PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ^{6} : G06T 13/00	A1	**(11) International Publication Number:** WO 98/29835 **(43) International Publication Date:** 9 July 1998 (09.07.98)

(21) International Application Number: PCT/US95/13433

(22) International Filing Date: 11 October 1995 (11.10.95)

(30) Priority Data:
08/321,332 11 October 1994 (11.10.94) US

(71) Applicant: STARNET, INCORPORATED [US/US]; 1332 Enterprise Drive, West Chester, PA 19380 (US).

(72) Inventors: KUNKEL, Gerard; 558 Heritage Oak Drive, Yardley, PA 19067 (US). HEYDT, Michael; 1509 Reed Street, Coatesville, PA 19320 (US). CROSS, Jerry; 725 South 9th Street, Philadelphia, PA 19147 (US). NOCKS, Jason; Apartment 305, 1511 Manchester Court, West Chester, PA 19380 (US). PORTUGAL, Howard; 1315 Grove Road, West Chester, PA 19380 (US). MCGLADE, Alan; 3644 Royal Palm Avenue, Coconut Grove, FL 33133 (US).

(74) Agent: BLOOM, Allen; Dechert Price & Rhoads, Princeton Pike Corporate Center, P.O. Box 5218, Princeton, NJ 08543-5218 (US).

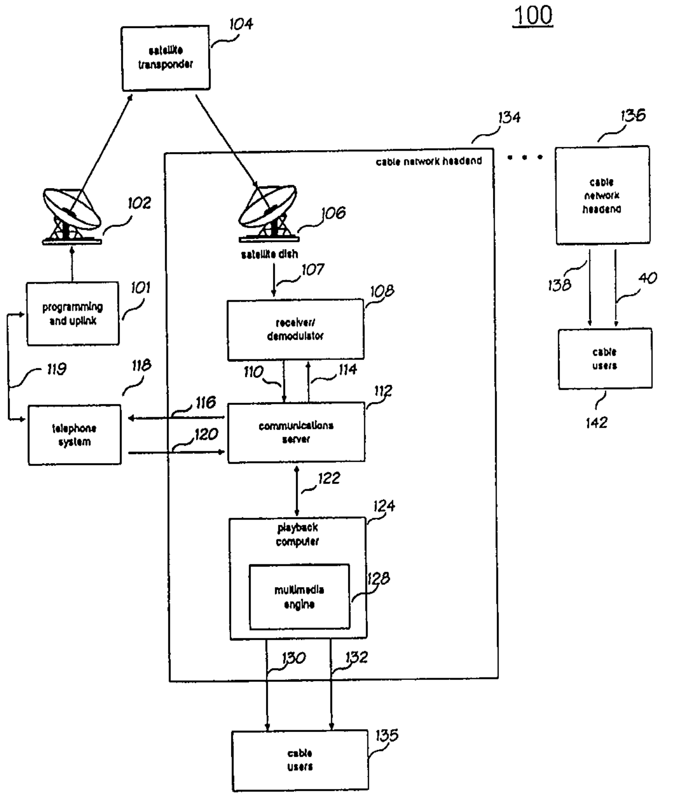
(81) Designated States: AM, AU, BB, BG, BR, BY, CA, CN, CZ, EE, FI, GE, HU, IS, JP, KG, KP, KR, KZ, LK, LR, LT, LV, MD, MG, MN, MX, NO, NZ, PL, RO, RU, SG, SI, SK, TJ, TM, TT, UA, UZ, VN, ARIPO patent (KE, MW, SD, SZ, UG), European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: REMOTE PLATFORM INDEPENDENT DYNAMIC MULTIMEDIA ENGINE

(57) Abstract

A multimedia presentation system for providing a multimedia presentation has a playback processor (124) for receiving and storing information. The received and stored information includes video information, graphics information, audio information and text information. The received and stored information further includes multimedia information. A multimedia engine (128) accesses and merges the stored information in accordance with the multimedia information to provide the multimedia presentation. The multimedia information includes scripting information (302) and segments (304) which may include stories (306) or advertisements (308). The stories include story events having event starting times and event durations. The stories also include video files (316) and audio files (314). Additionally, the stories include templates (310) which are used for rendering images by positioning them.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

REMOTE PLATFORM INDEPENDENT DYNAMIC MULTIMEDIA ENGINE

The present invention relates to the preparation and presentation of multimedia presentations and, in particular, to a multimedia engine for the preparation and presentation of such multimedia presentations.

The present invention comprises a system and process for constructing a continuous multimedia presentation from source data. While there are a number of systems known in the art for creating multimedia presentations and for replaying them with similar hardware, the multimedia engine of the present invention operates on a 24-hour basis assembling multimedia presentations based upon supplied data. This raw data is in the form of component data files, including image files, video files, audio files and text files. The multimedia engine dynamically constructs the presentation based upon this data as instructed by a multimedia scripting language. This language instructs the multimedia engine where, when, and how to assemble multimedia presentation. The language further describes when and how to stitch together multiple presentations. The intent of this invention is to create a low cost means of presenting multimedia information in a continuous form. The novel multimedia engine may be used in cable television systems where the cost of creating programming content for the viewer is of great concern. Using this system and method allows for the creation of programming content dynamically and inexpensively.

Another feature of the present invention allows the multimedia presentation to be organized based on scripting files that are transmitted to the multimedia engine, for instance via a satellite feed, or based on function calls that are either inputted at the site of the multimedia engine or are inputted remotely, typically using a telephone line and a modem.

The versatility of the present invention allows for a single multimedia engine design to operate, for numerous cable television systems, (1) products that promote pay-per-view broadcasting, (2) products that operate a news outlet that can be facilely tailored to highlight local content news, (3) television programming guides and (4) advertising programming. These various programming formats can be controlled from a single remote site.

Summary of the Invention

A multimedia presentation system for providing a multimedia presentation has a playback processor for receiving and storing information. The received stored information includes video information, graphics information, audio information and text information. The received stored information further includes multimedia information. A multimedia engine accesses and merges the stored information in accordance with the multimedia information to provide the multimedia presentation.

The invention is directed to a multimedia presentation system capable of dynamically constructing a multimedia presentation comprising: a receiving and storing means for receiving and storing commands and data files, which commands comprise script files or function calls; and a multimedia engine comprising a renderer and an application programming interface or a script manager, which multimedia engine comprises: a translating means for translating human-readable script files or human-readable function calls into commands usable by the renderer; and an executing means for executing the translated commands to create a multimedia presentation comprising audio and video components for broadcast, cable transmission or display. The presentation system of the invention advantageously provides a multimedia engine that has both an application programming interface and a script manager. The presentation system of the invention preferably stores and processes, segment files, story files, template files, advertisement files, video files and audio files as described hereinbelow.

In another embodiment, the invention is directed to a multimedia presentation system capable of dynamically constructing a multimedia presentation for cable transmission to a plurality of cable television users comprising: a receiving and storing means for receiving and storing commands and data files, which commands comprise script files or function calls; at least one segment file, at least one story file and at least one template file stored in the receiving and storing means, wherein said at least one segment file encodes the name of said at least one story file, and said at least one story file encodes text and the name of said at least one template file; and a multimedia engine comprising a renderer, an application programming interface and a script manager, which multimedia engine comprises: a translating means of translating human-readable script files or human-readable function calls into commands usable by the renderer; and an executing means of executing the translated commands to create a multimedia presentation comprising audio and video components for cable transmission.

Definitions

The following terms shall have, for the purposes of this application, the meaning set forth below. In particular, for the purpose of interpreting the claims, the term definitions shall control over any assertion of a contrary meaning based on other text found herein:

Advertisement file	- lists the advertisements that are scheduled for a defined period of time, typically a day.
Application programming interface (API)	- the code module in the multimedia engine that translates the human-readable function calls into instructions that are capable of being executed by the multimedia engine.

Dynamically constructing a multimedia presentation	constructing the multimedia presentation as the script and data files are being input from the receiving and storing means, for instance, within about 10 seconds of the input of the script and data files. In a preferred embodiment, the script and data files already within the receiving and storing means can be substituted before the scheduled broadcast, cable transmission or display time of a multimedia presentation, so long as the substitution occurs before the time and date that the segment, as defined by the segment file, in which the script and data files are to be presented is scheduled for broadcast, cable transmission or display.
Function call, call or command	- an instruction, with zero or more arguments, to perform some task.
Hunt mode	- one of the multimedia engine's modes of operation, in which the engine autonomously searches for, then parses, a sequence of segment files to produce a multimedia presentation.
MPEG	- a compressed digital audio/video bitstream format developed by the Motion Pictures Expert Group, a part of the International Standards Organization (ISO).

Message file	- lists all of the messages, such as public interest information, together with their start times and durations, which are scheduled for broadcast during a given period of time, such as a day.
Metafile	- a set of translated commands which are to be executed by the renderer.
Multimedia	- pertaining to or employing one or more media, including (but not limited to): full-motion video, audio, computer graphics, text and still images.
Multimedia engine	- a computer programmed to read in command information (which can, for example, be in the form of scripting files or function calls) and various data files and to output a multimedia presentation, for example, for display or broadcast.
Page	- consists of a collection of audio and visual elements which begin to be presented in a multimedia presentation at the same time.
Parsing	- taking a stream of information and breaking it out into the commands and arguments which a program can understand.
Rasterizer	- a program or subroutine which takes an abstract command, e.g., "put a line from x to y" and converts it into the actual pixel information that will appear on a screen, for instance in a multimedia presentation.

Renderer	- is made up of a graphics renderer, graphics rasterizer, font rasterizer, video decoder interface, graphics hardware interface, digital video subsystem and graphics generator/video overlay card, or the functional equivalents of these elements. Preferably, the renderer further includes one or both of an audio decoder interface and an audio card, or the functional equivalents of these elements.
Scripting file	- a file containing commands which define the format, appearance, timing and content of a multimedia presentation that will be outputted by the multimedia engine. The scripting files also contain references to any other files that may be needed for presentations, such as audio or video files.
Script manager	- the code module in the multimedia engine that translates script files into commands, which can be organized into metafiles, which commands or metafiles are sent to the renderer - the script manager performs the following functions: parsing; hunting; control; and timing.
Segment file	- a list of the story and advertisement files that defines the content for a given period of time in a multimedia presentation.
Story file	- a file which defines a sequence of multimedia events which make up a portion of a multimedia presentation.

Tags	- an identifier for an element in a template of a story file which defines what kind of content will be presented in a page: text, audio file, video file, image file or graphical element such as line, box, and the like. Tags mark the start of a line in a script file and are indicated by an opening "<" bracket and a closing ">" bracket.
Template file	- a file which defines the size, position, style, color, layout, and general appearance of the visual elements which appear on a page in a multimedia presentation. In a preferred embodiment, each of the elements in a template file has a tag, which matches a tag for that element in the story file.

Brief Description of the Drawings

The foregoing summary, as well as the following detailed description of preferred embodiments of the invention, will be better understood when read in conjunction with the appended drawings. For the purpose of illustrating the invention, there is shown in the drawings embodiments which are presently preferred. It should be understood, however, that the invention is not limited to the precise arrangements and instrumentalities shown. In the drawings:

Fig. 1 is a block diagram representation of a data delivery uplink and downlink system having a cable television distribution headend downlink including the multimedia engine of the present invention;

Fig. 2 is a more detailed block diagram representation of the cable system headend of the data delivery uplink and downlink system of Fig. 1;

Fig. 3 is a block diagram representation of the hierarchy of the textual script files which are applied to the playback computer memory of the multimedia engine of Fig. 1;

Fig. 4 is a block diagram representation of the relationship of types of multimedia scripting language textual files which are applied to the playback computer memory of the multimedia engine of Fig. 1;

Fig. 5 is a block diagram representation of a parsing process for the parsing of textual script files which are applied to the playback computer memory of the multimedia engine of Fig. 1;

Fig. 6 is a block diagram representation of a textual script file line parsing process for the parsing of the individual lines of textual script files which are applied to the playback computer memory of the multimedia engine of Fig. 1;

Fig. 7 is a block diagram representation of a segment event generation process for generating the segment events of a segment file which is applied to the playback computer memory of the multimedia engine of Fig. 1;

Fig. 8 is a block diagram representation of a story event generation process for generating the story events of a story file which is applied to the playback computer memory of the multimedia engine of Fig. 1;

Fig. 9 is a block diagram representation of the determination and opening of an advertising list file and the retrieving of an ad record from an advertising list which is applied to the playback computer memory of the multimedia engine of Fig. 1;

Fig. 10 is a block diagram representation of an advertising event generation process for generating the advertising events within an advertising record which is applied to the playback computer memory of the multimedia engine of Fig. 1;

Fig. 11 is a block diagram representation of a process control overview of the multimedia engine of Fig. 1; and

Fig. 12 is a block diagram representation of the queue events generation for a page of a multimedia presentation.

Detailed Description

Referring to the drawings, wherein the same reference numerals are used to designate the same elements throughout, there is shown in Fig. 1 a data delivery uplink and downlink system 100 including a cable television downlink site 134 (referred to as the headend). The cable television distribution headend 134 receives and processes multimedia information (including full motion video files, image files, audio files and script files) which is transmitted from a remote uplink site 101 within the system 100. The transmission of the multimedia information from the remote uplink site 101 to the cable television distribution headend 134 is performed by way of an uplink dish 102 at the remote programming uplink site 101 and a satellite transponder 104. Broadcast multimedia scripting data included within the multimedia information for use in multimedia presentations may be transmitted in this manner to a plurality of cable television distribution headends 134, 136.

Each of the cable television headends 134, 136 which receives the broadcast multimedia scripting data includes a receiver/demodulator unit 108, a communications server computer 112 and at least one playback computer 124 in order to provide multimedia presentations according to a predetermined broadcast multimedia scripting language for the users of a cable television system such as the cable television users 135, 142. The broadcast multimedia scripting data from the remote programming uplink site 101 is received by a satellite receive dish 106 at the cable television headend 134 and passed to the receiver/demodulation unit 108.

It will be understood that the broadcast multimedia data may be provided in accordance with any one of a number of multimedia scripting languages understandable to those skilled in the art which are suitable for indicating how to merge and present received information to provide a multimedia presentation. Preferably, the scripting language allows for story, segment, advertisement and template files to be organized as described in the Appendix and hereinbelow. Since the broadcast multimedia scripting data is transmitted by way of the satellite transponder 104 the differing cable television network headends 134, 136 may receive and present the same or different broadcast multimedia scripting data simultaneously to their cable television network users 135, 142.

The broadcast multimedia scripting data is received by the cable television headend 134 from the satellite receive dish 106 in an analog form in the preferred embodiment of the cable television network headend 134. It is applied by way of a coaxial transmission cable 107 to the receiver/demodulator unit 108. The receiver/demodulator unit 108 converts the analog signal data received by way of the transmission cable 107 into a digital data stream representing the broadcast multimedia scripting data from the remote uplink site 101. The digital data stream formed by the receiver/demodulator unit 108 is applied to a transmission cable 110. The digital data stream of the transmission cable 110 is applied to the communications server (computer) 112. The communications server 112 receives the digital data stream and stores it within the communications server 112. The data stored in this manner is distributed later within the cable television headend 134.

The communications server 112 also communicates with a conventional telephone system 118 through a telephone modem line 116 which is coupled to conventional telephone system lines 119. The telephone modem line 116 allows the communications server 112 to send diagnostic information to the remote programming uplink site 101 by way of the telephone system lines 119. In addition, the remote uplink site 101 may dial into the communications server 112 of the cable television network headend 134 by way of the telephone system lines 119 and a telephone modem line 120. This allows the remote uplink site 101 to perform remote diagnostics of the cable television distribution headend 134.

When new broadcast multimedia scripting data is stored within the communications server 112 it is moved across a bidirectional peer-to-peer network cable 122 within the cable television headend 134 to the playback computer 124. As described in more detail hereinbelow, the playback computer 124 is provided with its own playback computer disk for storage of the data received in this manner. The playback computer 124 places the new data into predetermined subdirectories on the disk. This permits the data within the playback computer data storage to be accessed and merged as required by the multimedia engine 128 according to the received scripting information in order to assemble and provide multimedia presentations for the cable television network users 135.

When the multimedia engine 128 provides such a multimedia presentation, the playback computer 124 outputs analog audio signals to the cable television network users 135 by way of the audio presentation line 130. Additionally, the playback computer 124 outputs video signals to the cable users 135 of the cable television headend 134 by way of the video presentation line 132. The cable users 142 of the cable television headend 136 within the data delivery uplink and downlink system 100 may receive a multimedia presentation in a similar manner by way of the presentation lines 138, 140.

Referring to Fig. 2, there is shown a more detailed block diagram representation of cable television headend 134. The broadcast multimedia scripting data transmitted from the remote programming uplink site 101 is received by a satellite receive dish 106 at the cable television headend 134 and passed to the receiver/demodulation unit 108 as previously described. As also previously described, the data is then transmitted by way of the transmission cable 110 to the communications server 112. The communications server 112 receives the digital data by way of a high-speed transport adapter 210 and writes it to a hard disk drive 216 within the communications server 112. The data written to the hard disk drive 216 in this manner is controlled by a disk drive controller 214 and stored for later distribution within the cable television headend 134.

Also present within the communications server 112 is a modem 212, such as a modem that transmits at 14,400 baud or better. The modem

212 is used for telecommunication of information such as status information by way of the telephone line 116.

A hardware watchdog adapter 218 is provided in the communications server 112 to detect any system lock-up errors. System lock-up errors are defined as errors in which operation of the software fails to communicate with the hardware watchdog adapter 218 in a predetermined manner at predetermined intervals. When the software fails to do this, an error is indicated and the error condition is determined by the hardware watchdog adapter 218. When this occurs, the watchdog adapter 218 resets the computer system.

A video graphics array (VGA) display adapter 220 is also provided within the communications server 112. The VGA display adapter 220 permits a user of the cable network headend 134 to couple a conventional video monitor (not shown) to the communications server 112 in order to view the operations of the system.

A network interface card 222 is provided within the communications server 112 to permit peer-to-peer communications between the communications server 112 and any other computers within the cable network headend 134. For example, the peer-to-peer network interface card 222 permits communication between the communications server 112 and the playback computer 124. When new broadcast multimedia scripting data is received and stored by the communications server 112 some of the new data is moved across the peer-to-peer network cable 122 within the cable television headend 134 to the playback computer 124. The data is transmitted through the network interface card 224 within the playback computer 124.

The playback computer 124 may be provided with a separate playback computer hard disk 242 for storage of the broadcast data received by way of the communications server 112. Storage of data on the playback computer disk 242 is controlled by a disk drive controller 240. The playback computer 124 places the new data into predetermined subdirectories on the hard disk 242. Alternately, if the data is MPEG encoded it may be stored on a digital video hard disk 238 by way of an MPEG digital video playback decoder board 234, in accordance with a packing list which may be included within the transmitted broadcast data. Storage on disks 238, 242 permits the new data within the playback computer 124 to be accessed as required by the multimedia engine 128 in order to assemble and provide multimedia presentations for the cable television network users 135.

Also in the playback computer 124 is a hardware watchdog 226. The hardware watchdog 226 of the playback computer 124 performs in substantially the same manner as previously described with respect to the hardware watchdog adapter 218 of the communications server 112.

Audio signals for presentation to the cable television users 135 may be stored on either the playback computer disk 238 or the local hard disk 242 depending on whether they are MPEG encoded. When the multimedia engine 128 provides a multimedia presentation, the playback computer 124 outputs analog audio signals to the cable television network users 135 by way of a digital audio board 228 if the digital audio data is accessed from the playback computer disk 242. If the digital audio is embedded within an MPEG digital video stream, the audio signal is first output via the MPEG digital audio playback board 236, and is then routed to the input of the digital audio board 228 in analog form. A mixer (not shown) within the digital audio board 228 combines the analog audio signals to create the final audio out signal in analog form.

Video output from the playback computer 124 may operate in substantially the same way as the audio output. When the multimedia engine 128 provides a multimedia presentation where the digital video data is accessed from the playback computer disk 242, the playback computer 124 outputs analog video signals to the cable television network users 135 by way of a video overlay board 230 (which may be an NTSC video overlay board [for video consistent with North American broadcast standards] or a PAL video overlay board [for video consistent with European broadcast standards] or another board designed to accommodate another broadcast standard such as a digital or analog high definition television standard). These files are normally stored as still frame bit maps that get displayed by the multimedia engine 128 for use as backgrounds. If the video is stored as full-motion video in MPEG format on the MPEG digital video disk drive 238 it is accessed by way of the MPEG digital video playback board 234 and applied to the video overlay board 230 in analog form. Video stored in the receiving and storing means of the multimedia engine and video output of the multimedia engine preferably has the resolution of NTSC (512 x 486 pixels) or PAL, or better.

The multimedia engine software for performing the operations of the multimedia engine 128 by the playback computer 124 is stored on the playback computer disk 242. The multimedia engine 128 operating on the playback computer 124 may be adapted to constantly run in a hunt mode wherein the multimedia engine 128 hunts or searches for the arrival of new data. The information files which make up this new data can be formatted according to the multimedia scripting language specification set forth herein in the Appendix.

There are four types of information files and multimedia scripting files which may be stored within the local storage of the playback computer 124 and processed by the multimedia engine 128. The four types are: textual script files, bit mapped graphic image files, digitized audio files and video files. Within the category of textual script files there are four types which may be read and interpreted by the multimedia engine 128. The four types of textual script files are: segment files, story files, template files and advertisement files.

The segment files contain a basic sequence of events for a presentation of a multimedia sequence by the multimedia engine 128. They reference two of the four other types of textual script files: story files and advertisement files.

SAMPLE segment file:

```
;c:\mmdata\segment\09012125.seg - 09/01/94 - 09:25PM
<stry>00:00:00|01:00||0\Ev121000.sty
<stry>00:01:00|00:15||0\H2121260.sty
<stry>00:01:15|00:30||0\Sp121261.sty
<stry>00:01:45|00:30||0\HT121263.sty
<stry>00:02:15|00:30||0\Lo121271.sty
<stry>00:02:45|01:00||0\Ev121023.sty
<stry>00:03:45|00:30||0\Sp121283.sty
<stry>00:04:15|00:30||0\HT121291.sty
<stry>00:04:45|00:15||0\Sy121293.sty
```

Note that the opening line of the segment file recites the time of day and the date when the presentation elements defined by the segment file should be broadcast. Each line in the segment file recites a relative start time (e.g., "00:00:00" for the start time of the segment, or "00:01:15" for 1 minute, 15 seconds after the start time of the segment) and a duration (e.g., "00:15" for 15 seconds and "01:00" for 1 minute).

Story files contain all of the details for executing the individual audio and video transitions and events which represent a series of pages or a single page of a presentation.

SAMPLE Story file:

```
; FileName:                C:\Windows\NTVWork\0003552.STY
; Date & Time Modified: 08-02-1994  17:24:26
<page>00:00:00|00:06|1
<tmp1>00:00:00|00:06|d.tpl
<trns>00:00:00|00:06|F0500
<back>00:00:00|00:17||bkapna.tga
<wav1>00:00:00|00:17||00035521.WAV
<hed1l>00:00:00|00:06||Altman\napologizes
```

```
        <txt1>00:00:00|00:06||WASHINGTON - The Treasury's number two
        official is apologizing to senators who've questioned his    honesty in
Whitewater.
        <box1>00:00:00|00:06|
        <img1>00:00:00|00:06||00035521.TGA
        <page>00:00:06|00:07|2
        <tmp1>00:00:06|00:07|d.tpl
        <hed1>00:00:06|00:07||Altman\napologizes
        <txt1>00:00:06|00:07|| In prepared testimony, Roger Altman says he
never meant to mislead the Senate Banking Committee. \n   He also says he
didn't try to hinder
        <box1>00:00:06|00:07|
        <img1>00:00:06|00:07||00035521.TGA
        <page>00:00:13|00:04|3
        <tmp1>00:00:13|00:04|d.tpl
        <hed1>00:00:13|00:06||Altman\napologizes
        <txt1>00:00:13|00:06||the probe of an Arkansas savings and loan with
ties to the Clintons.
      <box1>00:00:13100:06|
      <img1>00:00:13100:06|00035521.TGA
```

In the above example, the lines tagged "<page>" indicate the start of each page and indicate the time length of the page (6, 7, 4 seconds respectively for the three pages exemplified above). The lines tagged "<tmp_>" identify the template file that will be used for a time defined on the line. The line tagged "<trns>" identifies a protocol for fading into a page. The lines tagged "<back>" and "<img_>" identify image files encoding images that will be broadcast as part of the multimedia presentation for prescribed periods of time that are defined on the corresponding line. The image files identified in the "<back>" lines are for full-screen images. The lines tagged "<wav_>" identify audio files that will be played for a period of time defined on the line. The lines tagged "<hed_>" and "<txt_>" identify text that will be overlaid on the broadcast images for the periods of time prescribed on the corresponding line. The lines reference "<box_>" indicates that a box should be overlaid on the broadcast images for the period of time indicated on the line. The template file for the page will have a corresponding tagged "<box_>" line that will include a definition of the box geometry.

Template files contain the geometry for rendering a page in a multimedia presentation. Each of the tags ("<txt1>", "<img1>", "<hed1>", etc.) contained in a template file corresponds to a matching tag on the corresponding page in a story file.

SAMPLE template file

```
<Ver>2.0
<txt1>260|190|200|200|1|24|24|hvb____|0|0|0|0|0|0|0
<img1>46|102|204|284|0|0|0|0
<hed1>260|90|200|84|0|42|42|uvbli____|0|0|0|0|0|0|0
<box1>44|100|208|288|0|1|0|0|0|0|0|0|0
```

A single advertisement file lists all of the advertisements which are scheduled for an individual day of the year. The multimedia engine 128 proceeds sequentially through the advertisement file and keeps track of which advertisements have been presented.

SAMPLE advertisement file:

```
; filename: 0216.AD
; advertising play list
; 01/28/94
; creationdate: 01/28/93
; createdfor: NEWSCHANNEL
; systemid: 2442, Coatesville
00:10|17|18|19|20
00:20|21|22|24|25
```

```
00:30|17|18|19|20
00:40|21|22|24|25
00:50|17|18|19|20
.
.
.
23:50|17|18|19|20
24:00|21|22|24|25
```

In the above example, the entries following the time entry which indicates the time of day at which to play an advertisement, are one or more video files or story file names separated by pipe symbols "|".

The generalized format of textual script files stored within the local playback computer disk of the playback computer 124 within the multimedia engine 128 may be as follows:

<TAG>StartTime|Duration|Device|Other

wherein the pipe symbol "|" designates a delineation between fields, the prefix <TAG> (e.g. "<box_>", "<txt_>", etc.) indicates the object type within the textual script file, StartTime indicates the starting time of the textual script file, relative to the start time for a segment file, in the conventional hours, minutes, seconds format, HH:MM:SS, and Duration indicates the time duration or dwell in the conventional minutes, seconds, format, MM:SS, during which the object remains active. Also within the generalized textual script file format, Device refers to the physical devices which may be referenced in the textual script files. The physical devices may include a host computer, a digital video subsystem, an external tape deck and a laser disc player. The other field may include additional information required to complete the current scripting line format. For instance in the case of digital video file reference, the other entry is the name of the digital video file.

Referring to Fig. 3, there is shown a block diagram file representation 300 of the hierarchical relationship of the four types of textual script files within the playback computer 124 of the multimedia engine 128. Referring to Fig. 4, there is shown block diagram representation 400 illustrating the relationship between the various scripting files. As previously described, the textual script files are one of four types of information files which may be stored within the playback computer 124. The application program 302 creates a number of segment files such as the segment files 304 for use in providing multimedia presentations within the multimedia engine 128. The application program 302 may be any entity operated by any user of the multimedia engine 128 which is able to specify a multimedia presentation for assembly and presentation by the multimedia engine 128.

The segment files 304 shown in the block diagram representation 300 may contain any number of story files 306. A story file 306 included within a segment file 304 is a collection of story events wherein each of the story events has a relative start time and a predetermined duration. Each story file 306 referenced within the segment file 304 must have a reference to a specific time at which the story file 306 is to be presented by the multimedia engine 128 to the cable users 135.

A story file 306 is thus a logical sequence of multimedia events with a specific application within a multimedia presentation by the multimedia engine 128. The application may be, for example, a news story, a movie promotion, an event promotion, or any multimedia information presentation. The story events within the story files 306 contain the information required for the multimedia engine 128 to execute the audio and visual transitions within the presentation of a multimedia story by applying signals to the audio line 130 and the video line 132 of the cable television headend 134 as previously described.

The following is an example of a number of story files 306 which may be associated with each other to form a portion of a segment file 304 by the multimedia engine 128:

```
<stry>00:00:00|01:00||highlite.sty
```

```
<stry>00:01:00|00:15||schedule.sty
<stry>00:01:15|00:15||howtoord.sty
<stry>00:01:30|00:30||upnexttv.sty
          .    .    .
          .    .    .
          .    .    .
```

The prefix <stry> at the beginning of each line identifies the file types of the four entries in this segment file 304 as story files 306. The various fields within the story files 306, are delineated by pipe symbols "|" as previously described. The time values 00:00:00 after the <stry> prefix indicate the starting times of the story files 306 relative to the starting time for segment file 304. For example, 00:00:00 in this field identifies the first story file 306 as the one within the segment file 304 which begins at time zero. The 01:00 in the duration field, delineated by a single pipe symbol, indicates that the first story file 306 has a duration of one minute. The device fields in this example are empty. The last field indicates that the name of the first story file 306 is "highlite.sty".

The second story file 306, named "schedule.sty", has 00:01:00 in its second field, indicating that it starts one minute into the segment file 304 represented by this example. The story file 306 named "schedule.sty" lasts for fifteen seconds as indicated by 00:15 in the duration field.

Thus, in the multimedia presentation represented by the segment file 304 of this example, the story file 306 named "highlite.sty" starts at time zero which is the beginning of the presentation of the segment file 304 and runs for one minute. The story file 306 named "schedule.sty" begins at one minute and runs for fifteen seconds. The story file 306 named "howtoord.sty" begins at one minute and fifteen seconds and runs for another fifteen seconds, and the story file 306 named "upnexttv.sty" begins at one minute and thirty seconds and runs for thirty seconds.

In addition to the story files 306, the block diagram file representation 300 of textual script files stored within the playback computer

124 also sets forth advertising list files 308. The advertising files 308 are sequential lists of advertising insertion lines and other advertising identifications. Advertising files 308 are used to insert advertising from a separate device or from a story file 306 into the multimedia presentation assembled and prepared by the multimedia engine 128. The advertisement information to be presented to the cable users 135 in this manner during a multimedia presentation is accessed from the local memory within the playback computer 124 by the multimedia engine 128 as required in order to properly insert it into the presentation. The accessing of advertising information by the playback computer 124 in accordance with an advertising file 308 occurs when the multimedia engine 128 encounters a <advr> prefix within a segment file 304 as shown in the following example:

```
<stry>00:00:00|00:00||highlite.sty
<stry>00:01:00|00:15||schedule.sty
<stry>00:01:15|00:15||howtoord.sty
<stry>00:01:30|00:30||upnexttv.sty
<advr>00:02:00|00:15||
            .     .     .
            .     .     .
            .     .     .
```

In this example after the four story files 306 are processed and displayed as previously described, an advertisement file, whose file name is described by the current month and day (as detailed below), is referenced. The contents of samplead.ad are read and scanned for a time pointer. This time pointer or time reference may point to a digital video filename or to a another story file 306. By separating advertising references into independent files identified in the advertising file 308, the advertising content may be scheduled separately from the rest of the presentation. This permits multimedia content to be managed by an editorial group which is separate from those managing the advertising material. This kind of separation is common in the art of commercial print publishing. It also permits the advertisement information to be used independently of the advertisement information time slots of the multimedia presentation.

The file name format for advertising list files 308 is of the form MMDD.ad in order to indicate the month and day of playback. For example, a list of advertising list files 308 may be of the form:

0101.ad
0102.ad
0103.ad
.
.
.
1231.ad

The last filename "1231.ad" is an advertising list file 308 which is played on December 3lst of the current year.

Fig. 5 shows a block diagram representation of a script file parsing method 500 for parsing various textual script files within the multimedia engine 128 of the present invention. The script file parsing method 500 may be used to parse segment files 304, story files 306 and template files 310. Therefore within the script file parsing method 500 a determination is made at decision block 502 what type of textual script file is being parsed by the parsing method 500.

If the textual script file being parsed is a story file 306, execution of the script file parsing method 500 proceeds by way of decision node 504 of decision block 502 to the script file line parsing process 600 for parsing of the individual file lines within the story file 306. Execution of the script file line processing process 600 described hereinbelow with respect to Fig. 6.

When execution exits the script file line parsing process 600 by way of the exit pathway 614, it reenters the file parsing method 500 at block 510. In block 510, the pages of the parsed story file 306 are identified. The appropriate records within the local storage of the playback computer 124 are associated with the various identified pages as shown in block 512. In block

514 a determination is made as to which template files 310 are included within the story file 306 in order to present the story file 306. The template files 310 are then loaded opened, parsed, and converted into metafile objects as described with other text files,block 524.

If the textual script file being parsed by the script file parsing method 500 is a segment file 304, as determined in decision block 502, execution of the script file parsing method 500 proceeds from decision block 502 by way of decision node 506. The segment file 304 being parsed is then processed by the file line parsing process 600 as previously described. Execution exits the file line parsing process 600 by way of exit pathway 614 and reenters the script file parsing process 500 at block 518. The duration of the segment file 304 is then calculated in block 518. All of the story files 306 within the segment file 304 being parsed by the parsing process 600 are identified, loaded and updated as shown in blocks 520, 522.

If the textual script file being parsed by the script file parsing method 500 is a template file 310, as determined in decision block 502, execution of the script file parsing method 500 proceeds from the decision block 502 by way of the decision node 508. The template files 310 are then opened and parsed as shown in block 600. They are converted into metafile objects with other text files as shown in block 524 in a manner described hereinbelow.

Referring to Fig. 6, there is shown a block diagram representation of the script file line parsing process 600. As previously described the script file line parsing process 600 may be used for the parsing of individual lines within the various textual script files. This parsing may occur within the parser block 1106 (Fig. 11) of the presentation process overview 1100 operating within the multimedia engine 128 as further described hereinbelow. The script file line parsing process 600 may be used to separate the lines of textual script files including segment files 304, story files 306, and template files 310 for processing by the script file parsing method 500.

Execution of the script file line parsing process 600 begins with a textual file being opened as shown in block 602. A determination is made whether the last line of the textual file has been parsed by the line parsing process 600 in decision block 604. If the current line is not an end of file marker, a line of the file is read as shown in block 606. If the last line of the file has been parsed the file is closed as shown in block 610. After the parsed file is closed execution of the script file line parsing process 600 exits by way of the exit pathway 614.

If the end of the textual script file being parsed by the line parsing process 600 has not been encountered, as determined in decision block 604, the items between the pipe delineation characters in the current file line of the textual script file are separated as shown in block 608. Execution of the file line parsing process 600 then returns to decision block 604 where another line of the textual script file is examined to determine whether it is an end-of-file marker.

Fig. 7 shows a block diagram representation of the segment event generation process 700 for generating segment events to prepare multimedia presentations by the multimedia engine 128. The first step in generating the events for a segment 304 by the process 700 is preparation of an advertising record for the segment 304 as shown in block 702 if required. Only applications that take advantage of dynamic advertising insertion use this feature. In most cases block 702 is not executed. The next step is the reading of a record as shown in block 708. If decision block 712 determines that the file is not done, the record is checked to see if it is a story record in decision block 732. If not, it is an advertising record and an advertisement event for the multimedia event queue is generated 748. The process continues with the next record being read in block 708 and checked in decision block 712. If the next record read is a story record the process for generating story events 800 is executed as described hereinbelow. Upon completion of story event generation, the next line is checked to see if it is the end of the file as shown in decision block 712.

Once the end of the file is reached, as determined by block 712, a determination is made whether the system is currently in hunt mode as shown in decision block 716. If the software is in hunt mode it calculates the specified offset event times to run as shown in block 720 and exits at terminal 724. If the software is not in hunt mode the multimedia engine 128 adds a stop event to the queue in block 728 and exits the operation at terminal 724.

Fig. 8 shows the story event generation process 800 for generating events corresponding to a story file 306 by the multimedia engine 128 of the present invention. Within the story event generation process 800 a determination is made in decision block 804 whether the story file 306 being processed by the multimedia engine 128 is present within the directory of the playback computer 124. If the story file 306 is not present in the playback computer memory 124, the events necessary for a predetermined default presentation are generated as shown in block 820 and execution by the multimedia engine 128 exits the story event generation process 800 at terminal block 822.

If the story file 306 does exist in the playback computer 124, as determined in decision block 804, the story event generation process 800 begins sequentially processing the pages within the story file 306. In order to do this, a record is read in block 806 and a determination is made in decision 808 whether the end of the story file 306 has been encountered. If the end of the story file 306 has been encountered, a predetermined end story event is added to the output of the story by the event generation process 800 as shown in block 824. A determination is then made in decision block 828 whether a stop is specified within the story file 306 being processed. If a stop has not been specified execution of the story event generation process 800 ends at terminal block 822. If a stop is specified, as determined in decision block 828, a stop event is scheduled as shown in block 832 and execution exits the story event generation process 800 by way of terminal block 822.

If the end of the story file 306 being processed by the story event generation process 800 is not encountered, as determined in decision block

808, a determination is made whether the page of the file being processed should be displayed. This page display determination is made in decision block 812. (For instance, the multimedia engine may decide to skip a page if needed to stay in synch with timing indicated in the script file.) If the page should not be displayed, execution proceeds to decision 808 where another determination is made whether the end of the story file 306 has been encountered. If the page should be displayed, as determined in decision block 812, execution proceeds by way of path 813 to the block 1200 (hereinafter described) where the queue events for the page are generated.

Fig. 9 shows the segment event generation process 900 for determining whether a segment file 304 being processed by the multimedia engine 128 contains an advertising list file 308 and for processing an advertising list file 308 when it is present. Within the segment event generation process 900 a determination is made in decision block 902 whether the segment file 304 contains an advertising list file 308. If the segment 304 does not contain an advertising list file 308, execution exits the segment processing process 900 by way of terminal block 904. If the segment file 304 does contain an advertising list file 308, a determination of the advertising file name is made, as shown in block 908.

A determination is then made in decision block 912 whether the advertising list file 308 indicated can be opened. If no advertising list file 308 can be opened an advertisement file directory within the local memory of the playback computer 124 is scanned by the event generation process 900 to determine the most recently used advertising list file 308, as shown in block 916. If the advertising list file 308 found in block 916 is not open, as determined in decision block 920, execution exits the segment processing process 900 by way of terminal block 904.

If an advertising list file 308 is open, as determined by either decision block 912 or decision block 920, the record for the current HH:MM is obtained in block 924. The record obtained in block 924 is returned by the segment processing process 900, as shown in the block 928 and the process 900 is exited by way of terminal block 904.

Fig. 10 shows the advertising event generation process 1000 for generating advertising events within the multimedia engine 128 of the present invention. Execution of the advertising event generation process 1000 begins with a determination whether the advertising record to be processed is present in memory. This determination uses the advertising record process 900 described hereinabove. This determination is indicated in decision block 1004. If the event is not present a default static screen event is scheduled as shown in block 1008 and execution exits the advertising event generation process 1000 by way of terminal block 1010.

If the advertising record to be processed by the event generation process 1000 does exist within memory, as determined in decision block 1004, a determination is made in block 1012 which advertising event should be played. This determination is made in a round robin method. The advertising record that is returned via record generation 900 is a series of file pointers that point to one or more digital video files or story files in memory. One advertisement is selected from the record returned. The first time the multimedia engine 128 accesses an advertisement in this method a round-robin counter is set. Upon subsequent calls to this record the counter is incremented and the next advertisement is played. When the end of the series is encountered the round-robin counter is reset to one and the first advertisement is replayed.

A determination is then made at decision block 1016 whether the advertising event being processed is a story file 306. If the event is a story file 306, the story events for the story file 306 are generated as shown in block 1020. After the story events are generated in block 1020 the advertising event generation process 1000 is complete and execution exits by way of terminal block 1010.

If the advertising event being processed by the event generation process 1000 is not a story file 306, as determined in decision block 1016, video file events are scheduled by the multimedia engine 128 as shown in block 1024. After scheduling of the video file events in block 1024, execution exits the advertising event generation process 1000 by way of terminal block 1010.

Referring to Fig. 11, there is shown a process overview 1100 describing the operations of the multimedia engine 128 of the present invention. Within the process overview 1100 an application programming interface 1102 provides an interface for applications to allow them to control the multimedia engine 128. The application programming interface 1102 applies commands scripting information from the external applications (not shown) to the various primary control subsystems of the multimedia engine 128. Commands can be applied, for example, to the parsing subsystem 1106, the hunting subsystem 1108, the control subsystem 1114 and directly to the graphics rendering subsystem 1118. The application programming interface 1102 permits a consistent view of the multimedia engine 128 for the external applications (i.e., a computer program whose purpose is to control the multimedia engine).

Through the application programming interface 1102 external applications can issue commands to the parsing subsystem 1106 to parse input files and schedule multimedia events based upon the files. The parsing subsystem 1106 reads the passive operation data files 1104 and generates the metafiles 1110 which represent the non-timing related items in the passive operation data files (PODF) or script files 1104. A metafile 1110 is a set of records which represent a series of calls by the application programming interface 1102 which are to be made to the graphics rendering subsystem 1118. The records of the metafile 1110 are played sequentially when all of the corresponding calls of the application programming interface 1102 are made. The metafiles 1110 are then scheduled into the timing subsystem 1116.

Also through the application programming interface 1102 external applications may request that the hunting subsystem 1108 identify a set of passive operation data files or PODF's 1104 which should be run at a particular time. The hunting subsystem 1108 checks the system time and identifies the PODF's 1104 which have time encoded filenames corresponding to the current system time. When the set of data files 1104 have been identified the parsing subsystem 1106 schedules the set of files for presentation by the multimedia engine 128.

External applications may also use the application programming interface 1102 to directly access the graphics rendering subsystem 1118 as previously described. This permits an external application to directly control the output of a presentation. This direct control allows an alternate means of controlling the multimedia presentation rather than the scripted format provided by the parsing subsystem 1106 and hunting subsystem 1108 (generally, the components of the script manager 1150) of the process overview 1100.

External applications using the multimedia engine 128 by way of the application programming interface 1102 may control the timing of the multimedia engine 128. This is done by an external application (not shown) having a timing loop for calling a polling function in the application programming interface 1102. This polling function in the application programming interface 1102 distributes the polling into the control subsystem 1114 which controls the timing subsystem 1116 and the hunting subsystem 1108. When a poll in the control subsystem 1114 occurs, the poll is transferred to the timing subsystem 1116. If the timing subsystem 1116 identifies an event as requiring action, it returns that event to the control subsystem 1114. A metafile 1110 is then retrieved from the event and it is passed into the graphics rendering subsystem 1118 for processing. If the timing subsystem 1116 is then empty, either the hunting subsystem 1108 is called to schedule another set of files through hunting or playback is stopped. An external application can also directly stop the presentation by way of the application programming interface 1102.

Among the components within the renderer 1151, is the graphics rendering subsystem 1118, which operates in a plurality of modes. The graphics rendering subsystem 1118 supports a set of functions which directly control multimedia presentation hardware within the multimedia engine 128. These functions can be called by external applications by way of the application programming interface 1102, or by a metafile 1110.

The graphics rendering subsystem 1118 provides a number of features within the multimedia engine 128. It may provide a hardware independent application program interface which external applications can use. This is possible because all of the functions in the graphics rendering subsystem 1118 are propagated to a similar function in the hardware abstraction layer 1128. It provides high level control of the font rasterizer 1122, the graphics rasterizer 1120, and the digital video decoder 234 to provide a logical multimedia presentation. This logical presentation is coordinated by preventing the user from requesting an illogical sequence of actions, by providing control of transitions from complete graphical presentations to video presentations, and by handling exceptional cases such as video playback failure.

The graphics rasterizer 1120 of the process overview 1100 is called by the graphics rendering subsystem 1118 to provide raster display operations. This includes drawing lines and polygons as well as displaying graphics files 1124. The graphics rasterizer 1120 loads the graphics files 1124 if required and calls the graphics hardware interface 1134 section of the hardware abstraction layer 1128 to draw an item on the graphics hardware (graphics generator/video overlay card) 230.

The font rasterizer 1122 is called by the graphics rendering subsystem 1118 to provide anti-aliased font-based raster display operations. The font rasterizer 1122 is provided with a font specification, text content, and position and provides a font from a font file 1126. It also performs anti-aliasing and calls the graphics hardware interface 1134 section of the hardware abstraction layer 1128 to display the desired text in the appropriate font. The font rasterizer 1122 also permits caching of font rasterization to provide greater performance.

The graphics renderer 1118 directly calls functions in the hardware abstraction layer 1128 for controlling the digital video decoder 234 and the digital video file storage of disk 238. The digital video decoder 234 includes its own application program interface which is abstracted by the video decoder interface 1132 of the hardware abstraction layer 1128. Through the video decoder interface 1132, the graphics rendering subsystem 1118 can request that the video decoder 234 play a particular video file. The video decoder interface 1132 first verifies the existence of the digital video file on the disk 238, and if the file exists, starts the video decoder 234 playback of that file.

The video decoder 234 may send asynchronous messages to the multimedia engine 128. These messages are directed to the external application, whose responsibility it is to return them to the playback computer 124 by way of the application programming interface 1102. These messages are asynchronous and are relayed by the application programming interface 1102 to the control module 1114 and then to the graphics rendering subsystem 1118. The control module 1114 may monitor these messages in the case of serious errors on the video decoder 234 which require the halting or modification of the presentation in the timing subsystem 1116. When the graphics rendering subsystem 1118 receives an asynchronous message from the video decoder 234 it applies the message to the video decoder interface 1132 which determines how to handle the message. Based upon the decision made by the video decoder interface 1132, the graphics rendering subsystem 1118 may make modifications in the presentation.

Referring to Fig. 12, there is shown a block diagram representation 1200 of a method for generating queue events for a page of a story file 306 within the multimedia engine 128 of the present invention. Entry of the method of block diagram representation 1200 is by way of decision block 812 and path 813 of the story event process 800. As previously described, decisions block 812 determines whether a page of a story is to be displayed. If the page is to be displayed its queue events must be generated. Thus the story page is applied to block 1202 which determines the time of the page in the story, and possibly the position of the story within a segment. Process block 1204 then creates an empty metafile 1110. Records are added to the metafile 1100 to correspond to the items on the story page.

A page is read in block 1206. Decision block 1208 determines whether the file is finished. If the file is finished block 1226 creates an event in the event queue for the time identified by block 1202 and process 1228 places the metafile 1100 inside the event created by process 1226. The procedure of diagram 1200 is then terminated.

If a record exists in the story page, execution of process 1200 proceeds to decision block 1210. Decision block 1210 determines if the story page record represents a change of a template 310. If it is, block 1216 attempts to select the specified template. Process control is then returned to decision block 1208 from block 1216. If the story page record is not a change of template, execution proceeds to decision block 1212. Decision block 1212 determines if the story page record requires a template 310. If it does, control passes to block 1218, otherwise it passes to block 1214. Block 1218 looks up the corresponding template record specified by the story page record. If the record does not exist, the record is ignored and execution proceeds from block 1220 to block 1224 and block 1208. If the record does exist control process 1222 merges the story page and template information into a single record to be placed within the metafile 1100. Block 1214 takes the output of block 1212 and/or block 1222 and creates a record with the metafile created by block 1204 which represents the record identified of created by block 1212 or block 1222. Control then passes back to decision 1208.

It will be appreciated by those skilled in the art that changes could be made to the embodiments described above without departing from the broad inventive concept thereof. It is understood, therefore, that this invention is not limited to the particular embodiments disclosed, but it is intended to cover modifications within the spirit and scope of the present invention as defined by the appended claims.

APPENDIX

Starnet Multimedia Scripting Language (SMSL)

Language Specification

Version 1.5

Purpose of this document

The purpose of this document is to define the StarNet Multimedia Scripting Language (SMSL). This definition includes what the StarNet Multimedia Engine (SMME) does. how it does it. the input data necessary for it's operation and the output it produces as a result of that ongoing operation.

What is the SMME?

The SMME is a software command interpreter which runs in a product specific computer at a cable head end and produces output consisting of still pictures. full motion video. rendered text and graphics and sound which is in turn fed to the cable system for broadcasting.

What is the SMSL?

The SMSL is a the multimedia scripting language invented by Gerard Kunkel of StarNet. and refined by Michael Heydt of StarNet. The language addresses all of the capabilities of the SMME in a simple collection of file formats all stored as ASCII text files.

What is still to come in the SMSL?

The SMSL was orginally conceived as an all purpose multimedia scripting language for use in passive or interactive television. As such, the language will be expanded during the fall of 1994 to address the new interactive needs or television set-top boxes (STBs). Of note will be client-server nature of data access to feed the engine, expanded graphics capabilities of the new dedicated graphics chips to do graphic sprites. Other operator needs will be identified as the API for the set-top box is disclosed. In the meantime, the general Windows 3.1 API will be used as a model for additional operators.

One very important change coming to the SMSL is the support of VGA for graphics output. Currently the engine talks directly to graphics hardware for its graphics display. Since the STBs appear to be supporting VGA as a standard for graphics overlay, the engine will be modified to use all of the Windows 3.1 API calls for graphics rendering and control.

How Does the SMME Work?

The SMME takes as input the SMSL files which consist of presentation sequencing information. and data files in the form of graphic images, digital video sequences and digitized sound. Some of the control information that the SMME receives directs it to access several different types of playback devices.in order to produce the necessary output. These playback devices include. and are currently limited to: digital video subsystems (DVSS from Scientific Atlanta), digital video adapters (Optibase PC Motion), video tape recorders (VTRs) and laser disk players.

It is not within the scope of this document to describe the method by which information is received by the SMME. We will assume for the sake of simplicity that any necessary information is available for the engine's use before it is needed. For a detailed description of the communications methods and interactions between StarNet equipment in a cable head end, see the *StarNet Communications Server Specification* and *StarNet Multimedia Hardware Specification.*

Engine operating modes

SMMEFRNT.EXE is software application that interfaces between data files and the engine itself. SMMEFRNT is a software front end that allows a user to define one of two operating for multimedia playback. There are other uses for the SMMEFRNT application that are not relevant here. For more detail see the document *StarNet Multimedia Engine Front End Application.*

Normal mode

If a SEGMENT file plays, ends and is not replaced by a newer segment file, the current segment file will stop playing and the program will stop.

Hunt mode

If a SEGMENT file plays, ends and is not replaced by a newer segment file, the current segment file will continue fooping until a newer SEGMENT file is found.

SMME Input Files and Sources

The SMME takes as input 4 fundamentally different types of information: textual script files, bit mapped graphic images, digitized sound, and digital or analog full-motion video. This information comes from four different storage devices:hard disk, digital video subsystem (which may actually be a hard disk attached to another device), video tape and laser disk. This document is responsible for describing the contents of only the textual script files.

Textual Script Files

There are 4 types of script files that the SMME reads and interprets: Segment files, Story files, Control files and Advertisement files. The Segment file (.SEG) contains the basic sequence of events for the presentation of a multimedia sequence. It references two other types of files:Stories (.STY's) and the Advertisements file (.AD). The CONTROL files (.CTL) are used only in the News Channel product and have no bearing on the operation of the SMME.

Story Files (.STY)

The term Story comes from the News Channel project (see other documentation) and is defined as a logical sequence of multimedia events with a specific application. The application may be a news story, a movie promotion, an event promotion, an advertisement, etc. The Story file contains all of the details for executing the individual audio and visual transitions and events which represent a Story.

Segment Files (.SEG)

The SEGMENT file is a collection of Stories and Advertisements.

Advertisement Files (.AD)

There is only one Advertisements file which lists all of the Advertisements which are scheduled for a certain amount of time. The SMME keeps track of which ads have been played and simply runs sequentially through the file.

Template Files (.TPL)

The template file contains the geometry for rendering a page in the multimedia presentation. Each of the tags contained in the TPL file corresponds to a matching tag in the STORY file.

DESCRIPTION TO COME:

```
; story template
<hed1>40,78|432,61|0|55|55|uvcsc___|0|0|0,0,0 [illegible]
<img1>40,160|432,126|0
<txt1>40,290 432,130|0|22|22|hv_____|0|0|0,0,0
<vid1>0,0|512,486|0
```

File Formats

Each of the SMME textual control files is written using the language specification called the StarNet Multimedia Scripting Language (SMSL). This language borrows the construct of a number of existing languages: C, SGML, and the Windows INI file format.

The following line shows the general format of a line within either a Story or Segment file:

<TAG>StartTime|Duration|Device|Other (Filename etc.)|..

Where:

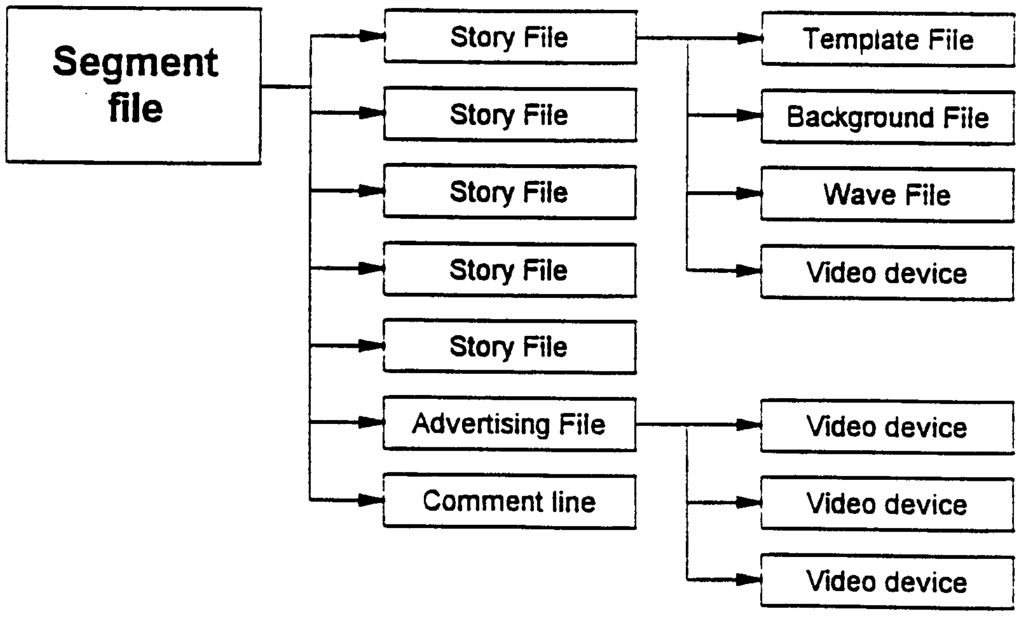
<TAG>	one of the tags described in the SMEPDL section of this document.
StartTime	HH:MM:SS
Duration	MM:SS
Device	described in a later section, defaults to COMP
Other	tag specific parameters

General standards for initialization and data files include the following:

- a semicolon (;) at the beginning of a line signals a comment
- all time magnitudes will be expressed in milliseconds where granularity of less than one second may be necessary and in MM:SS form where larger spans of time are needed.

The Segment File

The SEGMENT file is a list of events that the multimedia engine will play. The SEGMENT file points to STORY files that contain the actual data for display . The SEGMENT file is a higher level that simply sequences STORY files. This relationship is illustrated in the following diagram.



File naming convention

The name of the SEGMENT file is very important to the correct operation of the SMME. The DOS eight character filename is used to represent the exact minute of the day/month/year that the SEGMENT is to begin operation. Below is an example of the SEGMENT file naming convention:

Month	Day	Hour	Minute	Extension
1-12	1-31	00-23	00-59	.SEG

or

01080155.SEG

Below is a sample NewsChannel SEGMENT file.

```
<stry>00:00:00|00:10||0000019.STY
<stry>00:00:40|00:15||000002A.STY
<stry>00:00:55|00:19||0000002.STY
<stry>00:01:14|00:12||0000006.STY
<stry>00:01:26|00:15||000000D.STY
<stry>00:01:42|00:15||000000A.STY
<stry>00:01:57|00:19||0000002.STY
<stry>00:02:16|00:12||0000006.STY
<stry>00:02:29|00:11||000000D.STY
<stry>00:03:10|00:12||0000009.STY
<stry>00:03:22|00:11||0000005.STY
<stry>00:03:33|00:15||000000C.STY
<stry>00:03:48|00:24||0000018.STY
<stry>00:04:13|00:12||0000008.STY
<stry>00:04:25|00:11||0000005.STY
<stry>00:04:37|00:15||000000C.STY
<stry>00:04:52|00:17||0000019.STY
<stry>00:05:40|00:12||0000030.STY
<stry>00:05:52|00:12||0000030.STY
<stry>00:06:04|00:12||0000030.STY
<stry>00:06:16|00:12||0000030.STY
<stry>00:06:28|00:12||0000030.STY
<stry>00:06:40|00:12||0000030.STY
<stry>00:06:52|00:12||0000030.STY
<stry>00:07:04|00:12||0000030.STY
<stry>00:07:16|00:12||0000030.STY
<stry>00:07:29|00:11||0000030.STY
<advr>00:07:40|00:30|ad.ini
<stry>00:08:10|00:21||000002A.STY
<stry>00:08:31|00:21||000002A.STY
<stry>00:08:53|00:21||000002A.STY
<stry>00:09:15|00:21||000002A.STY
<stry>00:09:37|00:21||000002A.STY
<stry>00:09:58|00:01||000002A.STY
```

Below is a sample Barker SEGMENT file representing 5 minutes of presentation

```
<stry>00:00:00|01:00||highlite.sty
<stry>00:01:00|00:15||schedule.sty
<stry>00:01:15|00:15||howtoord.sty
<stry>00:01:30|00:30||upnextvi.sty
<stry>00:02:00|00:15||schedule.sty
<stry>00:02:15|00:15||howtoord.sty
<stry>00:02:30|01:00||highlite.sty
<stry>00:03:30|00:15||schedule.sty
<stry>00:03:45|00:15||howtoord.sty
<stry>00:04:00|00:30||upnextvi.sty
<stry>00:04:30|00:15||schedule.sty
<stry>00:04:45|00:15||howtoord.sty
```

The Story File

The STORY file is a collection of events and pointers. The events listed in the STORY file occupy one line per event. Each event reference must have a relative starting time and a relative duration (dwell). The starting time is relative to the start of the STORY sequence. The duration is relative to the start of the individual event. Each event is labeled with a tag denoted in greater and less-than symbols (<>).

A NewsChannel Story File:

```
; D:\ntv\edit\0000011.STY
; 03-22-1993  10:14:30
<page>00:00:00|00:00||1
<wav1>00:00:00|00:26||00000111.wav
<tmp1>00:00:00|00:00||vns2tic2.TPL
<hed1>00:00:00|00:26||Trade allies: "U.S. two-faced"
<sub1>00:00:00|00:26||Administration denies European charge
<txt1>00:00:00|00:10||WASH.D.C.--American trade partners, bracing for the Clinton admini-
stration's first set of trade sanctions, aré angry and confused.\n
   Officials within the European community's Executive Commission
<img1>00:00:00|00:26||00000111.tga
<txt1>00:00:10|00:10||feel the President has indicated he is committed to free trade but is also
receptive to protectionism.\n
   Administration officials responded there is no inconsist-
<txt1>00:00:20|00:06||ency in the emerging U.S. policy and that trade diplomacy can no longer be
neatly labeled.
```

A Barker Story File:

```
;schedule screen
;
;
<page>1
<tmp1>schedule.tpl
<vid1>00:00|00:15||schedule.vid
<txt1>00:00|00:07||10AM
<txt2>00:00|00:07||Ch 45
<txt3>00:00|00:07||A River Runs Through It \s030PG\n\s0401-800-CINEMA-1 $3.95
<txt4>00:00|00:07||10AM
<txt5>00:00|00:07||Ch 46
<txt6>00:00|00:07||Bram Stoker's Dracula \s030R\n\s0401-800-CINEMA-2 $4.95
<txt1>00:07|00:08||10AM
<txt2>00:07|00:08||Ch 47
<txt3>00:07|00:08||Toy's \s030PG\n\s0401-800-CINEMA-1 $3.95
<txt4>00:07|00:08||12PM
<txt5>00:07|00:08||Ch 45
<txt6>00:07|00:08||A River Runs Through It \s030R\n\s0401-800-CINEMA-1 $4.95
```

Language description by tag

PRESENTATION operators

menu

SYNTAX:

<menu*n*_*n*>title|director|description

DESCRIPTION: Signifies the creation of a new menu of choices. The subsequent menu items are listed in series following the initial menu item. The engine assumes base 0 for both the menu and item numbering scheme.

SAMPLE:

```
<menu0_0>Classified advertising|menu1|The latest in articles for sale, real
estate and personals
; the world's first interactive presentation file
; for the StarNet Multimedia Engine
;
; created by Gerard Kunkel at 11:56PM on 8/3/94
;
<page>1
<back>present.tga
<wav1>present.wav
<img1>logo.tga
<temp>present.tpl
<txt1>SME
<txt2>Multimedia for cable, telephony, computers, and settop boxes
<txt3>A proposal for a new way of doing business
<menu0_1>Multimedia Engine|menu2
<menu0_2>The Barker Service|barker.sty
<menu0_3>NewsChannel|menu3
<menu0_4>TV Guide Channel|video.sty
<menu0_5>Video Juke Box|vjn.sty
<menu0_6>SN Classifieds|menu1
<menu0_7>EXIT|EXIT
<menu1_1>Autos|auto.seg
<menu1_2>Real Estate|real.seg
<menu1_3>Services|service.sty
<menu1_4>Items for sale|items.sty
<menu1_5>Personals|persl.sty
<menu1_6>previous menu|menu0
<menu2_1>Multimedia's charter|charter.sty
<menu2_2>Product descriptions|products.sty
<menu2_3>Marketing approach|market.sty
<menu2_4>Business model|biz.sty
<menu2_5>Timelines|menu4
<menu2_6>previous menu|menu0
<menu3_1>Man cleared in killing|000355a.sty
<menu3_2>Jersey doctor keeps license|000355b.sty
<menu3_3>Jordan hears plea for kids|000355c.sty
<menu3_4>Hamilton girl murdered|000355d.sty
<menu3_5>Shore preservation|000355e.sty
<menu3_6>previous menu|menu0
<menu4_1>Template editor|menu2
<menu4_2>Story editor|menu2
<menu4_3>Segment editor|menu2
<menu4_4>Billing software|menu2
<menu4_5>Scheduler software|menu2
<menu4_6>previous menu|menu2
```

menutpl

SYNTAX:

<menutpl>filename

DESCRIPTION: Specifies the template file to use for building a menu list in the interactive presentation.

SAMPLE:

```
<menutpl>menudef.tpl
```

menuback

SYNTAX:
<menuback>filename

DESCRIPTION: Specifies the background bitmap file to use for building a menu list in the interactive presentation.

SAMPLE:
```
<menuback>default.tga
```

STORY operators

page

SYNTAX:
<page>hh:mm:ss|mm:ss|n

DESCRIPTION: Signifies a change of pages in the presentation. It is used primarily as a device for the News Channel but can be applied to any product that requires tracking the changing of pages.

SAMPLE:
```
<page>2
```

txt

SYNTAX:
<txt*n*>hh:mm:ss|mm:ss||text

DESCRIPTION: Denotes a line of text to be rendered to the display.

SAMPLE:
```
<txt1>00:00:00|00:15||PHILADELPHIA - The Phils Sweep the Division
```

msg

SYNTAX:
<msg*n*>hh:mm:ss|mm:ss

DESCRIPTION: Denotes a line of text to be rendered to the display.

SAMPLE:
```
<msg1>00:00:00|00:15
```

wav

SYNTAX:
<wav*n*>hh:mm:ss|mm:ss|||filename

DESCRIPTION: Signifies the inclusion of a WAV digital audio file. The actual file is referenced by the trailing filename.

SAMPLE:
```
<wav1>00:00:00|00:15||sample.wav
```

back

SYNTAX:

SYNTAX:
<back>hh:mm:ss|mm:ss||filename

DESCRIPTION: Filename of .TGA file which will be overlayed by live video. graphics or rendered text.

SAMPLE:
```
<back>00:00:00|00:15||sample.tga
```

img

SYNTAX:
<img*n*>hh:mm:ss|mm:ss||filename

DESCRIPTION: Signifies the inclusion of a TARGA raster file. The actual file is referenced by the trailing filename.

SAMPLE:
```
<img1>00:00:00|00:15||sample.tga
```

tmpl

SYNTAX:
<tmpl>hh:mm:ss|mm:ss|filename

DESCRIPTION: Specifies the template file to be used for the current page. The actual file is referenced by the trailing filename. This specification should come immediately after the <page> tag.

SAMPLE:
```
<tmpl>sample.tpl
```

hed

SYNTAX:
<hed*n*>hh:mm:ss|mm:ss||text

DESCRIPTION: Signifies the inclusion of a headline. The actual text is referrenced by the last text entry following the transition designation.

SAMPLE:
```
<hed1>00:00:00|00:15||Phillies Take New York by Storm
```

sub

SYNTAX:
<sub*n*>hh:mm:ss|mm:ss||text

DESCRIPTION: Signifies the inclusion of a subhead. The actual text is referrenced by the last text entry following the transition designation.

SAMPLE:
```
<sub1>00:00:00|00:15||Fans riot in the streets!
```

tim

SYNTAX:
<tim*n*>hh:mm:ss|mm:ss||time format|date format

DESCRIPTION: Signifies the inclusion of a time stamp on the video display. The location, color, font and size is determined by the corresponding reference in the template file. There are different types of display format for time. The following list describes the available formats.

0	none
1	hh:mm am/pm
2	hh:mm:ss am/pm
3	hh:mm military
4	hh:mm:ss military
5	mm:ss

There are different types of display format for time. The following list describes the available formats.

0	none
1	mm/dd
2	mm/yy
3	mm/dd/yy
4	mm/dd/yyyy
5	Month dd
6	Month dd, yyyy
7	Month yyyy
8	mm - dd
9	mm - yy
10	mm - dd - yy
11	mm - dd - yyyy

SAMPLE:
```
<tim1>00:00:00|01:00||1|0
<tim2>00:00:00|01:00||0|5
```

vid

SYNTAX:
<vid*n*>hh:mm:ss|mm:ss|device|startaddress|endaddress|filename

DESCRIPTION: Signifies the inclusion of a digital video clip file. The actual file is referenced by the trailing filename or by the starting and ending address on the device.

SAMPLE:
```
<vid1>00:00:00|00:15|DVS1|||sample.vid
<vid2>00:00:10|00:05|LASR|12345|34567|
<vid3>00:00:15|00:15|TAPE|12345|34567|
```

; **(used in ALL SMSL script files)**

SYNTAX:
; any ascii text

DESCRIPTION: Comment line used to annotate the files, ignored by the engine.

SAMPLE:
```
; J:\PROG\SRC\BARKER\SAMPLE.STY
; Last User: John Doe
; September 6, 1994  11:20:23
```

lin

SYNTAX:
<lin*n*>hh:mm:ss|mm:ss

DESCRIPTION: Signifies the drawing of a line.

SAMPLE:
```
<line>00:00:00|00:15
```

box

SYNTAX:
<box*n*>hh:mm:ss|mm:ss

DESCRIPTION: Signifies the drawing of a box.

SAMPLE:
```
<box1>00:00:00|00:15
```

clk

SYNTAX:
<clck*n*>hh:mm:ss|mm:ss|start time|interval|direction|time format

DESCRIPTION: Signifies the inclusion of a clock on the video display which will be updated at the specified interval. The location, color, font and size is determined by the corresponding reference in the template file. There are different types of display format for time. The following list describes the available formats.

0	none
1	hh:mm am/pm
2	hh:mm:ss am/pm
3	hh:mm military
4	hh:mm:ss military
5	mm:ss

The direction item specified whether the clock is to count upwards or downwards. A value of zero will cause the clock time to be incremented by the value specified by the interval parameter. Likewise, a value of 1 will cause the clock time to be decremented (to allow countdown timers)

Note: This operator will only work properly when used on top of a full screen video with text overlay.

SAMPLE:
```
<clk1>00:00:00|00:15|02:15:00|00:00|1|0
```

trns

SYNTAX:
<trns>hh:mm:ss|mm:ss|type

DESCRIPTION: Signifies the setting of the global variable for transition type. The SMME will read this tag and set its internal transition global variable. Therefore, all page transitions that follow this tag will transition according to the type specified. Please see the section on transitions for a complete description of transition types. The default transition is F1000 which equals a fade transition lasting for 1 second. The following example shows a transition of 1/2 second being directed to the system.

SAMPLE:
```
; story file
<trns>00:00:00|00:00|F0500
<subl>00:00:00|00:15||Fans riot in the streets!
```

blk

SYNTAX:
<blk*n*>hh:mm:ss|mm:ss|on time|off time|text

DESCRIPTION: Signifies the inclusion of a text item which will be flashed on and off on the screen at the defined intervals. On time specifies the duration that the text will be displayed on the screen, and off time specifies the time interval that the item will be removed from the screen before it is displayed again.

Note: This operator will only work propeny when used on top of a full screen video with text overlay.

SAMPLE:
```
<clck1>00:00:00|00:15|02:15:00|00:00|1|0
```

SEGMENT operators

stry

SYNTAX:
<stry>hh:mm:ss|mm:ss||filename

DESCRIPTION: Signifies the inclusion of a story within a segment file. The filename points to a complete story file that will in turn describe the individual multimedia sequences.

SAMPLE:
```
<stry>00:00:00|00:30||sample.sty
```

advr

SYNTAX:
<advr>hh:mm:ss|mm:ss|filename

DESCRIPTION: Signifies the inclusion of an advertisement in the sequence of events in the segment file. The filename points to the advertisement playlist supplied to the system. That playlist contains all of the information on where to find the advertisement and what device to get the ad from.

SAMPLE:
```
<advr>00:00:00|00:15|sample.ad
```

TEMPLATE operators

box

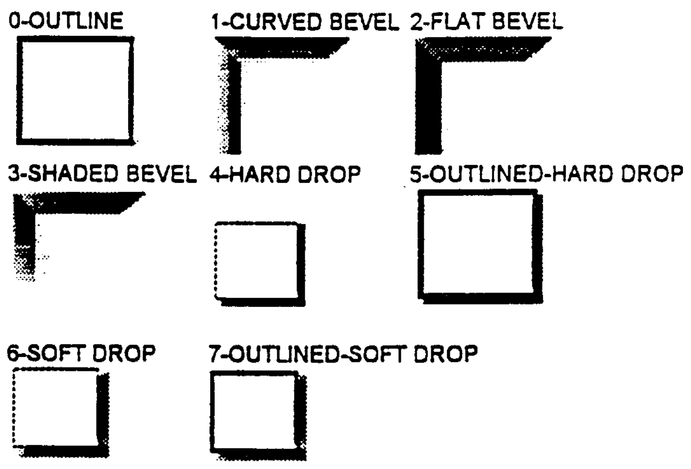
SYNTAX:

<box*n*>llx|lly|w|h|frametype|filltype|lineweight|line R|G|B|fill R|G|B

DESCRIPTION: Signifies the inclusion of a box on the video page. Below is a list of available box frame types:

0	outline frame
1	curved bevel
2	flat bevel
3	shaded bevel
4	hard drop shadow
5	outline w/hard droo shadow
6	soft drop snadow
7	outline w/soft drop shadow

Visual samples:



The last entry in this command line is the frame weight. It is measured in pixels. The color of the frame is determined by the current foreground color as set by the <colr> tag. If the box is filled the fill color is determined by the current background color.

An fill entry of 1 signifies that the box is to be filled. An entry of 0 signifies the box is not filled.

0	unfilled
1	filled

SAMPLE:

```
<box1>100|240|122:100|5|1|2|128|129|128|64|64|64
```

lin

SYNTAX:

<lin*n*>x1|y1|x2|y2|line weight|R|G|B

DESCRIPTION: Signifies the inclusion of a line.

SAMPLE:

```
<line>100|200!55|25|3|128|46|46
```

txt

SYNTAX:
<txtn>x|y|x|y|character spacing|size|leading|fontname|font color R|G|B|shadow size|shadow color R|G|B

DESCRIPTION: Signifies the inclusion of a line of text.

SAMPLE:
```
; story template
<hed1>40|78|432|61|0|55|55|uvcoo___ 0|0|0|3|0|0|0
<img1>40|160|432|126
<txt1>40|290|432|130|0|22|22|hv______ 128|0|0|3|0|0|0
<vid1>0|0|512|486
```

msg

SYNTAX:
<msgn>x|y|x|y|character spacing|size|leading|fontname|font color R|G|B|shadow size|snadow color R|G|B

DESCRIPTION: Signifies the inclusion of a message line. The text for this message is obtained from the message file on disk.

SAMPLE:
```
; story template
<hed1>40|78|432|61|0|55.55|uvcoc___ 0|0|0|3|0|0|0
<img1>40|160|432|126
<msg1>40|290|432|130|0|22|22|hv______ 128|0|0|3|0|0|0
<vid1>0|0|512|486
```

grd

SYNTAX:
<grd*n*>gradient type|starting R|G|B|ending R|G|B

DESCRIPTION: Specifies a gradient fill. This is a global reference that can be applied, but must also be restored when necessary. The following types may be applied to any filled box:

0	no gradient
1	top-down
2	left-right

Visual samples:

SAMPLE:
```
<box1>100|240|122|100|5|1|2|128|128|128|64|64|64
```

hed

SYNTAX:
<hedn>x|y|x|y|character spacing|size|leading|fontname|font color R|G|B|shadow size|shadow color R|G|B

DESCRIPTION: Signifies the inclusion of a line of text. This operator is identical to the txt operator.

SAMPLE:
```
; story template
<hed1>40|78|432|61|0|55|55|uvcoo___ 0|0|0|3|0|0|0
<img1>40|160|432|126
<txt1>40|290|432|130|0|22|22|nv______ 128|0|0|3|0|0|0
<vid1>0|0|512|486
```

img

SYNTAX:
<imgn>x|y|x|y|transparent|red|green|blue

DESCRIPTION: Signifies the inclusion of an image. A particular color in the image may be made transparent by assigning transparent=1, and specifying the RGB value of the color in the folowing three parameters. If transparent=0 then no colors in the image will be made transparent.

SAMPLE:
```
; story template
<hed1>40|78|432|61|0|55|55|uvcbo___ 0|0|0|3|0|0|0
<img1>40|160|432|126|0|0|0|0
<txt1>40|290|432|130|0|22|22|hv______ 128|0|0|3|0|0|0
<vid1>0|0|512|486|1.255|0|0|0
```

vid

SYNTAX:
<vidn>type|x1|y1|x2|y2

DESCRIPTION: Signifies the inclusion of an video clip. If the type field is 0, the video clip will be played full screen and all other items will be rendered ontop of the video. If the type field is 1, the video will be played in the window specified by (x1,y1)-(x2,y2).

SAMPLE:
```
; story template
<hed1>40|78|432|61|0|55|55|uvcbc___ 0|0|0|3|0|0|0
<img1>40|160|432|126
<txt1>40|290|432|130|0|22|22|hv______ 128|0|0|3|0|0|0
<vid1>0|0|0|512|486
```

sub

SYNTAX:
<subn>x|y|x|y|character spacing|size|leading|fontname|font color R|G|B|shadow size|shadow color R|G|B

DESCRIPTION: Signifies the inclusion of a subhead. This operator is identical to the txt operator.

SAMPLE:
```
; story template
<sub1>40|78|432|61|0|55|55|uvcbo___ 0|0|0|3|0|0|0
<img1>40|160|432|126
<txt1>40|290|432|130|0|22|22|hv______ 128|0|0|3|0|0|0
<vid1>0|0|512|486
```

tim

SYNTAX:
<timn>x|y|x|y|character spacing|size|leading|fontname|font color R|G|B|shadow size|shadow color R|G|B

DESCRIPTION: Signifies the inclusion of a subhead. This operator is identical to the txt operator.

SAMPLE:

```
; story template
<timl>40|78|432|61|0|55|55|uvcbo___ 0|0|0|3|0|0|0
<imgl>40|160|432|126
<txtl>40|290|432|130|0|22|22|hv______ 128|0|0|3|0|0|0
```

clk

SYNTAX:
<clckn>x|y|x|y|character spacing|size|leading|fontname|font color R|G|B|shadow size|shadow color R|G|B

DESCRIPTION: Signifies the drawing of a clock.

SAMPLE:
```
; story template
<clckl>40|78|432|61|0|55|55|uvcbo___ 0|0|0|3|0|0|0
<imgl>40|160|432|126
<txtl>40|290|432|130|0|22|22|hv______ 128|0|0|3|0|0|0
```

blk

SYNTAX:
<clckn>x|y|x|y|character spacing|size|leading|fontname|font color R|G|B|shadow size|shadow color R|G|B

DESCRIPTION: Signifies the drawing of a flashing text item.

SAMPLE:
```
; story template
<clckl>40|78|432|61|0|55|55|uvcbo___ 0|0|0|3|0|0|0
<imgl>40|160|432|126
<txtl>40|290|432|130|0|22|22|hv______ 128|0|0|3|0|0|0
```

Devices

The devices entry within the story and segment files refers to the physical device attached to the computer system. The four letter acronym tells the Multimedia Engine which device will be used to access data. Below is a description of all of the device types supported by the Engine.

comp

DESCRIPTION: Specifies that the data for this instruction is available on the host computer. It is assumed that the drive and directory are the same as the location of the Engine. If the application using the Engine has a separate set of directories setup by its INI file, then that drive and directory designation is used. Comp is the default device and should never actually be seen in any of these files. If the device parameter is null but the tag actually uses the device parameter, the engine will assume comp.

dvs*n*

DESCRIPTION: Specifies that the data for this instruction is available on the digital video subsystem. In this case the Scientific Atlanta Playback board is the source. The actual board designation is determined by the numeric value contained in the tag. For instance, if the tag specified dvs1 then the first SA Playback board will be used.

tape

DESCRIPTION: Specifies that the data for this instruction is available on an external tape deck.

lasr

DESCRIPTION: Specifies that the data for this instruction is available on an attached laser disc player.

Transitions

The transitions entry within the story and segment files refers to the type of video transition that is to be used between video pages. This transition is called by the Engine used the Targa transitional effects that are resident on the Targa board. A limited set of transitions exist for this display board, and a smaller number of transitions have been engineered into the Engine.

NOTE: Transition information has moved from an item within all tags to its own tag. The SMME will read this transition and set a global variable for use in all of the following transitions.

The duration of effects in the engine is determined in milliseconds, not frames.

F*nnnn*

DESCRIPTION: Specifies that the transition will be a fade. The numeric value contained in this tag instructs the Engine as to the duration of the fade from one page to the next. The value is measured in milliseconds. Therefore a value of 1000 would request a fade of one second.

SAMPLE:
```
<txt1>00:00:00|00:06||This is a headline
```

W*cnnnn*

DESCRIPTION: Specifies that the transition will be a wipe. The numeric value contained in this tag instructs the Engine as to the type of wipe from one page to the next. The available wipe types are requested from the following table of wipe types.

CODE	WIPE
WR*nnnn*	wipe to the right across nnnn milliseconds
WL*nnnn*	wipe to the left across nnnn milliseconds
WU*nnnn*	wipe up across nnnn milliseconds
WD*nnnn*	wipe down across nnnn milliseconds

D*nnnnn*

DESCRIPTION: Specifies that the transition will be a diagonal wipe. The numeric value contained in this tag instructs the Engine as to the type of wipe from one page to the next and its duration. The first n denotes which wipe type from the following table will be used.

CODE	WIPE
D1*nnnn*	wipe diagonally from top left to lower right across nnnn milliseconds
D2*nnnn*	wipe diagonally from lower left to upper right across nnnn milliseconds
D3*nnnn*	wipe diagonally from upper right to lower left across nnnn milliseconds
D4*nnnn*	wipe diagonally from lower right to upper left across nnnn milliseconds

Inline Formatting and Commands

Inline formatting allows the individual lines of the STORY file to address multiple character rendering formats. For instance, the News Channel may require the use of italic or underlined text in a single line of text. This is prevelent in headlines when emphasis is required or when the name of a publication is used within the sentence. All of the inline formatting or commands are identified by the backslash. This was used to help keep the Lenfest Multimedia Language in line with some of the other common languages, such as C.

The following list of format and command operators is in no way complete. Other will be added as needed. The current list of operators is in lowercase and all are a single character.

s*nnn*

DESCRIPTION: Specifies that the current point size should be changed to nnn points. The numeric value contained in this tag instructs the Engine as to the point size. Therefore a value of 030 would request a change in point size to 30 points. The value nnn is fixed in length to 3 characters.

RANGE:
6 to 999

SAMPLE:
```
<txtl>00:00:00|00:06||This is a \s030headline
```

ffontnamef

DESCRIPTION: Specifies that the current font should be changed to fontname. Since the fontname is of undetermined length there must be an accompanying \f to mark the end of the fontname.

RANGE:
Any legal and available font name. The default font is the current font.

SAMPLE:
```
<txt1>00:00:00|00:06||This is a s030\fHarrisburg\fheadline
```

lnnn

DESCRIPTION; Specifies that the current leading should be changed to nnn points. The numeric value contained in this tag instructs the Engine as to the leading size. Therefore a value of 030 would request a change in leading to 30 points. The value nnn is fixed in length to 3 characters.

RANGE:
6 to 999

SAMPLE:
```
<txt1>00:00:00|00:06!This is a s030\032headline
```

ann

DESCRIPTION: Specifies that a font attribute is being requested. The numeric value contained in this tag instructs the Engine as to the type of attribute change. The value nn is fixed in length to 2 characters. The attribute values can be added together to create composite attribute type such as 06, bold-italic; or 10, bold-underline.

ACCEPTABLE ATTRIBUTE TYPES:
normal = 01
bold = 02
italic = 04
underline = 08

SAMPLE:
```
<txt1>00:00:00|00:06||This is a \s030\l032\a04headline
```

rnnn

DESCRIPTION: Specifies that a the red component in RGB of the current font color is being requested to change. The numeric value contained in this tag instructs the Engine as to the type of the new red value, ranging from 0 to 255 with 0 being the darkest and 255 the brightest. The value nnn is fixed in length to 3 characters.

SAMPLE:
```
<txt1>00:00:00|00:06||This is a \s030\l032\a04\r255a headline which has full
red
```

g*nnn*

DESCRIPTION: Specifies that a the green component in RGB of the current font color is being requested to change. The numeric value contained in this tag instructs the Engine as to the type of the new red value. ranging from 0 to 255 with 0 being the darkest and 255 the brightest. The value nnn is fixed in length to 3 characters.

SAMPLE:
```
<txtl>00:00:00|00:06||This is a  s030\1032\a04\r255a headline which has full
green
```

b*nnn*

DESCRIPTION: Specifies that a the blue component in RGB of the current font color is being requested to change. The numeric value contained in this tag instructs the Engine as to the type of the new red value, ranging from 0 to 255 with 0 being the darkest and 255 the brightest. The value nnn is fixed in length to 3 characters.

SAMPLE:
```
<txtl>00:00:00|00:06||This is a  s030\1032\a04\r255a headline which has full
blue
```

The Advertising List File

The advertising list file is used to insert advertising from a separate device, or as a story file into the multimedia presentation. The reason for having a separate list is that the insertion of advertising can be more, or less, dynamic than the multimedia presentation. In the case of the News Channel, it requires news programming to be fluid throughout the day. It cannot be burdened by managing the trafficking of advertising.

To handle advertising, the advertising list file is a sequential list of advertising insertion times and other IDs. This file should be read during the multimedia presentation as needed to properly insert the next ad in the list whenever the <advr> tag placed in the SEGMENT file for that presentation.

It is the sole responsibility of the multimedia engine to keep track of the advertising play list and to record the log information describing the success or failure of the ad insertion.

The creation of advertising playlists is handled by the ADLIST.EXE application. The application functions very much like the SEGMENT.EXE application which combines stories to create segments. ADLIST combines individual advertisements into the advertisement list file and manages the files across multiple days.

The advertising play list must be delivered to the Playback Chassis for each day of service. The filename format of this file is as follows:

MMDD.AD

Sample:

0101.AD
0102.AD
0103.AD
.
.
.
1231.AD

which represents January 1 through December 31 of the current year.

This file must reside in the following directory:

C:\MMDATA\AD

The format for the actual list file is as follows:

HH:MM|barcode|barcode|barcode|... barcode

The barcode can be replaced by a Story filename if the user wants to playback a multimedia advertisement rather than a digital video file. It would appear as follows:

HH:MM|barcode|SAMPLE.STY|barcode|... barcode

The following is a sample advertising playlist:

```
; filename: 1229.AD
; advertising play list
; 09/06/94
; creationdate: 08/09/93
; createdfor: NEWSCHANNEL
; systemid: 2442, Coatesville
00:00|12345|23456|34567|45678|56789
00:10|12345|23456|0006123.STY|45678|56789
00:20|12345|23456|34567|45678|56789
```

```
.
.
.
23:30|000B123.STY|23456|34567|45678|56789
23:40|12345|23456|34567|000B123.STY|56789
23:50|12345|000B123.STY|000B123.STY|45678|56789
```

The local creation system must follow a treed directory structure that is referenced in the ADLIST.INI file. The tree should be as follows:

```
\NTV
        \AD
                \DST00313
                \DST00314
                \DST00315
                .
                .
                .
                \DST00456
```

Within each directory will reside the ADLIST files for each day for that site.

The ADLIST.INI file is as follows

```
; ADLIST.INI
; Created on 01/29/94
;
[sites]
Doylestown=DST00313
Coatesville=DST00314

[mpegfiles]
1=23455
2=23456
3=23457

[storyfiles]
1=SAMPLE.STY
2=JUNK.STY
```

The ADLIST.INI file will be maintained by the ADLIST.EXE software. This application will read and write the .AD files for each site. It will also be the list manager for the available MPEG spots and relative .STY files composed as advertisements.

The Message List File

The message list file is used to insert a dynamic message from a separate text file into the multimedia presentation.

To handle a message, the message list file is a sequential list of message insertion times and other IDs. This file should be read during the multimedia presentation as needed to properly insert the next message in the list whenever the <msg*n*> tag placed in the STORY file for that presentation.

It is the sole responsibility of the multimedia engine to keep track of the message play list and to record the log information describing the success or failure of the message insertion.

The creation of message playlists is handled by the MSGLIST.EXE application. The application functions very much like the ADLIST.EXE application. MSGLIST combines individual messages into the message list file and manages the files across multiple days.

The message play list must be delivered to the Playback Chassis for each day of service. The filename format of this file is as follows:

```
MMDD.MSG
```

Sample:

```
0101.MSG
0102.MSG
0103.MSG
.
.
.
1231.MSG
```

which represents January 1 through December 31 of the current year.

This file must reside in the following directory:

```
C:\MMDATA\MESSAGE
```

The format for the actual list file is as follows:

```
HH:MM:SS|HH:MM:SS|message|message|... message
```

The following is a sample message playlist:

```
; filename: 1229.MSG
; advertising play list
; 09/06/94
; creationdate: 08/09/93
; createdfor: NEWSCHANNEL
; systemid: 2442, Coatesville
00:00:00|00:10:20|The president has been shot|The president has died|Funeral on Friday
.
.
.
23:00:00|23:10:20|The president has been shot|The president has died|Funeral on Friday
```

The *nnnnnnnn*.LOG

The creation of an activities log file is extremely important to the overall operations of the system. The name of the log file is determined by the system ID. This way the person viewing the log file that comes back from the field will know which machine it has come from.

The information needed in the log file is as follows:

```
; 00123456.LOG
; 08/12/93
; Product = Barker
; Location = Coatesville
;
Initialization successful
01:55:00 LS 00002000.SEG
02:00:00 PS 000AF314.STY
02:01:00 PS 000AF316.STY
02:01:30 PS 000A3314.STY
02:02:00 PS 00022314.STY !ERROR! DVS1 file not found
02:02:30 PS 00017314.STY
02:03:30 PS 00083314.STY
02:04:00 PS 00076543.STY
ad1=1|00:00:00|00:30|1234|2345|Gillette
.
.
.
ad2=1|23:59:00|00:30|1234|2345|Gillette
23:59:30 PS 00076543.STY
```

The LS acronym stands for Load Segment. The PS acronym stands for Play Story.

The creation of advertising entries withing the log file is important from a billing perspective and from a basic system maintenance perspective. The log file should report back the basic success or failure of an advertising insertion. A successful insertion is referenced as a 1 and failure as a 0. It is the first entry item within an entry line. The remainder of the line is exactly the same as advertising INI file. This will allow for simple searches and retrieval of valuable diagnostic information. Advertisement logging will occur in the mm engine's log file and will be split out if necessary to a separate log using another program.

```
ad1=1|00:00:00|00:30|1234|2345|Gillette
.
.
.
ad2=1|23:59:00|00:30|1234|2345|Gillette
.
.
.
adn=1|23:59:30|00:30|4567|5678|Suburban Cable
```

The STARNET.INI File

```
; STARNET.INI
; StarNet, Inc.
; Multimedia Labs
; 1332 Enterprise Drive, Suite 200
; West Chester, PA  19380
;
;
; Purpose:  This file is used to describe the system profile for one of
;           any of the StarNet PC-based products.
;
; History:
;           05/26/93  gkk  created
;           06/02/93  gkk  modified the system id information to match
;                          what is happening in the VB FORM_SYS.FRM file.
;
;           6/24/93 HJP  Added [version] section with version=
;                        Major version changes (X) should only be made if the
;                          format or content of any existing section changes or if
;                          any section is removed.
;                        Minor version changes (.XXX) reflect addition of new
;                          sections
;
[version]
version=1

[system]
; the system id is used by communications to uniquely identify all
;   inbound communications.
; the system name is the text name for the system.
; the system phone data is the phone number for dialing into the machine.
; the system phone voice is the phone number for calling the system contact.
; the system phone contact is the name of the system representative
; time zone is measured in number of hours offset from GMT.
;
parentoperator=TCI
systemoperator=Suburban Cable
contact1=John Smith
phone1=(215) 555-1212
contact2=Jane Jones
phone2=(215) 555-1212
systemaddress=Coatesville, PA
numberofsubs=80,000
timezone=0

[directories]
template=c:\template
font=c:\fonts
inbox=c:\inbox
image=c:\image
background=c:\backgrnd
wave=c:\wav
message=c:\message
app=c:\barker

[barker]
; the starnet product line identifies which service is on this system
; the starnet phone home number is the number that this machine will call
;   when a problem occurs
; the mpeg line tells the system if MPEG files are to be used.
ppvnumber=(215) 555-1212
systemnumber=(215) 555-1212
phonehome=(215) 692-0000
systemid=123456.123456
stereo=yes
mpeg=yes
commtype=1
systemnotes=Modified the Barker INI to accomodate new system id.

; Channel Map
;
; The data structure:
; Days of week are listed as integer values that correspond to a bit-mask
; in the reading and writing software. This is done to conserve space.
; The individual integer values for days of the week are:
;   Sunday=1
;   Monday=2
;   Tuesday=4
;   Wednesday=8
;    Thursday=16
;    Friday=32
;    Saturday=64
; If all days of the week are turned on then the value is:
; 64+32+16+8+4+2+1 or 127. If Monday, Tuesday and Thursday
; are turned on then the value is 16+4+2 or 22.
;
; The different definitions per channel are always separated
; by a | (piping symbol).
;
; The time is expressed as a single integer between 0 and 47.
; Each unit represents a 1/2 hour time slice. So 25 would be
; equal to 12:30pm and 14 would be equal to 7:00am.
;
; The order of appearance is as follows:
; CHAN#=CALL LETTERS, DAY, TIME [|CALL LETTERS, DAY, TIME]...

[channels]
2=PRISM,127,0-0
3=KYW,127,0-0
4=PREV,127,0-0
5=HBO,127,0-0
6=WPVI,127,0-0
7=WCOJ,127,0-0
8=WGAL,127,0-0
9=WPHL,127,0-0
10=WCAU,127,0-0
11=WTXF,127,0-0
12=WHYY,127,0-0
13=WGBS,127,0-0
14=SHOW,127,0-0
16=DIS,127,0-0
17=MAX,127,0-0
18=MEU,127,0-0
19=QVC,127,0-0
20=LOCAL,127,0-0
21=HSN,127,0-0
22=CSPAN,127,0-0
23=REQ1,127,0-0
24=REQ2,127,0-0
25=PLAY,127,0-0
26=SPICE,127,0-0
27=USA,127,0-0
28=DSC,127,0-0
29=TNN,127,0-0
30=FAM,127,0-0
31=CNN,127,0-0
32=NICK,127,0-0
33=SC,127,0-0
34=A&E,127,0-0
35=CVS,127,0-0
36=LIFE,127,0-0
37=VC,127,0-0
38=CCM,127,0-0
39=CNBC,127,0-0
40=TBS,127,0-0
41=BET,127,0-0
42=AMC,127,0-0
43=MTV,127,0-0
44=CNN,127,0-0
45=VH1,127,0-0
46=TLC,127,0-0
47=TNT,127,0-0
48=WWOR,127,0-0
49=TWC,127,0-0
51=ENC,127,0-0
52=COURT,127,0-0
53=SCIFI,127,0-0
54=CART,127,0-0
```

CLAIMS

1. A multimedia presentation system capable of dynamically constructing a multimedia presentation comprising:

a receiving and storing means for receiving and storing commands and data files, which commands comprise script files or function calls; and

a multimedia engine comprising a renderer and an application programming interface or a script manager, which multimedia engine comprises:

a translating means for translating human-readable script files or human-readable function calls into commands usable by the renderer; and

an executing means for executing the translated commands to create a multimedia presentation comprising audio and video components for broadcast, cable transmission or display.

2. The multimedia presentation system of claim 1, wherein the data files comprise image, video, audio and text files.

3. The multimedia presentation system of claim 1, wherein the script files include at least one segment file.

4. The multimedia presentation system of claim 3, wherein the script and data files already within the receiving and storing means can be substituted before the scheduled broadcast, cable transmission or display time of a multimedia presentation, so long as the substitution occurs before the time and date that the segment, as defined in the segment file, in which the script and data files are to be presented is scheduled for broadcast, cable transmission or display.

5. The multimedia presentation system of claim 3, wherein the at least one segment file encodes the name of at least one story file.

6. The multimedia presentation system of claim 3 comprising an advertisement file stored in the receiving and storing means, wherein the at least one segment file encodes the name of said advertisement file.

7. The multimedia presentation system of claim 6, wherein said advertisement file encodes a list of video file names or story file names, each such video or story file having a duration of defined length.

8. The multimedia presentation system of claim 7, wherein the multimedia engine monitors which video or story files listed in an advertisement file have been broadcast, cable transmitted or displayed and stores the information in the receiving and storing means.

9. The multimedia presentation system of claim 6 comprising a video file stored in the receiving and storing means, wherein the advertisement file names said video file.

10. The multimedia presentation system of claim 5 comprising said at least one story file and at least one template file stored in the receiving and storing means, which story file encodes text and the name of said at least one template file.

11. The multimedia presentation system of claim 10, wherein every segment file encodes a story file starting time and a story file duration in association with each story file named in the segment file.

12. The multimedia presentation system of claim 10 comprising a video file stored in the receiving and storing means and wherein said at least one story file names said video file.

13. The multimedia presentation system of claim 10 comprising an audio file stored in the receiving and storing means and wherein said at least one story file names said audio file.

14. The multimedia presentation system of claim 10 comprising an image file stored in the receiving and storing means and wherein said at least one story file names said image file.

15. The multimedia presentation system of claim 14, wherein the renderer comprises a means of rendering the image stored in said image file by positioning the image in the broadcast, cable transmission or display according to the instructions in the template file.

16. The multimedia presentation system of claim 1, wherein the receiving and storing means comprises a remote receiving means for receiving the script files and data files from a remote source.

17. The multimedia presentation system of claim 16, wherein the remote receiving means comprises a satellite dish.

18. The multimedia presentation system of claim 17 further comprising a cable television system and an output for transmitting the multimedia presentation over the cable television system.

19. The multimedia presentation system of claim 1, wherein the translating means comprises the script manager.

20. The multimedia presentation system of claim 1, wherein the translating means comprises the application programming interface.

21. The multimedia presentation system of claim 1, wherein the translating means comprises the script manager and the application programming interface.

22. The multimedia presentation system of claim 21, wherein the script files encode the name of at least one segment file.

23. The multimedia presentation system of claim 22, wherein the at least one segment file encodes the name of at least one story file.

24. The multimedia presentation system of claim 23 comprising at least one story file stored in the receiving and storing means, which story file encodes text to be used in a multimedia presentation and the name of at least one template file.

25. The multimedia presentation system of claim 24, wherein the at least one segment file encodes the name of at least one advertisement file.

26. A multimedia presentation system capable of dynamically constructing a multimedia presentation for cable transmission to a plurality of cable television users comprising:

a receiving and storing means for receiving and storing commands and data files, which commands comprise script files or function calls;

at least one segment file, at least one story file and at least one template file stored in the receiving and storing means, wherein said at least one segment file encodes the name of said at least one story file, and said at least one story file encodes text and the name of said at least one template file; and

a multimedia engine comprising a renderer, an application programming interface and a script manager, which multimedia engine comprises:

a translating means of translating human-readable script files or human-readable function calls into commands usable by the renderer; and

an executing means of executing the translated commands to create a multimedia presentation comprising audio and video components for cable transmission.

27. The multimedia presentation system of claim 26, wherein the receiving and storing means comprises a remote receiving means for receiving the script files and data files from a remote source.

28. The multimedia presentation system of claim 27, wherein the remote receiving means comprises a satellite dish.

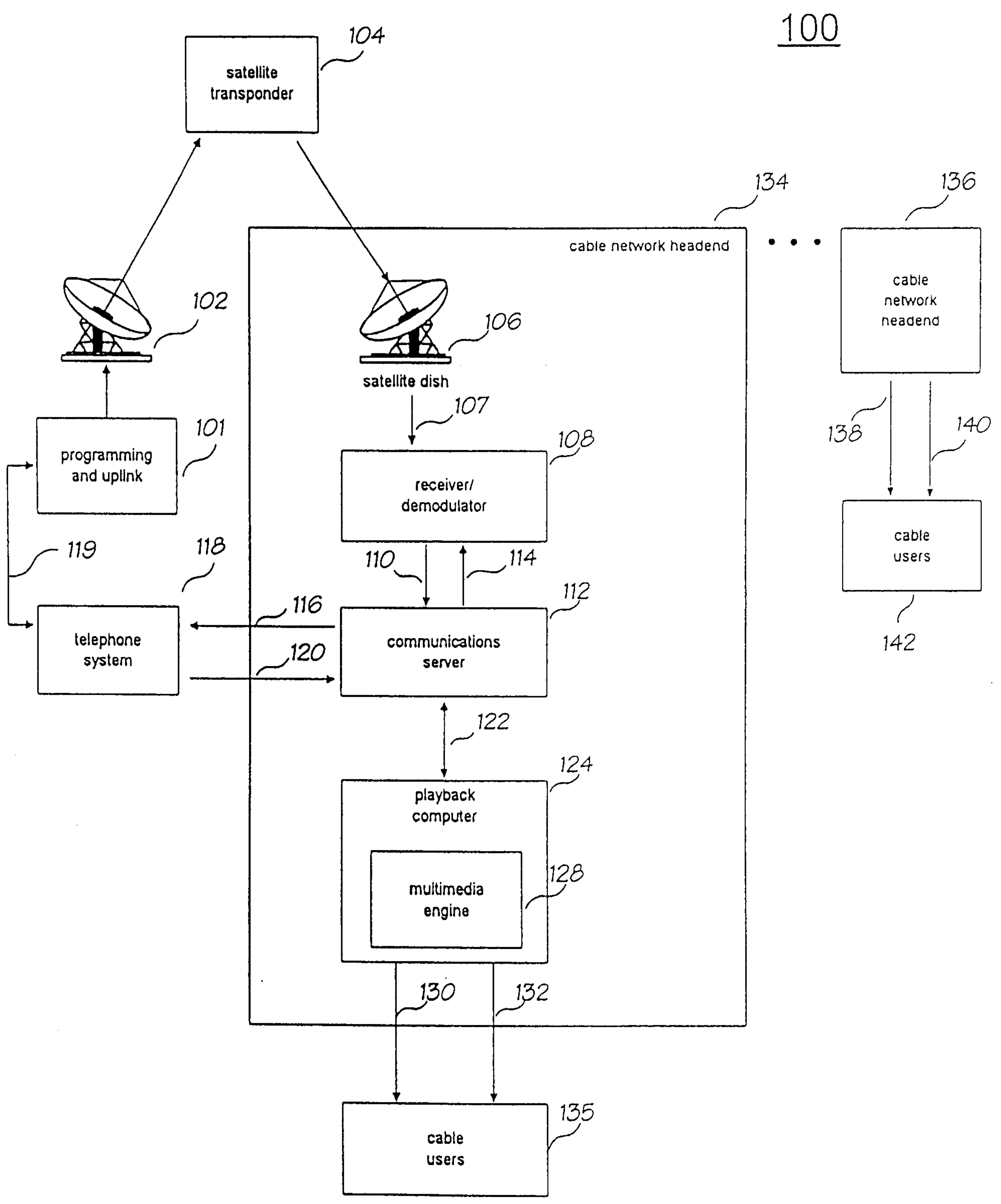

100
satellite transponder
104
134
136
cable network headend
• • •
cable network headend
102
106
satellite dish
107
programming and uplink
101
108
receiver/ demodulator
138
140
cable users
119
118
110
114
112
142
116
telephone system
communications server
120
122
124
playback computer
multimedia engine
128
130
132
cable users
135

Fig. 1

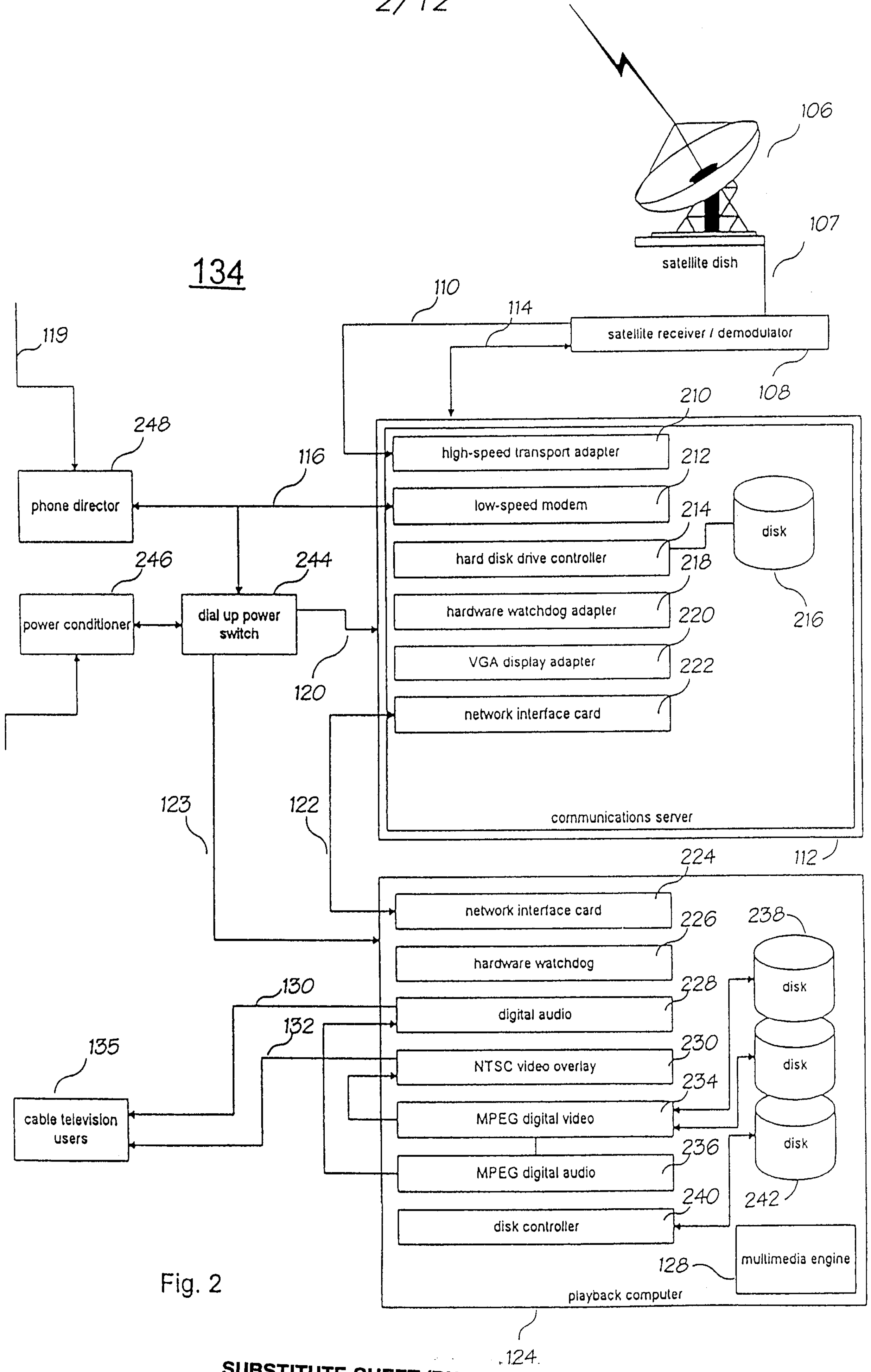

134
106
satellite dish
107
110
114
satellite receiver / demodulator
108
119
248
phone director
116
210
high-speed transport adapter
212
low-speed modem
214
hard disk drive controller
disk
216
218
hardware watchdog adapter
220
VGA display adapter
222
network interface card
246
power conditioner
244
dial up power switch
120
123
122
communications server
112
224
network interface card
226
hardware watchdog
238
228
digital audio
disk
130
132
230
NTSC video overlay
disk
234
MPEG digital video
disk
236
MPEG digital audio
242
240
disk controller
135
cable television users
128
multimedia engine
playback computer
124

Fig. 2

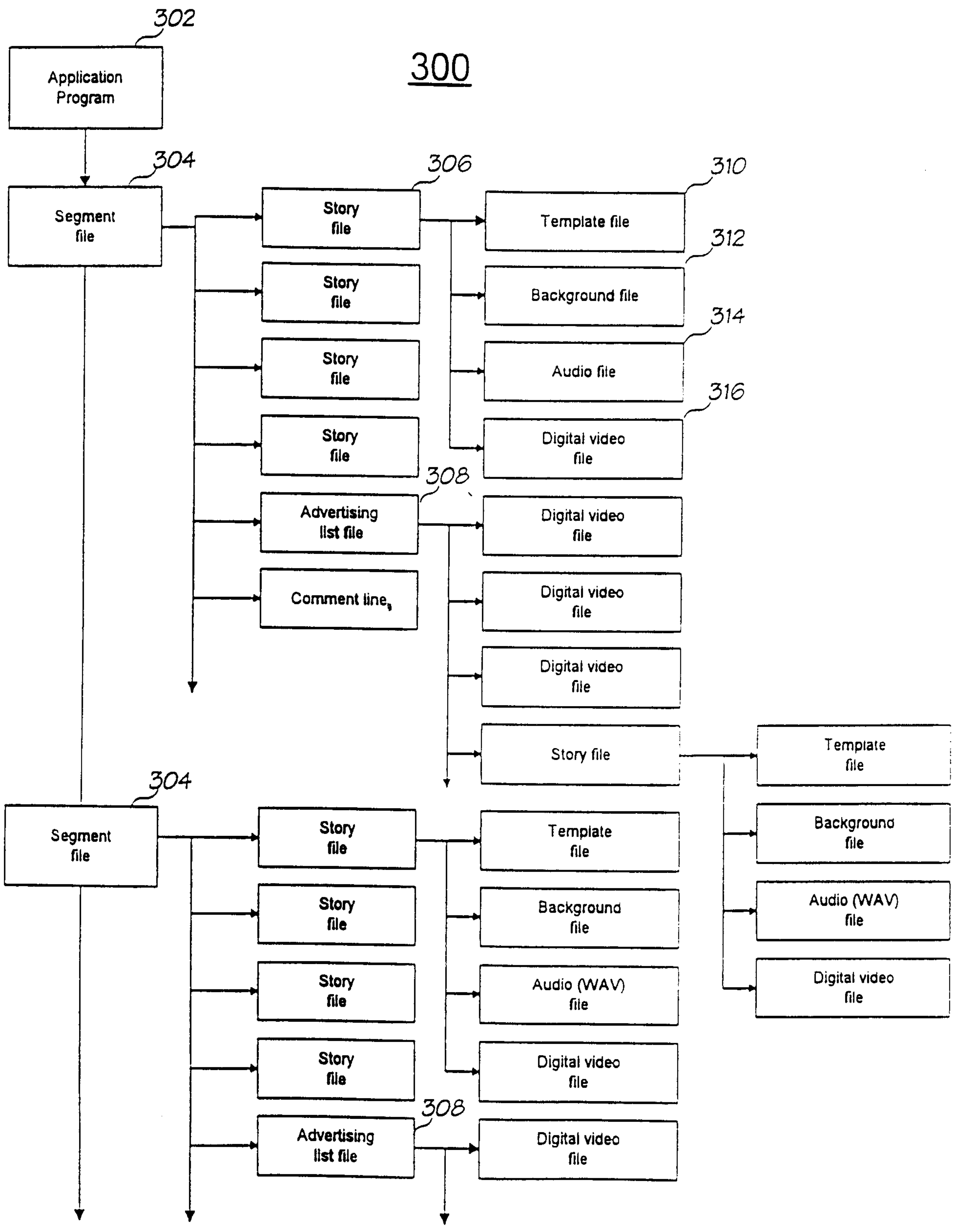

300
302
Application Program
304
Segment file
306
Story file
Story file
Story file
Story file
308
Advertising list file
Comment line,
310
Template file
312
Background file
314
Audio file
316
Digital video file
Digital video file
Digital video file
Digital video file
Story file
Template file
Background file
Audio (WAV) file
Digital video file
304
Segment file
Story file
Story file
Story file
Story file
308
Advertising list file
Template file
Background file
Audio (WAV) file
Digital video file
Digital video file

Fig. 3

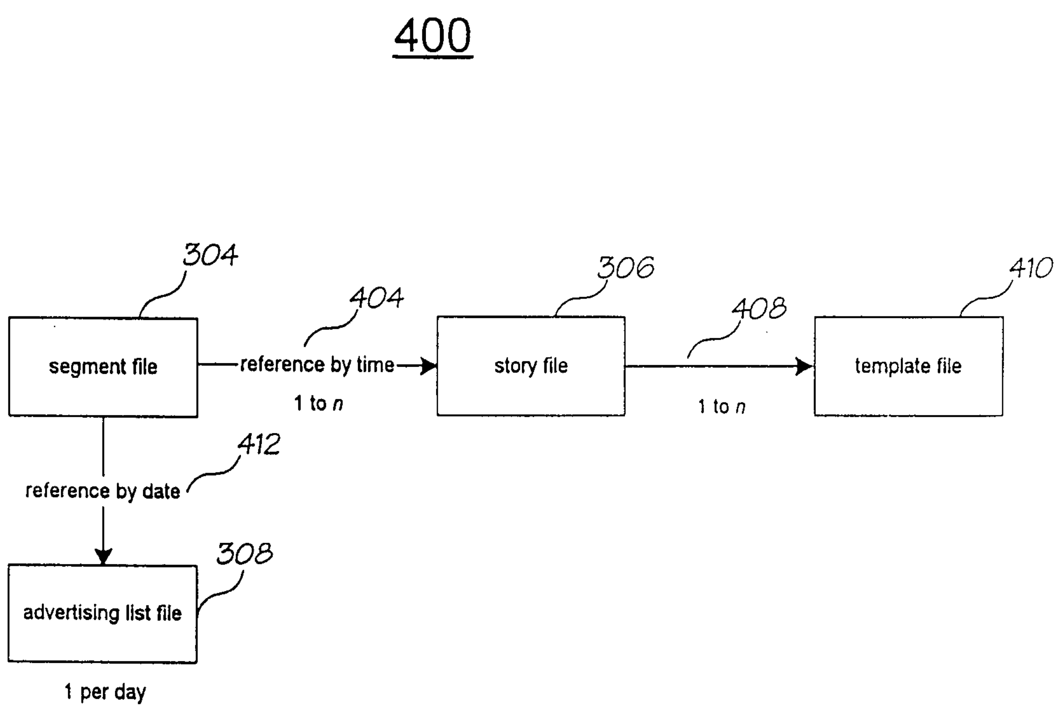

400
304
segment file
404
reference by time
1 to n
306
story file
408
1 to n
410
template file
412
reference by date
308
advertising list file
1 per day

Fig. 4

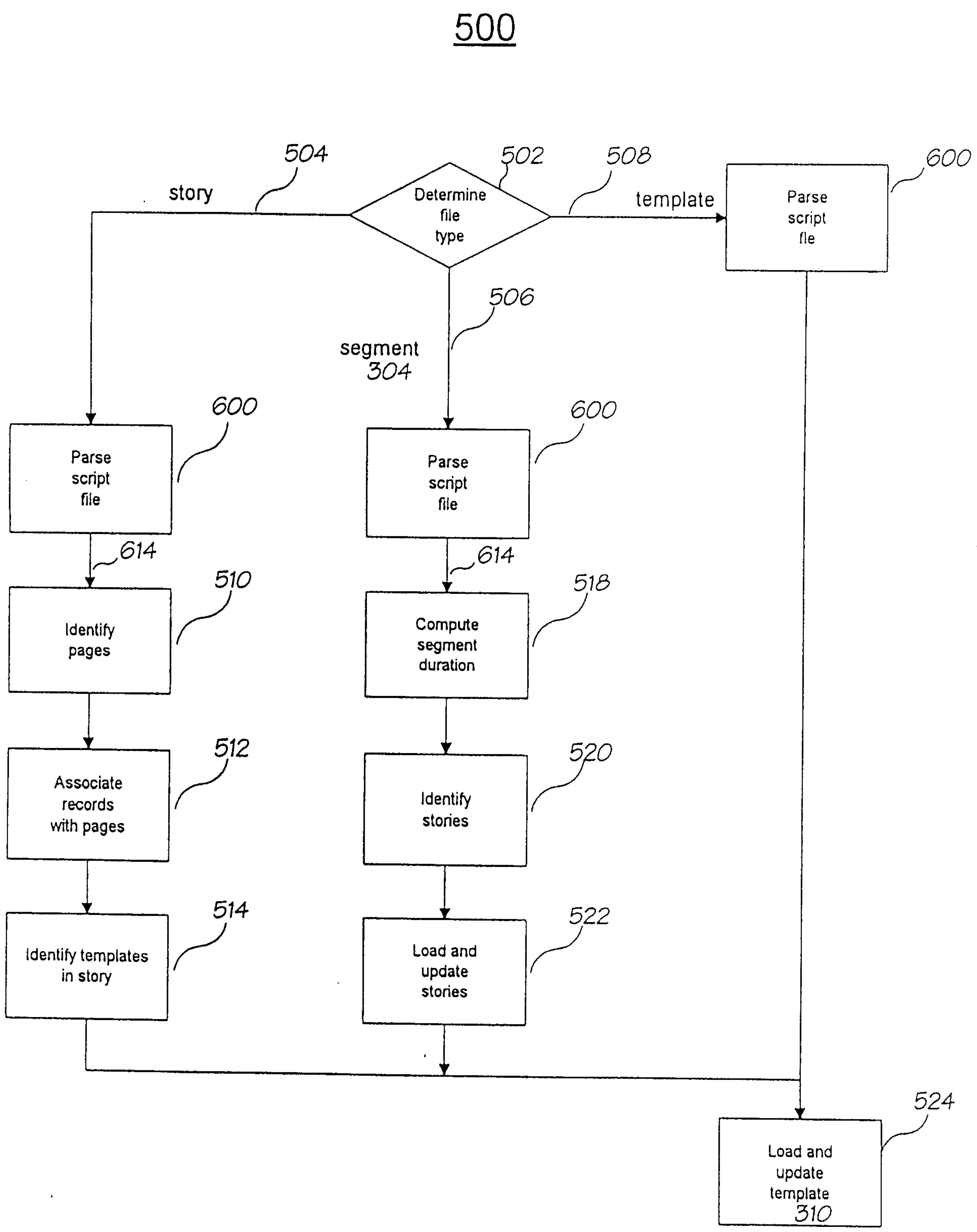

500
Determine file type
502
story
504
template
508
segment 304
506
Parse script fle
600
Parse script file
600
Parse script file
600
614
Identify pages
510
Associate records with pages
512
Identify templates in story
514
614
Compute segment duration
518
Identify stories
520
Load and update stories
522
Load and update template 310
524

Fig. 5

600

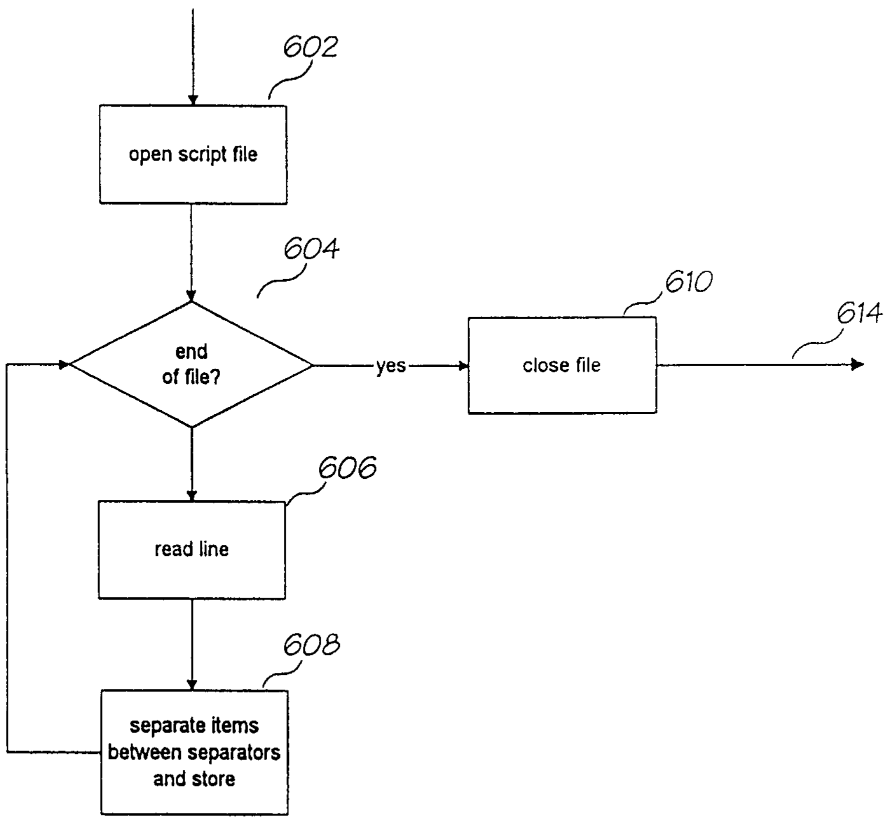

602
open script file
604
end of file?
yes
610
close file
614
606
read line
608
separate items between separators and store

Fig. 6

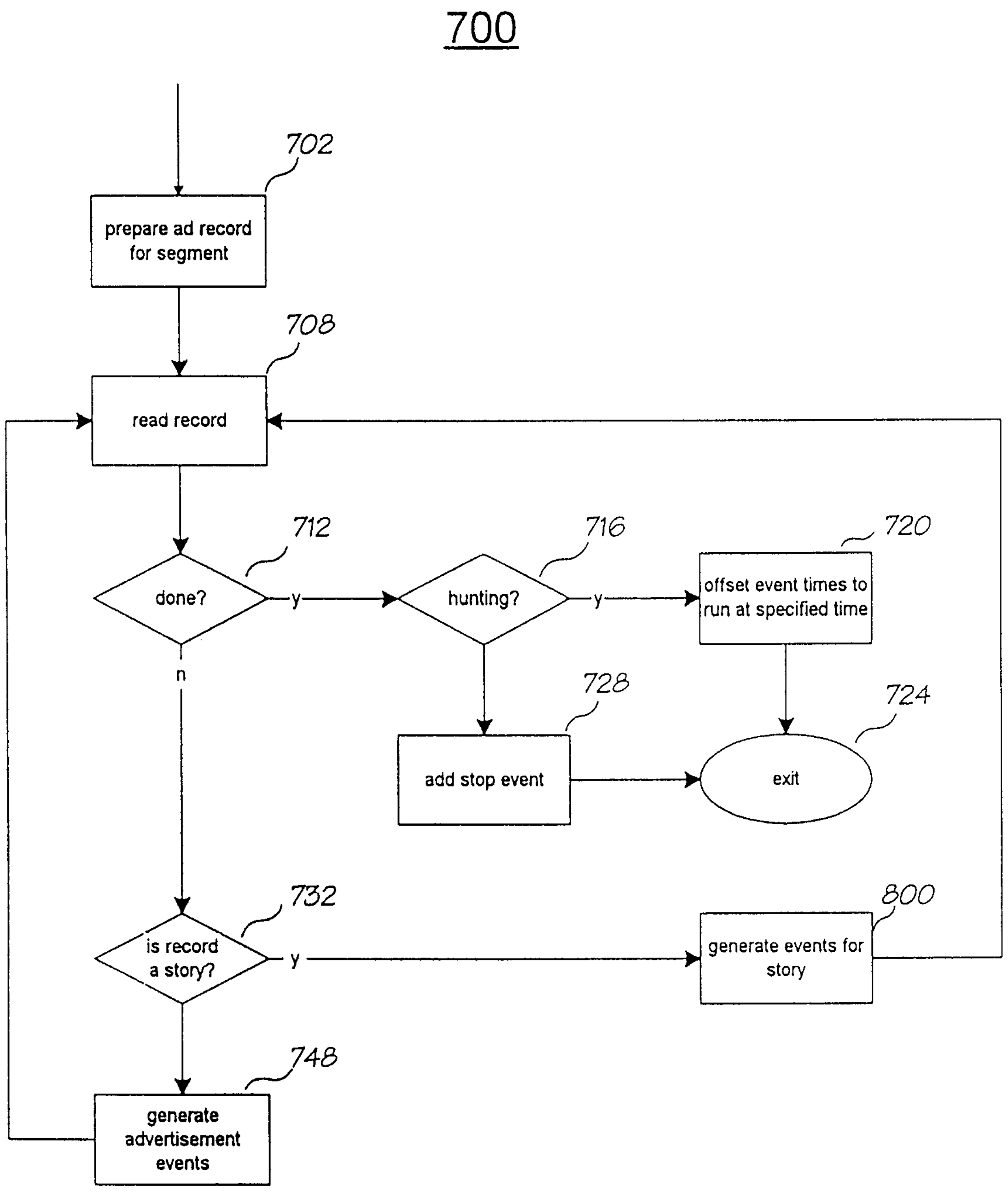

700
702
prepare ad record for segment
708
read record
712
done?
y
n
716
hunting?
y
720
offset event times to run at specified time
728
add stop event
724
exit
732
is record a story?
y
800
generate events for story
748
generate advertisement events

Fig. 7

800

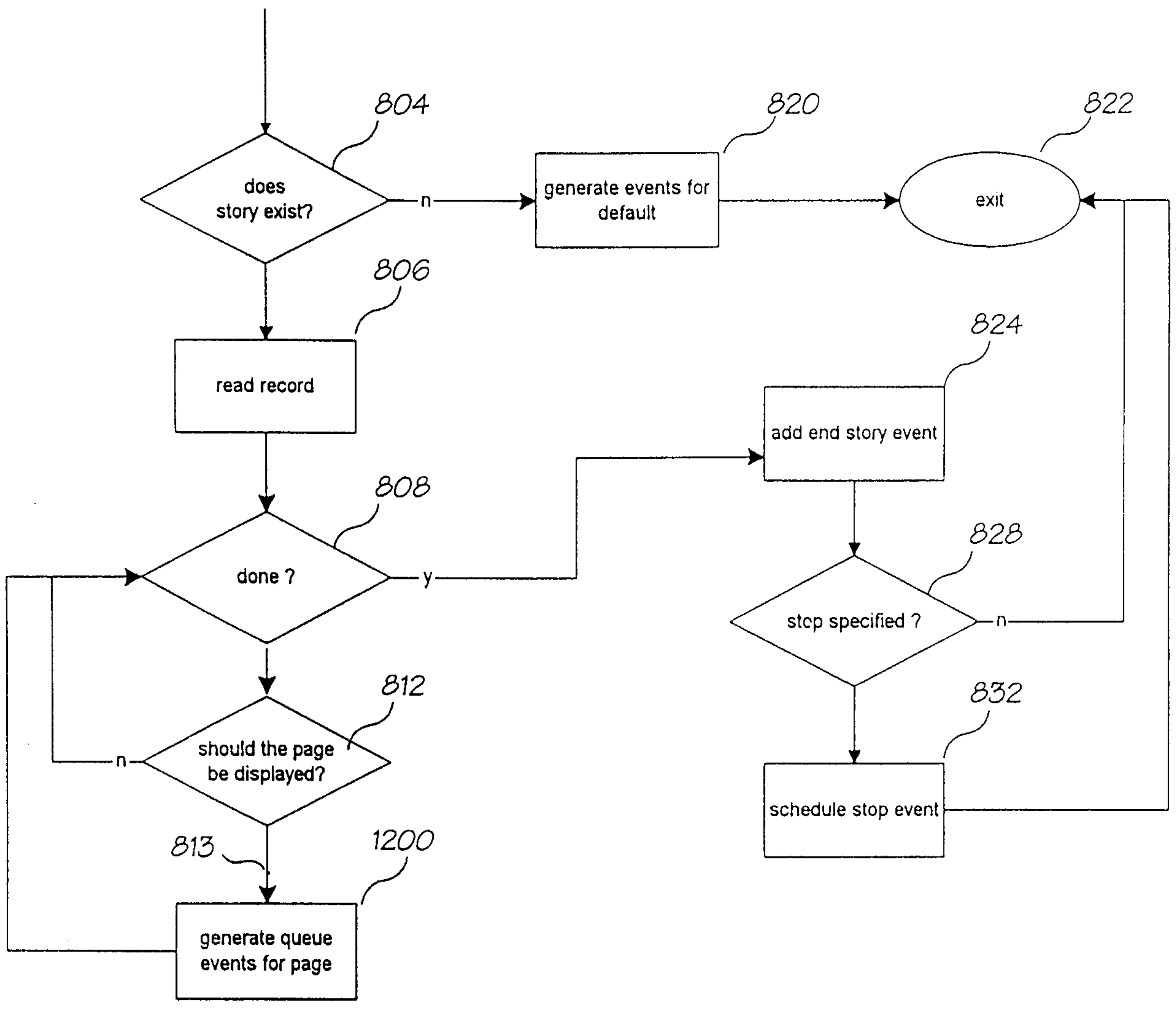

804
does story exist?
n
820
generate events for default
822
exit
806
read record
824
add end story event
808
done ?
y
828
stop specified ?
n
812
should the page be displayed?
n
832
schedule stop event
813
1200
generate queue events for page

Fig. 8

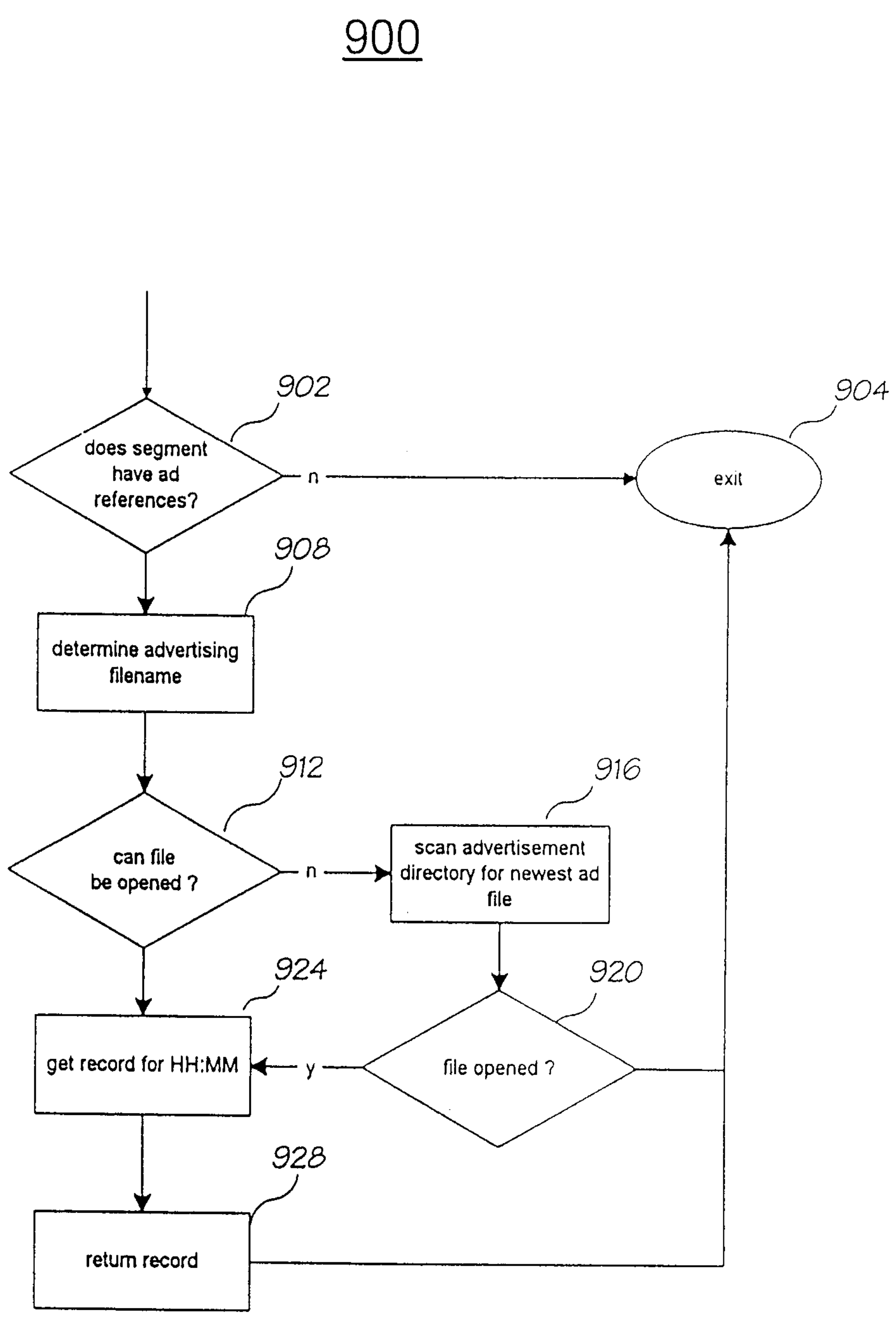

900
902
does segment have ad references?
n
904
exit
908
determine advertising filename
912
can file be opened ?
n
916
scan advertisement directory for newest ad file
920
file opened ?
y
924
get record for HH:MM
928
return record

Fig. 9

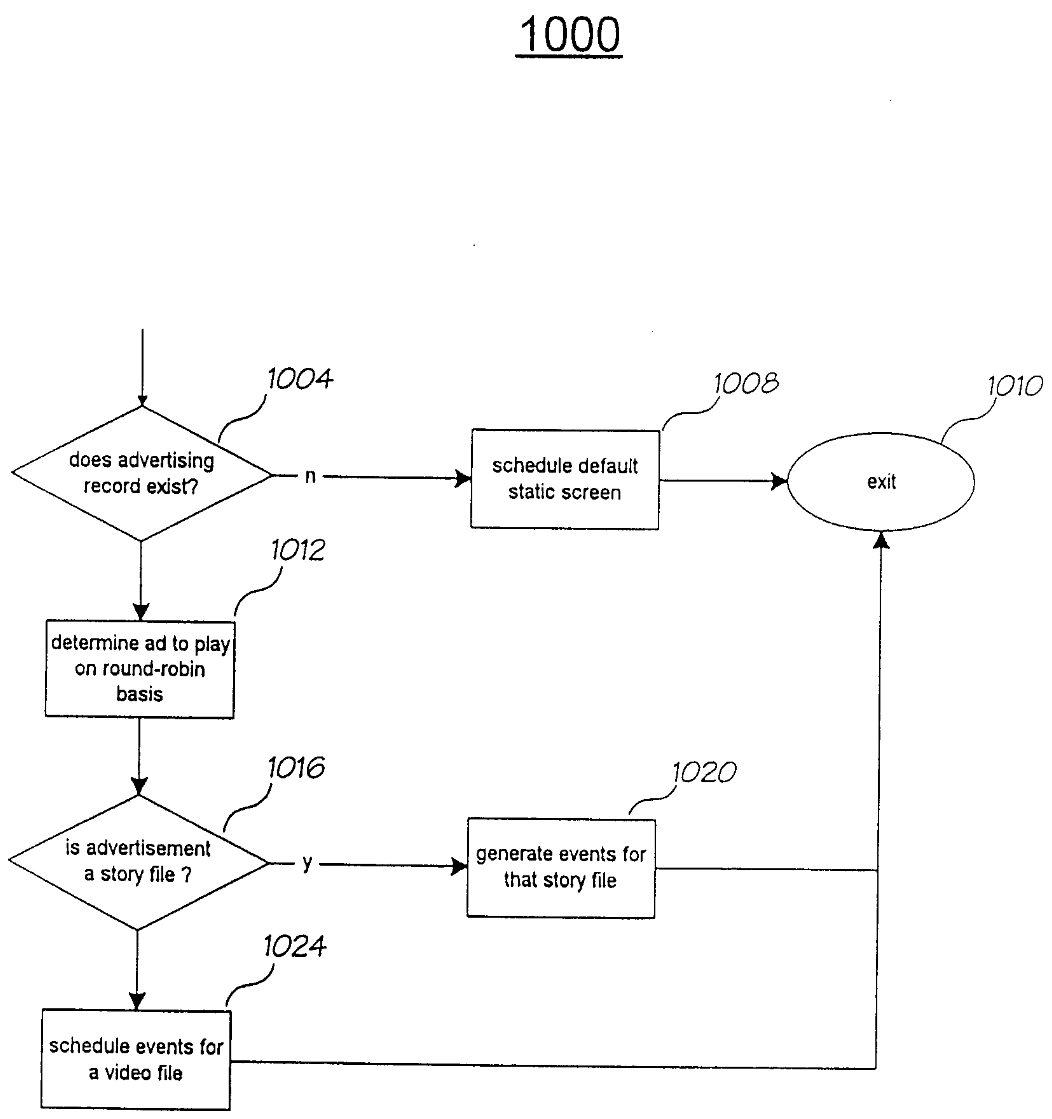

1000
1004
does advertising record exist?
n
1008
schedule default static screen
1010
exit
1012
determine ad to play on round-robin basis
1016
is advertisement a story file ?
y
1020
generate events for that story file
1024
schedule events for a video file

Fig. 10

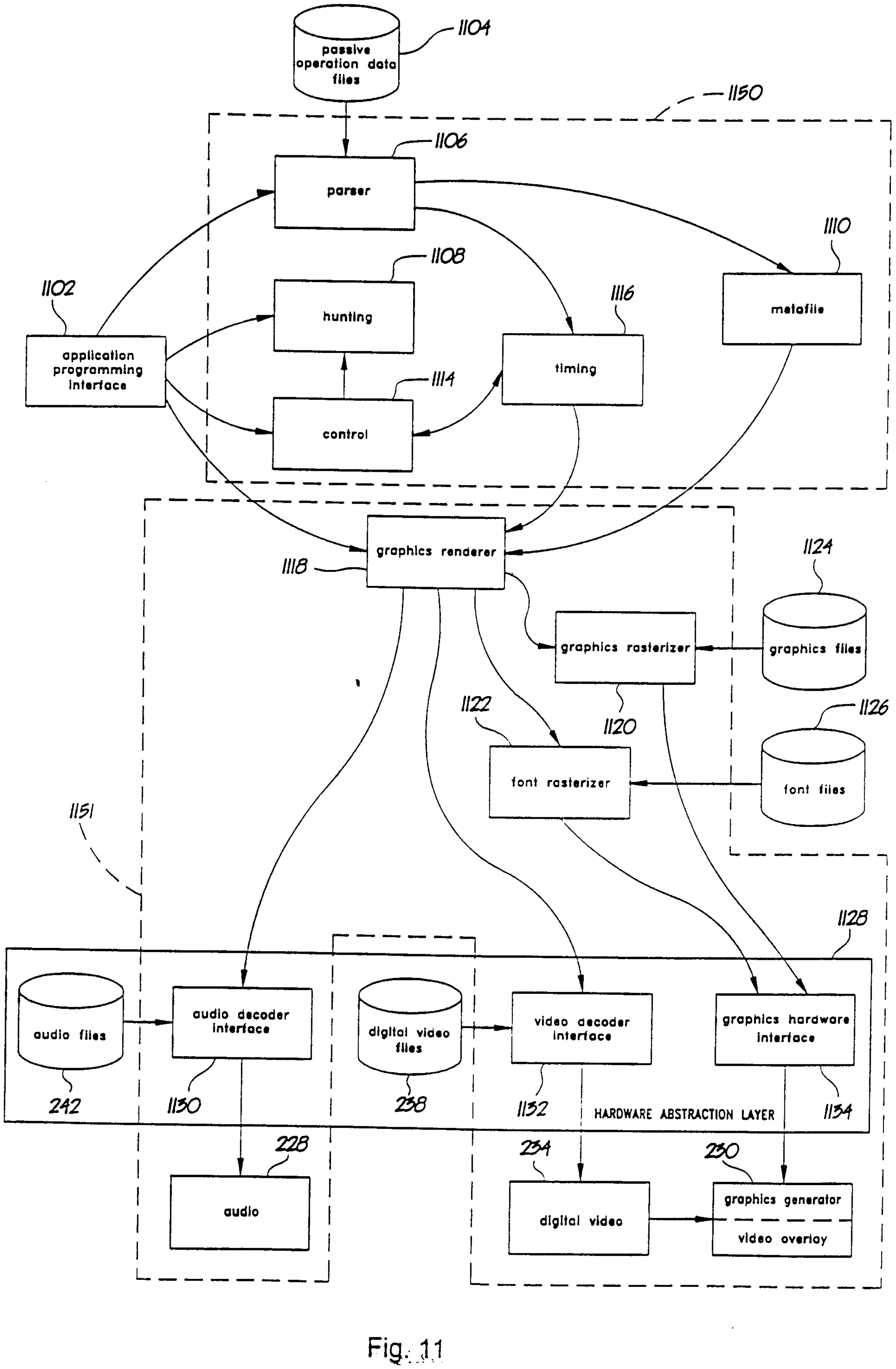

passive operation data files
1104
1150
1106
parser
1108
1110
1102
hunting
1116
metafile
application programming interface
timing
1114
control
graphics renderer
1118
1124
graphics rasterizer
graphics files
1122
1120
1126
font rasterizer
font files
1151
1128
audio files
audio decoder interface
digital video files
video decoder interface
graphics hardware interface
242
1130
238
1132
HARDWARE ABSTRACTION LAYER
1134
228
234
230
audio
digital video
graphics generator
video overlay

Fig. 11

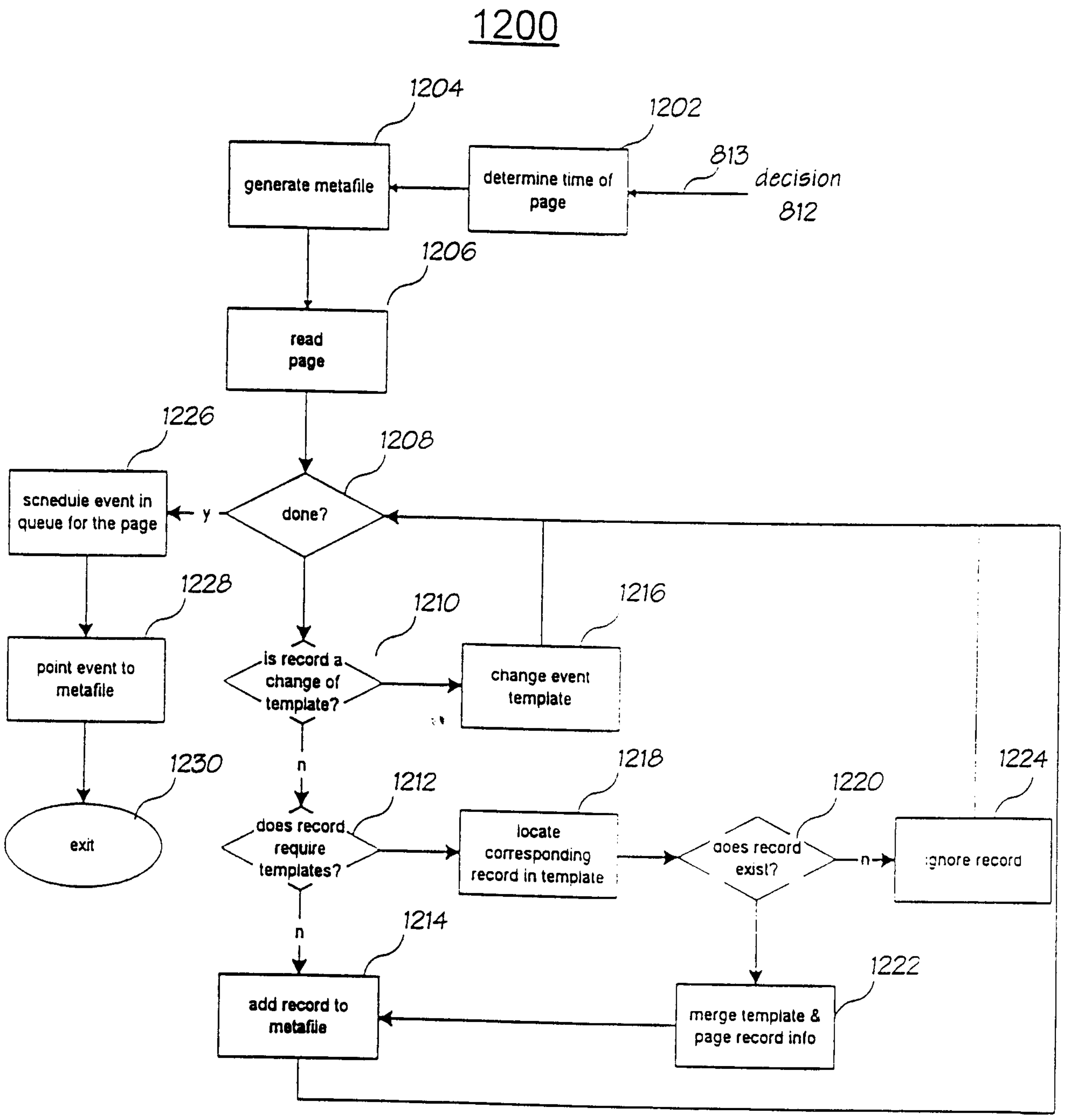

1200
1204
1202
generate metafile
determine time of page
813
decision 812
1206
read page
1226
1208
schedule event in queue for the page
y
done?
1228
1210
1216
point event to metafile
is record a change of template?
change event template
n
1230
1212
1218
1220
1224
exit
does record require templates?
locate corresponding record in template
does record exist?
n
ignore record
n
1214
1222
add record to metafile
merge template & page record info

Fig. 12

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US95/13433

A. CLASSIFICATION OF SUBJECT MATTER
IPC(6) :G06T 13/00
US CL : 395/152, 154; 370/70; 348/6, 13
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. : 395/152, 154; 370/70; 348/6, 13

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)

APS Search on US Patent Database

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X -- Y	US, A, 5,319,455 (HOARTY ET AL) 07 June 1994, col.2, line 50-57, col.3, line 53, col.4, line 16, col.7, line 31-68, col.8, line 6-31, col.17, line 54-60.	1-16, 18, 19, 22-27 -------------- 17, 20, 21, 28
A	US, A, 5,027,400 (BAJI ET AL) 25 June 1991, Figures 9, 11	1-28
Y	Computer Dictionary, Microsoft Press, 2nd Edition, 1993, page 24	20, 21
A, P	US, A, 5,381,412 (OTANI) 10 January 1995	1-28
A, P	US, A, 5,434,678 (ABECASSIS) 18 July 1995	1-28

[X] 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 part of particular relevance		
"E"	earlier document published on or after the international filing date	"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
"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)	"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
"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	"&"	document member of the same patent family

Date of the actual completion of the international search 16 JANUARY 1996	Date of mailing of the international search report 21 MAR 1996
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 STEPHEN HONG Joni Hill Telephone No. (703) 308-5465

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US95/13433

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y, P	US, A, 5,440,336 (BUHRO ET AL) 08 August 1995, see entire document	17, 28
A	US, A, 5,318,450 (CARVER) 07 June 1994	1-28
A	US, A, 5,325,423 (LEWIS) 28 June 1994	1-28