

01-22-01

A/Prov
PATENT

Docket No. 694231/0005

Express Mail Label No. EL 419 683 038 US

01/18/01
JC984 U.S. PTO**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE****COVER SHEET FOR FILING PROVISIONAL APPLICATION FOR PATENT
(37 C.F.R. § 1.51(c)(1))**JC658 U.S. PTO
60/263058
01/18/01BOX PROVISIONAL PATENT APPLICATION
ASSISTANT COMMISSIONER FOR PATENTS
Washington, D.C. 20231

Sir:

Transmitted herewith for filing is the provisional application for patent under 37 C.F.R. 1.53(c) entitled:

METHOD AND SYSTEM FOR MANAGING STREAMING MEDIA

Inventor(s):

1. Anthony	Joseph	Rodiger
Given Name	Middle Initial or Name	Family (or Last) Name

5004 Lake Vista Drive, The Colony, TX 75056 USA

Residence (Street Address, City or Town and either State and Zip code or Foreign Country)

2.		
Given Name	Middle Initial or Name	Family (or Last) Name

Residence (Street Address, City or Town and either State and Zip code or Foreign Country)

3.		
Given Name	Middle Initial or Name	Family (or Last) Name

Residence (Street Address, City or Town and either State and Zip code or Foreign Country)☐ Additional inventors are being named on separately number sheets attached hereto.

Enclosed are:

A. Documents required:

☒ 27 page(s) of specification☒ 18 sheets of drawing

B. Additional Documents:

☐ page(s) of claims No. of claims ____☐ page(s) of Declaration and Power of Attorney☐ Statement of "Small Entity" Status Under 37 C.F.R. §1.27☐ An assignment of the invention to _____

1073301v1

C. Fees:

- ☐ A check in the amount of \$150.00 is enclosed to cover the application filing fee For Other Than A Small Entity.
- ☐ A check in the amount of \$75.00 is enclosed to cover the application filing fee For A Small Entity.
- ☐ A check in the amount of \$40.00 is enclosed for recording the Assignment.
- ☐ No filing fee to be paid at this time.

AUTHORIZATION TO CHARGE DEPOSIT ACCOUNT

- ☒ Charge Fee to Deposit Account No. 19-4709. A DUPLICATE COPY OF THIS SHEET IS ATTACHED.
- ☒ The Assistant Commissioner is hereby authorized to charge any additional fees which may be required for filing this application, or credit any overpayment to Deposit Account No. 19-4709. A DUPLICATE COPY OF THIS SHEET IS ATTACHED.

The invention was made by an agency of the United States Government or under contract with an agency of the United States Government. ☐ yes ☒ no

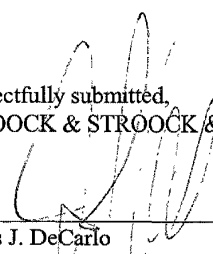
CORRESPONDENCE ADDRESS

Send all correspondence for this application to:

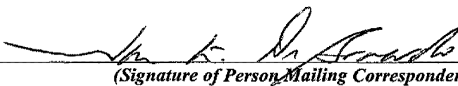
James J. DeCarlo
STROOCK & STROOCK & LAVAN, L.L.P.
180 Maiden Lane
New York, New York 10038

Respectfully submitted,
STROOCK & STROOCK & LAVAN, L.L.P.

Dated: January 18, 2001

By: 
James J. DeCarlo
Registration No. 36,120

Mailing Address:
STROOCK & STROOCK & LAVAN, L.L.P.
180 Maiden Lane
New York, New York 10038
(212) 806-5742 telephone
(212) 806-6006 facsimile

CERTIFICATE OF MAILING BY "EXPRESS MAIL" (37 CFR 1.10) Applicant(s): Anthony Joseph Rodiger			Docket No. 694231/0005	
Serial No. New Provisional	Filing Date 1/18/2001	Examiner	Group Art Unit	
Invention: METHOD AND SYSTEM FOR MANAGING STREAMING MEDIA CONTENT				
<p>I hereby certify that the following correspondence:</p> <div style="border: 1px solid black; padding: 10px; margin: 10px 0;"> Provisional Application Cover Sheet (two copies), Provisional Application (27 pages specification; 18 pages drawings) </div> <p style="text-align: center;"><i>(Identify type of correspondence)</i></p> <p>is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 in an envelope addressed to: The Assistant Commissioner for Patents, Washington, D.C. 20231 on</p> <div style="text-align: center; margin-top: 10px;"> January 18, 2001 <i>(Date)</i> </div> <div style="text-align: right; margin-top: 20px;"> <div style="text-align: center;"> Ian G. DiBernardo <i>(Typed or Printed Name of Person Mailing Correspondence)</i> </div> <div style="text-align: center; margin-top: 10px;">  <i>(Signature of Person Mailing Correspondence)</i> </div> <div style="text-align: center; margin-top: 10px;"> EL 419 683 038 US <i>("Express Mail" Mailing Label Number)</i> </div> </div>				
Note: Each paper must have its own certificate of mailing.				

P06A/REV02

Method and System for Managing Streaming Media Content

FIELD OF THE INVENTION

The present invention relates generally to a method and system for providing streaming
5 media 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.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a schematic illustrating the over all system of one embodiment of the present
invention;

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

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

Figures 4a - c are flowcharts illustrating processes of a uploading content files into the
content management system according to one embodiment of the present invention;

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

Figure 6 is a schematic illustrating a workflow of a content management system according
to an alternate embodiment of the present invention; and

Figure 7 is a schematic illustrating the database of the embodiment of Figure 6.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

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.
20 Turning first to the schematic of Figure 1, a system according to one embodiment of the present
invention is shown. In general, the system allow clients 102 to upload streaming media content to

the system, manage such content, and make a content available to end-users 104 via a web site on the Internet.

As will become apparent from the following discussion, both the client 102 and end-users 104 have computers, such as PC's, coupled to the Internet by any one of a number of known 5 manners. Furthermore, each client 102 includes an Internet browser, such as Internet Explorer or Netscape Navigator, as well as file transfer protocol (FTP) client software for interacting with various components of the system 100. Additionally, each end-user includes an Internet browser, such as Internet explorer or Netscape Navigator, and a media player, such as windows media player or real networks media. It is to be understood that although the present embodiment is described in 10 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.

As discussed in greater detail below, the client 102 uploads streaming media content to the repository server (and associated storage) 106 either directly, via an HTTP upload, or via an FTP ingest server (and associated storage) 108. Once the client 102 has uploaded its streaming media 15 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 functions: Upload new content; search the client's existing up-loaded content; create and edit playlists; browser the client's contents stored on the FTP server 108; and 20 perform a batch upload of content from the FTP server 108 to the repository 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.

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. The administrative terminals 114, in turn, are coupled to the file management server 112 to allow the administrative staff of the service provider maintaining the content management system to control the system. The script processor 116 works in conjunction with the file management server 112 to control operation of the over-all system. 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 uploads 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 of a number of programming languages, such as PERL, Visual Basic, JavaScript, C+, C++, and the like.

Central to the content management system 100 is the Content Management (CM) Database 118. As described in greater detail below with reference to Figure 2, the CM Database 118 includes numerous relational Databases. In general, the CM Database 118 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.

Under direction of the file management server 112, the uploaded content is eventually transferred from the repository server 106 to streaming media servers and associated 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.

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.

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

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 that other arrangements are within the scope of the present invention. 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.

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 contents in the event no other title or author information is provided by the client), parent account ID (in the event the current account is related to another account), and 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).

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 uploaded. Two representative sites are illustrated in Figure 1. 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 client's enduser's 104 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.

The CM Database 118 further includes a series of Databases 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 codec 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.

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, start time, duration, expiration date.

As illustrated in Figure 2, the CM Assets Table 212 has 3 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.

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.

	<u>Status</u>	<u>Meaning</u>	<u>Applicable to file located on:</u>
	Available	File is present on a media server and is available for streaming	Streaming server
10	Back up	File has been copied 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 upload or copy has failed	Repository server; media server
15	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 uploaded and not copied	Repository server

20 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.

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.

The Content Group Table 224 also includes records for each playlist, and identified by the playlist's 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.

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.

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.

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 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.

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.

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.

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.

Registration Process

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. 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 a 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

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 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.

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.

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 Streams 2 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.

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. As discussed in greater detail with reference to

Figure __, 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 account FTP file 308 as well as a listing of the client's file directories 310 and the client's content files 312.

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.

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.

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.

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.

As discussed in greater detail below, the client 102 may select one of the available batch files for either deletion or processing.

5 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 CG id's, each of which
10 identifies a different playlist for the client 102. For each CG id, the system identifies the CG description and format in the Content Group Table 224. The information is retrieved in XML format and displayed in the playlist list 318 in HTML format.

15 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.

20 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.

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
5 playing the streams associated with the playlist is described in greater detail with reference to Figures 1 and 2.

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, key words, stream id, creation date, and duration,
10 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
15 (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).

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
20 files associated with the client's username and password (and, thus, account id).

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.

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 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.

Uploading Content

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.

As described with reference to Figures 4a-c, the system of the present embodiment allows the client 102 to upload content to the repository servers 106 by any of four different processes. The first three processes (described with reference to Figures 4a and b) transfer content files from the client's FTP account on the FTP server log 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 directly to a repository server 106 via HTTP upload.

As an initial step in the HTTP upload process of Figure 4c, 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, Streams 2 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 in the form of an XML string, 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.

As will now be described with reference to Figure 4a, 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. Furthermore, although the process is described as an option on the CM web page, in alternate embodiments the script need not run.

As an initial step in the programmatic upload 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.

10 Step 410. The client 102 then logs into the system by entering its user name and password via a web page provided by the client web server 110. Step 414. Navigating the web page, the client 102 selects the "Browse FTP Content" option (Step 414), identifies a file, and submits an ingest request. Step 418.

As part of the ingest request, the client 102 is prompted to enter certain file details via a form similar to that in Figure 3b. In the present embodiment, 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").

The following optional file details may also be provided by the client 102: 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 description of an existing play list in the client's account ("PlaylistDesc"), an indication whether to add the file to the play list identified by the

aforementioned description or to replace the existing files associated with the aforementioned playlist with the file being ingested ("PlaylistAction"), 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"). The process of adding and replacing files in a playlist is discussed in greater detail below.

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.

By submitting the ingest request (for example, by clicking a button on the web page), the client 102 activates a basic command line that calls the script. Step 422. In general, the 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 and any file details entered in Step 418. 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
```

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."

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.

5

```
http://contentmgmt.broadcast.com/vbs_script_ingest.asp?username=Foo&Password=Bar&Title=
Test File&Author=Yahoo! Broadcast&CopyRight=(c)2000 All Rights Reserved&Description=
Testing The File System&KeyWords=test1|test2|test3& FileName=testfile.rm
```

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. Step 426. Based on whether the file was successfully uploaded (Step 430), 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.

Thus, if the file was not successfully uploaded, the following XML string is returned to the client 102, wherein an S code of "1" indicates a failure and the value of the S code_description indicates the reason for the failure. Step 434.

15

```
<?xml version="1.0" ?>
<response scode="1" scode_description="reason for failure" />
```

In the event that the file was successfully uploaded, the system returns the following XML string wherein the S code and S code_description indicate the successful upload (Step 438):

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

20

The system then renames the file and updates the appropriate fields in the CM database 118. Step 442. More specifically, with the Stream ID assigned to the file, a record is created in each of the CM Assets Table 212, Streams 2 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.

As will now be described with reference to Figure 4b, and continuing reference to Figure
3d, the client 102 may also perform two types of batch uploads, thereby transferring files from the
5 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.

10 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.

15 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
```

20 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::MYSECURITYKEY::122
```

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::ISSECURE::SECURITYKEY::SECURITYINTERVAL
```

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.

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.

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.

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.

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.

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, Streams 2 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

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.

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.

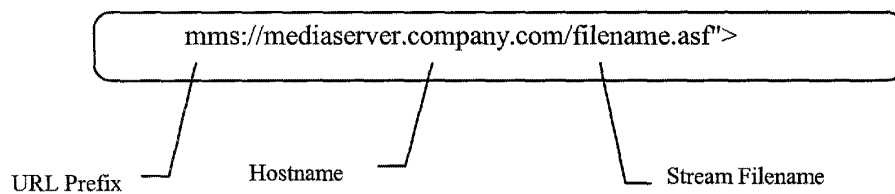
The content file being uploaded is placed on the streaming media server 120 (i.e., volume id) having the most available storage capacity. The system uses the selected volume id to create a new record in the CM Assets Locations Table 218 for the archived content file.

Creation of Playlist

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.

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 metafiles, as described below.

identifiers 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 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:



Using the individual stream URLs, the "makeplaylist" program then dynamically generates a metafile to be passed to the media player stored on the end user's computer 104. An example of a metafile for use with Windows Media Technologies 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>
```

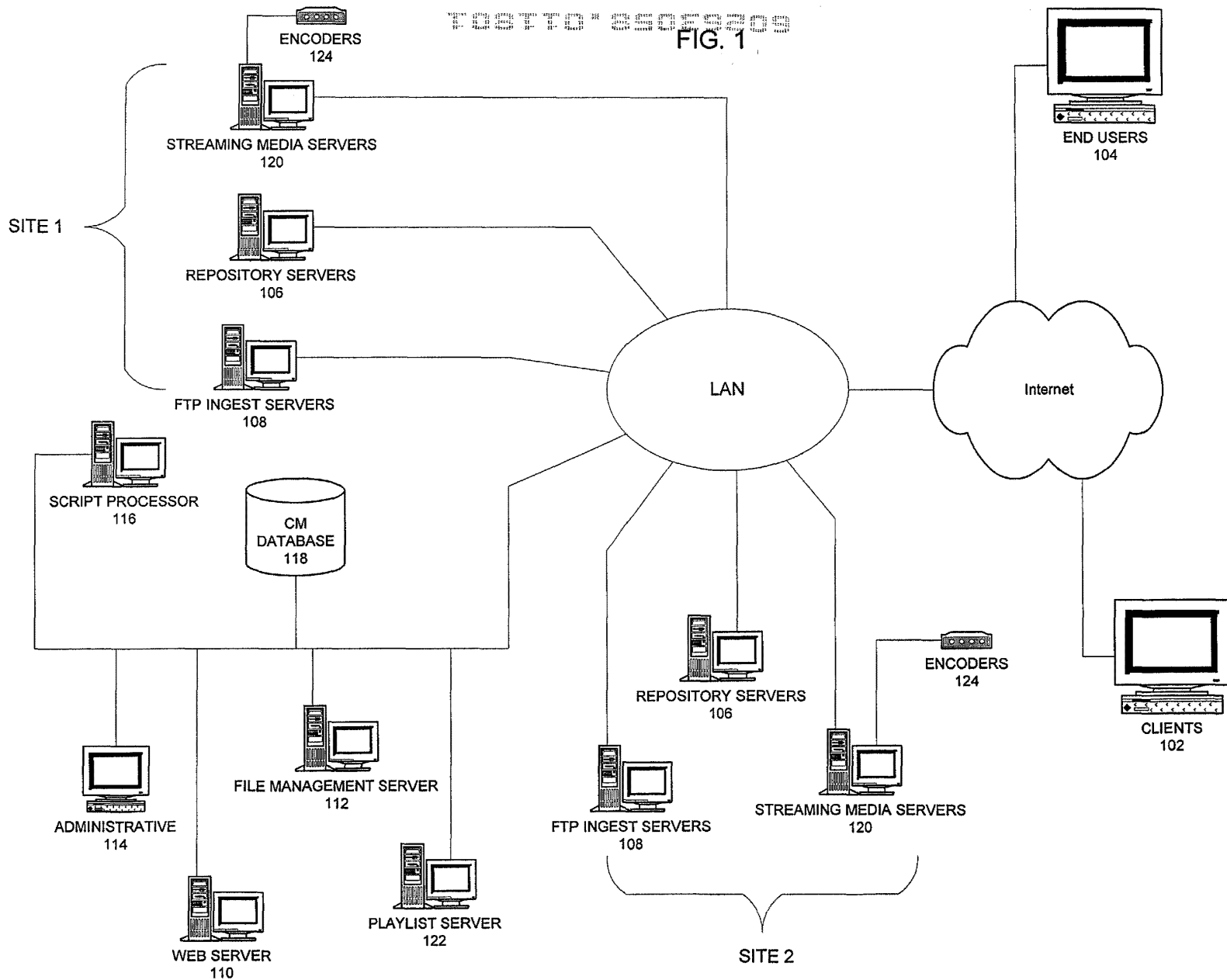
One skilled in the art will recognize, of course, that different media technologies utilize different formats of metafiles and, therefore, the term "metafile" is not limited to the ASX-type metafile shown above. Lastly, the end user's media player pulls each identified stream file from the media server 120 identified in the metafile in the order in which it appears in the metafile.

It is to be understood that other system architectures are within the scope of the present invention. For example, certain embodiment do not include repository servers and, therefore, do not maintain backed-up copies of files as in the embodiment of Figure 1. One such embodiment is shown and described in the workflow diagram of Figure 6 and the CM Database schematic of

5 Figure 7.

10
15

20



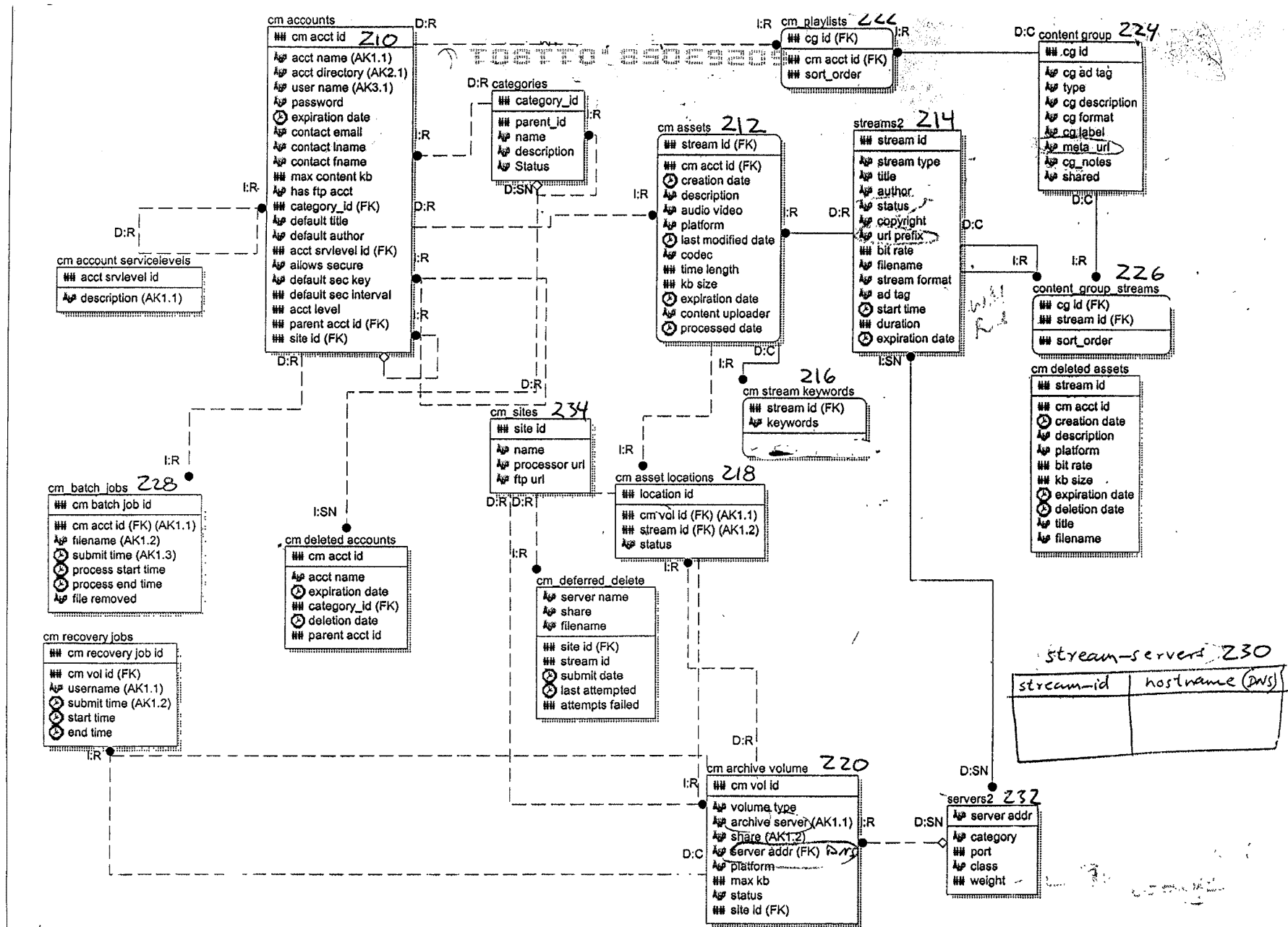


FIG 2

Content Management

MANAGE CONTENT / ADMIN ← 302

ACCOUNT PREFERENCES | LOG OFF

testuser

Account Preferences

Default Title:

Default Author:

Default Security Key:

Default Security Interval:

Upload Test Title

J Doe

Hours:

Minutes:

Set Defaults

FIG. 3a

304

1/18/01

Content Management

testuser

MANAGE CONTENT / ADMIN

[UPLOAD NEW CONTENT](#) | [BROWSE FTP CONTENT](#) | [BATCH UPLOADS](#) | [PLAYLISTS](#) | [SEARCH](#) | [RECYCLE](#)

Upload Content Form

☐ Media File:

☐ Bitrate (Kbps):

☐ Title:

☐ Author:

Copyright:

Description:

Keywords:

Current Keywords:

Secure Content? ☐

Security Key:

Security Interval:

 Hours: Minutes:

306

FIG. 35

1/18/01

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	
<input type="checkbox"/> .asx	1/16/01 15:13	0.1 KB	
<input type="checkbox"/> Copy of test.asf	1/16/01 15:13	0.7 MB	
<input type="checkbox"/> demo.asf	1/16/01 15:13	0.2 MB	
<input type="checkbox"/> File13.rm	10/9/00 12:10	10.0 KB	
<input type="checkbox"/> File131.rm	10/9/00 12:10	10.0 KB	
<input type="checkbox"/> File133.rm	10/9/00 12:10	10.0 KB	
<input type="checkbox"/> File134.rm	10/9/00 12:10	10.0 KB	
<input type="checkbox"/> File135.rm	10/9/00 12:10	10.0 KB	
<input type="checkbox"/> File136.rm	10/9/00 12:10	10.0 KB	
<input type="checkbox"/> File137.rm	10/9/00 12:10	10.0 KB	

310 {
 312 {

308

FIG.3C

Content Management

testuser

MANAGE CONTENT / ADMIN

[UPLOAD NEW CONTENT](#) | [BROWSE FTP CONTENT](#) | [BATCH UPLOADS](#) | [PLAYLISTS](#) | [SEARCH](#) | [RECYCLE](#)

Batch Uploads

Available Batch Files

PROCESS

DELETE

	File Name	Creation Time	Size	
314 {	<input type="checkbox"/> 1-100.bul	10/9/00 12:10	7.9 KB	
	<input type="checkbox"/> 101-200.bul	10/9/00 12:10	8.1 KB	
	<input type="checkbox"/> 28-anthony.bul	10/9/00 12:10	0.1 KB	
	<input type="checkbox"/> BillDefault.txt	10/6/00 10:38	0.1 KB	
	<input type="checkbox"/> BillsSmBatch.txt	1/18/01 12:28	0.3 KB	

Batch Process Summary

	Batch File	Submit Time	Process Start Time	Process End Time
316 {	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

1/18/01

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)

Manage

Search Playlists:

Search

Playlist ID:
Description:
Format:

} 320

1/18/01

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:

▪ Content Format:

Search Playlists:

Playlist ID:

Description:

Format:

Streams List

▪ Streams:

FIG. 3f

Content Management

testuser

MANAGE CONTENT / ADMIN

[UPLOAD NEW CONTENT](#) | [BROWSE FTP CONTENT](#) | [BATCH UPLOADS](#) | [PLAYLISTS](#) | [SEARCH](#) | [RECYCLE](#)

Search Content:

BROWSE

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. 39

Content Managementtestuser

MANAGE CONTENT / ADMIN

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

Browse Content:

BROWSE

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:

Create Playlist

Delete

	ID	Title	Type	Description	Creation Date	File Size	Status	
<input type="checkbox"/>	793735	<u>StevesTest</u>	WMT		11/1/00 2:59 PM	172 KB	Available	
<input type="checkbox"/>	943435	<u>ASF Upload Test</u>	WMT		12/8/00 10:57 AM	1,034 KB	Available	
<input type="checkbox"/>	943436	<u>ASF Upload 2</u>	WMT		12/8/00 10:57 AM	127 KB	Available	
<input type="checkbox"/>	943437	<u>ASF Upload 3</u>	WMT		12/8/00 10:58 AM	127 KB	Available	
<input type="checkbox"/>	943439	<u>ASF Upload 4</u>	WMT		12/8/00 10:58 AM	127 KB	Available	
<input type="checkbox"/>	943441	<u>ASF Upload x</u>	WMT		12/8/00 10:58 AM	127 KB	Available	
<input type="checkbox"/>	944032	<u>Upload Test Title</u>	G2		12/8/00 2:44 PM	10 KB	Available	
<input type="checkbox"/>	944036	<u>Upload Test Title</u>	G2		12/8/00 2:45 PM	10 KB	Available	
<input type="checkbox"/>	944503	<u>Upload Test Title</u>	WMT		12/8/00 5:25 PM	172 KB	Available	
<input type="checkbox"/>	944510	<u>Upload Test Title</u>	G2		12/8/00 5:33 PM	10 KB	Available	
<input type="checkbox"/>	960602	<u>Upload Test Title</u>	G2		12/15/00 12:57 AM	10 KB	Available	
<input type="checkbox"/>	961158	<u>Upload Test Title</u>	WMT		12/15/00 11:02 AM	172 KB	Available	
<input type="checkbox"/>	961165	<u>injust002 upload test</u>	WMT		12/15/00 11:04 AM	172 KB	Available	
<input type="checkbox"/>	961170	<u>injust003 upload test</u>	WMT		12/15/00 11:04 AM	172 KB	Available	
<input type="checkbox"/>	961171	<u>injust004 upload test</u>	WMT		12/15/00 11:05 AM	172 KB	Available	
<input type="checkbox"/>	961172	<u>injust005 upload test</u>	WMT		12/15/00 11:06 AM	172 KB	Available	
<input type="checkbox"/>	961174	<u>CA Upload test</u>	WMT		12/15/00 11:07 AM	172 KB	Available	
<input type="checkbox"/>	961176	<u>FTP Upload Test</u>	G2		12/15/00 11:08 AM	10 KB	Available	
<input type="checkbox"/>	962850	<u>Upload Test Title</u>	G2		12/15/00 11:29 PM	10 KB	Available	
<input type="checkbox"/>	966820	<u>upload test</u>	WMT		12/18/00 10:51 AM	243 KB	Available	
<input type="checkbox"/>	966825	<u>upload test 2</u>	G2		12/18/00 10:52 AM	90 KB	Available	
<input type="checkbox"/>	966882	<u>Upload Test Title</u>	G2		12/18/00 11:27 AM	32 KB	Available	
<input type="checkbox"/>	966900	<u>test</u>	G2		12/18/00 11:32 AM	32 KB	Available	
<input type="checkbox"/>	966959	<u>i2 test</u>	WMT		12/18/00 11:43 AM	155 KB	Available	
<input type="checkbox"/>	966984	<u>i4</u>	WMT		12/18/00 11:47 AM	155 KB	Available	

1/18/01

336

Create Playlist

Description:

Content Format:

RAM

Create

Add to Another Playlist

Available Playlists:

ChrisBinney1 (RAM)

Add To Playlist

Streams List

Streams:

Upload Test Title

332

334

FIG. 3i

FIG. 38

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

1/18/01

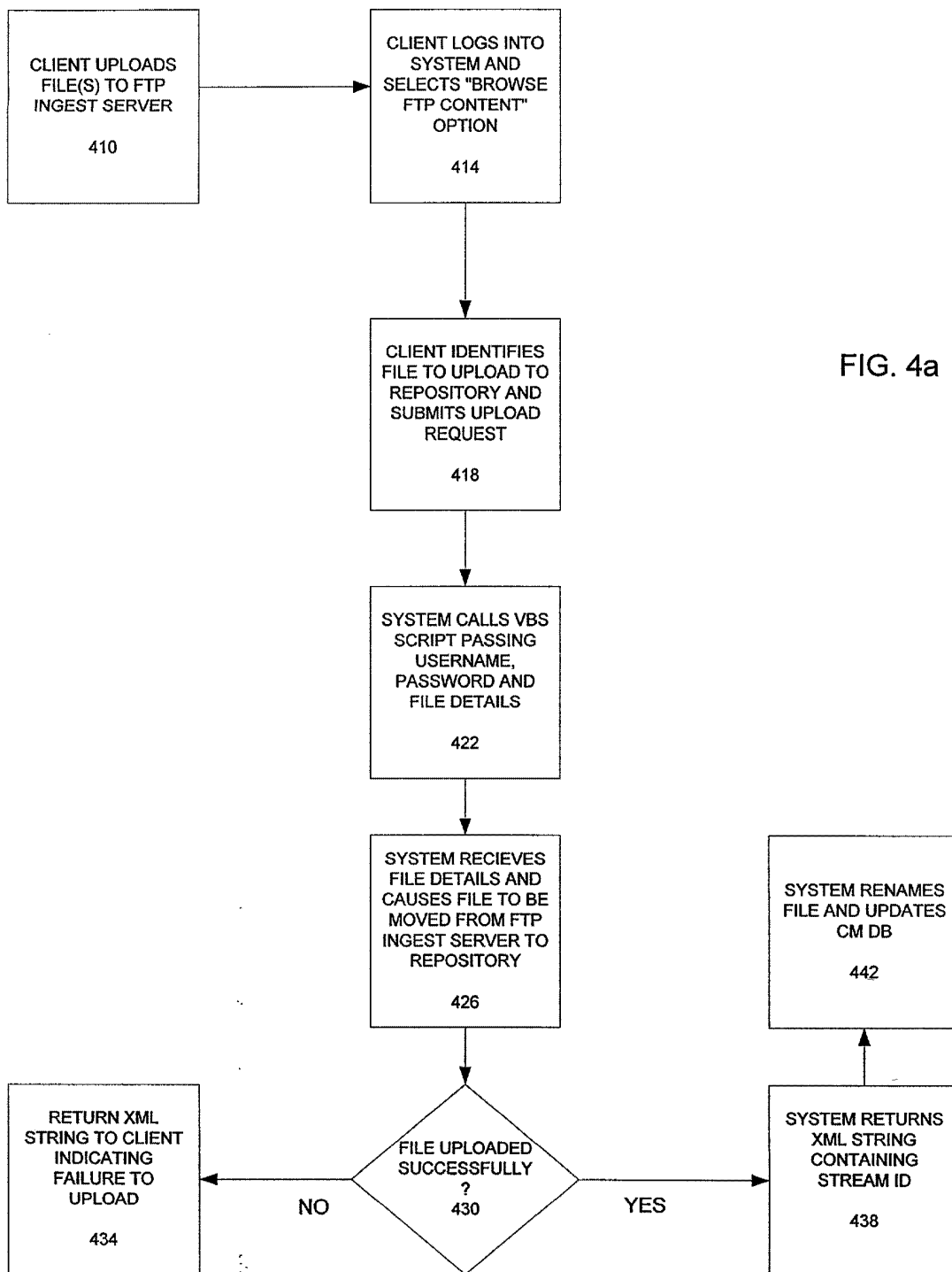
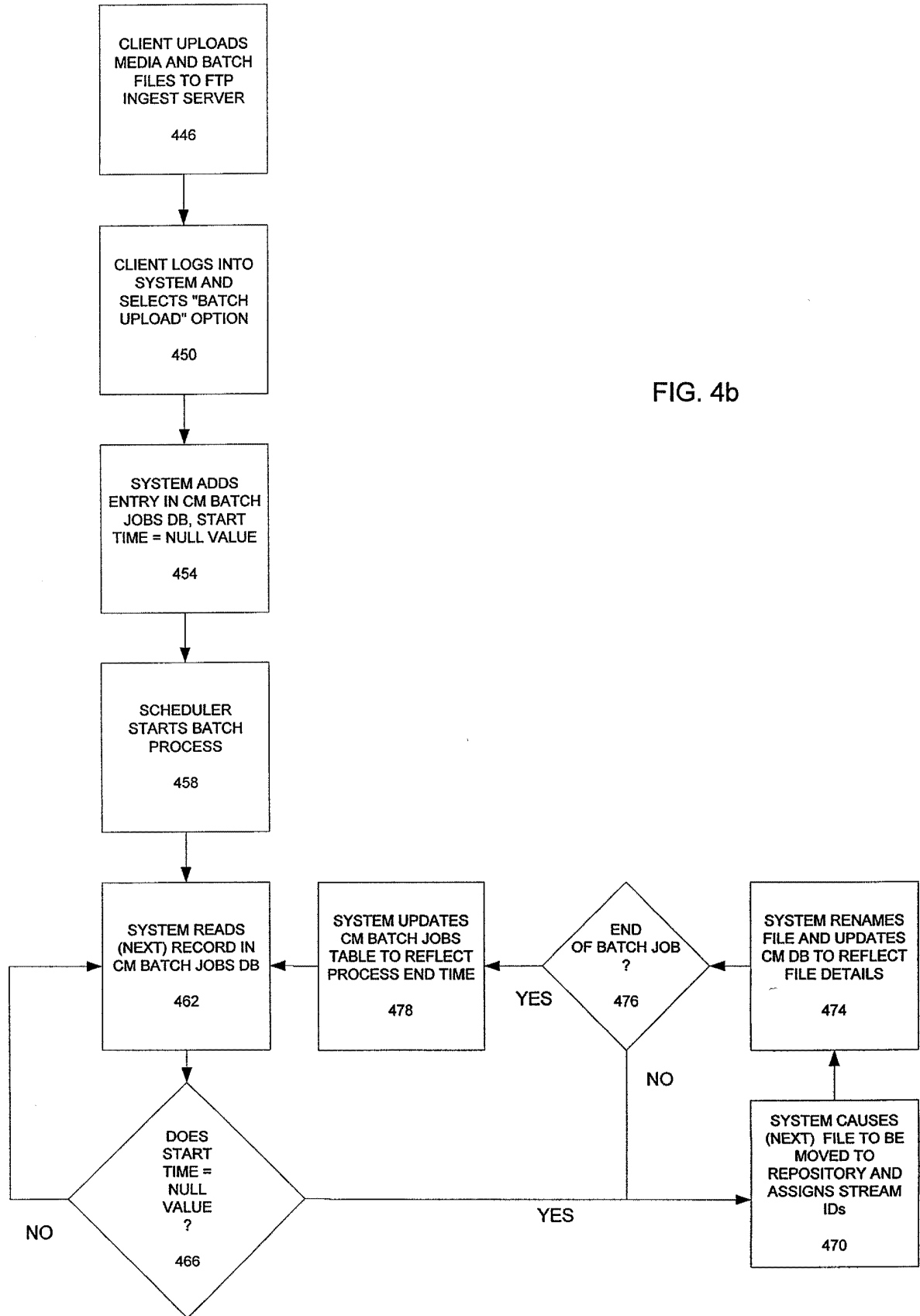


FIG. 4a

FIG. 4b



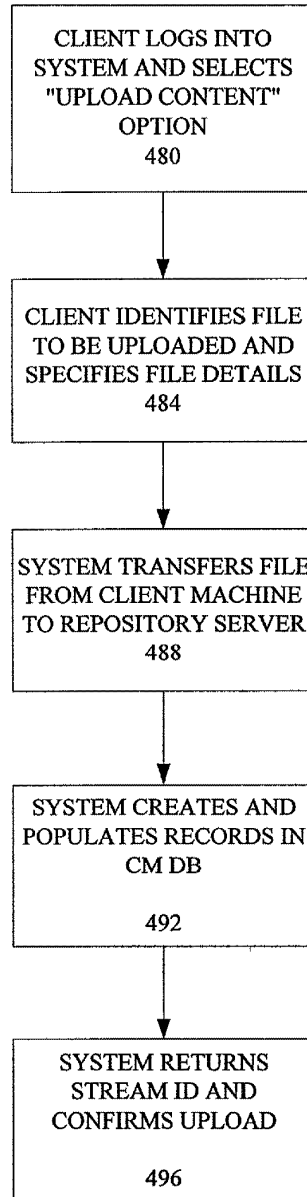


FIG. 4c

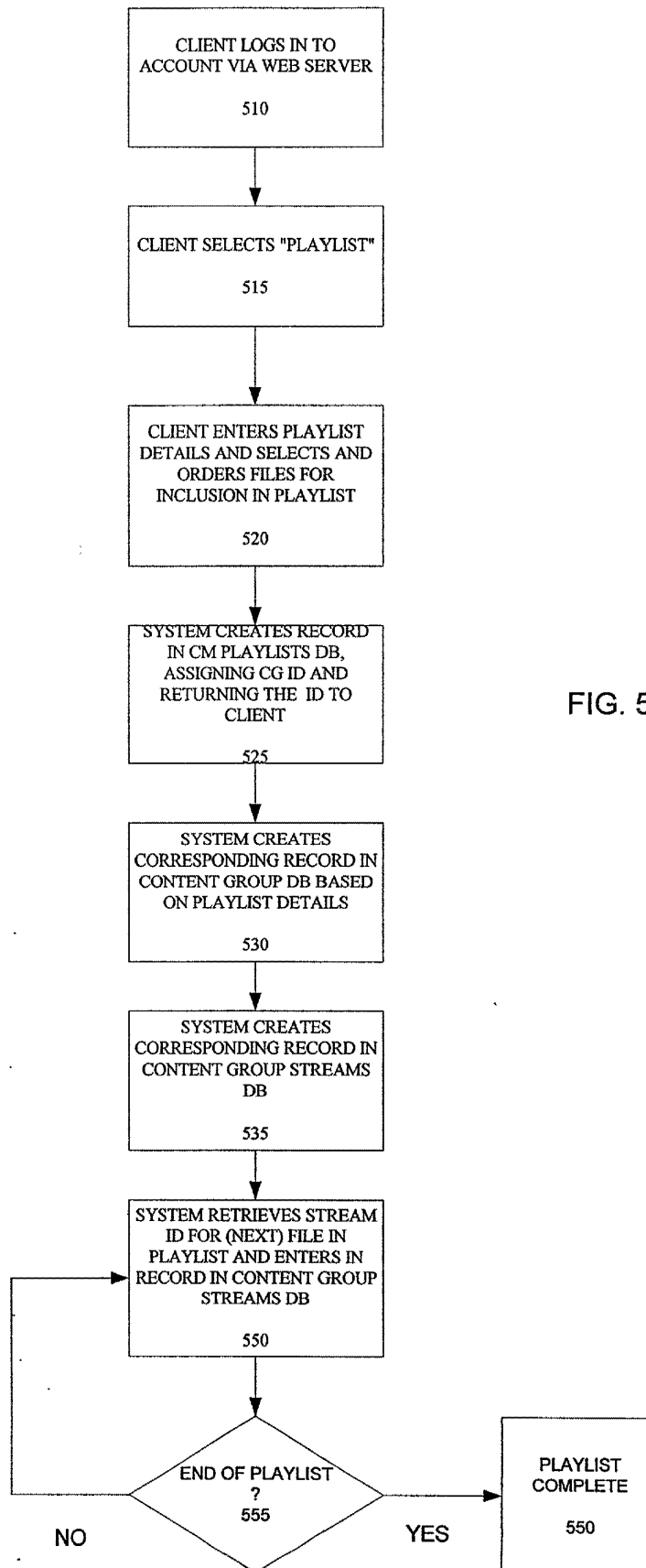


FIG. 5

FIG 6

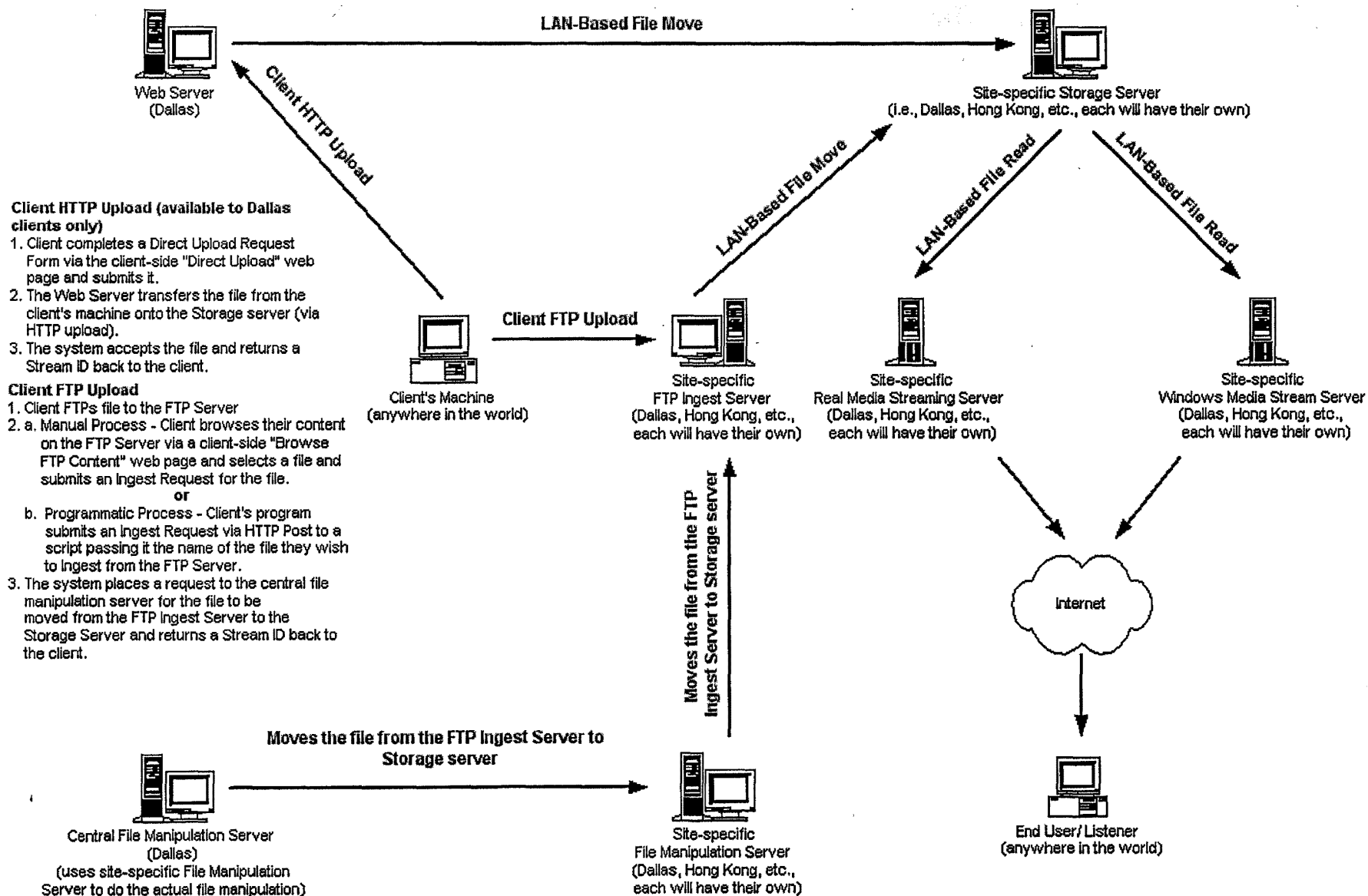


FIG. 7

