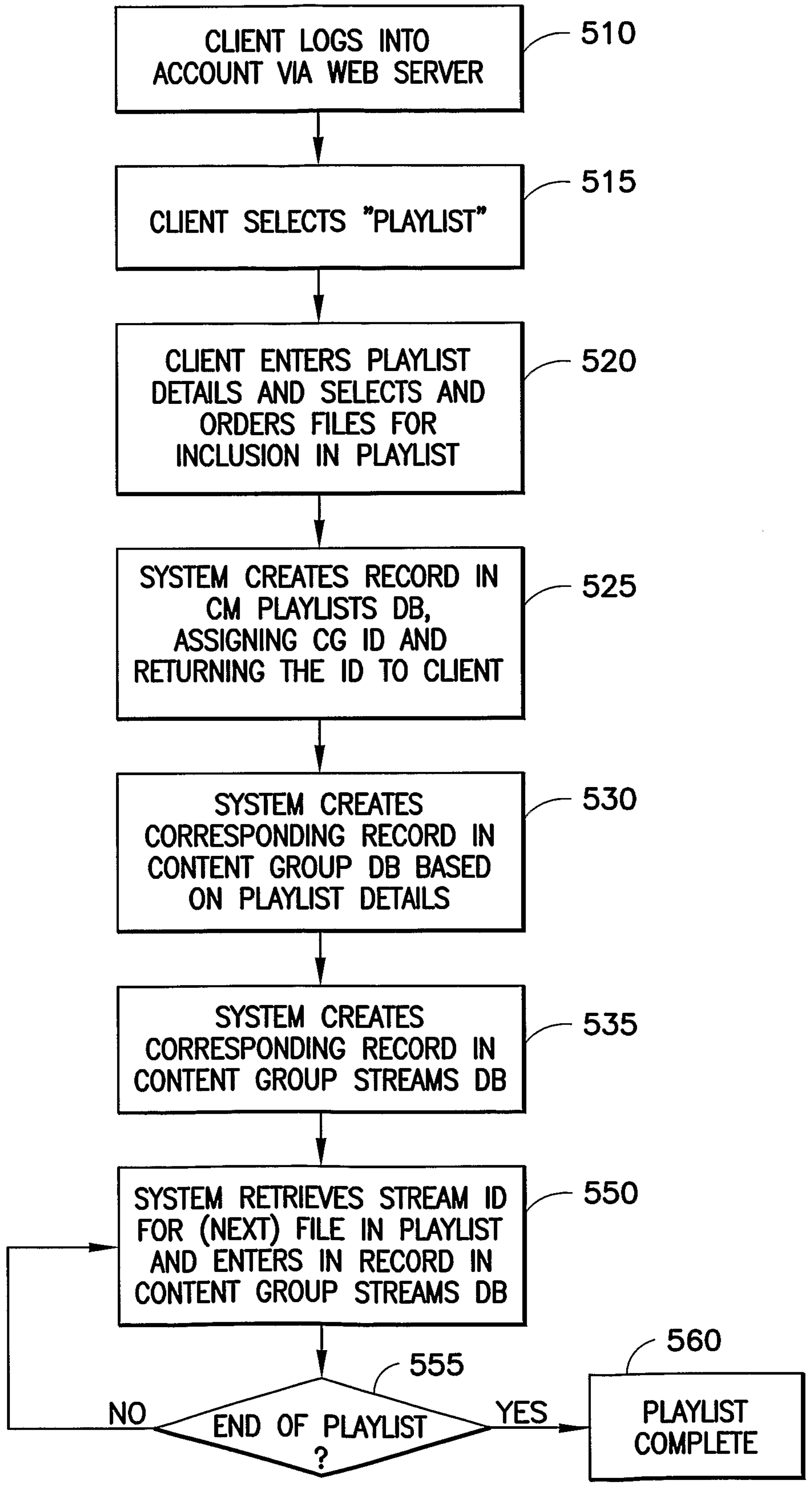

CLIENT LOGS INTO ACCOUNT VIA WEB SERVER
510
CLIENT SELECTS "PLAYLIST"
515
CLIENT ENTERS PLAYLIST DETAILS AND SELECTS AND ORDERS FILES FOR INCLUSION IN PLAYLIST
520
SYSTEM CREATES RECORD IN CM PLAYLISTS DB, ASSIGNING CG ID AND RETURNING THE ID TO CLIENT
525
SYSTEM CREATES CORRESPONDING RECORD IN CONTENT GROUP DB BASED ON PLAYLIST DETAILS
530
SYSTEM CREATES CORRESPONDING RECORD IN CONTENT GROUP STREAMS DB
535
SYSTEM RETRIEVES STREAM ID FOR (NEXT) FILE IN PLAYLIST AND ENTERS IN RECORD IN CONTENT GROUP STREAMS DB
550
555
END OF PLAYLIST ?
NO
YES
560
PLAYLIST COMPLETE

FIG.5

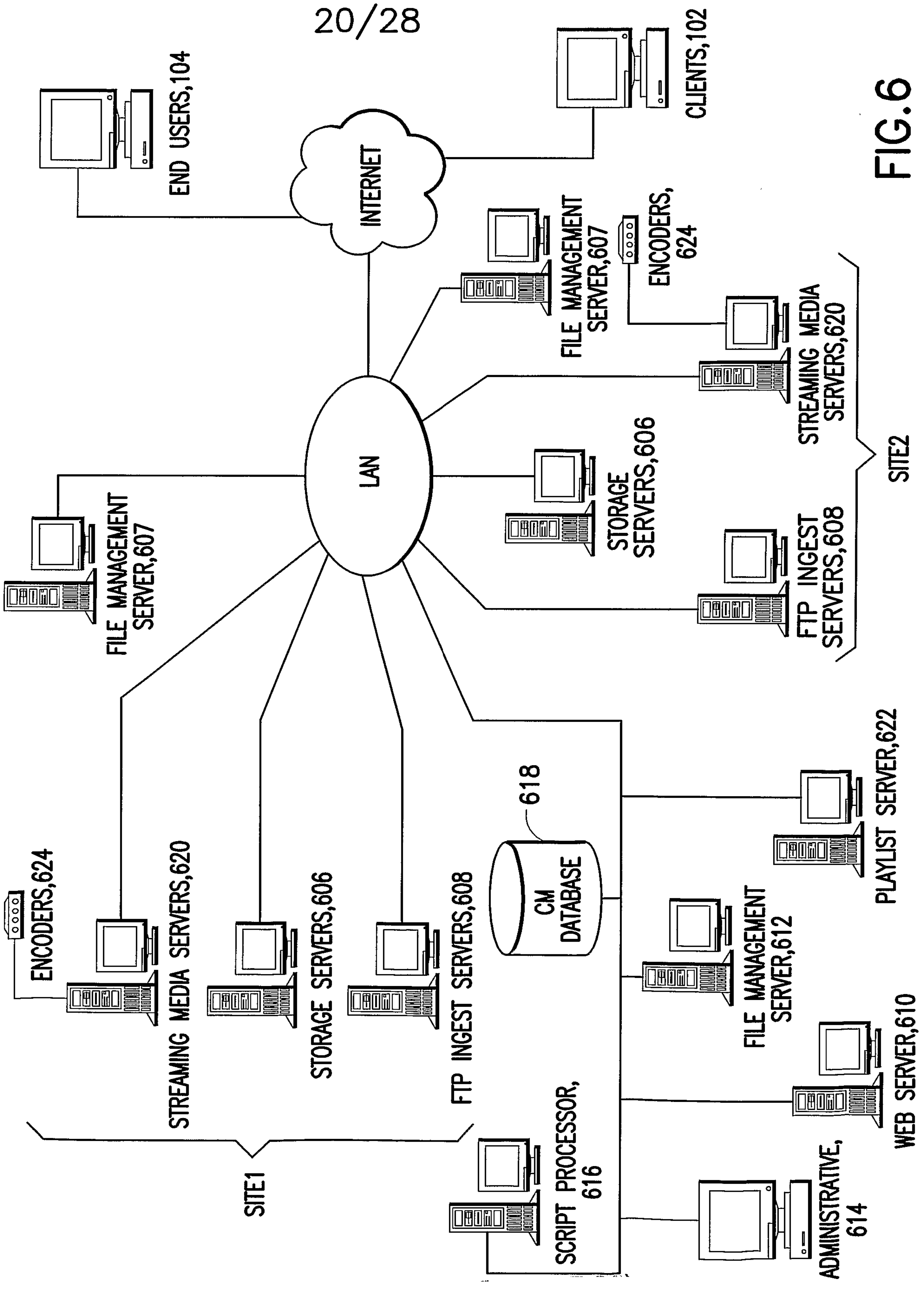

END USERS,104
CLIENTS,102
INTERNET
LAN
FILE MANAGEMENT SERVER,607
ENCODERS, 624
STREAMING MEDIA SERVERS,620
STORAGE SERVERS,606
FTP INGEST SERVERS,608
SITE2
FILE MANAGEMENT SERVER,607
ENCODERS,624
STREAMING MEDIA SERVERS,620
STORAGE SERVERS,606
FTP INGEST SERVERS,608
SITE1
618
CM DATABASE
PLAYLIST SERVER,622
FILE MANAGEMENT SERVER,612
WEB SERVER,610
SCRIPT PROCESSOR, 616
ADMINISTRATIVE, 614

FIG.6

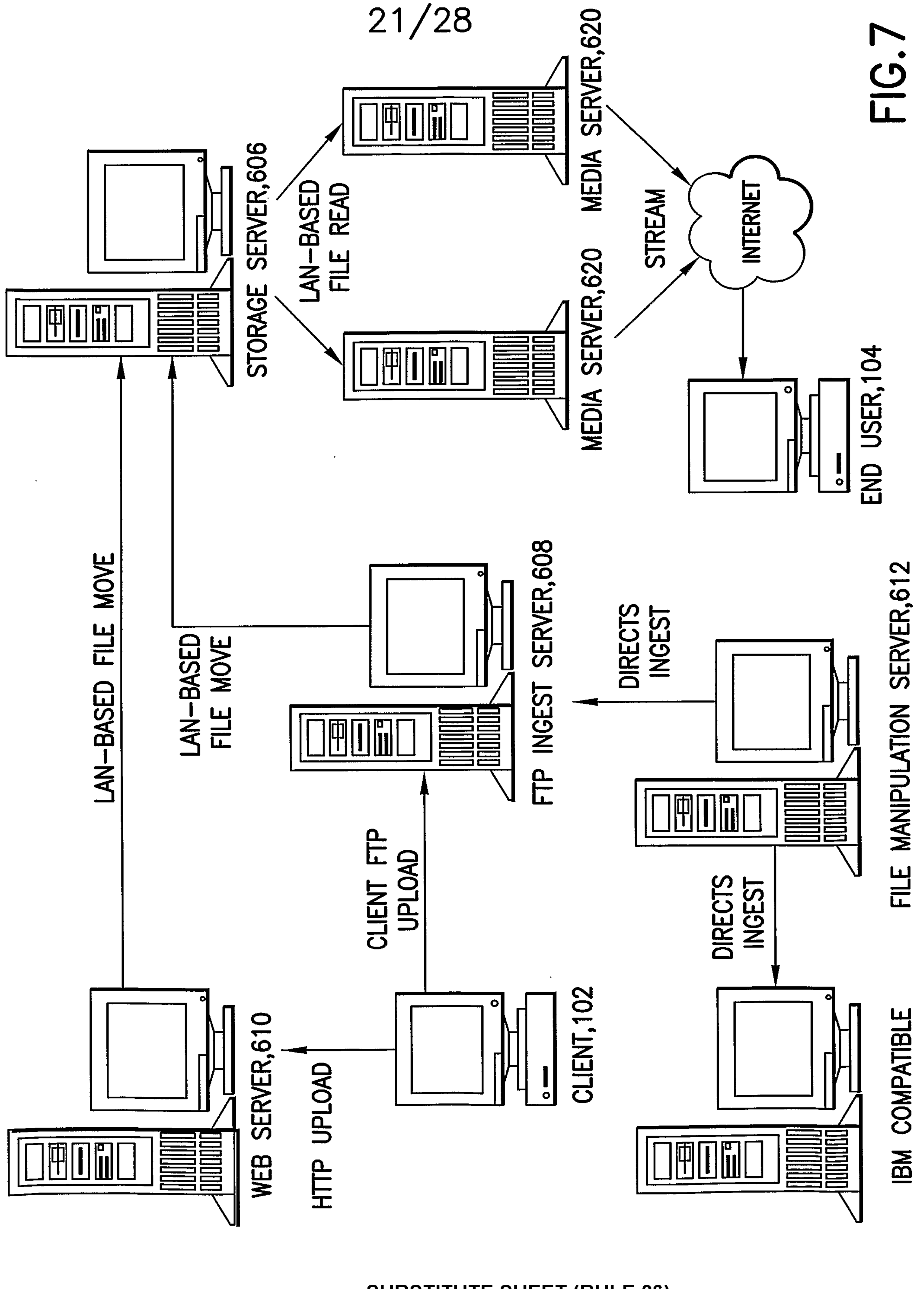

LAN-BASED FILE MOVE
LAN-BASED FILE MOVE
STORAGE SERVER,606
LAN-BASED FILE READ
MEDIA SERVER,620
MEDIA SERVER,620
STREAM
INTERNET
END USER,104
FTP INGEST SERVER,608
CLIENT FTP UPLOAD
DIRECTS INGEST
DIRECTS INGEST
FILE MANIPULATION SERVER,612
WEB SERVER,610
HTTP UPLOAD
CLIENT,102
IBM COMPATIBLE

FIG.7

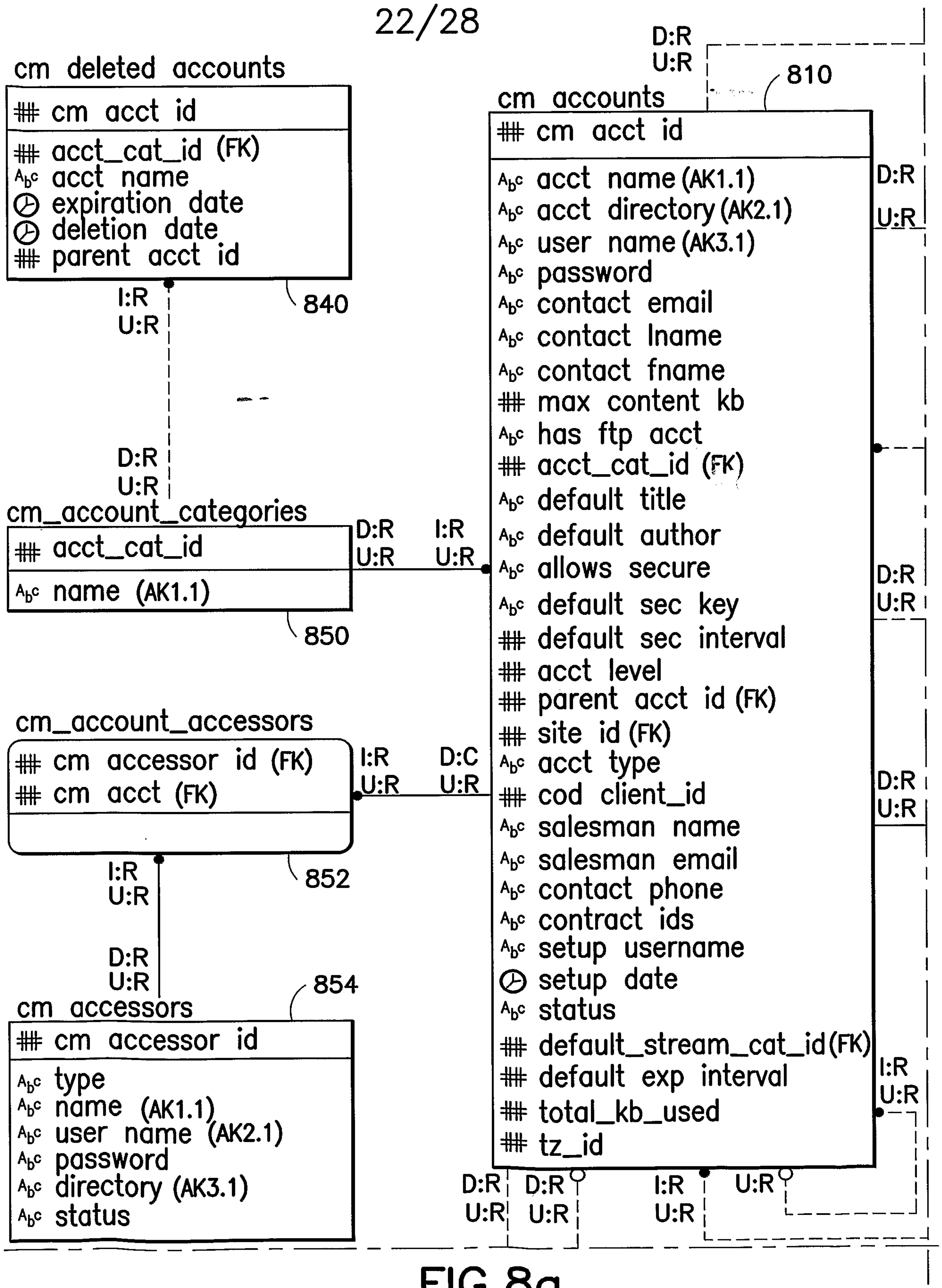
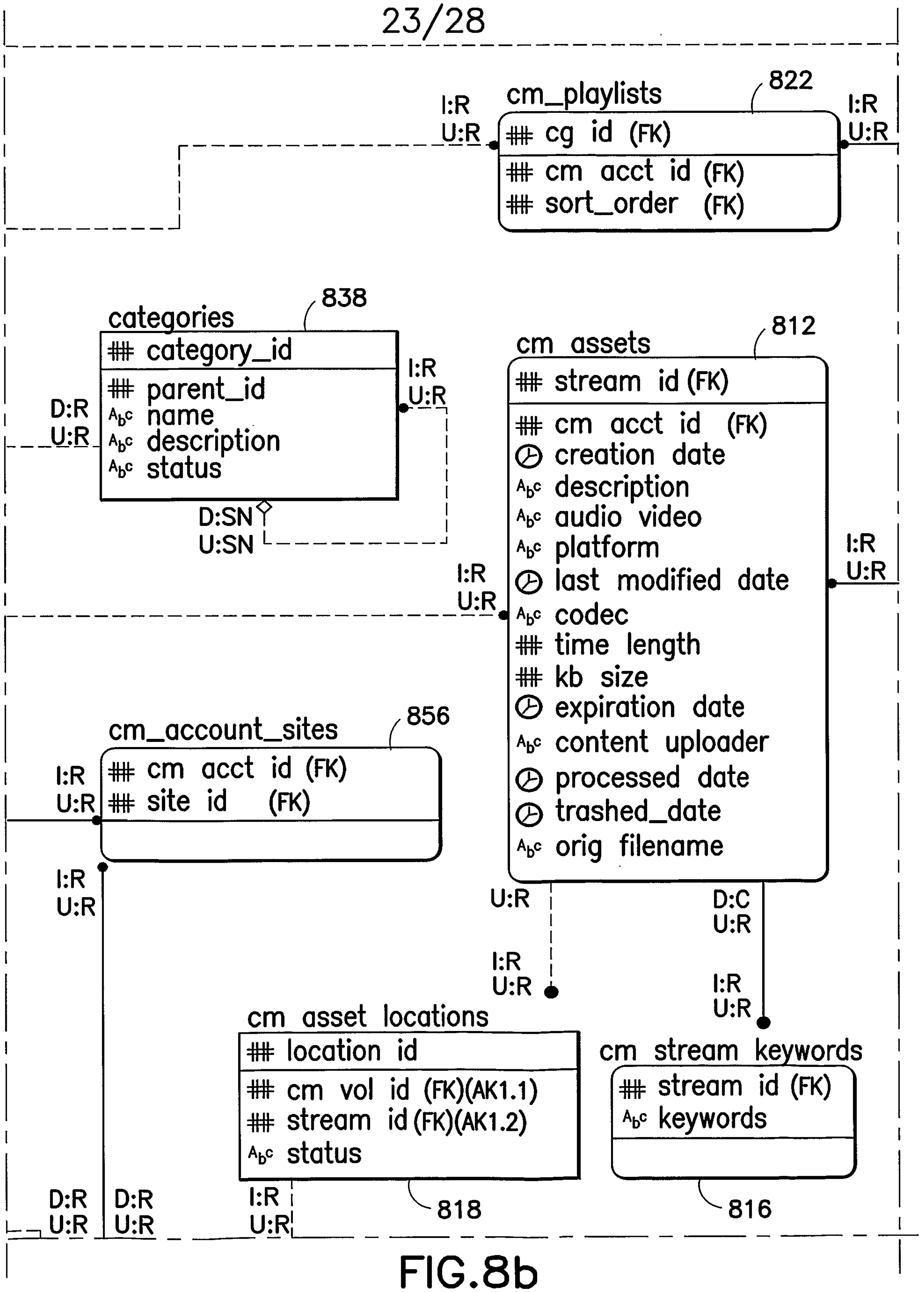

cm deleted accounts
cm acct id
acct_cat_id (FK)
acct name
expiration date
deletion date
parent acct id
840
I:R
U:R
D:R
U:R
cm_account_categories
acct_cat_id
name (AK1.1)
850
D:R
U:R
I:R
U:R
cm_account_accessors
cm accessor id (FK)
cm acct (FK)
852
I:R
U:R
D:C
U:R
I:R
U:R
D:R
U:R
cm accessors
854
cm accessor id
type
name (AK1.1)
user name (AK2.1)
password
directory (AK3.1)
status
D:R
U:R
810
cm accounts
cm acct id
acct name (AK1.1)
acct directory (AK2.1)
user name (AK3.1)
password
contact email
contact lname
contact fname
max content kb
has ftp acct
acct_cat_id (FK)
default title
default author
allows secure
default sec key
default sec interval
acct level
parent acct id (FK)
site id (FK)
acct type
cod client_id
salesman name
salesman email
contact phone
contract ids
setup username
setup date
status
default_stream_cat_id (FK)
default exp interval
total_kb_used
tz_id
D:R
U:R
D:R
U:R
D:R
U:R
I:R
U:R
D:R
U:R
D:R
U:R
I:R
U:R
U:R

FIG.8a



cm_playlists
822
I:R
U:R
cg id (FK)
cm acct id (FK)
sort_order (FK)
I:R
U:R
categories
838
category_id
parent_id
name
description
status
D:R
U:R
I:R
U:R
D:SN
U:SN
cm assets
812
stream id (FK)
cm acct id (FK)
creation date
description
audio video
platform
last modified date
codec
time length
kb size
expiration date
content uploader
processed date
trashed_date
orig filename
I:R
U:R
I:R
U:R
cm_account_sites
856
cm acct id (FK)
site id (FK)
I:R
U:R
I:R
U:R
U:R
D:C
U:R
I:R
U:R
I:R
U:R
cm asset locations
location id
cm vol id (FK)(AK1.1)
stream id (FK)(AK1.2)
status
818
cm stream keywords
stream id (FK)
keywords
816
D:R
U:R
D:R
U:R
I:R
U:R

FIG.8b

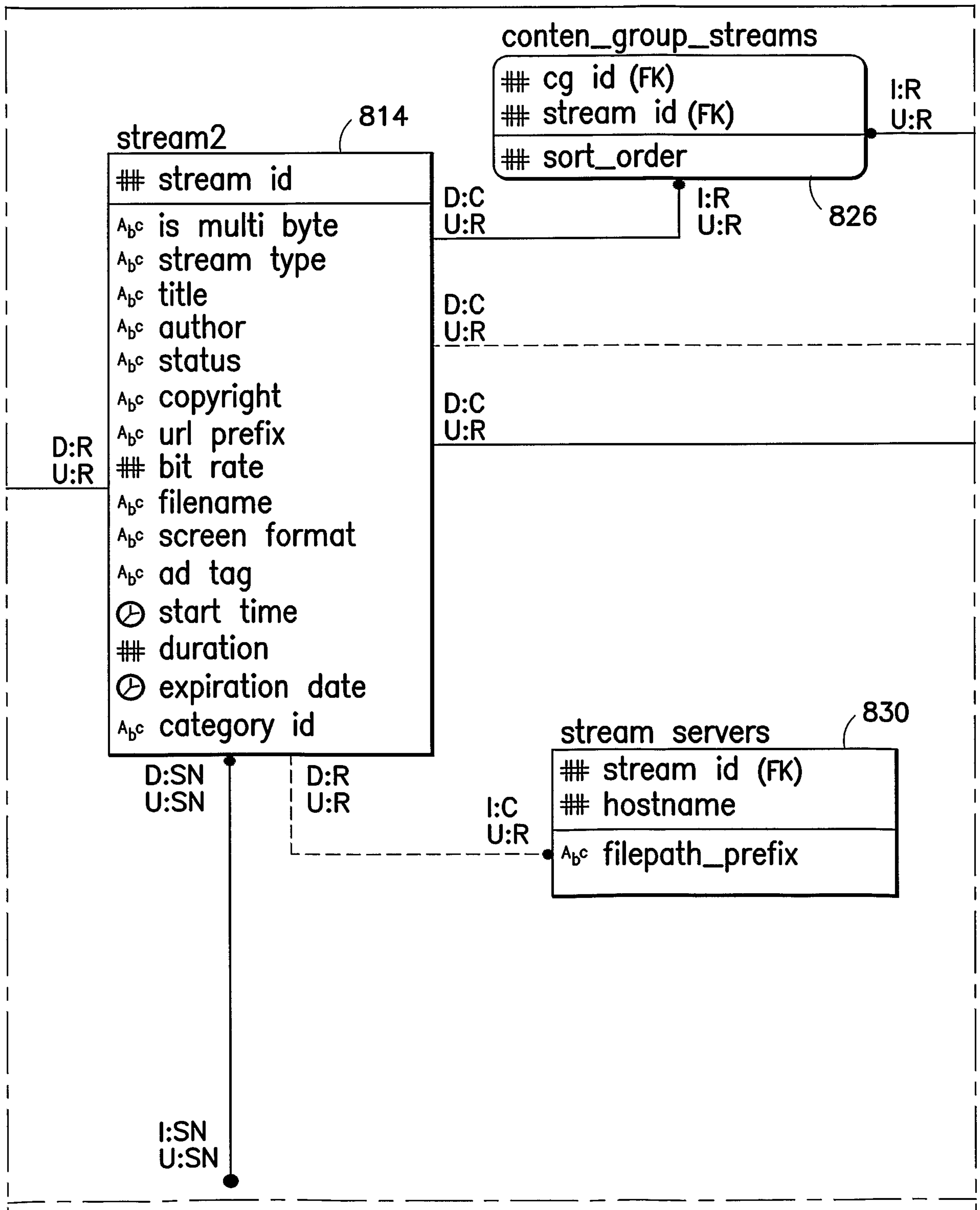
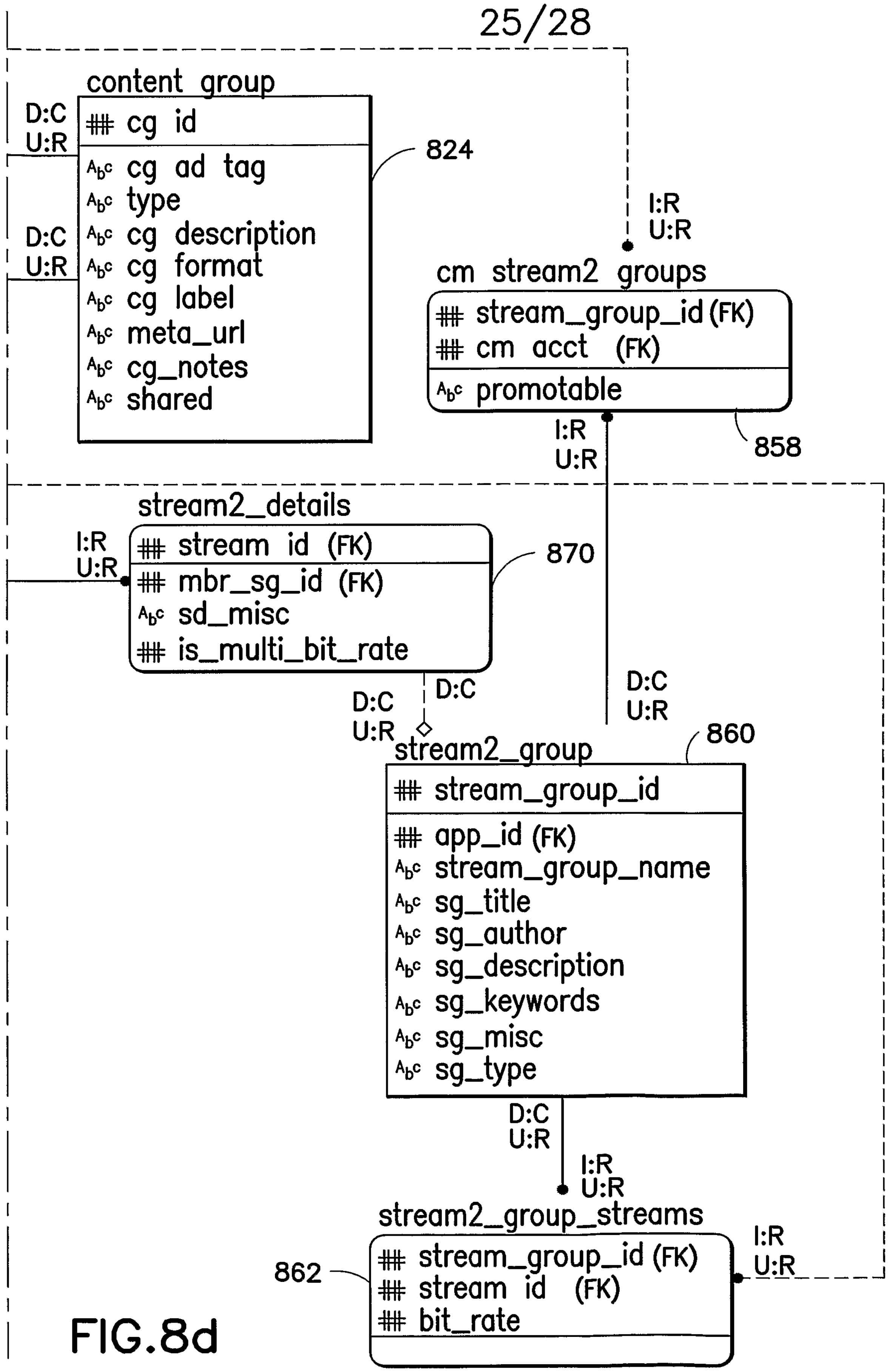

conten_group_streams
cg id (FK)
stream id (FK)
sort_order
826
I:R
U:R
I:R
U:R
stream2
814
stream id
is multi byte
stream type
title
author
status
copyright
url prefix
bit rate
filename
screen format
ad tag
start time
duration
expiration date
category id
D:C
U:R
D:C
U:R
D:C
U:R
D:R
U:R
D:SN
U:SN
D:R
U:R
I:C
U:R
I:SN
U:SN
stream servers
830
stream id (FK)
hostname
filepath_prefix

FIG.8c



content group
### cg id
Abc cg ad tag
Abc type
Abc cg description
Abc cg format
Abc cg label
Abc meta_url
Abc cg_notes
Abc shared
824
D:C
U:R
D:C
U:R
I:R
U:R
cm stream2 groups
### stream_group_id (FK)
### cm acct (FK)
Abc promotable
858
I:R
U:R
stream2_details
### stream id (FK)
### mbr_sg_id (FK)
Abc sd_misc
### is_multi_bit_rate
870
I:R
U:R
D:C
D:C
U:R
D:C
U:R
stream2_group
860
### stream_group_id
### app_id (FK)
Abc stream_group_name
Abc sg_title
Abc sg_author
Abc sg_description
Abc sg_keywords
Abc sg_misc
Abc sg_type
D:C
U:R
I:R
U:R
stream2_group_streams
862
### stream_group_id (FK)
### stream id (FK)
### bit_rate
I:R
U:R

FIG.8d

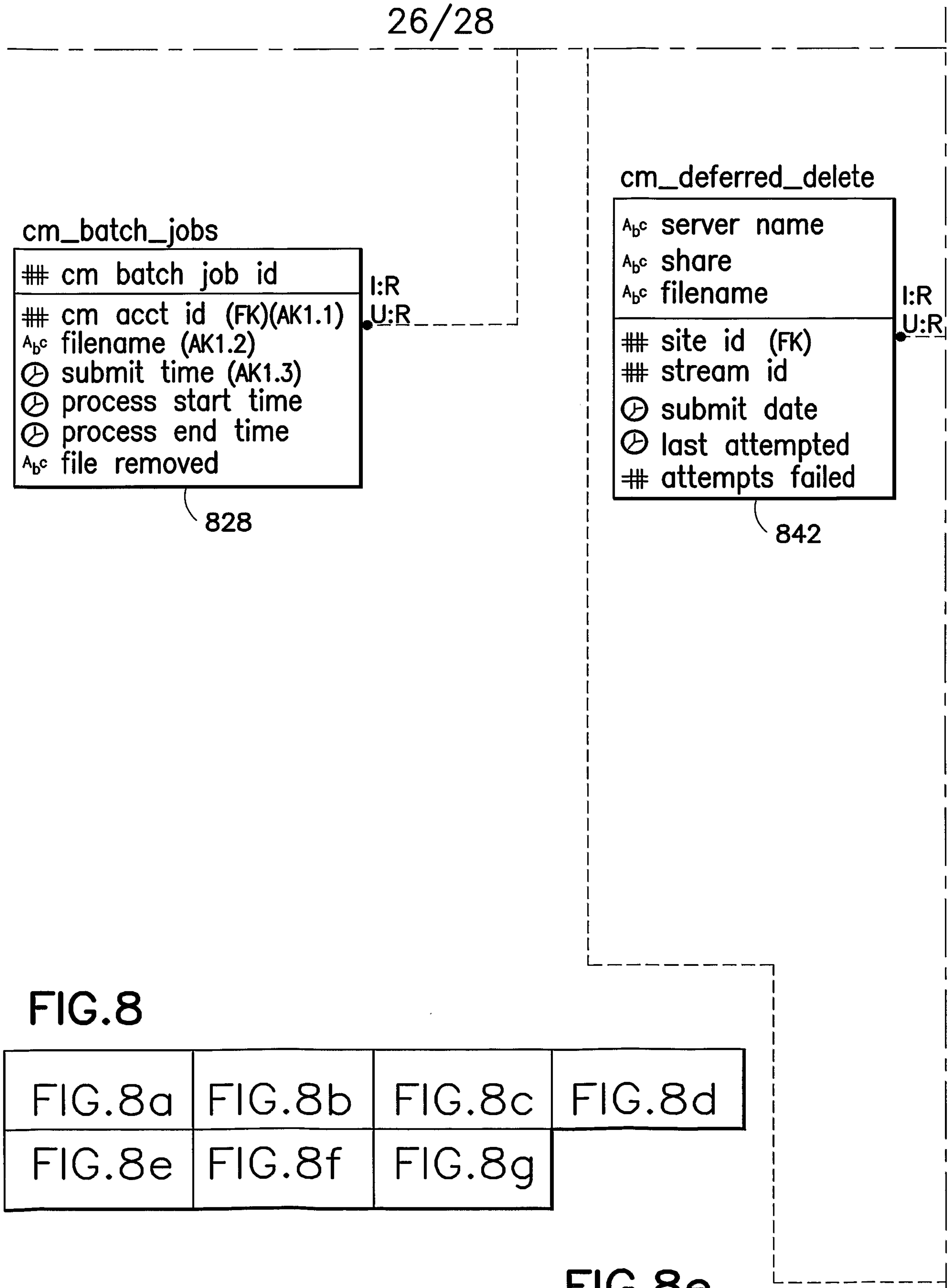


FIG.8

FIG.8a	FIG.8b	FIG.8c	FIG.8d
FIG.8e	FIG.8f	FIG.8g	

FIG.8e

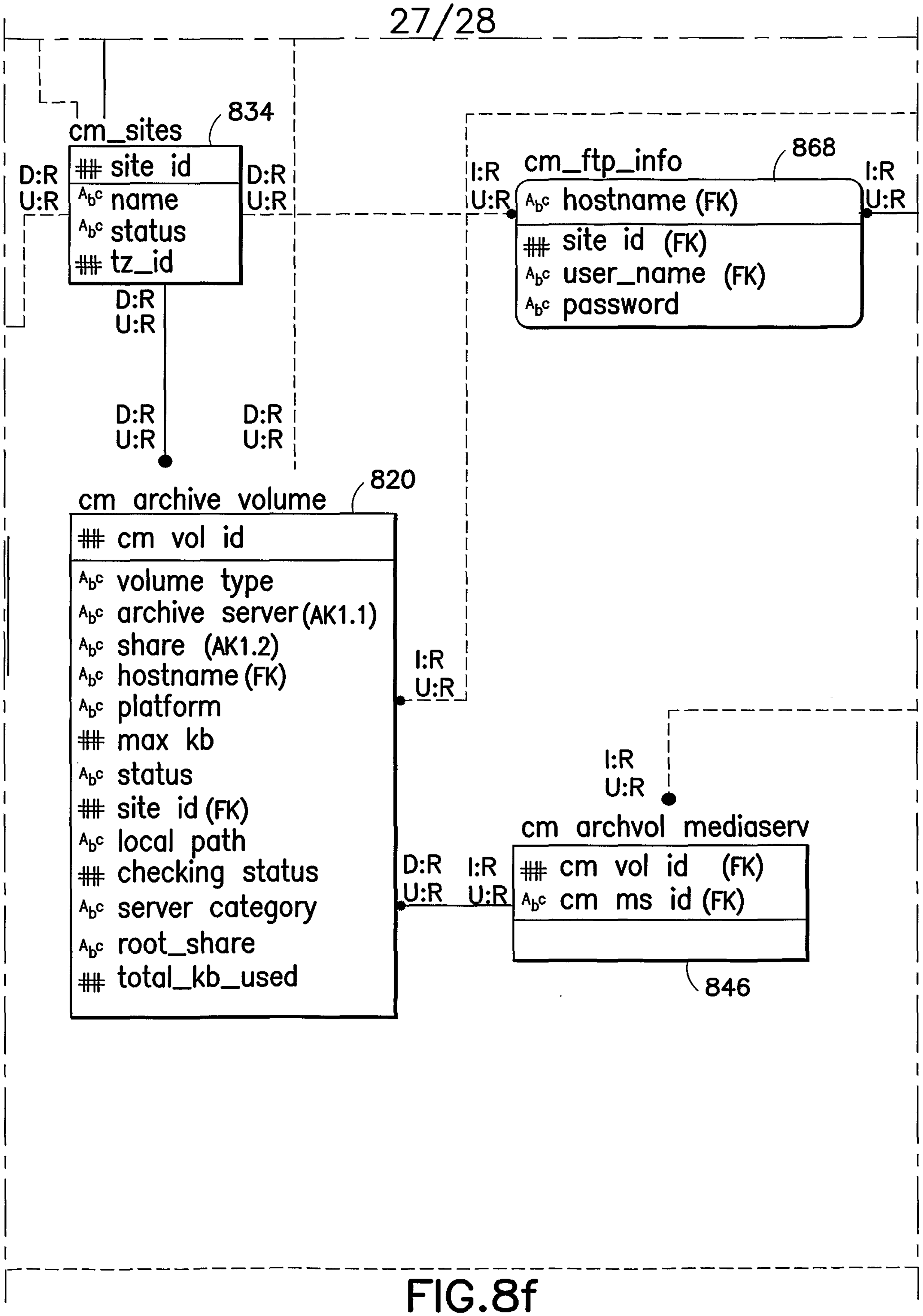

cm_sites
834
site id
name
status
tz_id
D:R
U:R
D:R
U:R
D:R
U:R
cm_ftp_info
868
hostname (FK)
site id (FK)
user_name (FK)
password
I:R
U:R
I:R
U:R
D:R
U:R
D:R
U:R
cm archive volume
820
cm vol id
volume type
archive server (AK1.1)
share (AK1.2)
hostname (FK)
platform
max kb
status
site id (FK)
local path
checking status
server category
root_share
total_kb_used
I:R
U:R
I:R
U:R
cm archvol mediaserv
cm vol id (FK)
cm ms id (FK)
846
D:R
U:R
I:R
U:R

FIG.8f

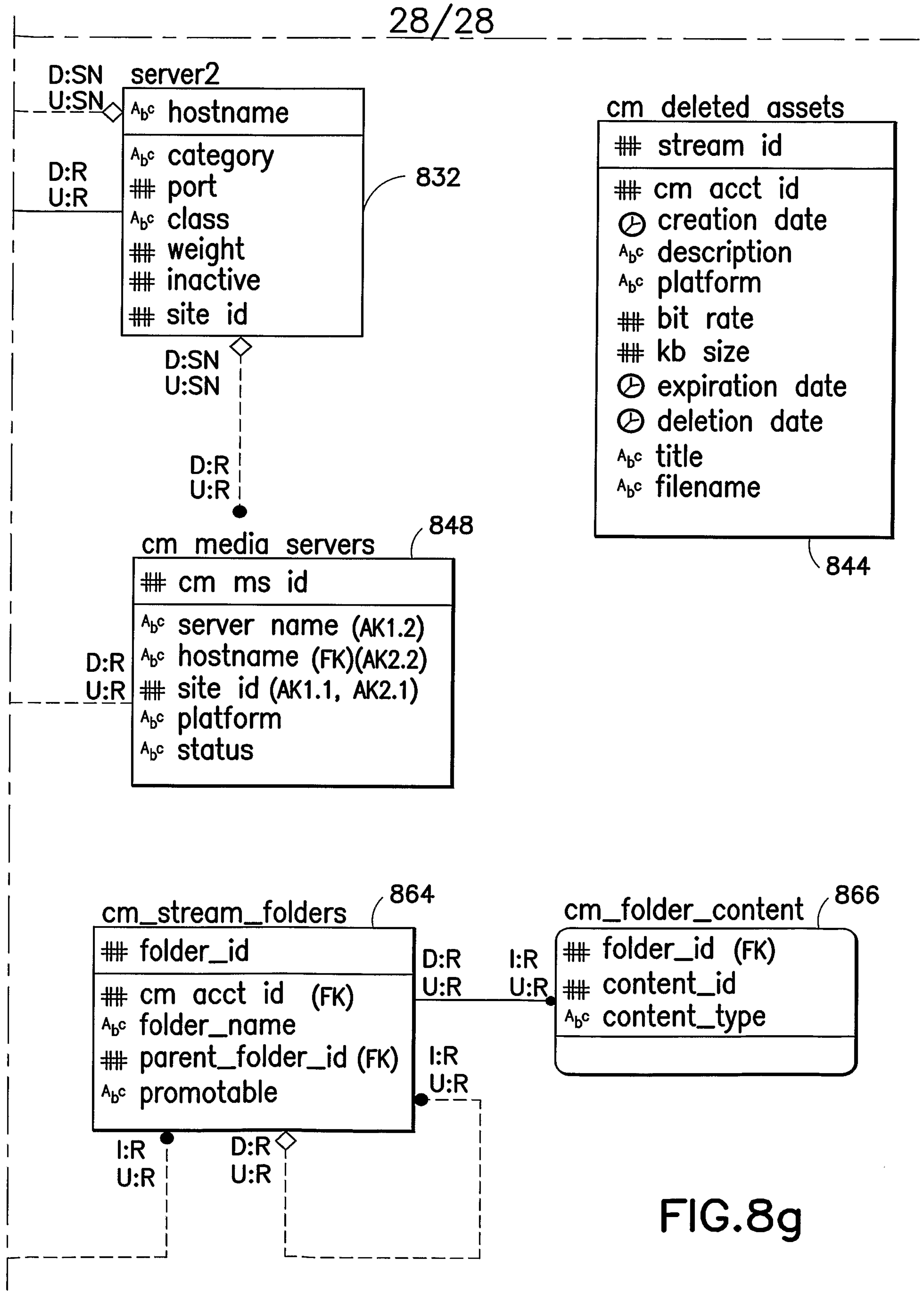

D:SN
U:SN
server2
hostname
category
port
class
weight
inactive
site id
832
D:R
U:R
D:SN
U:SN
D:R
U:R
cm deleted assets
stream id
cm acct id
creation date
description
platform
bit rate
kb size
expiration date
deletion date
title
filename
844
cm media servers
848
cm ms id
server name (AK1.2)
hostname (FK)(AK2.2)
site id (AK1.1, AK2.1)
platform
status
D:R
U:R
cm_stream_folders
864
folder_id
cm acct id (FK)
folder_name
parent_folder_id (FK)
promotable
D:R
U:R
I:R
U:R
cm_folder_content
866
folder_id (FK)
content_id
content_type
I:R
U:R
I:R
U:R
D:R
U:R

FIG.8g

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US02/01840

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) :G06F 15/16
US CL : 709/203, 231
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 709/203, 231

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EAST, IEEE Online
search terms: streaming, media, client, server, uploading, web site, internet

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	CHO et al. "Multimedia Service Interworking over Heterogeneous Networking Environments" IEEE 1999 pages 63-69,	1-34
X,E	US 2002/0010759 A1 (HITSON et al.) 24 January 2002 Figures 6-8	1-34
X,P	US 6,248, 946 B1 (DWEK) 19 June 2001, figures 1, 3a-3b	1-34
X,P	US 6,226,672 A (DEMARTIN et al.) 01 May 2001	1, 7, 12, 19, 28

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:
"A"	document defining the general state of the art which is not considered to be of particular relevance
"E"	earlier document published on or after the international filing date
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O"	document referring to an oral disclosure, use, exhibition or other means
"P"	document published prior to the international filing date but later than the priority date claimed
"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&"	document member of the same patent family

Date of the actual completion of the international search
22 MARCH 2002

Date of mailing of the international search report
10 MAY 2002

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231
Facsimile No. (703) 305-3230

Authorized officer
LARRY DONAGHUE
[illegible]
Telephone No. (703) 305-9675

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

CORRECTED VERSION

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
25 July 2002 (25.07.2002)

PCT

(10) International Publication Number
WO 02/057943 A1

(51) International Patent Classification⁷: **G06F 15/16**

(21) International Application Number: PCT/US02/01840

(22) International Filing Date: 18 January 2002 (18.01.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/263,058 18 January 2001 (18.01.2001) US

(71) Applicant *(for all designated States except US)*: **YAHOO! INC.** [US/US]; 701 First Avenue, Sunnyvale, CA 94089 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **JUSTIN, Paul, Madison** [US/US]; 1504 Concord Drive, Richardson, TX 75081 (US). **ANTHONY, Joseph, Rodiger** [US/US]; 5004 Lake Vista Drive, The Colony, TX 75056 (US). **SYED, Noman, Kazmi** [PA/US]; 12480 Abrams Road, #2601, Dallas, TX 75243 (US). **WILLIAM, David, Turner** [US/US]; 6326 Bryan Parkway, Dalls, TX 75214 (US). **STEVEN, Alexander, Roberts** [US/US]; 5406 Kingston Drive, Richardson, TX 75082-4067 (US). **GHOUSUDDIN, Syed** [US/US]; 504 Sheffield Drive, Richardson, TX 75081 (US). **VINAY, Polavarapu** [IN/US]; 6913 Valleyview Lane- Apt. #230, Irvin, TX 75039 (US). **BHARGAVARAM, Bulleyya, Gade** [IN/US]; 1213 Meadow Creek Drive, Apt. #J, Irving, TX 75038 (US).

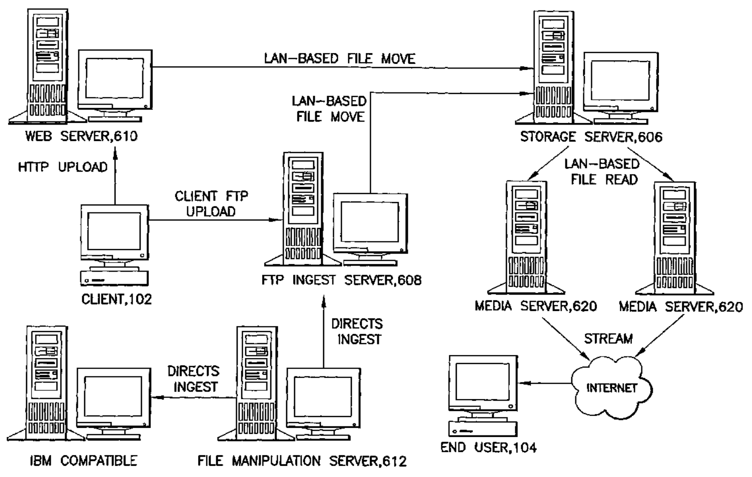
(74) Agents: DECARLO, James, J. et al.; Stroock & Stroock & Lavan LLP, 180 Maiden Lane, New York, NY 10038 (US).

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

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW),

[Continued on next page]

(54) Title: METHOD AND SYSTEM FOR MANAGING DIGITAL CONTENT, INCLUDING STREAMING MEDIA



(57) Abstract: Method and system for uploading, managing and delivering digital content, including streaming media. The system according to one embodiment allows receives digital content from the client (102), assigns a stream identifier (ID) to the content and stores the content. The client is given a playlist uniform resource locator (URL) for publishing on its web site (610), the URL including the stream ID. Activation of the URL by an end user (104) causes the stream to be served to the end user, without the client receiving or providing an indication of the specifics of where the content was stored. An embodiment of the present invention provides a system and method that permit clients to actively manage their content, including defining logical folders and subfolders containing item(s) of the content and defining logical stream groups, containing items of the content.

WO 02/057943 A1

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, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

(48) Date of publication of this corrected version:
23 January 2003

(15) Information about Correction:
see PCT Gazette No. 04/2003 of 23 January 2003, Section II

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

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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
20 March 2003 (20.03.2003)

PCT

(10) International Publication Number
WO 03/023781 A1

(51) International Patent Classification[7]: G11B 27/10, 27/32

(21) International Application Number: PCT/US02/28485

(22) International Filing Date: 6 September 2002 (06.09.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/318,721 10 September 2001 (10.09.2001) US

(71) Applicant *(for all designated States except US)*: **THOMSON LICENSING S.A.** [FR/FR]; 46, Quai A. Le Gallo, F-92648 Boulogne Cedex (FR).

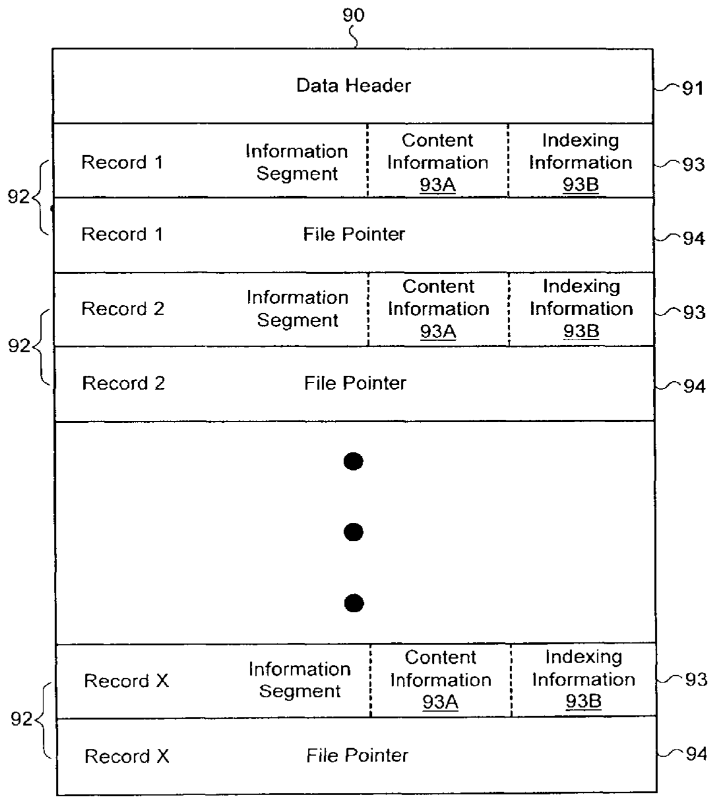
(72) **Inventors; and**
(75) **Inventors/Applicants** *(for US only)*: **GUILLEN, Newton, Galileo** [PA/US]; 742 Marana Drive, Carmel, IN 46032 (US). **JOHNSON, Lisa, Renee** [US/US]; 9109 Dew Circle, Indianapolis, IN 46260 (US). **CHEAH, Sin, Hui** [MY/US]; 14648 Brahms Drive, Apt. 2124, Carmel, IN 46032 (US).

(74) **Agents: TRIPOLI, Joseph, S.** et al.; Thomson multimedia Licensing Inc., P.O. Box 5312, Princeton, NJ 08540 (US).

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

[Continued on next page]

(54) **Title:** EXTENSION OF M3U FILE FORMAT TO SUPPORT USER INTERFACE AND NAVIGATION TASKS IN A DIGITAL AUDIO PLAYER



(57) **Abstract:** A digital audio player (10) and a data structure and method for providing an audio playlist (90). The data structure includes a playlist record (92) for each audio data file, each playlist record (92) including a file pointer segment (94) and an information segment (93) having a plurality of content information fields (93A) and at least one indexing information field (93B). A method of browsing audio data file content information in an audio data player (10) having a user interface includes providing at least one playlist (90) including records (92) stored in a predetermined sequence and including an information segment (93) including content information fields (93A) and indexing information fields (93B), outputting the content information fields (93A) to the user interface, receiving a playlist navigation signal from the user interface, and in response to the playlist navigation signal, using the indexing information fields (93B) to locate and output the content information fields (93A) for another record, the records (92) related by the predetermined sequence and the navigation signal.

WO 03/023781 A1

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*
— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

EXTENSION OF M3U FILE FORMAT TO SUPPORT USER INTERFACE AND NAVIGATION TASKS IN A DIGITAL AUDIO PLAYER

BACKGROUND OF THE INVENTION

1. Field Of The Invention.

The present invention relates to an apparatus and a method for processing digitally encoded audio data, and in particular, to a method, an apparatus, and a data structure related to an audio data file playlist.

2. Description Of The Related Art.

The use of portable audio data players capable of playing digitally encoded audio data has become commonplace. In particular, relatively small handheld devices that can process digitally encoded audio data stored on solid state memory devices have become popular. Additionally, as demand has increased for higher data storage capacity in portable audio data players, another generation of players that include miniaturized high capacity hard drives has been developed and is gaining popularity.

In an audio data player, the digital audio data is loaded into a data storage device by first downloading the data to a PC from an audio CD, the Internet, or another digital audio device. The data is then usually compressed according to a selected encoding format and loaded into the data storage device associated with the audio data player.

The audio data is decompressed/decoded by the audio data player during playback according to the selected encoding format. A variety of encoding formats for compressing and decompressing audio data is available. As used hereinafter, the term encoding format refers to any encoding/decoding scheme that specifies the syntax and semantics of a compressed bitstream and how the bitstream must be decompressed for reproduction. Such encoding formats include, but are not limited to, MP3 and MP3 Pro.

For MP3 encoded audio data files, the data file is prepended or appended with a special set of frames called an ID3 tag. The ID3 tag contains descriptive text and other data relevant to the audio data file. For example, the tag may include title, artist, album, year, comments, and genre. ID3 tag information is useful for searching, sorting, and selecting specific audio data files based on the information contained in the ID3 tag. Because ID3 tag information is often stored as textual characters, the information can be displayed on the display screen of an audio data player.

Most PC-based audio data file management programs allow the user to create and edit playlists that can then be downloaded to a portable audio data player and used for playing a select sequence of audio data files. One such form of playlist typically associated with MP3 audio data files is known as an M3U playlist. An M3U playlist consists simply of a text file containing a sequential list of paths or locations of data audio files included in the playlist. Thus, a playlist created on a PC and downloaded to an audio data player may be used to selectively play a sequence of audio data files that are contained in the data storage of the audio data player. However, the M3U file format includes only the file location or path information and a comment field. Thus, the M3U file format allows the player to playback a predetermined sequence of audio data files, but does not contain other audio data file information such as the information contained in an ID3 tag of an MP3 audio data file.

PC-based audio data file management programs also allow the user to sort available audio data files by their content, such as by ID3 fields for MP3 audio data files. PCs generally have the processing power to quickly extract the content description information from the audio data files and also have the necessary memory to store this information and display it in a timely manner to the user. However, such processing is generally not practical in non-PC-based audio data players, particularly portable or hand-held players, which have limited processing power and memory. This limitation is especially acute in audio data players having high-capacity data storage that is able to store several hundred or thousand audio data files. Therefore, browsing available audio data files in various sequences according to their ID3 information has not been available in non-PC-based audio data players.

BRIEF SUMMARY OF THE INVENTION

The present invention addresses some of the above-noted limitations of audio data players, particularly handheld audio players, by providing a data structure for audio playlist records having both content information and indexing information. The method of browsing audio data file content information utilizes the playlist's content and indexing information. The resulting audio data player having a microcontroller coupled with data storage and an audio decoder for processing encoded audio data files and audio playlist files may then quickly and conveniently allow a user to review, select, and modify audio file playlists and save those modified playlists on other systems, e.g., a user's PC.

In particular, the present invention provides a data structure that is an extension of the M3U file format used to store audio playlists. The data structure according to the present invention uses an M3U comment field format to add audio content information descriptive of the content of the audio data file, and indexing information indicating the relative location of related playlist records. Content information can include, for example, ID3 tag information found in MP3 files. Additionally, the data structure can be sorted by one or more of the content information fields.

The present invention also provides a method of adding to an audio playlist content and indexing information for each playlist record. The playlist files can be used by an audio data player to later access audio content information for all available audio data files in data storage without having to again access the data directly from individual audio data files. Additionally, multiple audio playlists can be created and stored, each being sorted by a different content information field, for example, artist, album, title, genre, etc.

The present invention also provides a method of browsing audio data file content information in an audio data player by providing a playlist having records stored in a predetermined sequence and including a content and indexing information segment. The content information includes fields descriptive of the content of the related audio data file and the indexing information includes fields providing the relative location of related playlist records. For example, a playlist sorted by genre may for a single genre include several albums by a particular artist. The indexing information provides quick and efficient navigation among records that are related, for example, by genre, artist, and album. Additionally, the content information fields may be provided to an audio data player output device, for example a display, and navigation to the content information of other playlist records may be provided in response to a playlist navigation signal and guided by the indexing information fields.

The present invention also provides an audio data player having a microcontroller coupled with data storage capable of storing audio data files and playlist files and having software capable of reading the playlist file records and outputting a navigable list of at least a portion of content information fields of the playlist records according to a predetermined sequence.

The audio data player generally includes a microcontroller coupled with a user interface, data storage, buffer memory, and an audio decoder. The user interface includes an LCD and a keyboard having various multi-way and multi-function switches. The audio data player also provides a universal serial bus ("USB") port for connection to a PC or other USB-equipped device. By connecting the audio data player to a PC via the USB port, audio data files and audio playlists may be downloaded to the audio data player and stored into data storage. In one embodiment, the data storage comprises a 10 GB hard drive; however, other moving data storage media or solid state memory devices, such as flash memory cards, may also be used. In this embodiment, the user interface provides menu driven selection, sorting, and playback of audio data files. Additionally, during playback of an audio data file, the LCD displays ID3 tag information such as title, artist, album, and genre. The LCD screen may also display other information such as elapsed playback time, volume level, and preset DSP mode.

The disclosed embodiment of the audio data player is a portable handheld unit having a rechargeable battery, 5 volt DC input, headphones output port, and line out port. Therefore, the audio data player may be used for portable applications using headphones, or for fixed applications using AC power and headphones or another audio device.

In one form thereof, a data structure stored on a computer-readable medium is disclosed including a playlist record for each audio data file, each playlist record including a file pointer segment, each playlist record including an information segment having a plurality of content information fields descriptive of the content of the audio data file and including at least one indexing information field indicating the relative location of related playlist records, and the playlist file including a data header indicating a first content information field upon which the playlist records are sorted.

In another form thereof, a method is disclosed for adding to an audio data file playlist content and indexing information for each playlist record by locating content information descriptive of the content of each audio data file, determining for each playlist record indexing information providing the relative location of related playlist records, and formatting the content and indexing information for storage in the playlist.

In yet another form thereof, in an audio data player having a user interface including an output device and a user input, a method is disclosed for browsing audio data file content information by providing at least one playlist including at least a first and second record relating to audio data files available for playback, each record stored in a predetermined sequence including a content and indexing information segment, the content information including fields descriptive of the content of the related audio data file, and the indexing information having fields providing the relative location of related playlist records, outputting via the output device at least one of the content information fields for at least a first record, receiving a playlist navigation signal from the user input, and in response to the playlist navigation signal, using at least one of the indexing information fields to locate and output at least one of the content information fields of at least a second record, the second record related to the first record by the predetermined sequence and the navigation signal.

In another form thereof, an audio data player is disclosed comprising a microcontroller coupled with data storage capable of storing audio data files and playlist files, the audio data files each having attributes descriptive of the audio content of each audio data file, the playlist files including records for each of at least a portion of the audio data file, the records in a predetermined order based on at least one of the attributes, the records including content information fields storing the attributes of each audio data file, and indexing information fields indicating the relative location of related playlist records, and the microcontroller having software capable of reading the playlist records and outputting a navigable list of at least a portion of the content information fields according to the predetermined order.

Advantageously, the disclosed data structure supports and enhances user interface and navigation tasks in viewing and selecting audio data files stored on a high-volume data storage device. Additionally, the present invention allows non-PC-based audio data players with limited processing power and memory to provide sophisticated user interface and navigation features that allow players to display the audio data files stored in data storage sorted by content information such as ID3 fields.

A further advantage of the present invention is that non-PC-based audio data players may access the audio content information for all audio data files stored in data storage without having to read the data directly from each audio file. Therefore, the user may quickly and easily sort and display the stored audio data files in a specified manner.

Another advantage of the present invention is generating a playlist file in an audio data player that contains content information and indexing information for the purpose of reducing memory and processing power requirements, and thus the cost of producing audio data players. Yet another advantage of the present invention is that the audio playlist files maintain compatibility with standard M3U playlist files and thus may be used with other PC and non-PC-based applications.

BRIEF DESCRIPTION OF THE DRAWINGS

The above mentioned and other features and objects of this invention, and the manner of attaining them, will become more apparent and the invention itself will be better understood by reference to the following description of one embodiment of the invention taken in conjunction with the accompanying drawings, wherein:

Fig. 1 is a block schematic diagram of a portable audio data player according to the present invention;

Fig. 2 is a top view of a portable audio data player according to the present invention;

Fig. 3 is a back view of the portable audio data player of Fig. 2;

Fig. 4 is a right side view of the portable audio data player of Fig. 2;

Fig. 5A is a plan view of the main sort-by menu displayed on the audio data player of Fig. 2;

Fig. 5B is a plan view of the artist menu displayed on the audio data player of Fig. 2;

Fig. 5C is a plan view of the album menu displayed on the audio data player of Fig. 2;

Fig. 5D is a plan view of the song or track menu displayed on the audio data player of Fig. 2;

Fig. 6 is a schematic diagram of a data structure for a playlist according to the present invention;

Fig. 7 is a flowchart diagram illustrating the steps for adding content and indexing information to an audio playlist file according to the present invention; and

Fig. 8 is a flowchart diagram illustrating the steps for creating an audio playlist file according to the present invention.

Corresponding reference characters indicate corresponding parts throughout the several views. Although the drawings represent embodiments of the present invention, the drawings are not necessarily to scale and certain features may be exaggerated in order to better illustrate and explain the present invention. The exemplification set out herein illustrates one embodiment of the invention, in one form, and such exemplifications are not to be construed as limiting the scope of the invention in any manner.

DETAILED DESCRIPTION OF THE INVENTION

The embodiment disclosed below is not intended to be exhaustive or limit the invention to the precise form disclosed in the following detailed description. Rather, the embodiment is chosen and described so that others skilled in the art may utilize its teachings.

Fig. 1 shows a block diagram of portable audio data player 10 according to the present invention. The general arrangement and operation of the various elements are described hereinbelow. However, the details of the various elements of audio data player 10 are well known to those skilled in the art and will not be discussed here. Audio data player 10 comprises DSP 12 that controls the various elements and the overall operation of audio data player 10, including transferring data from data storage 32, through buffer memory 25, and decoding compressed audion files. DSP 12 includes a suitable amount of memory 23 and 11, for storing various instruction sets and programs for controlling the operation of audio data player 10.

DSP 12 may be programmed to perform a variety of signal processing functions during playback of a selected audio data file. In this case, the functions that DSP 12 performs during playback include, but are not limited to, decoding audio data files, volume control, digital sound equalization, and sample conversion. In that regard, DSP 12 includes onboard memory 11, wherein the decoder files, audio data files, equalizer mode selection, and various other required data are loaded during playback.

The decoder files comprise programs that control the decoding operations of DSP 12 and the audio data files include data associated with the audio content. Both the audio data files and the decoder files are stored in data storage 32. The decoder file including the programs are transferred to DSP memory 11 from data storage 32.

Audio data and decoder programs stored in data storage 32 may be encrypted, requiring that decoding program files and audio data files be decrypted by DSP 12 using one or more decryption keys. The decryption keys may also be stored in data storage 32 and may be security linked to the particular storage device or some other coded component of audio data player 10 so that audio data files encrypted for use on a particular audio data player may only be decrypted and played by that particular audio data player.

As a selected audio data file is decoded, DSP 12 provides the decoded data stream to digital to analog converter 14. D/A converter 14 converts the digital output of DSP 12 into an analog signal and provides the analog signal to headphones amplifier 16 and lineout pre-amp 40. The analog signals are amplified and provided to lineout jack 41 and headphones jack 17, both disposed on housing 13 of audio player 10.

Audio player 10 is adapted to operate with data storage 32. In this embodiment, data storage 32 is a moving data storage device, specifically a hard drive, that can be used to store various data files, including encoded audio data files, decoder files for controlling the decoding operation of DSP 12, playlist files, and computer data files, such as, for example, word processing files, presentations, and spreadsheets. A large amount of data can be readily transferred between data storage 32 and DSP 21 through data bus 33. Buffer memory 25 operates as a circular data buffer to prevent interruption of audio playback caused by a skip or other similar moving data storage device data transfer delays. Using the present invention, decoder files, playlists, and relatively large amounts of audio data may be stored on data storage 32.

In accordance with the present invention, audio data files are loaded into data storage 32 via USB port 42 from a PC, or other similar device, using music management software that encodes the audio data files in accordance with a selected encoding format, such as MP3, or MP3 Pro, and then stores the encoded data files. Such music management software is implemented using programming methods known in the art. The music management software transmits the audio data files and appropriate decoder files to audio data player 10 across data buses 43 and 33 and into data storage 32. The music management software also generates, and modifies as necessary, a system configuration file and a file attribute table to provide information regarding the various data files and decoder files stored in data storage 32. Using the configuration file and the file attributes table, audio data player 10 is able to display audio data files sorted by various groupings on display 21, determine the correct encoding format for each audio data file, and download the appropriate decoder file for each content file in response to a user selection.

Fig. 6 is a schematic diagram of an exemplary embodiment of data structure 90. Data structure 90 generally comprises data header 91, individual audio data file records 92, each record 92 including information segment 93 and file pointer 94. In the exemplary embodiment, data structure 90 is an audio playlist file that includes an M3U format. Data header 91 includes information for identifying the file and of relevance to all the individual records 92. For example, in the exemplary embodiment, the data header has the following format:

#EXTLYRAM3U <Sorting_Field> Vx.xx

The "#EXTLYRAM3U" keyword is used to identify the file as a LyraHD system playlist file intended for use with the exemplary embodiment of audio data player 10. The Sorting_Field is enclosed by the "<" and ">" characters and contains the name of the content information field, for example, an ID3 tag field, used to sort records 92 in playlist file 90. The version of the LyraHD playlist follows the Sorting_Field and is in the form of Vx.xx where x.xx represents a 3 digit decimal version number. The applications that utilize the extended M3U file format may be configured to recognize the specific order of content and indexing information shown below.

Information segment 93 includes a plurality of content information fields 93A descriptive of the content of the audio data file, for example, ID3 tag fields, and also includes at least one indexing information field 93B indicating the relative location of related playlist records 92. Table 1 includes the content information fields 93A included in the exemplary embodiment and Table 2 includes the indexing information fields 93B included in the exemplary embodiment.

TABLE 1

Field Name	Description
audioInfoKeyword	#EXTLYRAINF indicates the start of a two-line audio file record
numberOfLevelsInTrackInfo	The number of sorted levels in a particular playlist file
trackIndexInPlaylist	Indicates the order of the current record in the playlist
charsInCurrentTrackInfo	The number of characters in the current two-line audio record
albumInfoField	Album name
artistInfoField	Artist name
titleInfoField	Title
genreInfoField	Genre
trackNumberInfoField	Track number in a given album
genericInfoField	For future expansion

TABLE 2

Field Name	Description
trackIndexInLevel[level]	The order of a specific entry in the current sorting level
totalTracksInLevel[level]	The total number of different entries in the current sorting level
charsToTopOfLevel[level]	Number of characters from the end of the current record to the start of the first record in the current sorting level
charsToNextTrackInSameLevel[level]	The number of characters from the end of the current two-line record to the start of the first record in the next entry that is in the same sorting level and shares the same parent sorting level
charsToPreviousTrackInSameLevel[level]	The number of characters from the end of the current two-line record to the start of the first record in the previous entry that is in the same sorting level and shares the same parent sorting level

Playlists 90 are sorted by at least one of the content information fields 93A of Table 1. However, the playlist 90 may also have multiple sorting levels and thus each record 92 may have several levels of index information fields 93B shown in Table 2. For example, playlist 90 sorted by artist may contain three sorting levels. The first sorting level refers to all audio data files sorted by artistInfoField. The second sorting level groups all files by a particular artist and sorts each group by albumInfoField. The third sorting level groups all files by album and sorts each group by either the value of the trackNumberInfoField or the value of the titleInfoField. Although the exemplary embodiment includes five sorting levels, any number of content information and indexing information fields and sorting levels may be used by the present invention.

In the exemplary embodiment, the data structure of Fig. 6 includes data written in the typical M3U format. Specifically, file pointer segment 94 stores data in accordance with