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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
2 May 2008 (02.05.2008)

PCT

(10) International Publication Number
WO 2008/052050 A2

(51) International Patent Classification:
H01R 13/58 (2006.01)

(21) International Application Number:
PCT/US2007/082373

(22) International Filing Date: 24 October 2007 (24.10.2007)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/867,763 24 October 2006 (24.10.2006) US
60/886,283 23 January 2007 (23.01.2007) US

(71) Applicant *(for all designated States except US)*: **SLACKER, INC.** [US/US]; 16935 W. Bernardo Dr., Suite 101, San Diego, California 92127 (US).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: **KINDIG, Bradley, D.** [US/US]; 9325 Laurentian Dr., San Diego, California 92129 (US).

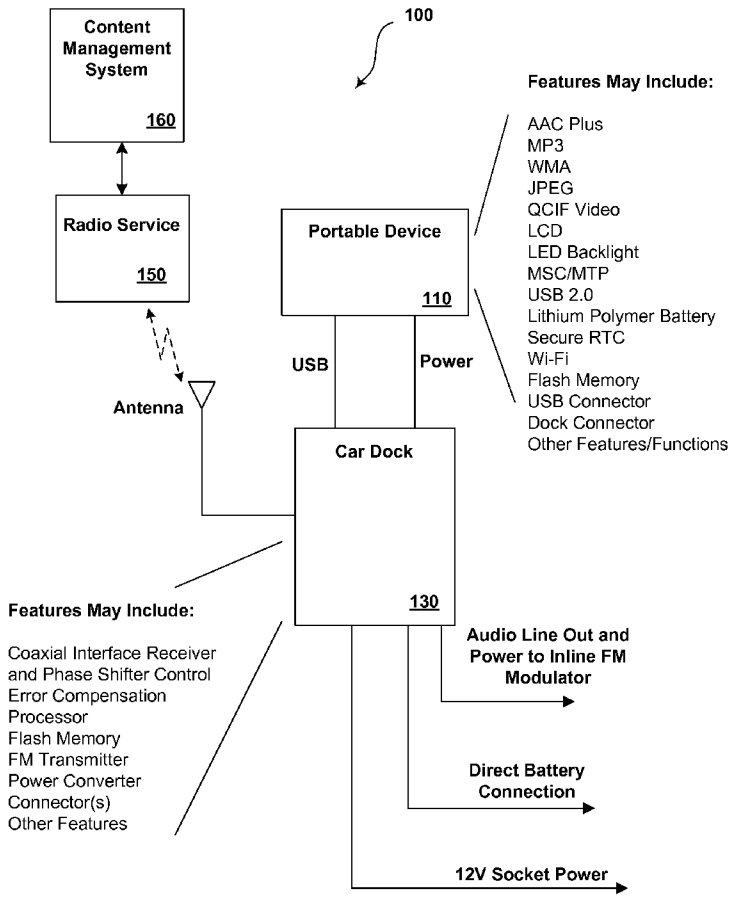
(74) Agents: **TIETSWORTH, Steven, C.** et al.; COOLEY GODWARD KRONISH LLP, Attn: PATENT GROUP, 777 6th Street, NW, Suite 1100, Washington, District Of Columbia 20001 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH,

[Continued on next page]

(54) Title: METHOD AND DEVICE FOR PLAYBACK OF DIGITAL MEDIA CONTENT



(57) Abstract: Systems and methods are disclosed for providing devices capable of the reception and playback of one or more crafted media channels transmitted over an intermittent connection. The one or more channels may be customized by the end user to bias the programming towards individual tastes or moods. The channels may be delivered over an intermittent, unidirectional link, for example a Satellite Radio receiver in a home or car. Content may also be delivered over an intermittent, bidirectional link, for example to a portable media player via a wireless 802.11 or other networked connection.

WO 2008/052050 A2

GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report*

METHOD AND DEVICE FOR PLAYBACK OF DIGITAL MEDIA CONTENT

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit of priority to United States Provisional Patent Application Serial No. 60/867,763, entitled "Method and Device for Playback of Locally-Stored Digital Media Content," filed on October 24, 2006, and to United States Provisional Patent Application Serial No. 60/886,283, entitled "Devices and Methods for Distributing Digital Content," filed on January 23, 2007. This application is also related to United States Patent Application Serial No. 11/637,300, entitled "Method and Apparatus for Distribution of Digital Content," filed on December 12, 2006.

FIELD OF THE INVENTION

[0002] This invention relates generally to methods, systems, and devices for receiving, storing and playing back digital media files and, more particularly, to methods, systems, and devices capable of acquiring digital media files over wired and wireless communications links that may be intermittently connected and potentially subject to noise, interference, or other forms of connectivity disruption.

BACKGROUND

[0003] Traditional AM, FM, television, and satellite radio receivers are configured to receive real-time broadcasts: i.e., the broadcast is rendered for playback contiguously with the reception of the signal. With the introduction of recording capabilities such as audio tape decks and VCRs, users were enabled to record broadcast content and play it back at their leisure.

[0004] More recently, the introduction of recording devices such as digital video recorders (DVRs) allows consumers to more conveniently "time shift" programming. These devices allow simultaneous recording and playback, thus enabling features such as the simulation of pausing, rewinding, or fast-forwarding of live television.

[0005] Portable media players and media-enabled phones are able to load content from personal computers over wired connections. A few media players and certain phones are able to load content over wireless networks. A few portable media players have connectivity to satellite radio systems. At least one such portable player also allows the user to record and store content from the satellite broadcast as if the device contained an

integrated tape deck. Unfortunately, traditional broadcast mediums such as television, radio, and even satellite radio tend to offer minimal opportunities for user customization and interactivity. While the number of channels continues to increase in each of these mediums, the mediums themselves fail to accommodate even simple customizations such as "Classic Rock with extra U2, and no Supertramp."

[0006] Moreover, existing portable music players also fail to address the needs of many users that want a professionally programmed listening experience akin to traditional radio, but want their user specific preferences adhered to. With these existing devices, the users must acquire their own content and program their own channels at substantial time, effort, and cost. In addition, existing media players capable of receiving broadcast transmissions, such as satellite radio receivers used in cars and other portable devices, can be frustrating to use in that temporary losses in signal result in interruption of the listening experience.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] For a better understanding of the nature and objects of various embodiments of the invention, reference should be made to the following detailed description taken in conjunction with the accompanying drawings, wherein:

[0008] FIG. 1 illustrates an embodiment of a portable device and associated system in a mobile application in accordance with aspects of the present invention.

[0009] FIG. 2 illustrates an embodiment of a portable device and associated system in a residential or office application in accordance with aspects of the present invention.

[0010] FIG. 3 illustrates an embodiment of a portable device and associated system in a mobile application in accordance with aspects of the present invention.

[0011] FIG. 4 illustrates an embodiment of an object configuration in accordance with aspects of the present invention.

[0012] FIG. 5 is a state diagram showing an embodiment of a content playback implementation in accordance with aspects of the present invention.

[0013] FIG. 6 is a state diagram showing an embodiment of a user interface implementation in accordance with aspects of the present invention.

[0014] FIG. 7 is a state diagram showing an embodiment of a wireless connection implementation in accordance with aspects of the present invention.

[0015] FIG. 8 is a state diagram showing an embodiment of content processing in a unidirectional connection in accordance with aspects of the present invention.

[0016] FIG. 9 is a flowchart showing an embodiment of a start station process in accordance with aspects of the present invention.

[0017] FIG. 10 is a flowchart showing an embodiment of a stop station process in accordance with aspects of the present invention.

[0018] FIG. 11 is a flowchart showing an embodiment of a skip logic process in accordance with aspects of the present invention.

[0019] FIG. 12 is a flowchart showing an embodiment of a next track process in accordance with aspects of the present invention.

[0020] FIG. 13 is a flowchart showing an embodiment of an add to sequence process in accordance with aspects of the present invention.

[0021] FIG. 14 is a flowchart showing an embodiment of a select track process in accordance with aspects of the present invention.

[0022] FIG. 15 is a flowchart showing an embodiment of a playback scoring process in accordance with aspects of the present invention.

[0023] FIG. 16 is a graph of song aging results for an 80's station based on an embodiment of the present invention.

[0024] FIG. 17 is a graph of song aging results for a Classic Rock station based on an embodiment of the present invention.

[0025] FIG. 18 is a graph of song aging results for an Alternative station based on an embodiment of the present invention.

[0026] FIG. 19 is a graph of track sequence scoring results as a function of rest for an embodiment of the present invention.

[0027] FIG. 20 illustrates an allocation of inventory pools for media content storage for an embodiment of the present invention.

[0028] FIG. 21 is a flowchart showing an embodiment of an add track process in accordance with aspects of the present invention.

[0029] FIG. 22 is a flowchart showing an embodiment of an inventory cull process in accordance with aspects of the present invention.

[0030] FIG. 23 is a graph of duty cycle adaptation for various types of stations based on an embodiment of the present invention.

[0031] FIG. 24 is a flowchart showing an embodiment of a delete tracks process in accordance with aspects of the present invention.

[0032] FIG. 25 is a flowchart showing an embodiment of a scan playlists process in accordance with aspects of the present invention.

[0033] FIG. 26 is a flowchart showing an embodiment of a save tracks process in accordance with aspects of the present invention.

[0034] FIG. 27 is a table illustrating inventory management symbols and description based on an embodiment of the present invention.

[0035] FIG. 28 is a flowchart showing an embodiment of a ban tracks process in accordance with aspects of the present invention.

DETAILED DESCRIPTION OF EMBODIMENTS OF THE INVENTION

Introduction

[0036] Embodiments of a digital media content reception and playback device that overcomes many of the shortcomings described previously, as well as others, is disclosed herein. In one or more embodiments, the digital media content reception and playback device (hereinafter also denoted for brevity as a "device" or a "player") is capable of receiving content over intermittent connections and organizing and sequencing the content into a program that leaves a listener with the impression that traditional programming is being delivered, but which nonetheless accommodates the listener's specific preferences and may be customized and tailored dynamically to user preferences. Content may be created, managed, and provided by a content management system including one or more servers configured to receive user information as well as content and content requests, store content, manage content, customize content, distribute content via wired or wireless channels, and provide other related management and distribution functions. Such a content management system and any associated servers or other components is also described herein simply as the "server" for purposes of brevity.

[0037] In one embodiment, the device leverages the expertise of professional content programmers that acquire and filter a broad base of relevant content and define the rules that determine the mix and sequencing of content to be played on the device. Thus, the user is freed from the difficult task of acquiring content and creating playlists.

[0038] In typical embodiments the device will be configured to filter incoming content for transmission errors and discard content that contains artifacts that would be

perceived by the user. Thus the ability to render pristine content is preserved even when the device is in areas with poor or no reception.

[0039] In one or more embodiments the device comprises a personal digital audio player for use in and out of automotive or other mobile environments. While this embodiment is described hereinafter in terms of audio playback, it will be apparent to those skilled in the art that the invention is equally applicable to photos, graphics, images, video or other types of multimedia content.

[0040] In some embodiments the device may be configured to dock in an automobile or other vehicle. When the device is docked in an automobile, it may be connected to a satellite antenna that is capable of receiving a unidirectional broadcast of audio content and data.

[0041] Additionally, the device may be configured with built-in wired or wireless capabilities using networks such as those based on IEEE standard 802.11, or other local or wide area networks capable of providing connection to wireless networks. Those skilled in the art will readily appreciate that the invention is not constrained to any particular type of network connection or client server configuration.

[0042] In some embodiments the device may be configured to dock or otherwise connect with home or office based systems such as personal computers or other devices capable of networking, or with home entertainment or similar systems.

[0043] Additional aspects of the invention are also contemplated as further described and illustrated herein.

Operational Overview

Operation with Unidirectional Connectivity

[0044] In typical unidirectional embodiments, when the device is operating with a unidirectional connection as it typically is when receiving content over a satellite link, the device receives content that is broadcast on the connection, with content typically originating from a server or servers that are part of a content management system. In accordance with one aspect of the invention, the device compares the content against one or more station profiles and decides whether to keep the content or whether to discard it. In essence, the device decides if the content is of interest to its user, based on a set of user preferences, and stores the content of most interest for playback. One aspect of the present invention relates to systems and methods for implementing decision processes for selecting, storing, and playing

content and is described in detail in the sections that follow in relation to Inventory Management.

[0045] The process of receiving, analyzing, and caching or discarding content typically takes place independently of device playback. The device will continuously receive and process incoming content so long as it is sufficiently powered and has an operational connection. In typical embodiments, the device is configured to be capable of simultaneously playing back content from the cache while receiving and processing new content into the cache.

Operation with Bidirectional Connection

[0046] In typical bidirectional embodiments, when connected to an associated content broadcasting service with a bidirectional connection, the device is configured so that it can transmit its preferences to a content management system so that only content of interest is provided and downloaded to the device. This is done by either fully or partially offloading the Inventory Management function to the server side of the connection (where the device is part of the client side).

[0047] In a preferred embodiment, the device transmits its station profiles to the server and receives a prioritized list of content to download. The device downloads the content in priority order. The list is prioritized so as to maximize the improvement to the device's inventory should the connection be lost before all tracks are downloaded.

Playback Operations

[0048] In typical embodiments, whether or not connected, the device can play the content in its cache. The cached content may be organized such that the device's algorithms can efficiently generate playlists that approximate the crafted song sequences created by professional radio programmers. Professional radio programmers may be used to support the device population by categorizing the content transmitted to the device into stations and station categories. The radio programmers may also specify the rules used by the device to sequence the content for a given station. This may be done in conjunction with the content management system, with the information being input, stored, processed, and output by one or more servers or other data storage and distribution elements.

[0049] The user may then access these sequences on the device by selecting the station by name or preset button, similar to the way a user selects a station on a traditional car

stereo or satellite radio receiver. In addition, however, the user may also be provided with options and selections to customize the station and interact with playback order in a way that is not possible using traditional broadcast receivers. Embodiments of these processes are further described and illustrated below.

Playback Features

[0050] Because the device may be configured to select the next song for playback independently of the broadcast stream, each device can customize playback order according to the preferences of its owner or user. For example, some of the unique features of the device that are not possible with traditional radio receivers may include:

- Ability to skip content. If the user does not like the current song, they may skip to the next song on the station.
- Ability to replay content. The user may skip backwards to replay songs.
- Ability to pause content. The user may pause the currently playing track.
- Random access to content. The user may seek forward or backward through the content.
- Ability to ban content. If the user does not like a particular song, artist, or other content characteristic (composer, commentator, etc.) they may ban the content from the device. Content matching the banned characteristics will no longer be played on the device.
- Ability to save favorites. If the user hears a song they like, they may save it as a favorite. Favorites can be stored in a special folder or playlist for future access.
- Ability to bias station playback towards certain content. The user may adjust station knobs or "sliders" to bias the track selection towards songs with certain attributes. Examples of station slider controls include (but are not limited to):
    - Newer ←→ Older
    - Harder ←→ Softer
    - Popular ←→ Obscure
    - Favorite ← → Non-Favorite
- Ability to create user tailored stations. While most stations can be created by professional radio programmers, users that are so inclined can create their own stations using these same tools or variants thereof.
- Very large station catalog. While a limited number of stations are supported through the unidirectional channel, users that update their players via bidirectional channels can have access to a very large catalog of stations. Potentially, this catalog can contain stations created by the user community as well as stations created by programming professionals. This can also include content published by Really Simple Syndication (RSS) including audio-blogs or "podcasts."
- Other unique features and functions are also contemplated as further described herein.

Physical Architecture

[0051] Attention is now directed to FIGS. 1-3, which depict the physical architecture of embodiments of aspects of the invention in a variety of contexts. Specifically, FIGS. 1 and 2 provide representations of embodiments of systems in accordance with the invention in cases where the digital media playback device is docked within an automobile and in a home location, respectively. For example, as shown in FIG. 1, one embodiment of a system 100 may include a portable device 110 connected to a car dock 130, wherein the car dock 130 may include the capability of receiving and/or sending content to a content management system 160 through a radio service 150. This connection will typically be via a wireless connection using systems based on satellite communication and/or wireless networks such as those based on IEEE 802.11 standards (Wi-Fi) or other local or wide area networking standards. As shown in FIG. 1, the docking may include standard interfaces between the portable device 110 and the dock 130 such as USB, or may be based on proprietary interfaces.

[0052] FIG. 2 illustrates one embodiment of a system 200 configured for home, office, or business use of portable device 110. System 200 may include connection through a home dock 230 and/or via other connection methods such as through a home or office networking device 220, a PC 250, home A/V equipment 260, or via other connections configured to send information and receive content from a content management system 160.

[0053] FIG. 3 illustrates one embodiment of a system 300 wherein portable device 110 is used as a mobile or portable device in conjunction with a wireless network connection 220, such as a router or other networking device based on IEEE standard 802.11 or other local or wide area networking standards. A typical portable system 300 will include a listening device (headphone, ear buds, etc.) 310 allowing the user to move about and isolate audio content from the environment if desired.

[0054] Those skilled in the art will appreciate that it is possible to realize a wide range of additional embodiments of systems in accordance with the invention using a variety of hardware platforms as well as hardware, software, and network configurations.

Software / Firmware Architecture

[0055] Aspects of the present invention are related to playback and content management on the device. Before describing embodiments of algorithms used for playback sequencing and inventory management, a set of core objects used in a preferred embodiment are described below and illustrated in FIG. 4. It is noted that the object-oriented architecture described below is not unique and other objects and associated object configurations may also be used in keeping within the spirit and scope of the invention.

[0056] ***PlayerContext*** - The parent object that implements the overall player behavior.

[0057] ***Station*** - As described herein, a Station is a content channel analogous to a terrestrial radio station. The user may store one or more favorite stations as 'presets' on the device to optimize their ability to access the channel. The user will have access to the broader catalog of all available Stations through the device. The Station holds the settings / customizations that are used to bias the content cached or played on the station (sliders) as well as the engine used to sequence the Station's content (Clock).

[0058] ***StationProfile*** – As described herein, a StationProfile is a definition of the rules and settings used to assemble a station. A set of StationProfiles is loaded onto the device. Some settings in the StationProfile may be customized by the device user. Such customization may involve, for example, setting sliders in accordance with user preferences. In one embodiment the station profile includes the current biases (as set by sliders) towards selecting content based on Popularity, Energy (hard/smooth), Favorites, and Age. Providing a setting to control whether or not certain types of content will be permitted to be played enables an alternate form of customization to be effected.

[0059] StationProfiles for a core set of stations are typically created and maintained by professional radio programmers. These station profiles may be transmitted on either the bidirectional or unidirectional links to update the profiles in the device from time to time.

[0060] Optionally, users may create their own custom station profiles (typically using an application external to the device, such as a web based application, but devices may also be configured to allow users to create simple stations on the device through user interfaces). These profiles would typically be loaded and provisioned through the bidirectional connection. A StationProfile includes the definition of the Clock and the Buckets that comprise the Clock that will be used on the device.

[0061] The StationProfile may be thought of as a set of rules or instructions for creating a station or a serialization of the Station and its associated objects. In one embodiment, the StationProfile is an XML document. In an alternative embodiment, the StationProfile is a more compact binary format with a corresponding parsing schema. Other embodiments of StationProfiles based on any of a variety of data formatting and storage methods may also be used.

[0062] ***Clock*** **-** As described herein, a Clock is utilized in sequencing content. In one embodiment content for a station is categorized. For example, songs that are the current hits and are getting the most airplay might be categorized as 'Current' songs. Content that was formerly a Current song, but is fading in popularity might be categorized as 'Recurrent.' Content that is introduced for variety might be categorized as 'Library.'

[0063] The Clock is an ordered list of these categories (called Buckets) used to create a song sequence. To create a sequence, a song is chosen from the first Bucket in the Clock, followed by the second Bucket and so on until the last Bucket is reached, whereupon sequencing continues by returning to the first Bucket. In practice, the creator of a Station may define as many Buckets as desired and sequence them as they see fit.

[0064] ***Bucket*** – As described herein, a Bucket is a category of content used in Clock sequencing as described above. In typical embodiments two types of buckets, SongBuckets and RuleBuckets are utilized. SongBuckets are buckets for which songs are explicitly categorized. That is, a BucketId attribute is associated with the song (i.e., by a content programmer) that categorizes the song explicitly into the Bucket with that matching identifier.

[0065] RuleBuckets categorize content by scoring one or more attributes of the song against the Bucket's rules. For example, a RuleBucket might select content that is 'older than 1975' and 'in the Hard Rock genre.'

[0066] A further specialization of the SongBucket is the HeavyRotation bucket. A HeavyRotation bucket is a bucket whose songs are designed to repeat at a given rate. Most buckets will implement logic to avoid the repetition of songs to ensure variety. HeavyRotation buckets contain the 'new, hot' songs that are desirable to repeat so long as they are still 'new and hot.'

[0067] ***Header*** **-** As described herein, a Header is content metadata that is used to describe the attributes of the content/song. A Bucket manages its inventory by maintaining a

list of Headers. Set forth below is a table containing the type of information included within an exemplary Header of a given media file.

SOF	0xAA55AA55	4 byte
Number of Frames	Number of frames comprising the music file and including the header(s)	2 byte
Data bytes in last frame	Number of data bytes in the last frame	1 byte
Opcode	0x0 – Data	1 byte
Version	The version number of this header format	1 byte
Format Flags	0x01 = Display as Classical Track 0x02 = Multipart 0x04 = Can Overlay Ads 0x08 = Can Overlay Information	1 byte
SongId	The Song Id used for song separation rules	4 byte unsigned integer
PerformanceId	The Id of the performance used for content management	4 byte unsigned integer
LicensorId	The Id of the content owner (i.e. record label)	4 byte unsigned integer
Song Name Length	The length in bytes of the song name that follows	2 byte short integer
Song Name	UTF-8 encoded song name	Song Name Length bytes
ArtistId	The Id of the featured artist	4 byte unsigned integer
Artist Name Length	The length in bytes of the artist name that follows	2 byte short integer
Artist Name	UTF-8 encoded artist name	Artist Name Length bytes
Composer Name Length	The length in bytes of the composer name that follows	2 byte short integer
Composer Name	UTF-8 encoded composer name	Composer Name Length bytes
AlbumId	The Id of the album associated with the track	4 byte unsigned integer
Album Name Length	The length in bytes of the album name that follows	2 byte short integer
Album Name	UTF-8 encoded album name	Album Name Length bytes

Release Year	The year of the original release of the song	2 byte short integer
Popularity	Relative measure of song popularity	2 byte short integer`
Style Length	Number of Style,Style Fit pairs to follow	Byte
Style	Id of a style associated with this song	2 byte short integer
Style Fit	Relative strength of style association	2 byte short integer
	Additional Style,Style Fit pairs	
Bucket Length	Number of Bucket,Date pairs to follow	Byte
Bucket	Id of a radio programming bucket	2 byte short integer
Date	Date the song was programmed into the bucket as seconds since 1/1/1970	4 byte unsigned integer
	Additional Bucket,Date pairs	
Energy	-1 or 0-4 value representing energy level of the track	Byte
Multipart Length	Number of Offset, Type, Name triples to follow	Byte
Offset	Multipart offset	4 byte unsigned integer
Offset Type	0=Audio Chapter 1=Play Once Intro 2=Play Always Intro 3=Play Once Outtro 4=Play Always Outtro 5=Album Art [if not in mp4 container]	
Part Name Length	Length in bytes offset name to follow	2 byte short integer
Part Name	UTF-8 encoded part name (for audio chapters only)	
	Additional parts	
	Padding or Piggyback data to fill frame	
CRC	CRC checksum on the frame contents	4 bytes

Table 1 – Exemplary Header Information

[0068] ***Attribute*** – As described herein, an Attribute is a logical name/value pair embodying a fact or piece of metadata about the content. The Header stores Attributes. For efficiency of transmission and storage, attributes may be stored in a fixed layout in the Header or stored as Id,Value pairs or ClassId, Id, Value triples. In the latter case, ClassId

serves to specify the identifier namespace for Id so that multiple, overlapping identifier spaces can be used.

[0069] ***Rule*** – As described herein, Rules serve to combine Attributes and other derived information (for example, the last time the track was played) in order to form scores. Scores are ultimately used to determine playback order and priority during inventory management. Specializations of Rules include SequenceRules for scoring Headers based on the history of matching artists, albums, or tracks in the Sequence; AttributeRules for scoring Headers based on attribute matching; as well as RuleSets for combining a set of Rules into a single score. Other specializations of Rules may also be used.

[0070] ***RuleSet*** – As described herein, a RuleSet executes a set of rules and combines their scores in a specific way. There are specialized RuleSets for combining SequenceRules into sequence scores (SequenceRuleSet) and for combining AttributeRules into attribute scores (AttributeRuleSet).

[0071] ***MediaFile*** - As described herein, a MediaFile is the content actually rendered by the media player to affect playback of the content.

[0072] ***Sequence*** – As described herein, a Sequence is the list of content already played and in the queue to be played. The Sequence is examined to implement rules to limit repetition.

[0073] ***SequenceElement*** – As described herein, a Sequence Element is an element stored in a Sequence. Binds a Header with the timestamp when it was played for use with time-based SequenceRules.

[0074] FIG. 4 is a UML diagram showing an exemplary aggregation of the core objects described above.

Processing States

[0075] State transition diagrams of embodiments of the device as shown in FIGS. 5-8 illustrate high-level functional behavior and provide a roadmap for the detailed descriptions that follow. In typical embodiments the device will have separate threads of control running simultaneously under control of an operating system that handles thread and/or process scheduling. Thus, in typical embodiments the operations associated with the state diagrams will be operating simultaneously in their own contexts.

[0076] In the description that follows embodiments of state diagrams as illustrated in the associated figures are described. The program execution associated with the various state diagrams are typically implemented on one or more modules within the device, such modules typically including computer hardware, software, firmware and/or combinations of these elements. The computer software is typically stored on a computer readable medium such as memory and includes instructions configured to be executed by one or more processors. It is noted that, while the operations associated with the figures include events and states shown in the figures and described in the associated written description, it is apparent that other events and states including fewer, more, or different events and/or states than those shown in the figures are within the spirit and scope of the present invention. Accordingly, the state diagrams shown in the following figures are provided for purposes of illustration, not limitation.

Player States

[0077] Attention is now directed to FIG. 5, which illustrated an embodiment of a state diagram 500 representative of the processing states and events related to device playback of stored media files. These processing states are described in detail as applicable. An overview is as follows:

- Playing – The device renders content by transferring audio or other data from the current track to the decoder. The decoder decodes the data and sends the raw digital audio or other content to a digital to analog converter (DAC) or other output device such as a display, and to the device's audio amplifiers and line outputs.
- Stopped – The device is not rendering content. The previous Playing sub-state may be preserved so that a return to the Playing state resumes where it left off.
- Next Track – The device determines the next track to render. A new file/track is selected for rendering and control returns to the Playing state.

[0078] As shown in FIG. 5, process execution may first begin in an initialization state 510. Execution may then transition to a stopped state 520, wherein execution is stopped pending user input or other execution initialization. Upon receipt of a Play instruction or other input operation execution transitions to a Playing state 530 wherein playback of content such as audio files/tracks (MP3s, WMAs, etc.) is performed (i.e. the track is played/rendered on the device). A user may be able to provide input to direct execution of the device to Skip a track, or the track or file may end, wherein a Next Track state is entered, with execution proceeding to determine a new track to be played and then return execution to the Playing

state 530. A user and/or other input may also be enabled to enter a Stop instruction wherein device execution transfers from Playing state 530 to Stopped State 520.

User Interface Handling States

[0079] FIG. 6 illustrates an embodiment of a state diagram 600 showing some of the states used in an implementation of a state machine providing a device user interface. An associated event processing table is shown below in Table 2. Additional states/functions that may be peripheral are not shown for purposes of clarity. After turn-on at state 610, execution proceeds to an initialization phase at state 620, where the application registers event handlers for the various input devices and other events generated by the device. Execution then proceeds to a Wait state 630. When an event is received, the handler is dispatched at state 640. The event handler may create a new thread to handle the event or place the event in the queue of an existing thread. Table 2 below further illustrates state and event processing of one embodiment of the invention.

State	Event	Handling	New State
Stopped	Play	Send "Play" event to Player thread. Update display for Playing state.	Playing
Playing	Stop	Send "Stop" event to Player thread. Update display for Stopped state.	Stopped
Playing	Skip	Skip handling. See Skip	Playing
Playing	Ban	Ban handling. See Banning Tracks Processing	Playing
Playing	Save	Save handling. See Saving Favorites Processing	Playing
*	Station Slider Update	Update the corresponding slider settings for the current stations. See <insert reference>	Unchanged
Playing	Progress	Update track progress indicator	Playing
Playing	NewTrack	Update display with current track info	Playing
*	NewChannel	Stop Player, change channel, update display, Start Player	Playing
*	*	Update the display with current UI view	Unchanged, Changes UI Sub-states

Table 2 – Event Processing

Wireless (Bidirectional) Processing States

[0080] State diagram 700 of FIG. 7 illustrates one embodiment of a set of processing states and events relating to connection of the device to wireless networks and to updating of media file content stored by the device via such networks. Once initialized at state 710, device execution may proceed to a Not Connected Idle state 740. When the device is not connected to a wireless network, the thread may be configured to wait for an event to trigger a connection, for example a timer expiring or an explicit request by the user to connect. Power management may also affect this logic. From this state, execution may proceed to a Scanning state 720 where potential connections may be detected. A time out may also be applied allowing execution to return to the Not Connected Idle state 740 after a predetermined period. If a connection is established execution may proceed to state 730, where a networking update may be performed based on desired network utilization. For example, if the connection succeeds, an update exchange sending information and/or receiving content with the content management system and/or any associated servers may be executed. If the update completes before disconnection of the device, the device may enter the Connected Idle state 750 and wait for triggers to initiate an additional update at state 730. Upon disconnection based on, for example, timeouts, user requests to disconnect, or other events, execution may be returned to Not Connected Idle state 740.

Antenna Processing States

[0081] FIG. 8 is a state diagram 800 illustrating an embodiment of various processing states related to the processing of content in the case of unidirectional connections to a source of content. In one embodiment such a unidirectional connection is established between the playback device and a satellite-based delivery system through an Antenna Unit (AU) capable of being operatively connected to the playback device. The Antenna Unit is responsible for receiving tracks and data over the unidirectional (satellite) connection and creating regular files from the received signal.

[0082] In a preferred embodiment, the AU is implemented as a docking station that the device plugs into, for example, when the device is in a car. As shown in FIG. 8, after initialization at state 810, execution may proceed to a Not Connected Idle state 820. The device thread may remain in the idle state until it detects that it has been connected to the AU by, for example, a user request or AU trigger event. The device and the AU may then perform a secure handshake at Player/AU Handshake state 830 to authenticate one another

and pass decryption keys or other information. In the event of authentication failure, the device may return to Not Connected Idle state 820 to repeat the process.

[0083] Once authenticated, thread execution may proceed to a Connected Idle state 840. In this state a Data Available trigger event may transition execution to an Update Data state 850 wherein a data file or files may be provided, after which execution may be returned to Connected Idle state 840. Likewise, track or other content availability may be signaled by a Track Available Trigger to transfer execution to Add Track state 860. In this state tracks or other content may be added, with execution then returned to Connected Idle state 840. The Connected Idle state may also allow for a transition to Not Connected Idle state 820 based on a disconnection signaled triggered by a user input, timeout, or other event.

Playback Processing

[0084] Playback processing concerns the implementation of various processes that are related to the User Interface and Player State Diagrams previously described. In the description that follows embodiments of processes as illustrated in the associated figures are described. These processes are typically implemented on one or more modules within the device, such modules typically including computer hardware, software, firmware and/or combinations of these elements.

[0085] The computer software is typically stored on a computer readable medium such as memory and includes instructions configured to be executed by one or more processors. It is noted that, while the processes associated with the figures include particular stages shown in the figures and described in the associated written description, it is apparent that other processes including fewer, more, or different stages than those shown in the figures are within the spirit and scope of the present invention. Accordingly, the processes shown in the following figures are provided for purposes of illustration, not limitation.

Start Station

[0086] Attention is now directed to FIG. 9, which is a flowchart 900 representative of one embodiment of a Start Station process in accordance with aspects of the invention. Stations may generate the sequence of content to be played one or more items ahead. This allows the player to more efficiently pipeline content into the decoders. It may also allow for

potentially faster skip operations as well as displaying to the user a song or songs that are to come in the playback sequence.

[0087] As shown in FIG. 9, execution may begin with a Start Station initialization stage 910 wherein a user may select a particular station for execution or a station for execution may otherwise be selected. Stage 920 may then determine whether the desired station is the current station. If it is, the station selection process may be concluded by transitioning to stage 970. Alternately, if the desired station is not the current station, stage 930 may determine whether a current station is playing. If a station is playing, it is stopped at stage 950 (an implementation of this stage is further shown starting at stage 1010 as shown in FIG. 10), and execution transferred to Start Player stage 960. Alternately, if a current station is not playing as determined at stage 930, Start Player stage 960 may be directly executed. Playback/rendering of audio tracks and/or other content are then initiated at Start Player stage 960, after which execution of the Start Station process is completed at stage 970.

Stop Station

[0088] Referring now to FIG. 10, a flowchart 1000 representative of one embodiment of a Stop Station process in accordance with aspects of the invention is shown. Stop station execution begins at stage 1010, with execution transferring to Stop Player stage 1020 wherein station playback/rendering is halted. The portion of the sequence pre-generated for future playback may be truncated at stage 1030. The stop station process is then completed at stage 1040.

Skip

[0089] Skip handling is typically triggered by a user interface request to skip the current track. In typical embodiments, the device will allow users to skip certain tracks. However, the device logic must enforce certain business rules such as only allowing users with a specific tier of service to skip tracks, applying certain limits on the number of skips, or preventing users from skipping certain content such as advertisements.

[0090] Referring now to FIG. 11, a flowchart 1100 representative of one embodiment of a Skip process in accordance with aspects of the invention is illustrated. As shown in FIG. 11, a skip process is typically initiated at stage 1110, such as by a user request. Execution proceeds to stage 1120, where a determination is made as to whether the track can be skipped (i.e. whether the track is "skippable"). Tracks may not be skippable based on user service

limitations, content limitations, DMCA or other statutory requirements, or for other reasons. If a track is not skippable, a Can't Skip Dialog may be presented at stage 1140 providing a display, audible indication, or other indication that the track cannot be skipped, after which execution of the skip process is completed at stage 1160. Alternately, if a track is skippable, a second test may be applied at stage 1130 to determine whether the particular user's service allows skipping the track. This may be based on reasons such as those described above, as well as others such as the number of tracks skipped previously and the like. If the user is not allowed to skip the particular track a Not Allowed Dialog 1150 stage, providing a display, audible indication, or other indication that the track cannot be skipped, may be executed, with execution of the skip process then completed at stage 1160.

[0091] Alternately, if the track is both skippable and the user is allowed to skip the track, playback of the track may be skipped by transferring execution from stage 1130 to a next track stage 1170 wherein the next playable track may then be played/rendered (an implementation of the next track stage is further shown starting at stage 1210 as shown in FIG. 12).

Next Track

[0092] Turning now to FIG. 12, a flowchart 1200 representative of one embodiment of a Next Track process in accordance with aspects of the invention is provided. The device triggers Next Track processing when it needs a new track to play. This logic pre-generates the sequence "high water" tracks ahead whenever the sequence falls to "low water" tracks into the future. More specifically, Next Track processing may begin at stage 1210 wherein execution proceeds to a test stage 1220 wherein the pre-generated sequence is tested to see whether the sequence has fallen below a low water mark. If the sequence has not reached a low water mark the Next Track process is completed at stage 1240 where the next track in the sequence is returned for playback/rending. Alternately, if the pre-generated sequence has reached a low water mark, new content may be added to the sequence at stage 1230 (an implementation of an add to sequence stage is further shown starting at stage 1310 as shown in FIG. 13) and then the sequence is again tested at stage 1235 to see if the track content has risen above a high water mark. If it has not, execution is returned to stage 1230 where another track is added to the sequence and the sequence is again tested at stage 1235. Once the sequence has risen above the high water mark execution is completed at stage 1240 with the next track in the sequence returned for playback/rendering.

Add To Sequence

[0093] Adding a track to a sequence is a core process in the playback processing logic. In a typical embodiment, this logic is driven by a clock object that orchestrates buckets to select the next track. As described previously and further detailed below, clocks are ordered, sequential, and cyclical lists of buckets. However, there are two special cases of clocks worth noting for alternate embodiments: a Dynamic Clock dynamically chooses the next bucket based on special purpose bucket sequencing logic, and a Trivial Clock contains a single bucket from which the next track is always chosen.

[0094] By scoring semantics convention, the Clock allows buckets to provisionally decline to return a track. If the track returned by a Bucket Selection process scores less than 0.0 (i.e., by convention is an undesirable track), the bucket is skipped. However, if all buckets return tracks that score less than 0.0, the highest scoring track (denoted as the Best Worst Track) is returned.

[0095] FIG. 13 is a flowchart 1300 depicting one embodiment of an Add to Sequence process further depicting this process. As shown in FIG. 13, an Add to Sequence process may begin at a start stage 1310, with execution proceeding to stage 1320 where the bucket at the index of the clock for the current station is examined. A track is selected from the bucket at stage 1330 (an implementation of a select track from bucket stage is further shown starting at stage 1410 as shown in FIG. 14), and the index is incremented at stage 1340. The index is tested at stage 1350 to determine if it is greater than the number of buckets. If it is, the index is then reset to 0 at stage 1355, with execution then proceeding to stage 1360. Alternately, if the index is less than the number of buckets execution proceeds to stage 1360 where the track score is tested. If the track score as tested at stage 1360 is greater than zero, the track is added to the sequence at stage 1364 and the add to sequence process is completed. Alternately, if the track score is less than or equal to zero the track score is tested against the best worst score at stage 1366. If the track score is greater than the best worst track score the tested track is then assigned to the best worst track. Otherwise, execution proceeds to step 1370 where examination of the number of buckets tested is done. If all buckets have not been tested, execution is returned to stage 1320 where the process is repeated. Alternately, if all buckets have been tested and no track has been added to the sequence at stage 1364, the best worst track is added to the sequence at stage 1380 and execution of the Add to Sequence process is completed.

Select Track from Bucket

[0096] Select Track from Bucket is a process by which the best track is selected from a bucket. Selection involves scanning the bucket for the best scoring track. There are obvious performance optimization schemes for scanning an entire bucket for the best scoring track; however the maximum size of a bucket is typically assumed to be small enough that these are likely to be of small benefit, and therefore a simplified process 1400 as shown in FIG. 14 is illustrated. As shown in FIG. 14, the Select Track from Bucket process begins at a start stage 1410, with execution proceeding to stage 1420 where the next track in the bucket is examined. A track playback score may then be calculated at stage 1430 (an implementation of a playback scoring stage is further shown starting at stage 1510 as shown in FIG. 15) and this playback score may then be compared to a Best Track score at stage 1440. If the current track score is greater than the Best Track score, the current track is set to the Best Track at stage 1445, and execution is returned to stage 1450. Alternately, if the current track scores lower than the Best Track at stage 1440 execution may proceed to stage 1450 where a determination of remaining tracks in the bucket is made. If there are remaining tracks in the bucket execution returns to stage 1420 where the process may then be repeated. If there are no remaining tracks in the bucket, execution proceeds to stage 1460 where the identified Best Track is returned/selected.

[0097] When there are significant differences in the quality of tracks available to be selected, that is, the available tracks have a wide range of scores, it is desirable to choose the best track. In typical embodiments, it is generally considered a bad idea to 'ration' the best tracks by mixing in poorer tracks; however, such an approach may be employed in some embodiments. In an exemplary embodiment, front-loading the sequence with the best tracks is considered preferable for at least the following reasons:

1. Additional good tracks may be received before the next bucket selection.
2. The user may end the session or switch to a different channel at any time.
3. The scores of tracks change as the session progresses and previously played 'good' tracks become sufficiently rested to be repeated.

[0098] It is also considered desirable that the scoring methodology not be so rigid that the sequence becomes deterministic or is perceived as such by the user. For this reason, a random 'noise' component may be added to the scores to reduce the likelihood of this

perception. It is noted that the description of certain preferred embodiments defines certain conventions for scoring. These are designed to create semantic relationships between the scores by defining conventions relating to the meaning of the scores. It will, however, be apparent to those skilled in the art that other scoring conventions are possible assuming they preserve a consistent interchange between the different score types.

[0099] For instance, one example of a useful semantic relationship consistent with the invention involves the relationship between minimum and target rest to content fit. In particular, when a track has already been played, it may be "rested" for a certain minimum number of plays (i.e., a "minimum rest" period) before becoming eligible to be played again. Upon expiration of a "target rest" period, the track is deemed to be sufficiently "rested" to be nearly as eligible for playback as a track that has never been played. Content fit determines how well a track matches the ideal track and is typically a function of the sliders in SongBuckets or a combination of the slider scores and the rule scores in RuleBuckets.

[00100] Continuing with this example, a balance between content fit and sequencing rules like "rest" may be achieved by defining a convention (i.e., a semantic relationship) applicable to the meaning of the scores. In general, a content fit score of 1.0 may be used to define a perfect fit. A sequence score of -1.0 or less means the track is not eligible for play, as does a combined score of 0.0. Relating this convention to rest, a track gets a large negative sequence score after having been just been played. This decays to -1.0 as the rest approaches target and further decays toward 0.0 as the rest exceeds the target.

Playback Scoring

[00101] Attention is now directed to FIG. 15, which is a flowchart 1500 depicting one embodiment of a Playback Scoring process in accordance with the invention. When scoring the tracks in a bucket to determine Bucket Selection, it may be desirable to look at the fit of the track to the bucket, the fit of the track to the current slider settings, and how well the track obeys the sequencing rules relative to the history of tracks that have already been played or are queued for playback. In addition, a large negative score may be assigned to tracks that have been banned at either the performance or artist level. An overview of the process shown in FIG. 15 is provided below, followed by additional details of aspects of scoring.

[00102] As shown in Fig. 15, a Playback Score process may begin at start stage 1510 with execution proceeding to stage 1515 where score is assigned a calculated Bucket Fit

Score. The Bucket is next tested to determine if it has sliders for user customization at stage 1520. If it does, a slider score is calculated at stage 1530. At stage 1532 the track is tested for simple Boolean membership. If it does not have simple Boolean measurement, the track score is assigned the average of the fit and slider scores at stage 1540. If it does, the track is tested to determine if it is a bucket member. If it is a bucket member the score is assigned the value of the slider score at stage 1542. Alternately, if it is not a bucket member, the score is assigned a value of zero at stage 1544. Following stages based on whether the bucket has a slider, execution resumes at stage 1550 where the track is tested for banning. If the track is banned it is assigned a very low score (for example, -10) at stage 1555. If it is not banned a sequence score is calculated at stage 1560, a noise score is calculated at stage 1565, and the score, sequence score, and noise score are returned/provided to other stages at stage 1570. Additional details and features related to FIG. 15 are further described below.

Bucket 'Fit' Score

[00103] The Bucket Fit score determines how well the track fits the category of the Bucket. In a typical embodiment, for SongBuckets, all songs assigned to the bucket score a perfect value of 1.0. All songs not assigned to the Bucket score the minimum value of 0.0. This is also denoted herein as a 'Boolean' membership. For RuleBuckets, the fit score is the result of executing the AttributeRules against the song attributes. The fit score for RuleBuckets, by convention, is a score between 0.0 (worst fit, not in Bucket) to 1.0 (best possible fit).

Slider Score

[00104] Slider Rules allow the user to customize the station. When applied during sequencing, they bias track selection towards tracks with certain attributes. When applied during Inventory Management (as discussed in successive sections), they bias the addition of new tracks towards tracks with certain attributes. As the name implies, Slider Rules may be conceptually or actually connected to sliders or knobs in the User Interface. Exemplary slider implementations are further described in the related applications incorporated herein by reference. In particular, United States Provisional Patent Application Serial No. 60/886,283 describes exemplary embodiments of such sliders.

[00105] In typical embodiments Sliders are associated with a Station, although the effects of the slider may differ depending on the type of Bucket. The following

exemplary sliders bias track selection in a preferred embodiment:

- Newer ←→ Older
- Harder ←→ Softer
- Popular ←→ Obscure
- Favorite ← → Non-Favorite

Slider Attributes

[00106] By convention, slider attributes are defined as either 'Raw' or 'Cooked.' A 'Raw' slider attribute contains an actual value, typically associated with the attribute. For example, AlbumReleaseYear is a Raw slider attribute that contains a value based on release date between approximately 1950 and 2006.

[00107] Cooked slider attributes return a value between (by convention) -1.0 and 1.0. For example, a Popularity Cooked slider attribute is shown below in Table 3.

Popularity	**Cooked Slider Attribute Value**
Extremely popular, smash hit	1.0
Very popular	0.75
Well known	0.50
Somewhat known, not quite obscure	0.0
Not well known, deep, obscure	-1.0

Table 3 – Embodiments of Slider Attributes

[00108] Raw slider attributes may be converted into cooked slider attributes by rescaling them using a Rescaling Rule that includes a Min, Max, and Median parameter. This allows, for example, for an 80's radio station to define 'newer' as 1989 and 'older' as 1980, with 1985 as 'median' (the term median is used herein not in the strict mathematical sense, but to denote a value that should map to the center of the slider range - that is, a value that is neither 'old' or 'new' for the given station). For a simple linear interpolation (allowing the Median to not be centered between Max and Min) we have:

For raw sliders values v where v > Median:

$$v' = (v - Median) / (Max - Median)$$

For raw slider values v where v < Median

$$v' = (v - Median) / (Median - Min)$$

[00109] This puts the -1.0 score on Min, the 0.0 score on Median, and the 1.0 score on Max, with a linear interpolation for values between these points. However, a linear interpolation does not account for outlying data points beyond Min and Max. Such points will exceed the -1.0..1.0 convention or will cease to be differentiated if capped. In some embodiments, a better way of cooking sliders during rescaling may be to apply an exponential decay function such that the cooked scores decay towards 1.0 as the raw score increases from the Median towards the Max. Symmetrically, scores decay towards -1.0 as the raw score decreases from the mean toward the min.

For values >= Median:

$$v' = 1.0 - e^{-\lambda v}$$

And for values < Median:

$$v' = -1.0 + e^{-\lambda' v}$$

[00110] As shown here, v represents the raw slider value and v' is the cooked slider value. λ is the decay constant that controls the speed of decay. We can determine a suitable value for λ by determining how close we want the score to be to 1.0/-1.0 for values of Max and Min. We can determine this in terms of how many half-lives of decay should remain at the Max/Min. For example, 3 half-lives is a decay of .5 + .25 + .125 = .875 leaving a residual decay of .125 for scores that out lie the max.

$$t_{1/2} = (Max - Median) / n_{half-lives}$$

Half-life is related to λ as follows:

$$t_{1/2} = \frac{\ln(2)}{\lambda}$$

Re-arranging for λ:

$$\lambda = \frac{\ln(2)}{t_{1/2}}$$

Substituting into the above we have:

$$\lambda = \frac{\ln(2)}{(Max - Median)/3}$$

Similarly:

$$\lambda' = \frac{\ln(2)}{(Median - Min)/3}$$

[00111] FIGS. 16-18 are plots of age versus score profile for several embodiments of representative stations generated using the above equations. FIG. 16 shows a plot 1600 illustrating age scoring for an "80's" station characterized by the following parameters: Min=1980, Median=1985, Max=1989. Similarly, FIG. 17 shows a plot 1700 generated for a "Classic Rock" station characterized by the parameters of Min=1960, Median=1977, Max=1995. Finally, FIG. 18 shows a plot 1800 for an "Alternative" station characterized by the parameters of Min=1975, Median=2000, Max=2006.

Slider Scoring Rule

[00112] Once the slider attribute is 'Cooked', that is, converted to a standard range (-1.0…1.0 by convention), it can be converted into a slider score using the default slider scoring rule:

*Score = 1.0 – (SliderWeight / 2.0) + ((CookedAttributeValue * SliderWeight) / 2.0)*

A more generalized version of this formula is:

*Score = MaxScore – (DynamicRange / 2.0) + ((NormScore * DynamicRange) / 2.0)*

Where MaxScore is the maximum possible score. DynamicRange is the maximum delta from MaxScore, such that MinScore = MaxScore – DynamicRange. NormScore is a normalized score in the range of -1.0 to 1.0. Certain properties of this formula make it desirable for a slider application. These include:

- Generating scores in the 1.0 – 0.0 range that match desired conventions.
- The SliderWeight can be modulated along with the position of the slider to control the magnitude the attribute value contributes to the score.
- A SliderWeight of 0.0 generates a uniform 1.0 score for all attribute values, which is ideal for a centered slider.

- The negation of a slider weight inverts the scores in a symmetrical way. That is, the score order produced by a SliderWeight of 1.0 will be exactly reversed with a SliderWeight of -1.0. The score deltas in both lists will be symmetric.

Combining Multiple Sliders

[00113] If multiple sliders are in effect simultaneously, they may be combined to form a single score. In one embodiment, the average of all slider scores for which the SliderWeight is not 0.0 is taken (if all sliders are weighted 0.0, the slider score is set to 1.0). In some circumstances, a weighted average of the slider scores or other techniques to balance the contribution of unequal sliders may also be used. However, this approach should not be necessary if the attribute scores are well balanced across different types of sliders.

Sequence Rules

[00114] Sequence rules are used to score a candidate track for bucket selection against the current sequence. In one embodiment, sequence rules generate scores according to the following conventions:

Score	Semantic
< -1.0	Disqualifies track for playback
>= -1.0 and < 0.0	Technically eligible for playback, although suboptimal to the degree of the negative score.
0.0	Track is fully eligible for playback. Nothing in the sequence history discourages the track from being played
> 0.0	Not typically used for Sequence Scores. May indicate a HeavyRotation track that is overdue for playback.

Table 4 – An Embodiment of Sequence Rules Scoring

[00115] Recall that the Clock may be configured to skip a Bucket if the best track scores <= 0.0. By convention, the best possible Bucket/Slider fit is 1.0, so the sequence score must be greater than -1.0 to render an otherwise perfect track playable.

Artist Sequence Rules

[00116] Artist Sequence Rules can be configured to typically generate a 0.0 or -1.0 score. The default Artist Sequence rule is parameterized by the number of times an

Artist can appear in a sub-sequence, and the size of the sub-sequence to check. If the artist associated with the track being scored appears more than the given number of times in the interval, the rule returns -1.0. Otherwise the Artist Sequence Rule returns 0.0. Intervals may be specified in terms of time or number of tracks. Number of tracks is generally preferred for simplicity of implementation and better overall functionality (i.e. the device won't play an artist back-to-back regardless of how much time elapses between tracks). However, time based rules may be required to implement the sequencing rules specified by the Digital Millennium Copyright Act (DMCA) or by other statutory or regulatory content playback requirements.

Album Sequence Rules

[00117] Album Sequence rules are analogous to Artist Sequence Rules, except it is the Album associated with the track that is checked against the sequence. Album Sequence Rules are not typically used except in enforcement of DMCA rules or other statutory or regulatory requirements.

Track Sequence Rules

[00118] Track Sequence Rules are used to ensure that a track is not repeated too frequently. The term "rest" is used herein to denote the number of intervening tracks that are played between plays of a given track. Each Bucket has the notion of minimum rest and target rest. Minimum rest is the minimum number of tracks that must be played between plays of a track. Target rest is the ideal number of tracks that must be played between repetitions.

[00119] By convention, when a track has less than the minimum rest, its Track Sequence Score is less than -1.0. When a track reaches minimum rest, its Track Sequence Score is -1.0. As a track approaches target rest, its score moves towards 0.0. In typical embodiments it is desirable to heavily discourage the playing of tracks with less than minimum rest. Between minimum rest and target rest, there is a trade-off between optimal rest and best fitting track. Beyond target rest, the influence of the sequence score should wane, but there should still be a differentiation between played and unplayed tracks.

[00120] In an exemplary embodiment, to model this intended behavior, an exponential decay function may be used to decay the sequence score penalty towards zero. Exponential decay is a function where the rate of decay of a quantity is proportional to the

amount of the quantity remaining. For a quantity N and decay constant λ we have:

$$\frac{dN}{dt} = -\lambda N$$

The solution to this differential equation is the generalized function for exponential decay at time t where N_0 is the initial quantity.

$$N(t) = N_0 e^{-\lambda t}$$

[00121] It is desirable to choose N_0 and decay constant λ such that N($T_{Minimum\,Rest}$) = -1.0 and N($T_{Target\,Rest}$) = noise floor. The noise floor represents a score low enough that the random noise added to the scores to induce variety will dominate over the influence of additional rest. An exemplary approach begins by calculating how many half-lives are required between $T_{Minimum\,Rest}$ and $T_{Target\,Rest}$:

$$n_{half-lives} = -\log_2(|noise_floor|)$$

For example, if the noise floor is -.0625, we need 4 half-lives to decay the score from -1.0 to -.0625. The size of a half-life in number of tracks is can be determined as follows:

$$t_{1/2} = (T_{Target\,Rest} - T_{Minimum\,Rest}) / n_{half-lives}$$

Half-life is related to λ as follows:

$$t_{1/2} = \frac{\ln(2)}{\lambda}$$

Re-arranging for λ:

$$\lambda = \frac{\ln(2)}{t_{1/2}}$$

Substituting into the above we have:

$$\lambda = \frac{\ln(2)}{(T_{Target\,Rest} - T_{Minimum\,Rest}) / -\log_2(|noise_floor|)}$$

N_0 can then be determined as the initial score that will decay to -1.0 at $T_{Minimum\,Rest}$

$$N_0 = -1.0 \cdot 2^{T_{Minimum\,Rest}/t_{1/2}}$$

FIG. 19 shows a graph 1900 of sequence score as a function of track rest for one embodiment under the following conditions: Target Rest is 300, Minimum Rest is 150, Noise floor is .0125.

[00122] Having determined the values of N_0 and λ based on constants from the track sequence rules associated with the bucket, track sequence scores for each track can be calculated. If the track has not been played, the track sequence score is 0.0. If the track has been played, the number of tracks since the last play can be counted and used as 't' in the track sequence scoring formula:

$$N(t) = N_0 e^{-\lambda t}$$

[00123] To ensure maintaining differentiation between played and unplayed tracks, a minimum sequence penalty for played tracks can optionally be introduced and used as a ceiling for sequence scores of played tracks.

$$TrackSequenceScore = MIN(MinimumPenalty, N(t))$$

In a typical embodiment, MinimumPenalty is preferably a function of MaxNoise; for example it can be set to MaxNoise / 2.0. This ensures that the penalty of a played track, no matter the rest, is not washed out by the noise used to randomize the system. A particularly efficient way of computing rest for a track may be to keep a global variable that is incremented with each track played on the device. When a track is played, the value of the variable is recorded in the track header. Thus the rest is the difference between the current value and the value stored in the track.

Combined Scoring

[00124] The Fit, Slider, and Sequence scores are typically combined to generate a score that ranks the tracks in the Bucket for playback priority. Different combining rules may be applied. In an exemplary embodiment, for playback, different

scoring may be used to rank the tracks for RuleBuckets and SongBuckets.

For RuleBuckets:
PlaybackScore = ((FitScore + SliderScore)/2) + SequenceScore + Noise

For SongBuckets:
PlaybackScore = SliderScore + SequenceScore + Noise

That is, the FitScore and the SliderScore for Buckets are averaged to calculate fit, then the SequenceScore and Noise are added.

[00125] By convention, SongBuckets have no FitScore; all songs assigned to the Bucket are assumed equally fitting. That is, the Bucket fit is a Boolean or filtering function. For these buckets, the SliderScores are used to determine playback fit and add the SequenceScore and Noise.

[00126] For an exemplary embodiment, the main conventions for combined scores are as follows:

1. The combined score must be greater than 0.0 for the track to be eligible for play. If all tracks in the Bucket score <= 0, the Bucket will be skipped.

2. Fit and Slider scores are combined to form a 0.0 – 1.0 score that reflects the relevancy of the content to the desired content programming for the Bucket. 1.0 is perfect fit, 0.0 is no fit.

3. Sequence scores are used to enforce repetition and other ordering rules. A sequence score of -1.0 or lower is used to make a track ineligible for play.

4. Noise is added to the combined score. Noise is a random value between +/– MaxNoise/2.0. MaxNoise is a tunable system parameter, and represents the significance of score deltas. That is, a track that scores MaxNoise better than another track will always be selected ahead of the other track. However, tracks that score within MaxNoise of each other may be selected in either order (with a probability determined by the delta and the random distribution).

Detailed Inventory Management Processing

[00127] As described herein, Inventory Management processing concerns the implementation of the Add Track, Save, and Ban processing states shown in the state diagrams. In certain embodiments, Effective Inventory Management may be crucial to optimal operation, particularly with lower storage capacity devices. It may be less of an issue

for higher capacity devices with large amounts of storage. However, limitations on the amount of content that may be cached (for example, as dictated by record label agreements) may make inventory management important for all device sizes.

[00128] The goal of inventory management is to optimize the storage utilization of the device for the stations in use. FIG. 20 illustrates a chart 2000 of Inventory Pools for a representative embodiment. As shown in FIG. 20, device inventory can be categorized as follows:

- **Service Content** is the storage pool for files added by the service through either the unidirectional or bidirectional links.
- **'Favorites'** is the storage pool for files from the service that have been 'saved' by the user.
- **User Content** is the storage pool dedicated to storing files explicitly transferred to the device from the user's personal collection, such as user owned MP3 or other media files.
- **Unused** is the unused storage available on the device.

Managing Service Content Inventory Concepts

[00129] In typical embodiments, Favorites and User Content are considered separate storage pools, because once the user saves a favorite or loads their own content the storage is deemed largely untouchable. That is, only the user can free this storage by explicitly removing items. The Service Content pool, on the other hand, is dedicated to providing the best possible enhanced radio experience and must optimize its use of storage autonomously. Basic concepts behind the managing of the service inventory are as follows:

1. The service storage pool will grow to the capacity of the device (for smaller capacity devices) or a fixed upper limit (for larger capacity devices).

2. Add Track processing from the connections fills the pool towards its upper limit (high water).

3. When enough tracks have been added to reach Inventory "high water" the worst tracks can be removed until inventory "low water" is reached.

Add Tracks Processing

[00130] Attention is now directed to FIG. 21, which is a flowchart 2100 depicting an embodiment of an Add Tracks process in accordance with aspects of the invention. In summary, an incoming track is 'offered' to each station on the device, which in turn offers the track to each of its buckets. If the station is programmed using only

SongBuckets, it may be possible to route the track directly to the station/buckets to which the track is targeted as an optimization. Stations that contain RuleBuckets must score the track against their rules to determine if the track fits or not. In a typical embodiment, there are 3 possible results of offering a track to a Bucket:

1. The Bucket does not want the track
    a. The Bucket already has this exact same track
    b. The track does not meet the Bucket's criteria

2. The Bucket has added the track (growing in size to do so).

3. The Bucket has added the track by replacing one of its existing tracks:
    a. The Bucket replaced an older version of the same track
    b. The Bucket replaced a different track with this track

[00131] If some Bucket on some station adds the incoming track, the media file may then be stored on the device. If no Bucket on any station adds the track, the media file is discarded. In a typical embodiment, the device monitors available storage space using a high and low water mark model. Once storage usage reaches the high water mark, an operation to cull inventory is undertaken to bring storage utilization back down to the low water mark.

Eviction

[00132] HeavyRotation buckets typically contain the most popular and frequently played songs on standard stations. The bucket has a fixed maximum size that, combined with the significant station duty cycle, may yield a fairly high rate of individual song repetition. When new tracks are received into a full HeavyRotation bucket, old tracks must be removed from the bucket to make room for the newer tracks. Since the displaced track is still potentially very valuable to the station, the station can be programmed to allow tracks 'evicted' from one bucket to be automatically assigned to a new bucket. If no eviction bucket is specified, the evicted track is removed (providing no other stations hold references).

[00133] In more detail, process 2100 may begin with an add track start stage 2110 based on reception of a new track. Execution may then proceed to a banned selection stage 2115 where an assessment is made as to whether the particular track or artist has been banned. If the track or artist has been banned, execution of the process is terminated at stage 2118 with the track being discarded. If the track or artist is not banned, a station may next be selected for examination at stage 2120, with the next available bucket examined at stage 2125. The bucket fit score of the track may be examined at stage 2130, and the score tested

at stage 2135. If the score is less than zero, execution returns to the next bucket stage 2125. Alternately, if the score is greater than zero the track header may be added to the bucket at stage 2140, and tested to see if it replaces an earlier version of the header at stage 2142.

[00134] If there is an earlier version in the bucket, it will be removed at stage 2150 with execution proceeding to stage 2160. If not, the bucket may be tested for fullness at stage 2144 and, if full, execution proceeds to stage 2160. Finally, at stage 2146 if the bucket is not full and the bucket has an eviction bucket, the worst track header will be moved to the eviction bucket at stage 2152, with execution returned to stage 2160. If there is no eviction bucket execution proceeds from stage 2146 to stage 2154, where the worst track header is removed and execution proceeds to stage 2160.

[00135] At this point, a test is made at stage 2160 to determine if more buckets need to be tested. If so, execution proceeds to stage 2125. If no more buckets need to be tested for the current station, a test is made at stage 2170 to determine if additional stations need to be tested. If so, execution returns to stage 2120 to test the next station. Alternately, a test is made at stage 2172 to determine if any bucket wants the track and if not, the track is discarded at stage 2180. At stage 2173 a test is performed to determine if the service inventory has exceeded the capacity (or high water mark). If it has, the inventory may be culled at stage 2176 (an implementation of a cull inventory stage is further shown starting at stage 2210 as shown in FIG. 22). If capacity (i.e. high water) has not been exceeded, the track will be kept/added to inventory at stage 2182.

Inventory Culling

[00136] Attention is now directed to FIG. 22, which is a flowchart 2200 depicting one embodiment of an Inventory Culling process in accordance with aspects of the invention. In accordance with this embodiment, once the service pool reaches high water, an inventory culling process may be employed to remove the 'worst' tracks and bring the inventory levels down to a desired 'low water' mark. This process may be used to aid in ensuring that there is always free storage to handle the next 'Add Track' event. Other reallocation of device storage may also be implemented through Inventory Culling. For example, to free up space for additional user content to be transferred onto the device, the service pool allocation may be adjusted by changing the high and low water marks. Inventory culling then removes the tracks to free up the required space. Other operational situations may also employ the inventory culling process.

[00137] In summary, in process 2200 a scan traverses all stations and buckets and scores each track based on an estimate of the eventual order of playback. If a track appears in more than one bucket, the largest of its scores may be assigned to the track. Note that since the score indicates the predicted next time the track will play, it is possible to manage inventory by removing all tracks with a score greater than a calculated threshold. The calculation should take into account the repetition of tracks in HeavyRotation buckets as well as tracks that appear in multiple buckets. This approach may be slightly more efficient (i.e. a heap is not required), but is less flexible in regard to the scoring algorithm.

[00138] Process 2200 is further described below, with additional details related to specific aspects further described following the overall process description. The process may begin at an inventory cull start stage 2210, with execution proceeding to a calculate duty cycles stage 2215. The current score for all files may initially be set at the lowest possible score at stage 2220, with each station successively examined, starting at stage 2225. For each station the buckets in the station are examined at stage 2230, with the headers in the examined bucket examined at stage 2235. The inventory score for the header being examined is determined at stage 2240. If the determined score is greater than the file's current score, the current score is assigned to the file at stage 2248. If not (i.e. if the current is less than or equal to the file current score) the file current score is retained with execution proceeding to stage 2250. Each header within the bucket are then examined based on a test at stage 2250, each bucket in the station are examined based on the test at stage 2255, and each station is examined based on a test at stage 2260. Once all stations and their respective buckets and headers have been tested, based on reaching the end of the stations at stage 2260, a delete tracks stage 2270 (an implementation of this stage is further shown starting at stage 2410 as shown in FIG. 24) may then be employed.

[00139] Additional details of aspects of inventory management are further described below.

Scoring Tracks for Deletion

As noted above, in a typical embodiment, the overall strategy for Inventory Management is to rank each track according to its likely order of playback on the device. The tracks that are more likely to be played sooner are kept and the tracks that are likely to be played later are deleted. To score tracks, each track may be ranked in a bucket according to its probable play order within that bucket. Duty cycle is then accounted for with respect to

both the bucket within the station and the station within the player to achieve a score that ranks all tracks within the device. The worst tracks can then be deleted until inventory levels are within a desired range. In an exemplary embodiment, the first step in this process is to calculate a forward-looking duty cycle for each station.

Calculating Station Duty Cycles

[00140] The available inventory for the device may be managed according to the predicted "duty cycle" for each station. Duty cycle for a station represents the percentage of total device plays attributed to the station. A goal is to calculate a forward-looking duty cycle for each station. The forward-looking duty cycle must adapt to changing patterns of usage. For example, if a user starts listening to a new station, the forward looking duty cycle for that station should increase upwards even if the actual duty cycle of the station over the life of the device is quite low. Put another way, the duty cycle should be adapted based on a weighted average of recent and longer term observations; simply measuring the long term observed duty cycles of each station creates a system that fails to adapt quickly to new patterns, whereas a system based entirely on short-term observations neglects the information provided by the device's longer term history.

[00141] In an exemplary embodiment, an adaptive strategy that varies the rate of adaptation based on the delta between short term and longer-term observations is employed. The goal is for the duty-cycle of a new station to increase relatively rapidly based on short-term observations, at the same time taking into consideration an extended number of observations to completely overcome the longer-term usage patterns of the device. In addition, a tunable constant may be required to allow the rate of adaptation to be controlled.

[00142] One such approach is to use an exponential decay function on the delta between the short-term and long-term observations. As noted previously, exponential decay is a function where the rate of decay of a quantity is proportional to the amount of the quantity remaining. For a quantity N and decay constant λ we have:

$$\frac{dN}{dt} = -\lambda N$$

Integrating produces the generalized function for exponential decay at time t where No is the initial quantity.

$$N(t) = N_0 e^{-\lambda t}$$

Applying this technique to short and long term duty cycles we have

$$c_{t+1} = (o_{t+1} - c_t)\lambda + c_t$$

Where c_{t+1} is the adapted duty cycle at time t+1, o_{t+1} is the observed duty cycle at time t+1 and c_t is the adapted duty cycle at time t.

[00143] FIG. 23 shows a graph 2300 illustrating the general shape of the duty cycle adaptation starting with 4 equally played stations. At t0 (start time), the Rock station begins getting 100% of observed play events. Halfway through the displayed time interval, the Pop station begins getting 100% of observed play events. The value of λ chosen (here it is .05) controls the rate of adaptation.

Duty Cycle Renormalization

[00144] While short-term observations may provide a good predictor of future duty cycle, it may also be desirable to ensure that any station supported by the device is playable with a minimum allocation of inventory. To achieve this without disrupting the overall strategy of using duty cycle as the guide to inventory allocation, a minimum "baseline" duty cycle floor for all stations in the device may be set. The baseline duty cycle ensures that inactive stations get at least $T_{s\min}$ tracks where T_{total} represents the total number of tracks in the service.

$$c_{baseline} = T_{s\min} / T_{total}$$

To calculate the "normalized" duty cycles for active stations (those with predicted duty cycles > 0), the calculation is started by allocating a duty cycle of $c_{baseline}$ to all stations (active and inactive). The residual duty cycle available for allocation to the active stations is then:

$$c_{residual} = 1.0 - (c_{baseline} \cdot n_{stations})$$

Then, the residual duty cycle may be allocated to the active stations in proportion to the predicted duty cycle of each station:

$$c_i' = c_{baseline} + (c_i \cdot c_{residual})$$

A cap $T_{s\max}$ on the number of tracks in a station may also be established, beyond which the extra inventory for a station provides negligible benefit. After ensuring that the duty cycle accounts for baseline inventory for all stations, we can re-apportion the excess from stations that exceed the cap. The duty cycle cap for any station is:

$$c_{\max} = T_{s\max} / T_{total}$$

The duty cycle may be set to $c_{\max}$ for any station with a duty cycle greater than $c_{\max}$. For $u_0 \ldots u_{m-1}$ representing the duty cycles below the cap we have:

$$u_i' = u_i + (c_{\max} / m)$$

That is, the inventory excess is distributed equally among the m stations that are below the cap. The rationale is that additional inventory will have a larger positive impact on smaller stations (the reason for a cap), but these stations have less likelihood of being played. It may be assumed that factors cancel, therefore opt for an even reallocation. This technique can result in duty cycles greater than $c_{\max}$; a station just below $c_{\max}$ gets an allocation of excess inventory that could push it over $c_{\max}$.

Ranking of Tracks in Buckets

[00145] As described above, it may be preferable that each Bucket ranks its tracks in the likely order that the tracks will be played as part of the inventory management process. By convention, algorithms used for ranking the tracks versus the bucket vary based on the bucket type.

Heavy Rotation Buckets

[00146] HeavyRotation Buckets are buckets of fixed maximum size to achieve a desirable repetition rate. HeavyRotation buckets may be ranked for inventory management by fit/slider score first and secondarily by programming date. Because HeavyRotation buckets are small in comparison to their duty cycle, they tend to stay mostly in inventory.

Library Buckets and Medium Rotation Buckets

[00147] For larger buckets (> 10 tracks, target rest > 100), the sequence score becomes more important in determining the value of the track to the bucket. However, there is a tradeoff between bucket fit and the sequence penalty associated with repetition.

[00148] Slider settings dynamically affect this relationship. A distinction can be made between saved/locked slider settings and those used to temporarily tweak playback order. In the former case, the station is desirably optimized for the particular slider settings by favoring inventory that matches the sliders. In the latter case, the slider settings apply to playback only, and do not affect the ranking of tracks within the bucket for inventory purposes. In the latter case we assume the sliders may well be in a different position in the future, so they are irrelevant for ranking deep into the bucket.

[00149] For locked sliders, a combination of slider score and bucket fit score (for RuleBuckets) may be used to determine the fit component of the inventory score:

For RuleBuckets:
$InventoryScore = ((FitScore + SliderScore)/2) + InventorySequenceScore$

For SongBuckets:
$InventoryScore = SliderScore + InventorySequenceScore$

[00150] The calculation of a sequence score may be based on estimating what the sequence score might be when the track is played. In an exemplary embodiment, the sequence penalty associated with a track in relationship to the fit score may be discounted. The same sequence scoring algorithms that are used for playback may be used, but they can be calculated for a future time / track interval. Here t' represents the number of tracks into the future to calculate the sequence penalty. Nbucket is the number of tracks in the bucket.

$$t' = \frac{(n_{bucket} / 2)}{(c_{bucket} \cdot c_{station})}$$

For time-based rules (e.g., rules resulting from the requirements of the DMCA concerning the maximum number of times tracks for a given artist can be played in a given time period), t' can be multiplied by average track length. The inventory sequence score is then calculated at t':

$$InventorySequenceScore = sequenceScore(t')$$

The sequence score should have a maximum value of slightly less than zero for tracks that have been played, but are well rested, and a uniform value of 0.0 for tracks that have not been played.

[00151] As an example, the sequence score for a track that has recently been played will be a large negative number as a consequence of the minimum and target rest factors discussed above. If in this example the value of t' is "40," minimum rest is "30" and target rest is "50," the discounted sequence score will be between -1.0 and 0.0 according to the applicable equations set forth previously. The actual scores are -55.7 after being played (rest=1), and -0.25 with the t' discount.

Calculating the Score

[00152] The bucket rank may then be combined with bucket duty cycle and station duty cycle to obtain a score that represents an estimate of an overall track playback order for the device. If r represents the rank of the track in the bucket (1..n), let $c_{station}$ represent the predicted duty cycle of the station as described above, and let c_{bucket} represent the duty cycle of the bucket with respect to the station, that is, the number of times the bucket appears in the clock divided by the number of slots in the clock, this results in:

$$score = (r - 0.5) / (c_{station} \cdot c_{bucket})$$

Deleting Tracks

[00153] Attention is now directed to FIG. 24, which is a flowchart 2400 depicting one embodiment of a Track Deletion process in accordance with aspects of the invention. In general, after execution of an inventory cull pass through the stations (such as was previously described), the "worst" tracks are deleted from the device to free up storage space. In order to implement this, a process scanning all the files and their scores to accumulate a list of the at least N worst tracks, where:

$$N * averageTrackSize > currentInventorySize - lowWatermarkSize$$

may be done. Since tracks may be scored by estimated play order, by convention lower scores are better and higher scores are worse.

[00154] In an exemplary embodiment, a Min-Heap is used to efficiently accumulate the worst tracks. The process deletes each of the files in the Heap in worst to best order until the desired low water mark is achieved. Note that since the score indicates the predicted next time the track will play, inventory can conceivably be managed by removing all tracks with a score greater than a calculated threshold. The calculation must take into account the repetition of tracks in HeavyRotation buckets as well as tracks that appear in multiple buckets. This method may be slightly more efficient (i.e. a heap is not required), but is less flexible in regard to the scoring algorithm. In the described embodiment, this process may also be used to delete tracks that are no longer referenced by stations or playlists. As described previously, by convention a file will have a "worst possible" score from the initialization phase if it is not referenced by any station.

[00155] More particularly, the track deletion process 2400 illustrated in FIG. 24 may first begin at a start stage 2410, with execution proceeding to a scan playlists stage 2415 (details of an embodiment of a scan playlists stage are further illustrated starting at stage 2510 as shown in FIG. 25). A Min Heap sized at approximately the number of tracks to delete may be created at stage 2420. Examination of tracks begins at stage 2425. If the score is greater than the Heap Best Score, or the Heap is not full, the track may be added to the Heap at stage 2435, with execution proceeding to stage 2440. Alternately, the track is not added and execution likewise proceeds to stage 2440 where a test is made of whether additional tracks need to be tested. If they do, execution returns to stage 2425 for examination and testing of additional tracks. If no tracks remain to be tested at stage 2440, execution proceeds to stage 2450 where the heap is reverse sorted. A loop is then performed where the next track is deleted at stage 2460, the heap end of space is tested against the low water mark, and if it is greater than or equal to the low water mark execution returns to stage 2460 where an additional track is deleted and the process repeated. Once the heap end of space is below the low water mark the process may be completed at stage 2480.

[00156] If the user interface of a particular embodiment supports adding tracks from the service to user defined playlists, an additional pass is required to score the tracks that are referenced by playlists, but no longer referenced by stations. Each of these tracks can be given a "best possible" score to ensure that they are not removed.

[00157] An embodiment 2500 of a Scan Playlist process is illustrated by the flowchart of FIG. 25. The process of scanning files may begin with a scan playlists start stage 2510, with execution proceeding to a successive looped examination of playlists at stage 2520 and associated headers at stage 2530. A Score is assigned initially to the file at stage 2540 equal to the favorites score. The Score is tested at stage 2544 to determine if it exceeds the File Current Score. If it does, the Current Score is set to Score at stage 2548 and execution is transferred to stage 2550. If Score does not exceed the File Current Score at stage 2544, execution is transferred to stage 2550 without any further action. Once execution of the loops of stage 2550 and 2560 are finished the process may be completed at stage 2570.

Saving Favorites Processing

[00158] Attention is now directed to FIG. 26, which shows a flowchart 2600 illustrating a Saving Favorites process consistent with aspects of the invention. In a typical embodiment, the operation of saving favorites or adding tracks from the service into user-defined playlists essentially moves the track from one storage pool to another. Although these tracks are still part of the station inventory and still contribute to the station, this distinction is made because the tracks are protected so that they can no longer be removed from the system without the user explicitly doing so. Removal of a track from the service inventory pool can be thought of as essentially removing a re-usable slot. Constraints should be checked and enforced to ensure that the service cannot be substantially degraded by saving favorites.

[00159] As shown in FIG. 26, a save track (saving favorites) process may begin with a start stage 2610. This may be initiated by, for example, a user input based on a desire to save the track. The track may then be tested at stage 2620 to determine whether it is saveable. If it is not, a can't save dialog may be executed at stage 2625, along with a display not allowed error or other indication provided at stage 2638. Assuming a track is saveable, it may then be tested at stage 2630 to determine if the user is allowed to save the track. If not, a similar series of not allowed and error displays may be executed at stages 2635 and 2638 respectively. Assuming the track is not prohibited from saving, inventory constraints may be checked/determined at stage 2640 (details of inventory constraints checking are further described and illustrated below with respect to FIG. 27). Based on this information, device fullness may be tested at stage 2560. If the device is full, dialog stage 2654 and error stage 2658 may be executed. If, instead, the device is not full but close to capacity as determined

by device almost full stage 2660, a device nearly full warning stage 2665 may be executed. Finally, if the device has sufficient capacity for storage, the track may be added to the station favorites playlist at stage 2670, the header may be marked as a favorite at stage 2672, the service pool size may be decremented accordingly at stage 2674, the favorites pool size may be incremented at stage 2676, and the save track process completed at stage 2680.

Checking Inventory Constraints

[00160] Checking inventory constraints is the process of determining if there is enough space left in the service to accommodate a reallocation of storage. In typical embodiments, the minimum space required to operate the service may be defined in terms of a few tunable and derived parameters as set forth in FIG. 27. In general, the service inventory constraint is defined in terms of the constant $A_{\min}$: the minimum number of tracks needed to support the active stations. Since users divide their listening time between one or more stations, $A_{\min}$ could be described as the number of tracks needed to support a single station that gets 100% of the listening time.

[00161] In addition to tracks for the active stations, the service pool needs to reserve a minimum number of tracks for each inactive station that renders the station listenable as it transitions from an inactive station to an active station. This constant is denoted as $T_{s\min}$. Thus the minimum size of the services pool is:

$$T_{\min} = A_{\min} + (T_{s\min} * \text{inactive stations})$$

However, this is the minimum size of the services pool at low water. Space should also be reserved to account for the difference between low water and high water.

Thus the device is full when

$$T_{total} - T_{float} \leq T_{\min}$$

A device full warning message may be issued when the device approaches this threshold:

$$T_{total} - T_{float} \leq T_{\min} + T_{warn}$$

As noted above, when saving favorites or saving service tracks into user defined playlists, the service pool is decremented by one track. Therefore, the above constraints for T_{total} -1 should

be checked before allowing the operation.

Shrinking the Services Pool

[00162] In a typical embodiment, the services pool is designed to grow to the device capacity or some preconfigured maximum for large capacity devices. To free excess services storage for other purposes, the high and low water marks may be adjusted, and an inventory cull may be performed. To free N bytes of space:

$$W'_{low} = W_{low} - N$$

The number of tracks removed is approximately:

$$(W_{low} - W'_{low}) / t_{size}$$

The operation can be allowed if:

$$T_{total} - T_{float} - ((W_{low} - W'_{low}) / t_{size}) \geq T_{\min}$$

Banning Tracks Processing

[00163] Referring now to FIG. 28, a flowchart 2800 of an embodiment of a Ban Tracks process consistent with aspects of the invention is shown. In typical embodiments banning of tracks or artists is accomplished by marking the Headers of the tracks as being banned. In addition, the banned track may be added to a Banned Track List and the banned artist may be added to a Banned Artist List.

[00164] As shown in FIG. 28, a ban track process may begin at a start stage 2810, with execution proceeding to a test stage 2820 where artist/track ban-ability is tested. If the artist or track is not ban-able, a not ban-able dialog stage 2825 may be executed, followed by completion of the process at stage 2840. Likewise, a test may be applied at stage 2830 to determine whether the user is allowed to ban the particular track. If not, a similar not allowed dialog stage 2835 may be executed, and the process completed at stage 2840. Alternately, if the track or artist is ban-able, a test of track or artist ban may be done at stage 2850 (to determine whether ban applies to the single track or all of the artist's tracks). As a consequence, the track/artist may be added to a banned track/artist list at stages 2860 or 2870 respectively, and the banned attribute set in the track header or all headers with the artistID at stages 2865 or 2875 respectively. Finally, a skip stage 2810, such as was described

previously starting with stage 1110 of FIG. 11, may be executed.

[00165] In alternate embodiments, banning based on other criteria can be implemented in a similar or analogous fashion. Tracks that are marked as banned may be given 'worst possible' rankings in their buckets when scoring tracks for inventory culling, and thus will be deleted on the next inventory cull cycle.

[00166] The banned lists may also be consulted for other functions, such as when an incoming track is received so that banned artists or tracks are not re-added to device memory.

Updating Metadata

[00167] In typical embodiments, both bidirectional and unidirectional connections can provide updated metadata and media files for tracks. The connection may send metadata only (i.e. the Header) or may send an updated Header + Media File combination. While in future implementations it may be desirable to optimize the handling of Metadata updates, it is envisioned that the simplest way of handling Metadata updates is to process the updated Header with an Add Track process, such as process 2100 as illustrated in FIG. 21. Processing metadata updates via the Add Track process requires that all Buckets offered the Header scan their inventory for a reference to a previous version of the Header (as defined by having equivalent Performance Ids) even if they don't want the track. If the Bucket doesn't want the new version of the track, but holds a reference to the old version, the Bucket must remove the old version (possibly triggering the removal of the associated media file if this was the last reference). If the Bucket still wants the track, the old header must be replaced with the new one (if the media file is not updated, the new header must be pointed at the old media file; if the media file is updated, the old media file and header must be removed at the end of the update operation).

Adding New Stations

[00168] In typical embodiments, when the device is connected via a bidirectional connection the user will have the opportunity to add niche or custom stations that are only reasonable to populate across this connection type. When adding a new station, the inventory must be adjusted to accommodate the new station. The device must have room for at least an additional $T_{s\min}$ tracks. If this is not the case, the user may be prompted to facilitate making room for additional stations by removing stations, favorites, or media files

(i.e. mp3s, etc.).

[00169] In a typical embodiment a new station should receive a default initial duty cycle above and beyond the baseline, and predicted duty cycles should be re-calculated. (it is assumed that the user will listen to the newly added station). As tracks are added across the bidirectional connection and the high water mark is reached, inventory is freed according to the predicted duty cycles. See, for example, the process for culling inventory as is shown in process 2200 illustrated in FIG. 22.

Detailed Connections Processing

Adding Tracks from Unidirectional Connections

[00170] In typical embodiments, as tracks are received from a unidirectional connection (i.e. a broadcast type connection), they are added to inventory using Add Tracks processing as is, for example, described in process 2100 shown in FIG. 21.

Wireless (Bidirectional) Update

Station Refresh

[00171] In typical embodiments, when the device has a bidirectional connection to the broadcasting services/content management system, the device will attempt to obtain updates only for its active stations in priority order. Thus, the device can request targeted updates and utilize the connection only for tracks known to be of interest to the device. To obtain targeted updates, the device first re-calculates its predicted duty cycles as described above. The stations that have the largest negative delta between actual and predicted/allocated inventory are the first priority for update. The current inventory of the player relative to a station may be transmitted to the server along with the target inventory level for that station based on device capacity and predicted duty cycle. The server can then reply with an update to the station that may include the following information, as well as other information:

1. An updated Clock if the station definition has been changed on the server side.
2. Updated Headers for tracks currently on the device if metadata for those Headers has changed.
3. A list of additional tracks to download in priority order.

[00172] In response, the device processes the reply message by updating the station definition (if changed), and updating the Headers as indicated (see Updating Metadata above). The device then begins receiving/downloading each of the specified tracks in the order specified. Each track is 'added' to the player as if it where received on a unidirectional connection (this may be done to ensure that the player is consistent in the face of adds from both connection types, simplifies implementation, and handles the case where received tracks are targeted to multiple stations).

[00173] Rather than download all tracks to update a single station, it may be preferable for the device to get updates for its other active/high-priority stations and download the highest priority tracks across all the station updates to ensure that all active stations get some updates before the connection terminates. Should the connection persist, all stations will have the opportunity to process their updates.

[00174] Some aspects of the present invention may be embodied in the form or computer software and/or computer hardware/software combinations configured to implement one or more processes or functions of the present invention as described and illustrated herein. These embodiments may be in the form of modules implementing functionality in software, firmware, and/or hardware/software/firmware combinations. Embodiments may also take the form of a computer storage product with a computer-readable medium having computer code thereon for performing various computer-implemented operations, such as operations related to functionality as describe herein, on one or more computer processors. The media and computer code may be those specially designed and constructed for the purposes of the present invention, or they may be of the kind well known and available to those having skill in the computer software arts, or they may be a combination of both.

[00175] Examples of computer-readable media within the spirit and scope of the present invention include, but are not limited to: magnetic media such as hard disks, floppy disks, and magnetic tape; optical media such as CD-ROMs, DVDs and holographic devices; magneto-optical media; and hardware devices that are specially configured to store

and execute program code and/or data, such as application-specific integrated circuits ("ASICs"), programmable logic devices ("PLDs") ROM and RAM devices, Flash devices, and the like. Examples of computer code may include machine code, such as produced by a compiler, and files containing higher-level code that are executed by a computer using an interpreter. Computer code may be comprised of one or more modules executing a particular process or processes to provide useful results, and the modules may communicate with one another via means known in the art. For example, some embodiments of the invention may be implemented using Java, C#, C++, or other programming languages and software development tools as are known in the art. Other embodiments of the invention may be implemented in hardwired circuitry in place of, or in combination with, machine-executable software instructions.

[00176] The foregoing description, for purposes of explanation, used specific nomenclature to provide a thorough understanding of the invention. However, it will be apparent to one skilled in the art that specific details are not required in order to practice the invention. Thus, the foregoing descriptions of specific embodiments of the invention are presented for purposes of illustration and description. They are not intended to be exhaustive or to limit the invention to the precise forms disclosed; obviously, many modifications and variations are possible in view of the above teachings. The embodiments were chosen and described in order to best explain the principles of the invention and its practical applications, they thereby enable others skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated. It is intended that the following claims and their equivalents define the scope of the invention.

Claims

I claim:

1. A method for facilitating playback of media files through a device having access to a data link, the method comprising:

receiving, over the data link, media files and file metadata wherein the file metadata defines attributes of the media files;

storing, in a memory of the device, ones of the media files characterized by file metadata consistent with one or more channel profiles maintained within the memory of the device; and

sequencing the ones of the media files for playback based upon one or more of the channel profiles.

2. The method of claim 1 further comprising discarding, based upon the file metadata associated with ones of the media files, those media files that are non-responsive to the one or more profiles; and

categorizing the media files stored in the memory of the device into one or more categories and sequencing of media files for playback responsive to a sequencing of the categories.

3. The method of claim 2 where the ordering of the categories is specified by the channel profile.

4. The method of claim 2 where the ordering of the categories is dynamically determined by the device.

5. The method of claim 2 where the media files selected responsive to the category are selected responsive to the degree to which the attributes of the media file match the attributes targeted by the category.

6. The method of claim 5 wherein the media files selected responsive to the category are additionally selected responsive to the degree to which the attributes of the media files match one or more attribute biases specified by an end user.

7. The method of claim 5 wherein the media files selected responsive to the category are additionally responsive to repetition rules associated with the category.

8. The method of claim 7 wherein the repetition rule biases against more than x media files associated with a particular artist from being sequenced within a playback interval of y.

9. The method of claim 8 wherein y is specified in elapsed time.

10. The method of claim 8 wherein y is specified in elapsed time discounted by intervals that the device is not playing media files.

11. The method of claim 8 wherein y is specified in terms of the number of media files played.

12. The method of claim 7 wherein the repetition rule biases against more than x media files associated with a particular recordings collection from being sequenced within a playback interval of y.

13. The method of claim 12 wherein y is specified in elapsed time.

14. The method of claim 12 wherein y is specified in elapsed time discounted by intervals that the device is not playing media files.

15. The method of claim 12 wherein y is specified in terms of the number of media files played.

16. The method of claim 7 wherein the repetition rule biases against more than x media files associated with a particular recording from being sequenced within a playback interval of y.

17. The method of claim 16 wherein y is specified in actual elapsed time.

18. The method of claim 16 wherein y is specified in elapsed time discounted by intervals that the device is not playing media files.

19. The method of claim 16 wherein y is specified in terms of the number of media files played.

20. The method of claim 16 wherein the bias decays as the playback interval approaches y.

21. The method of claim 20 wherein the bias decays exponentially according to the equation $N(t) = N_0 e^{-\lambda t}$ where t represents the interval, N_0 represents the maximum bias, and λ represents the rate of decay.

22. The method of claim 21 wherein N_0 and λ are chosen responsive to a target bias N(t') where t' represents the minimal acceptable interval.

23. The method of claim 6 wherein at least one of the attributes used to bias selection represents the popularity of the content of the media file

24. The method of claim 6 wherein at least one of the attributes used to bias selection represents the preference of the user for the content of the media file.

25. The method of claim 6 wherein at least one of the attributes used to bias selection represents an intensity metric of the content of the media file.

26. The method of claim 6 wherein at least one of the attributes used to bias selection represents a tempo metric of the content of the media file.

27. The method of claim 6 wherein at least one of the attributes used to bias selection represents an age metric of the content of the media file.

28. The method of claim 6 wherein at least one of the attributes used to bias selection is re-scaled according to a channel specific scaling function.

29. The method of claim 28 wherein the scaling function is parameterized by a channel specific value of the attribute that maps to a neutral bias.

30. The method of claim 29 wherein the scaling function is additionally parameterized by the values of the attribute mapping onto highly positive and highly negative bias.

31. The method of claim 30 wherein the scaling function is of the form $v' = b - e^{-\lambda v}$ where b represents the maximum bias and λ represents the rate of decay of that bias from high to neutral, and v represents the attribute value being scaled.

32. The method of claim 1 wherein sequencing for playback is responsive to user preferences.

33. The method of claim 32 wherein the user preferences include identifying the content of a media file as a favorite.

34. The method of claim 32 wherein the user preferences include associating a rating with the content of the media files.

35. The method of claim 32 wherein the user preferences include a setting to exclude the content of the media file from playback.

36. The method of claim 32 wherein the user preferences include a setting to exclude content associated with one or more excluding attributes of the content.

37. The method of claim 36 wherein an excluding attribute is an artist associated with the content.

38. The method of claim 1 wherein the decision to store or discard a track is responsive to one or more user preferences.

39. The method of claim 38 wherein the user preferences include identifying the content of a media file as a favorite.

40. The method of claim 38 wherein the user preferences include associating a rating with the content of the media files.

41. The method of claim 38 wherein the user preferences include a setting to exclude the content of the media file from playback.

42. The method of claim 38 wherein the user preferences include a setting to exclude content associated with one or more excluding attributes of the content.

43. The method of claim 38 wherein an excluding attribute is an artist associated with the content.

44. The method of claim 1 wherein the amount of storage devoted to the local files associated with a channel is allocated responsive to the characteristics of the channel.

45. The method of claim 5 wherein the amount of storage devoted to the local files associated with a channel is allocated responsive to the characteristics of the categories associated with the channel.

46. The method of claim 45 wherein the storage devoted to the local files associated with a category is fixed.

47. The method of claim 45 wherein the storage devoted to the local files associated with a category is fixed by means of a maximum number of associated files.

48. The method of claim 46 wherein the local files associated with a category in excess of the allocation of the category are removed from the device.

49. The method of claim 46 wherein the local files associated with a category in excess of the allocation of the category are assigned to a different category.

50. The method of claim 44 wherein the amount of storage devoted to the local files associated with a channel is allocated responsive to the anticipated share of device playback usage attributed to the channel.

51. The method of claim 44 wherein the amount of storage devoted to the local files associated with a category is allocated responsive to the characteristics of the category defining the rate at which content in the category may be repeated in playback sequencing.

52. The method of claim 44 wherein files in excess of the allocation are selected for removal responsive to the inverse of their anticipated order of playback by the device.

53. The method of claim 44 wherein files in excess of the allocation are selected for removal responsive to their quality of fit to the characteristics of the channel.

54. The method of claim 44 wherein files in excess of the allocation are selected for removal responsive to their recency of playback.

55. The method of claim 45 wherein files in excess of the allocation for the category are selected for removal responsive to the inverse of their anticipated order of playback by the device.

56. The method of claim 45 wherein files in excess of the allocation for the category are selected for removal responsive to their quality of fit to the characteristics of the category.

57. The method of claim 45 wherein files in excess of the allocation for the category are selected for removal responsive to their recency of playback.

58. A digital media playback device, comprising:

a receiver capable of receiving, over a data link, media files and file metadata defining attributes of the media files;

memory in which are stored ones of the media files characterized by file metadata consistent with one or more channel profiles; and

a processor operatively coupled to the memory, wherein the processor executes instructions stored in the memory to:

automatically store certain of the ones of the media files stored within the memory,

automatically delete certain of the ones of the media files stored within the memory, and

automatically sequence the ones of the media files stored within the memory for playback based upon one or more of the channel profiles.

59. The device as in claim 58 wherein the removal of the media files is triggered by a storage utilization profile of the device reaching pre-configured thresholds.

60. The device as in claim 59 wherein the process of removal of the media files seeks to achieve the configured target storage utilization.

61. The device of claim 58 wherein the media files removed are responsive to the inverse of the anticipated playback order of all of the media files on the device.

62. The device of claim 58 wherein the media files removed are responsive to the anticipated percent of total device playback allocated to the channel or category associated with the media files.

63. The device of claim 62 wherein the anticipated playback of a channel or category is adapted based on observations of actual content playback.

64. The device of claim 63 wherein the share of anticipated playback (termed the "duty cycle") is calculated using an adaptation is of the form $c_{t+1} = (o_{t+1} - c_t)\lambda + c_t$ where c_{t+1} is the adapted duty cycle at time t+1, o_{t+1} is the observed duty cycle at time t+1 and c_t is the adapted duty cycle at time t.

65. The device of claim 63 wherein the files to be removed from the device are responsive to the formula: $score = (r - 0.5) / (c_{station} \cdot c_{bucket})$ where r is the rank of the media file within the category, $c_{station}$ is the anticipated duty cycle of the channel, and c_{bucket} is the anticipated duty cycle of the category within the channel.

66. A digital media playback device, comprising:

a transceiver configured to send, over a data link, one or more channel profiles to a remote computer having access to the data link and to receive, over the data link, media files and file metadata based upon the channel profiles;

a memory; and

a processor operatively coupled to the memory, wherein the processor executes instructions stored in the memory to:

automatically store within the memory ones of the media files consistent with certain ones of the channel profiles,

automatically delete certain of the ones of the media files stored within the memory, and

automatically sequence the ones of the media files stored within the memory for playback based upon a selected one of the channel profiles.

67. A computer readable media including instructions for execution on a processor in a digital media player device, comprising instructions to:

receive at the device, over a data link, media files and file metadata, wherein the file metadata defines attributes of the media files;

store, in a memory of the device, ones of the media files characterized by file metadata consistent with one or more channel profiles maintained within the memory of the device; and

sequence the ones of the media files for playback based upon one or more of the channel profiles.

68. A system for providing content for playback comprising:

a plurality of devices configured to receive media files over a network connection; and

a content management system configured to provide said media files and metadata relating to a plurality of characteristics of said media files to said plurality of devices, wherein said plurality of devices are disposed to render said media based at least in part on a set of user specific criteria in conjunction with a set of content specific criteria.

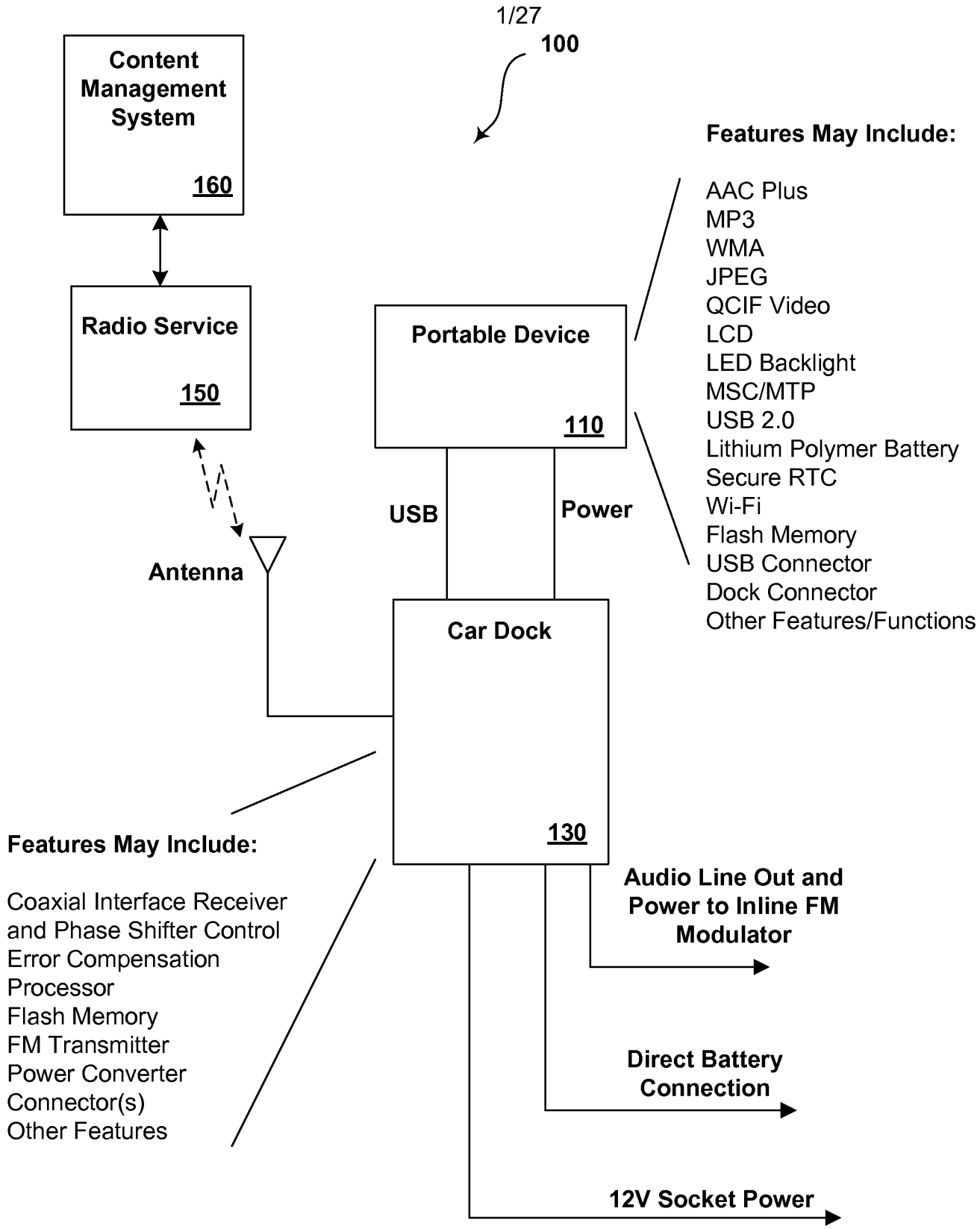

100
Content Management System
160
Radio Service
150
Antenna
Portable Device
110
USB
Power
Car Dock
130
Features May Include:
AAC Plus
MP3
WMA
JPEG
QCIF Video
LCD
LED Backlight
MSC/MTP
USB 2.0
Lithium Polymer Battery
Secure RTC
Wi-Fi
Flash Memory
USB Connector
Dock Connector
Other Features/Functions
Features May Include:
Coaxial Interface Receiver and Phase Shifter Control
Error Compensation Processor
Flash Memory
FM Transmitter
Power Converter
Connector(s)
Other Features
Audio Line Out and Power to Inline FM Modulator
Direct Battery Connection
12V Socket Power

FIG. 1

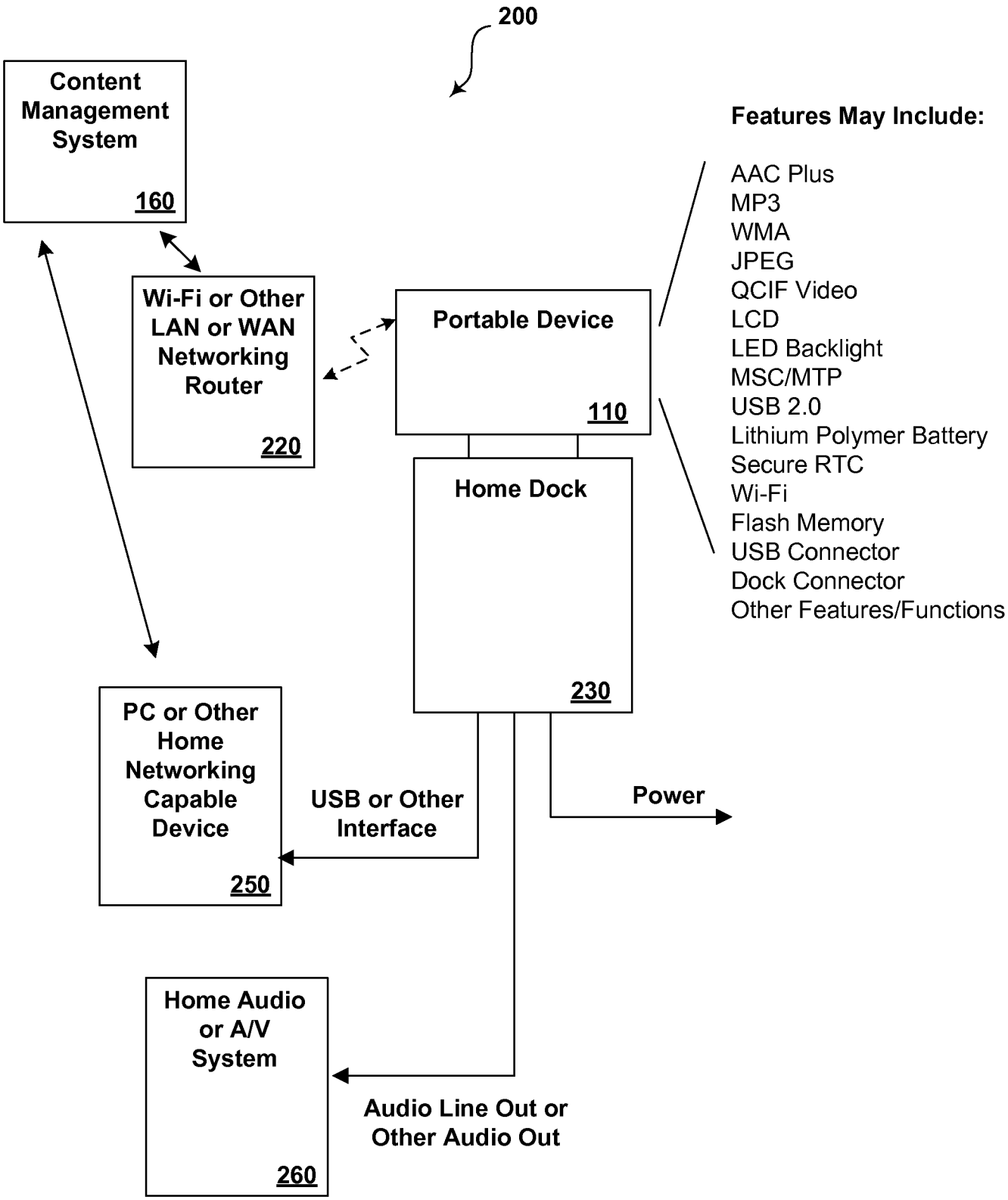

200
Content Management System
160
Wi-Fi or Other LAN or WAN Networking Router
220
Portable Device
110
Home Dock
230
PC or Other Home Networking Capable Device
250
USB or Other Interface
Power
Home Audio or A/V System
260
Audio Line Out or Other Audio Out
Features May Include:
AAC Plus
MP3
WMA
JPEG
QCIF Video
LCD
LED Backlight
MSC/MTP
USB 2.0
Lithium Polymer Battery
Secure RTC
Wi-Fi
Flash Memory
USB Connector
Dock Connector
Other Features/Functions

FIG. 2

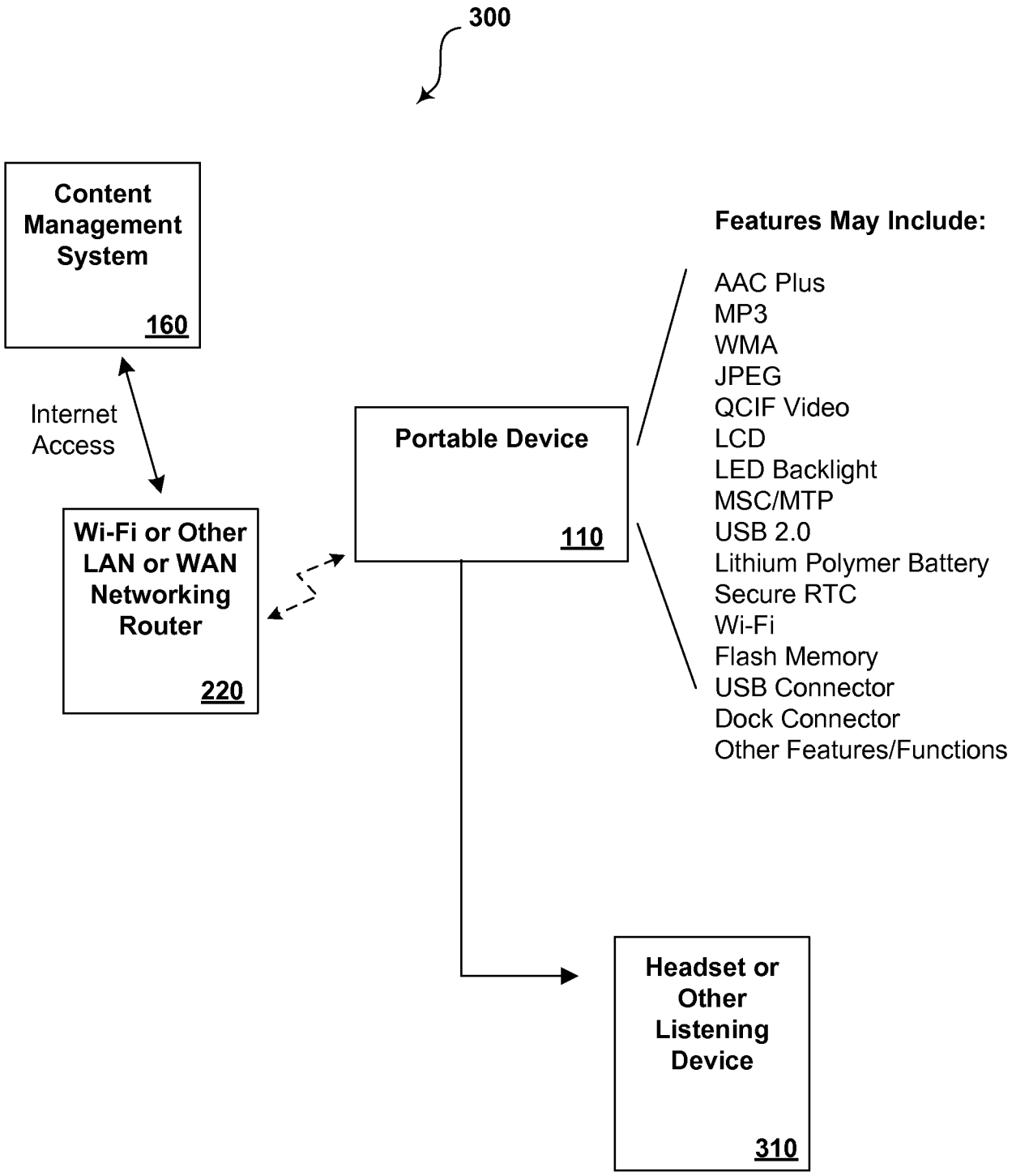

300
Content Management System
160
Internet Access
Wi-Fi or Other LAN or WAN Networking Router
220
Portable Device
110
Headset or Other Listening Device
310
Features May Include:
AAC Plus
MP3
WMA
JPEG
QCIF Video
LCD
LED Backlight
MSC/MTP
USB 2.0
Lithium Polymer Battery
Secure RTC
Wi-Fi
Flash Memory
USB Connector
Dock Connector
Other Features/Functions

FIG. 3

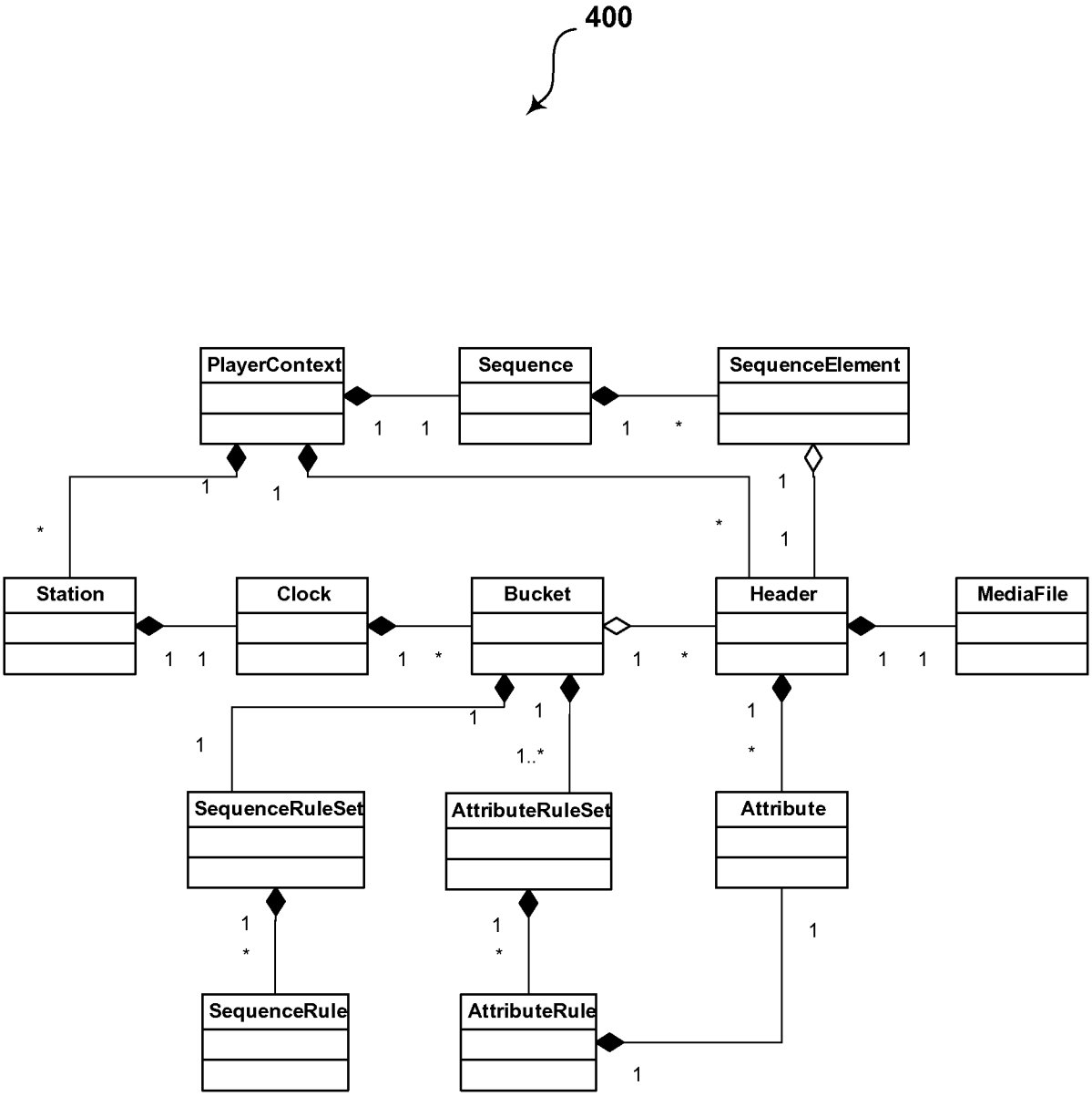

400
PlayerContext
Sequence
SequenceElement
Station
Clock
Bucket
Header
MediaFile
SequenceRuleSet
AttributeRuleSet
Attribute
SequenceRule
AttributeRule
1
*
1..*

FIG. 4

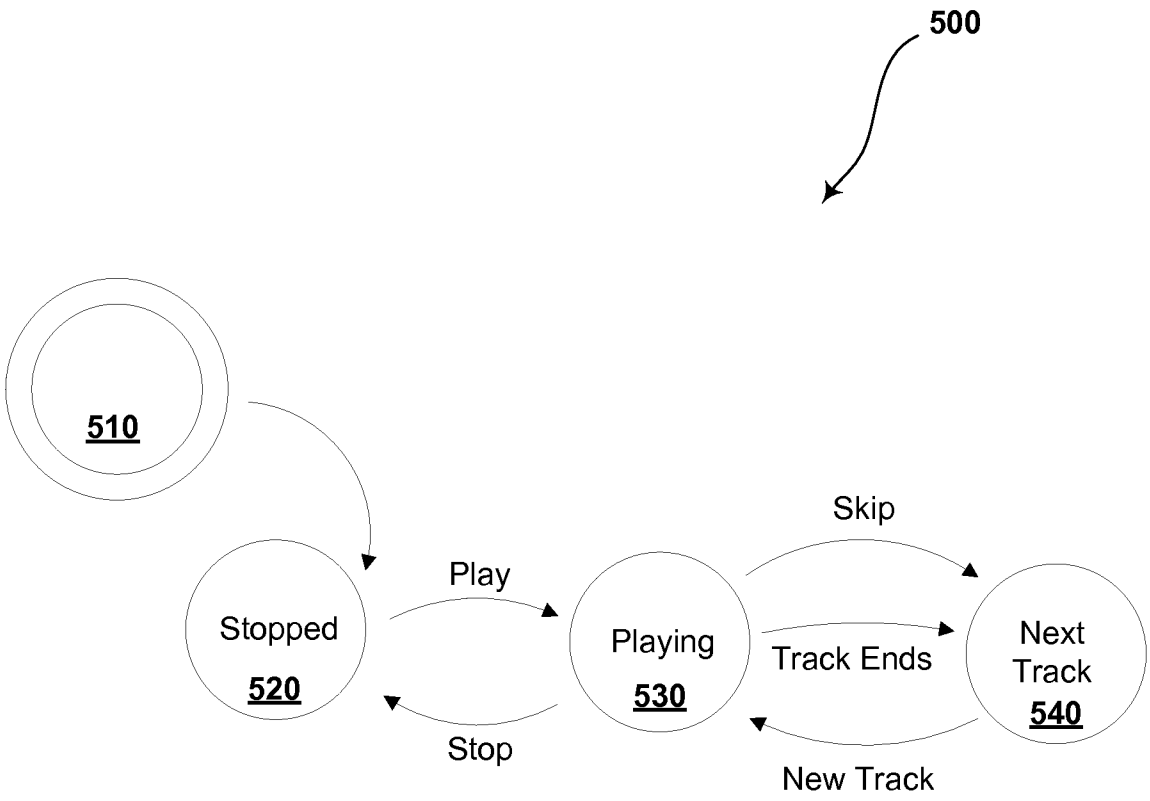

500
510
Stopped
520
Play
Stop
Playing
530
Skip
Track Ends
New Track
Next
Track
540

FIG. 5

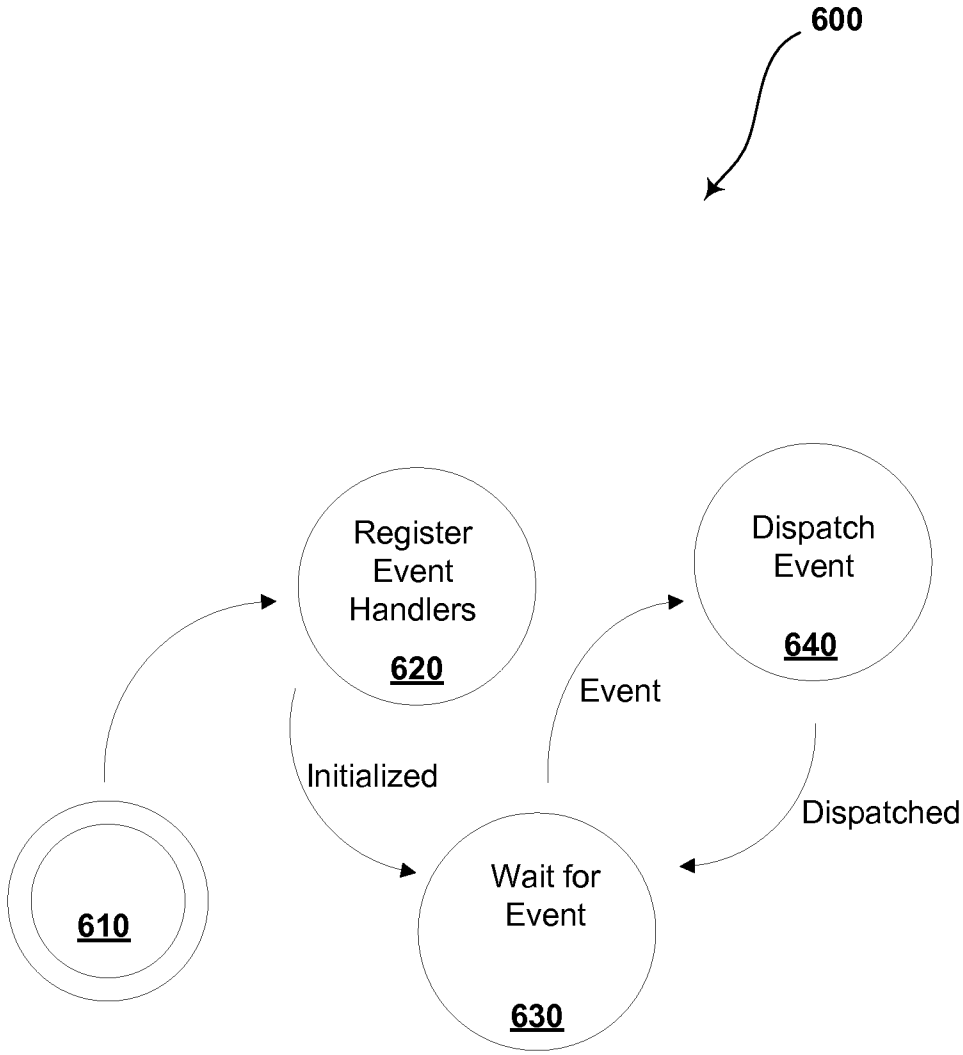

600
Register Event Handlers
620
Dispatch Event
640
Event
Initialized
Dispatched
610
Wait for Event
630

FIG. 6

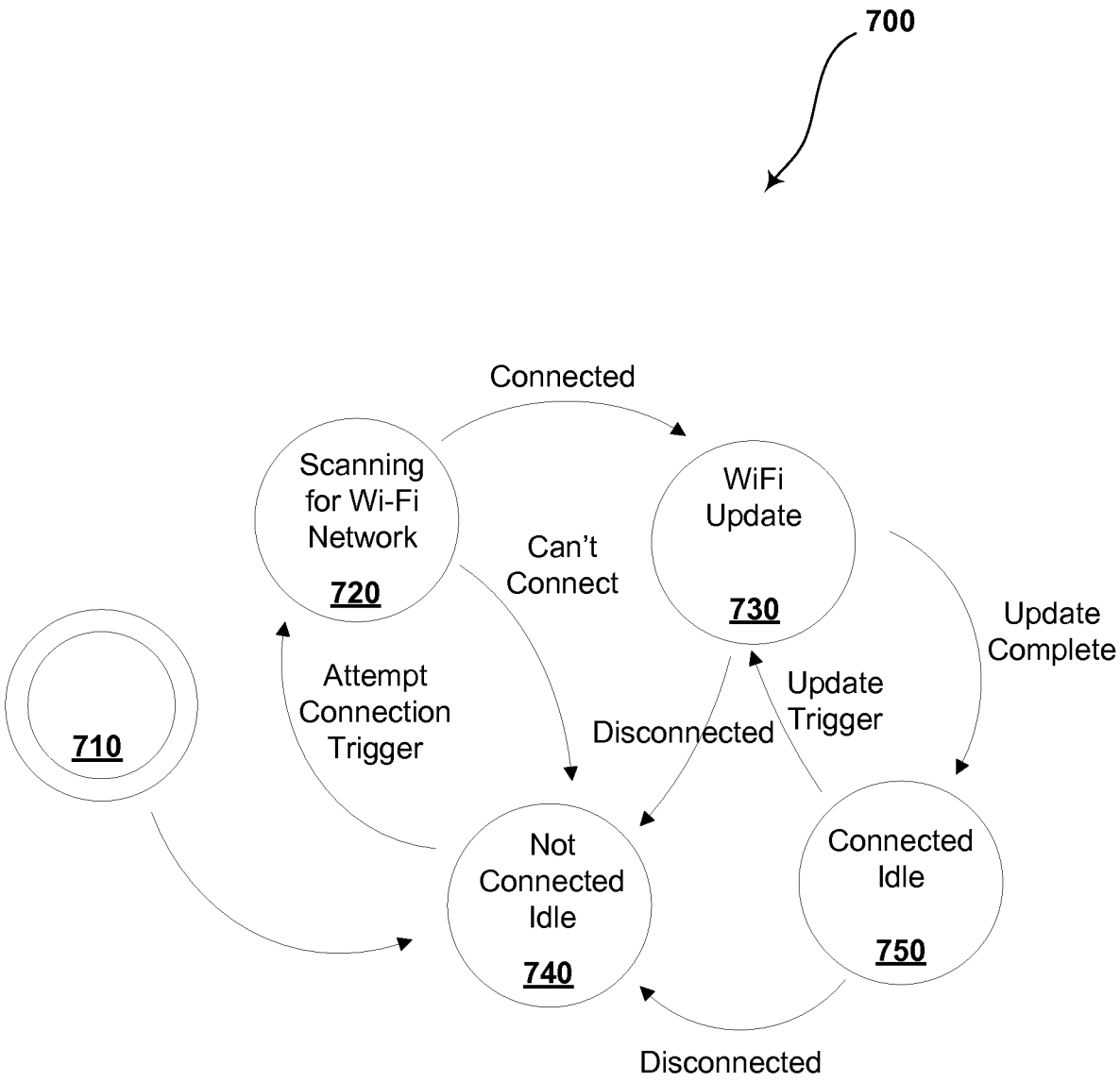

700
Connected
Scanning for Wi-Fi Network
720
WiFi Update
730
Can't Connect
Update Complete
Attempt Connection Trigger
Update Trigger
Disconnected
710
Not Connected Idle
740
Connected Idle
750
Disconnected

FIG. 7

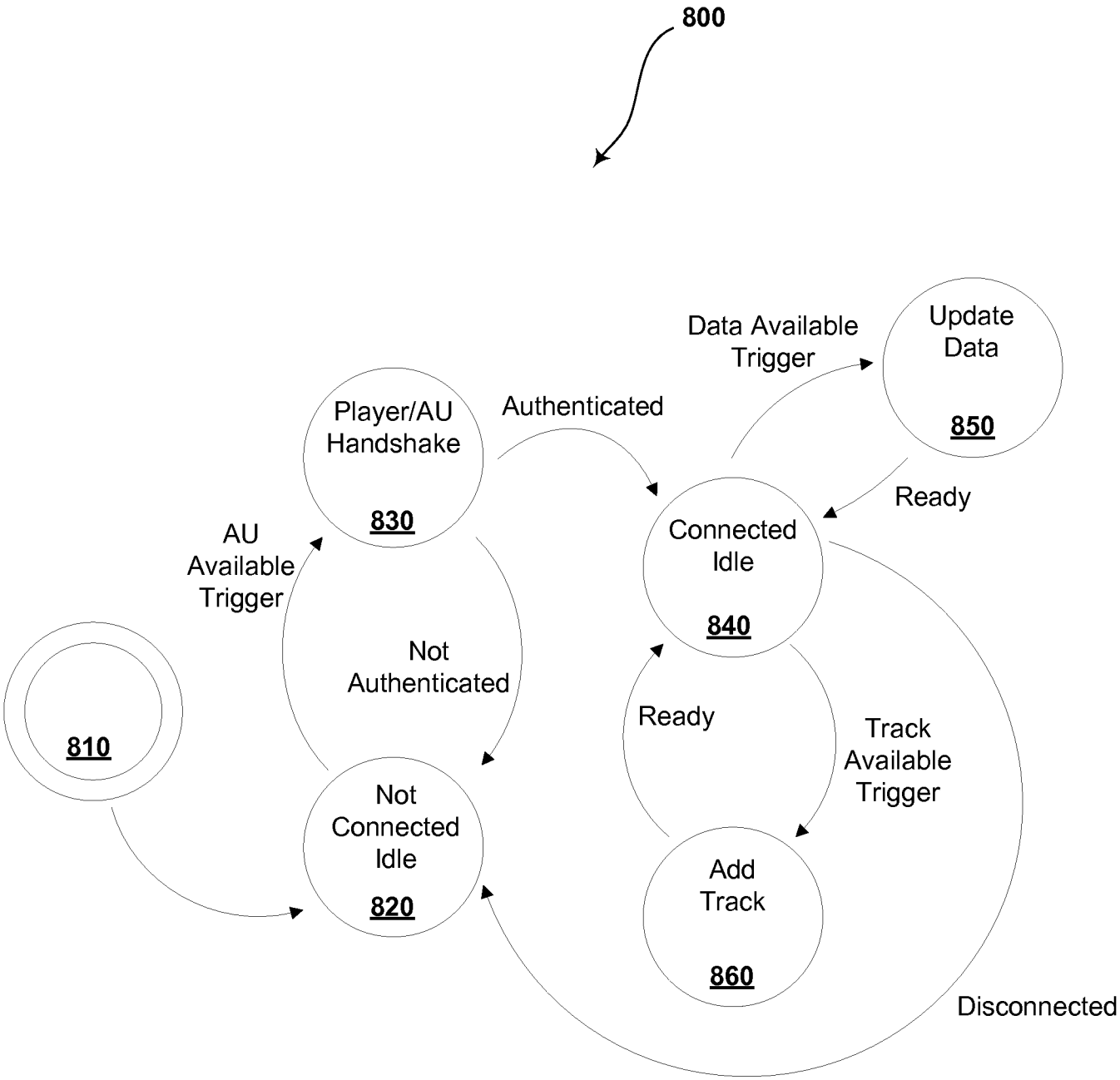

800
Data Available Trigger
Update Data
850
Player/AU Handshake
830
Authenticated
Ready
AU Available Trigger
Connected Idle
840
Not Authenticated
810
Ready
Track Available Trigger
Not Connected Idle
820
Add Track
860
Disconnected

FIG. 8

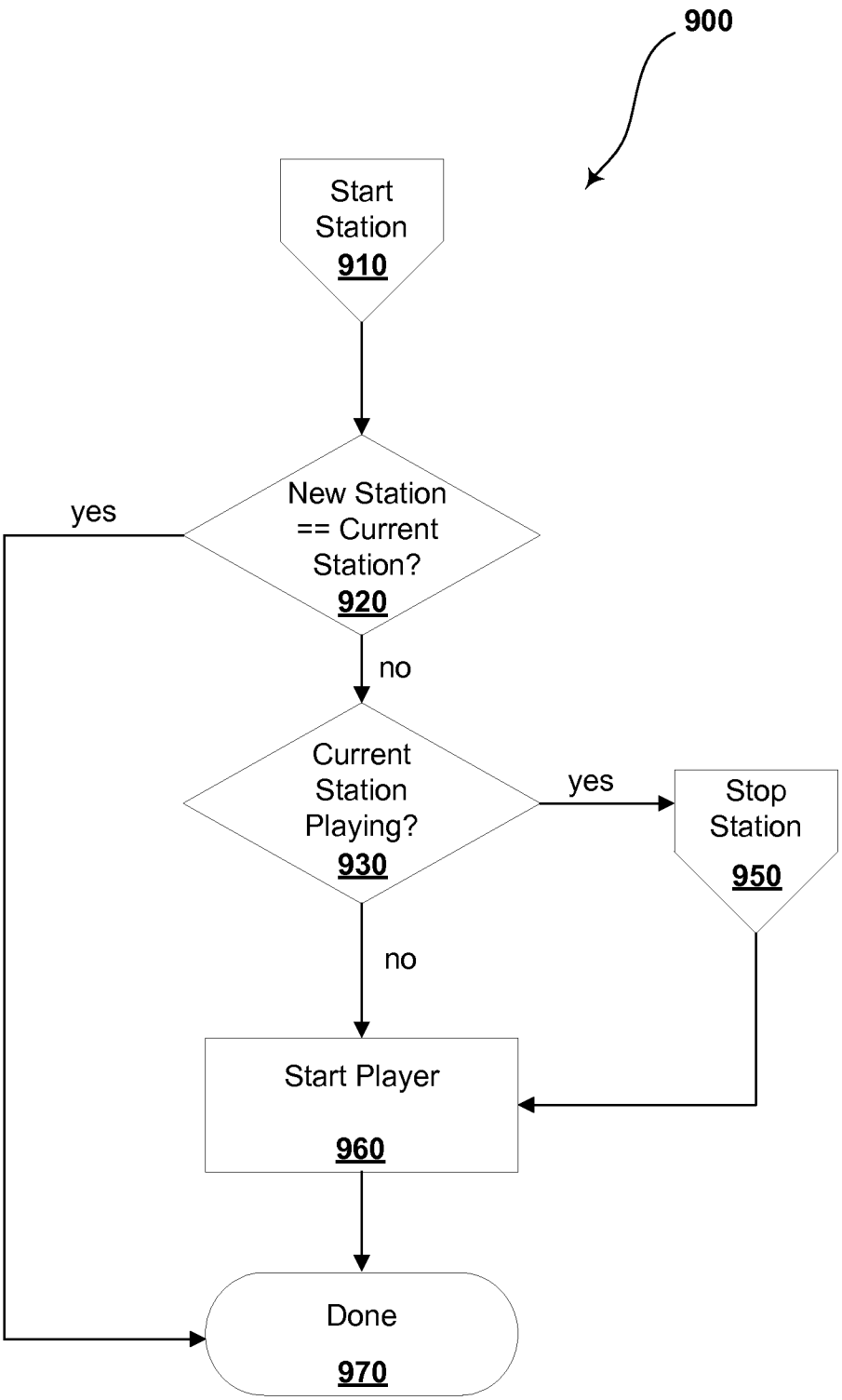

900
Start Station 910
New Station == Current Station? 920
yes
no
Current Station Playing? 930
yes
Stop Station 950
no
Start Player 960
Done 970

FIG. 9

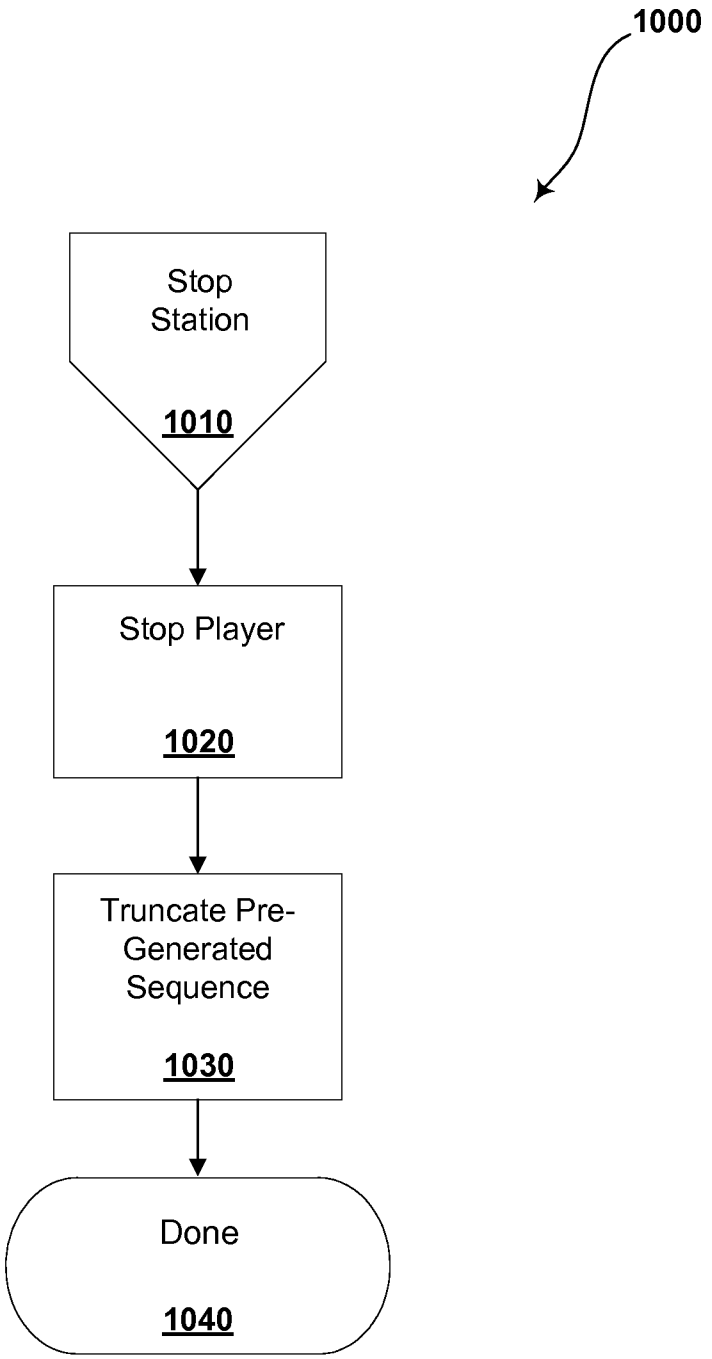

1000
Stop Station
1010
Stop Player
1020
Truncate Pre-Generated Sequence
1030
Done
1040

FIG. 10

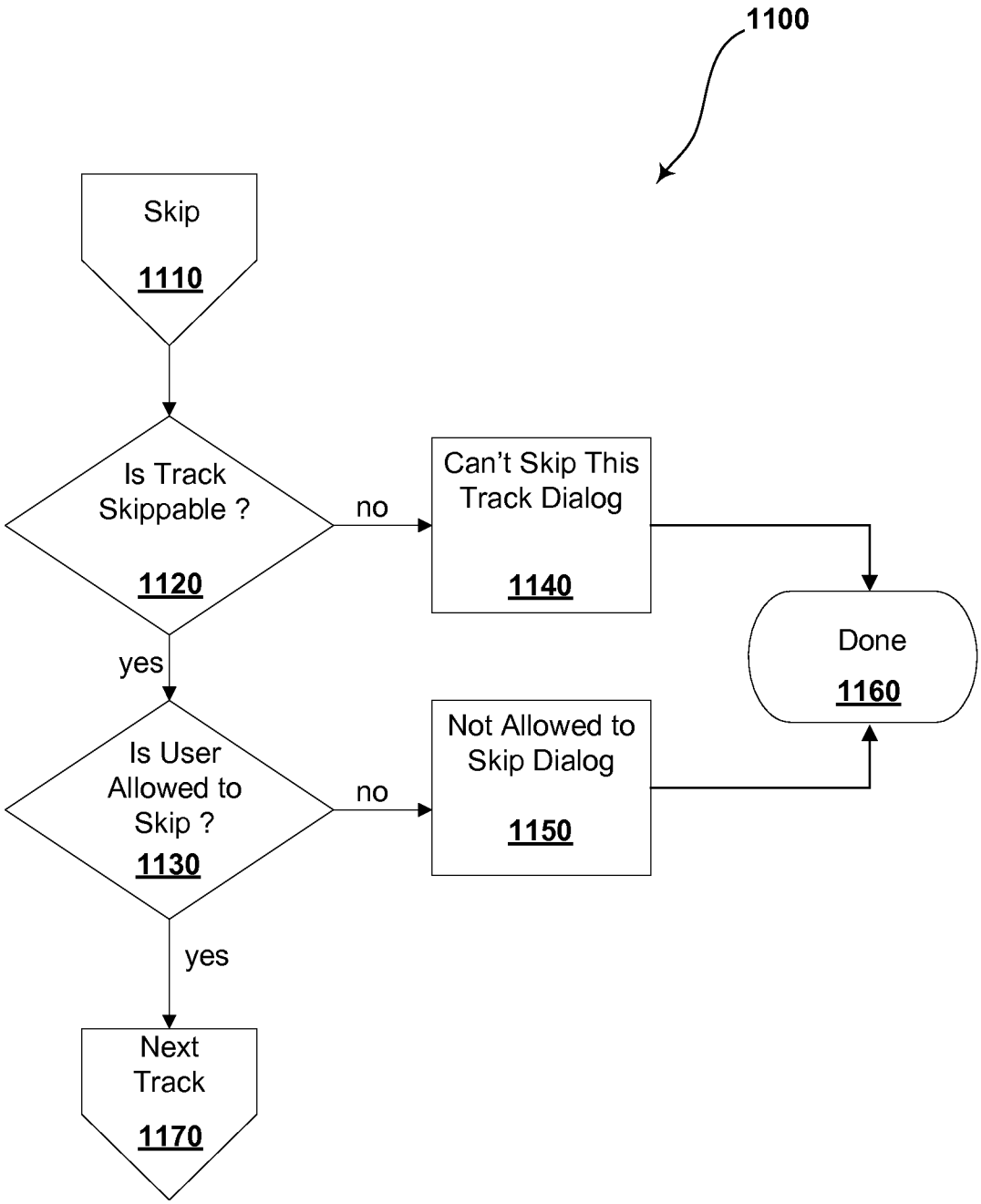

1100
Skip
1110
Is Track Skippable ?
1120
no
Can't Skip This Track Dialog
1140
yes
Done
1160
Is User Allowed to Skip ?
1130
no
Not Allowed to Skip Dialog
1150
yes
Next Track
1170

FIG. 11

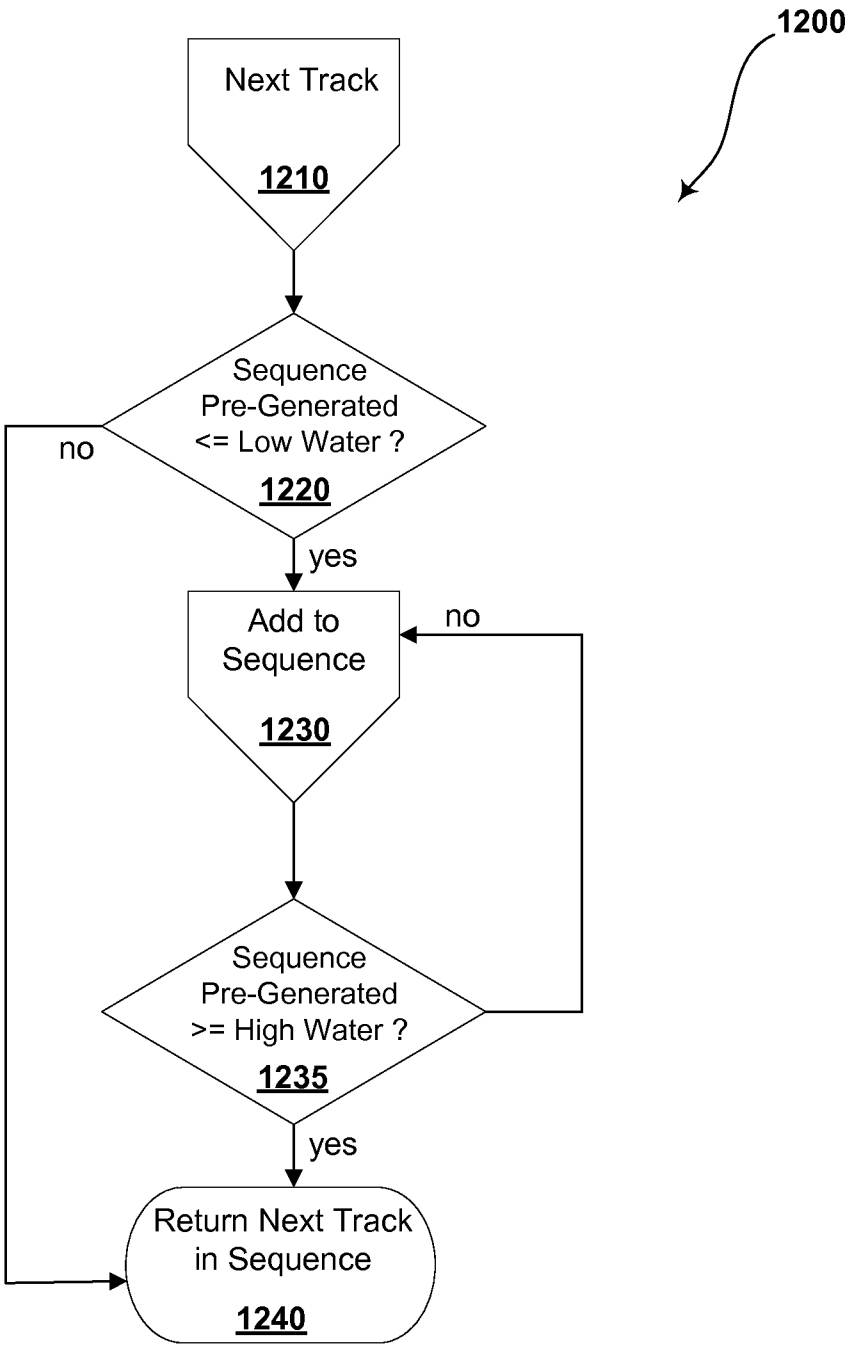

1200
Next Track
1210
Sequence
Pre-Generated
<= Low Water ?
1220
no
yes
Add to
Sequence
1230
no
Sequence
Pre-Generated
>= High Water ?
1235
yes
Return Next Track
in Sequence
1240

FIG. 12

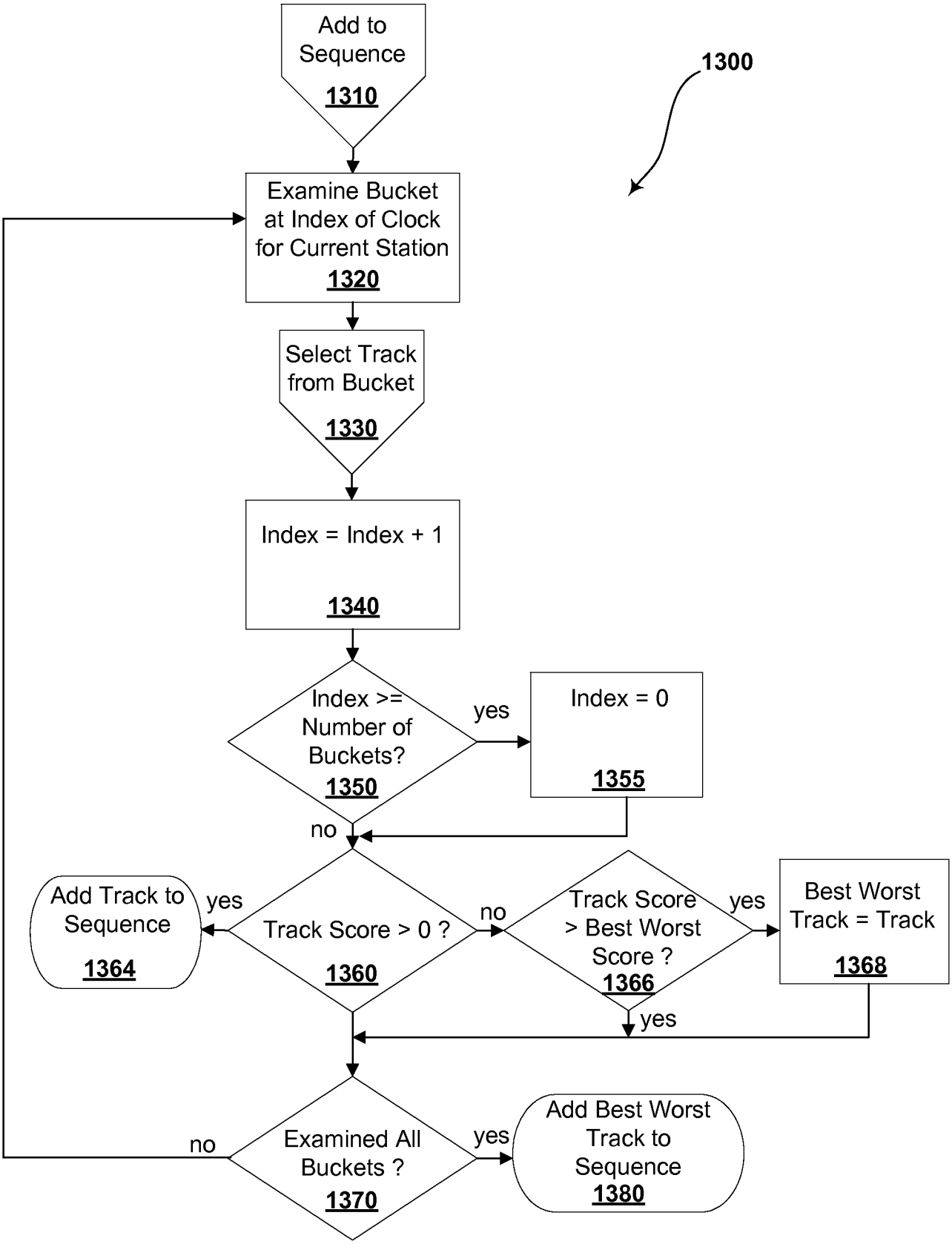

Add to Sequence
1310
1300
Examine Bucket at Index of Clock for Current Station
1320
Select Track from Bucket
1330
Index = Index + 1
1340
Index >= Number of Buckets?
1350
yes
Index = 0
1355
no
Add Track to Sequence
1364
yes
Track Score > 0 ?
1360
no
Track Score > Best Worst Score ?
1366
yes
Best Worst Track = Track
1368
yes
Examined All Buckets ?
1370
no
yes
Add Best Worst Track to Sequence
1380

FIG. 13

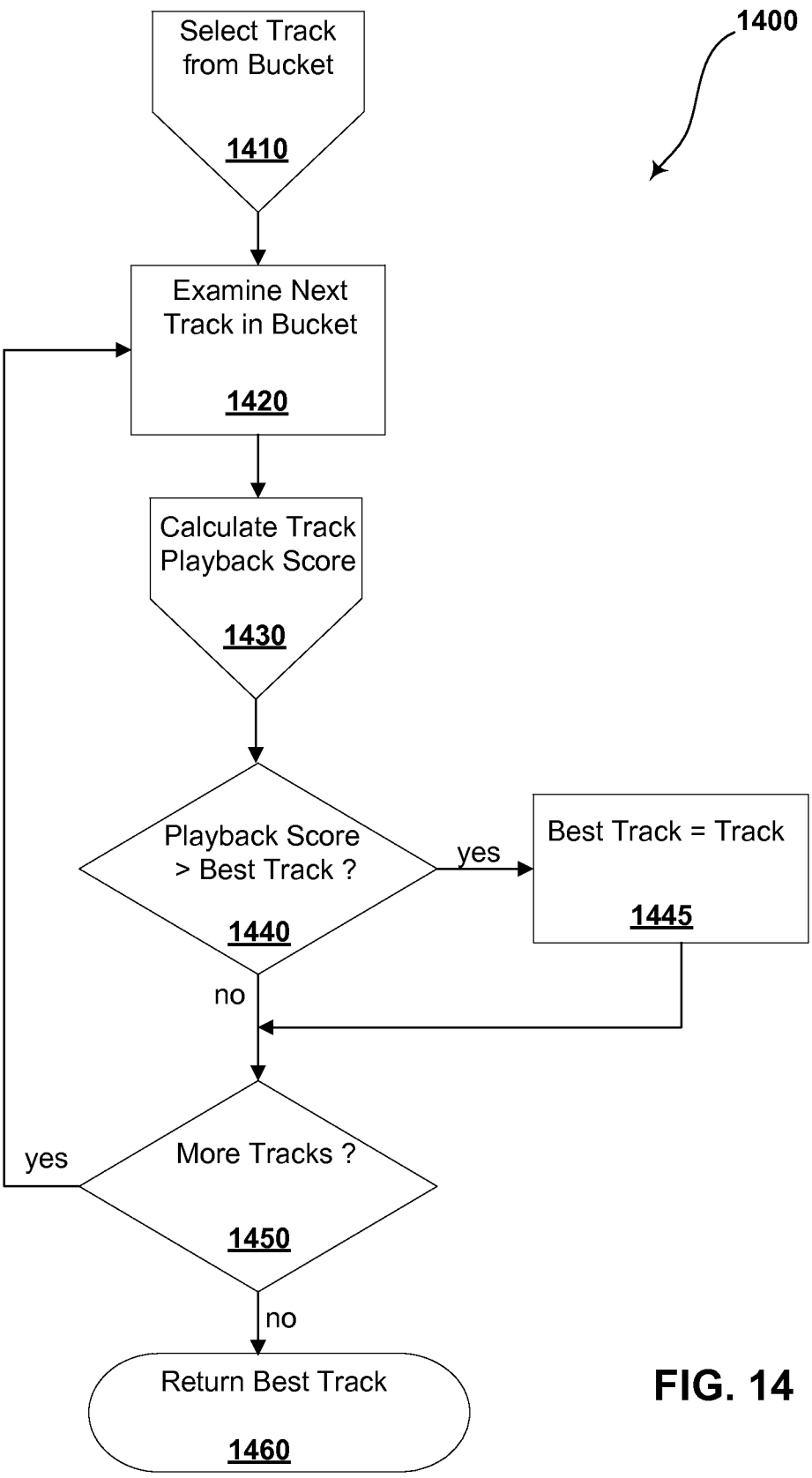

1400
Select Track from Bucket
1410
Examine Next Track in Bucket
1420
Calculate Track Playback Score
1430
Playback Score > Best Track ?
1440
yes
Best Track = Track
1445
no
More Tracks ?
1450
yes
no
Return Best Track
1460

FIG. 14

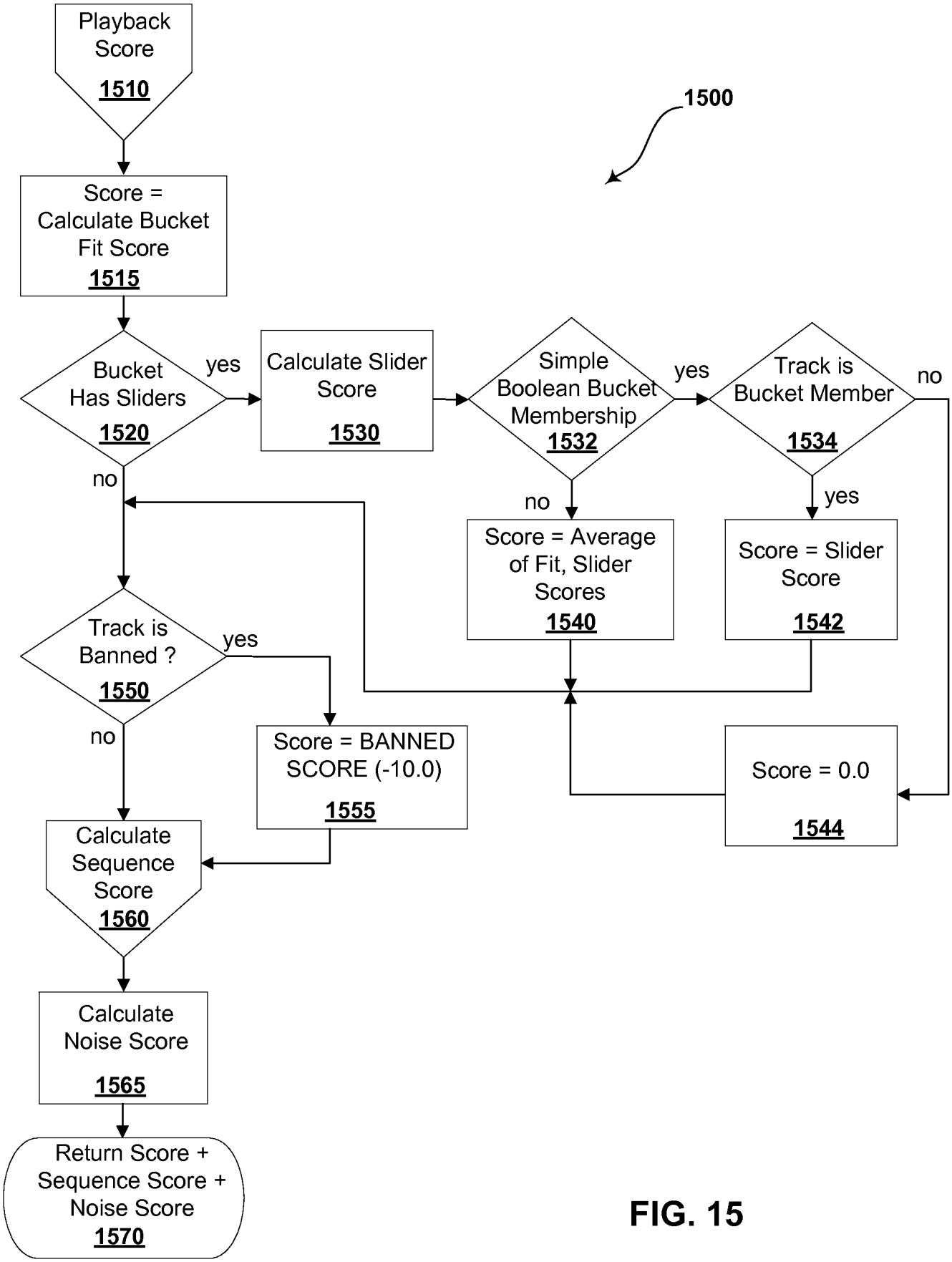
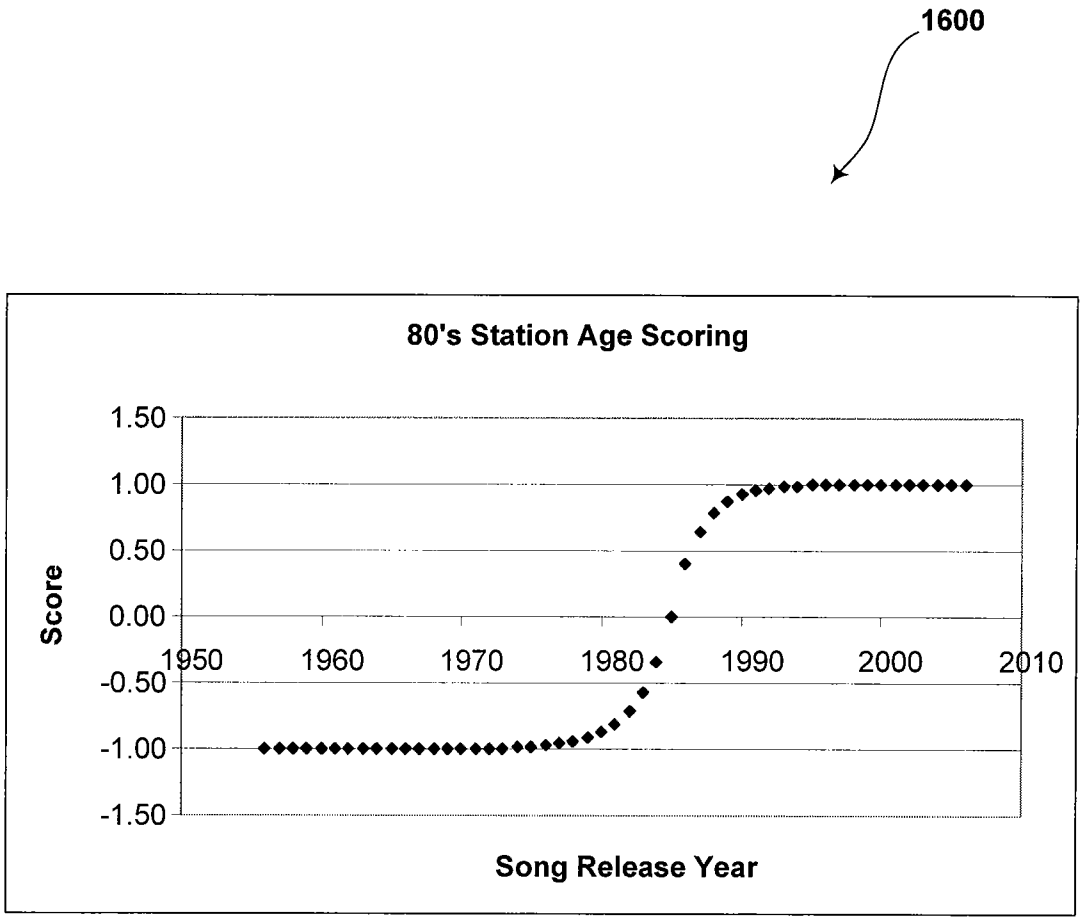

Playback Score
1510
1500
Score = Calculate Bucket Fit Score
1515
Bucket Has Sliders
1520
yes
Calculate Slider Score
1530
Simple Boolean Bucket Membership
1532
yes
Track is Bucket Member
1534
no
no
yes
Score = Average of Fit, Slider Scores
1540
Score = Slider Score
1542
Track is Banned ?
1550
yes
no
Score = BANNED SCORE (-10.0)
1555
Score = 0.0
1544
Calculate Sequence Score
1560
Calculate Noise Score
1565
Return Score + Sequence Score + Noise Score
1570

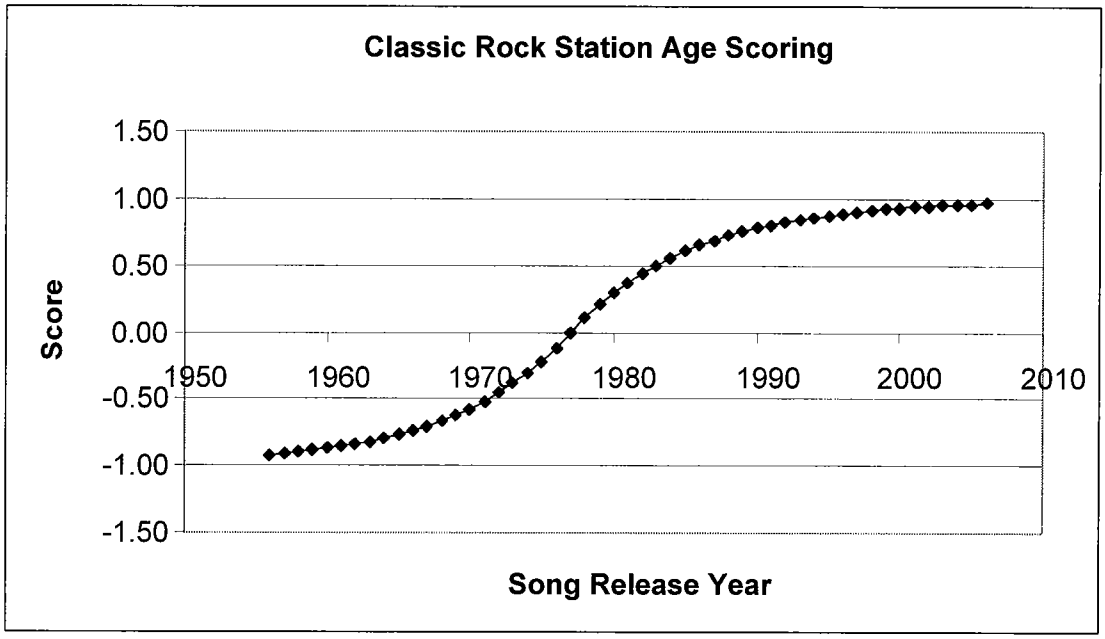
FIG. 15



Example Age scoring for an 80's station: Min=1980, Median=1985, Max=1989

FIG. 16

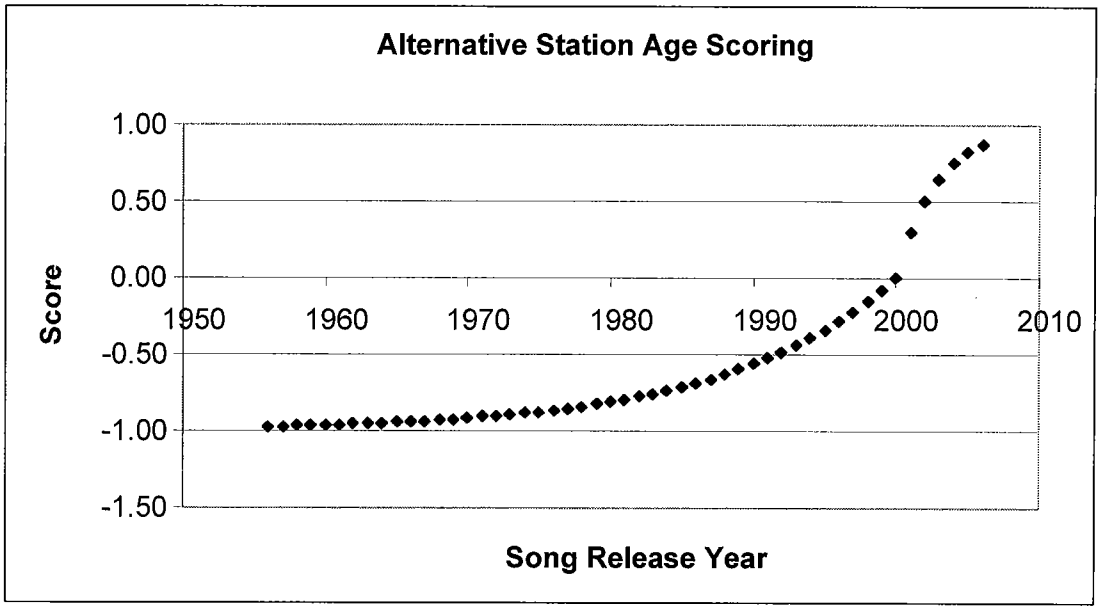
1700



Age scoring for a 'Classic Rock' station: Min=1960, Median=1977, Max=1995

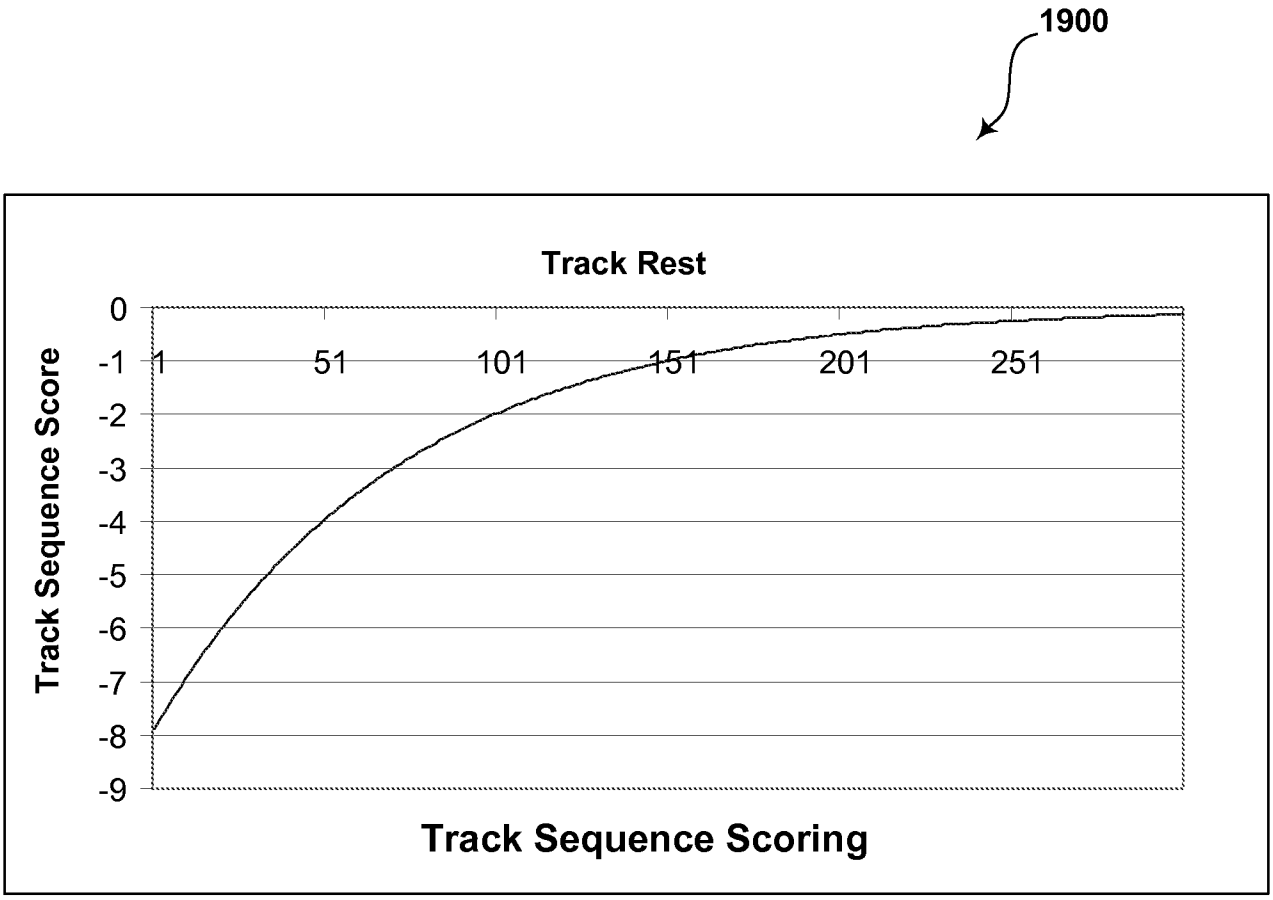
FIG. 17

1800



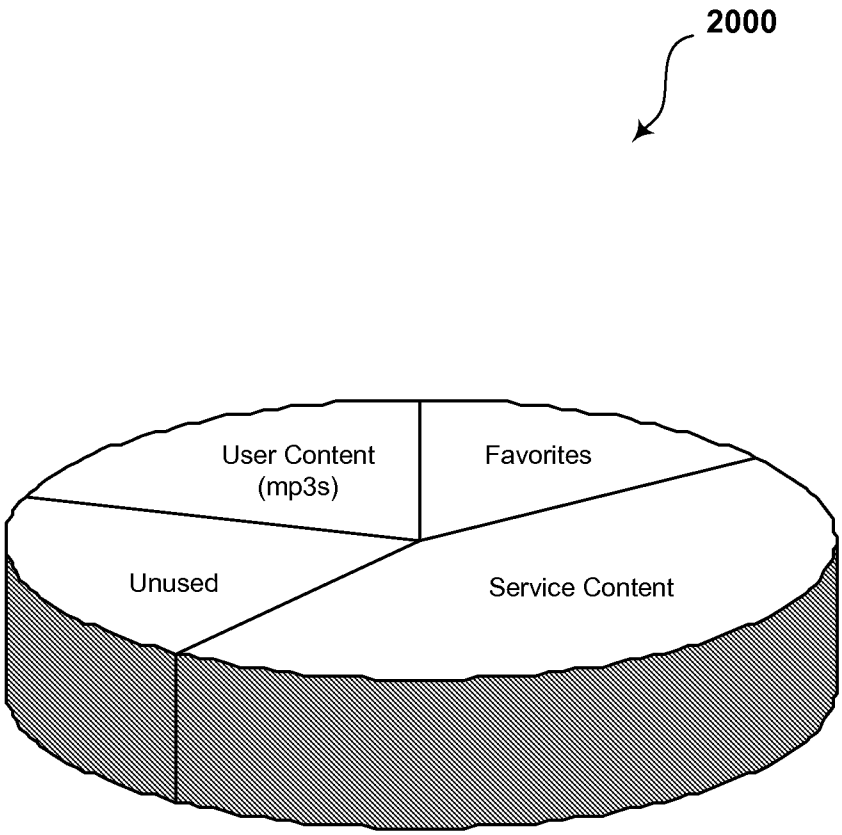
Age scoring for an 'Alternative' Station: Min=1975, Median=2000, Max=2006

FIG. 18



Sequence score as a function of track rest.
Target Rest is 300, Minimum Rest is 150. Noise floor is .0125.

FIG. 19



An Exemplary Allocation of Inventory Pools

FIG. 20

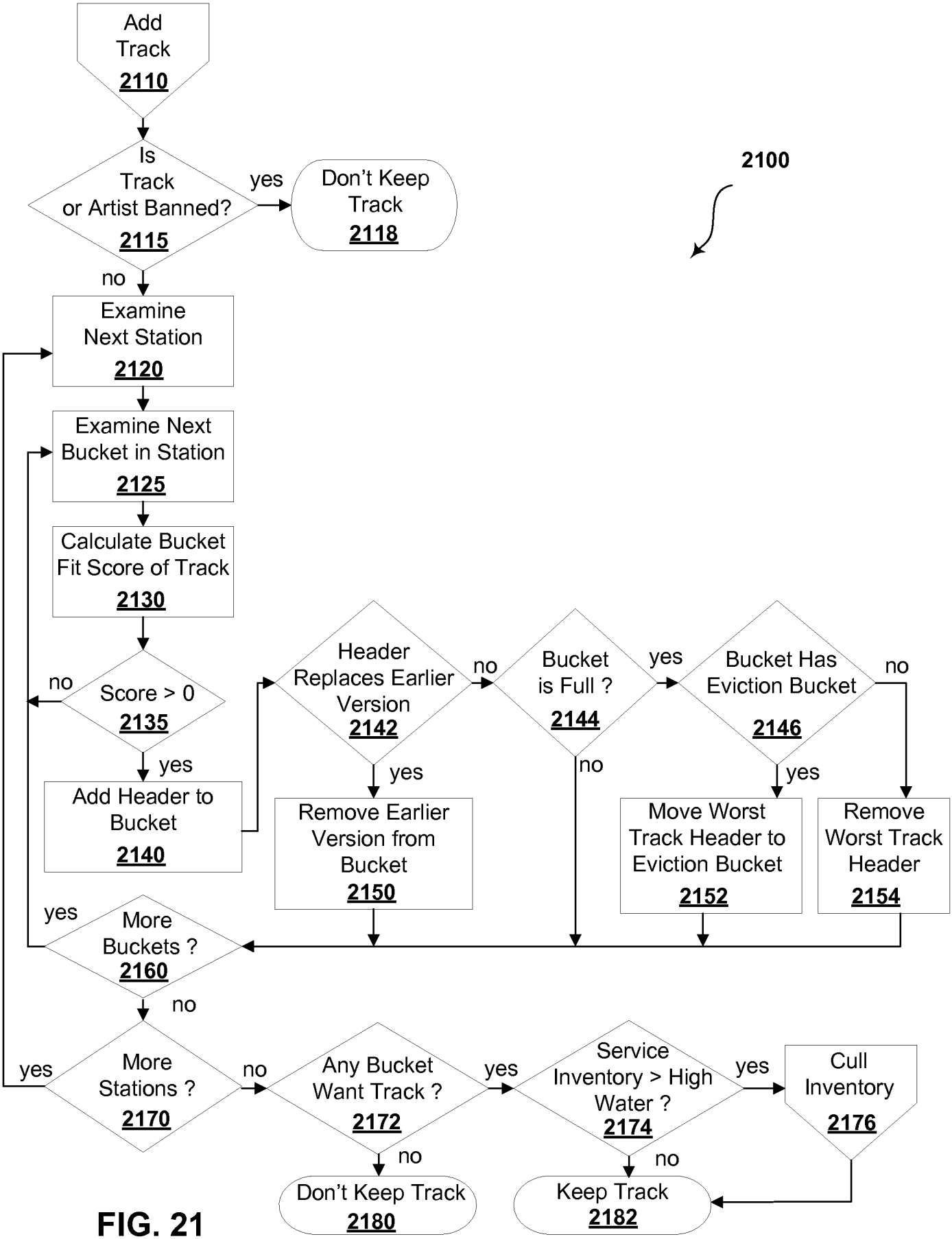

Add Track 2110
Is Track or Artist Banned? 2115
yes
Don't Keep Track 2118
2100
no
Examine Next Station 2120
Examine Next Bucket in Station 2125
Calculate Bucket Fit Score of Track 2130
Score > 0 2135
no
yes
Add Header to Bucket 2140
Header Replaces Earlier Version 2142
no
yes
Remove Earlier Version from Bucket 2150
Bucket is Full ? 2144
yes
no
Bucket Has Eviction Bucket 2146
yes
no
Move Worst Track Header to Eviction Bucket 2152
Remove Worst Track Header 2154
yes
More Buckets ? 2160
no
yes
More Stations ? 2170
no
Any Bucket Want Track ? 2172
yes
no
Don't Keep Track 2180
Service Inventory > High Water ? 2174
yes
no
Cull Inventory 2176
Keep Track 2182

FIG. 21

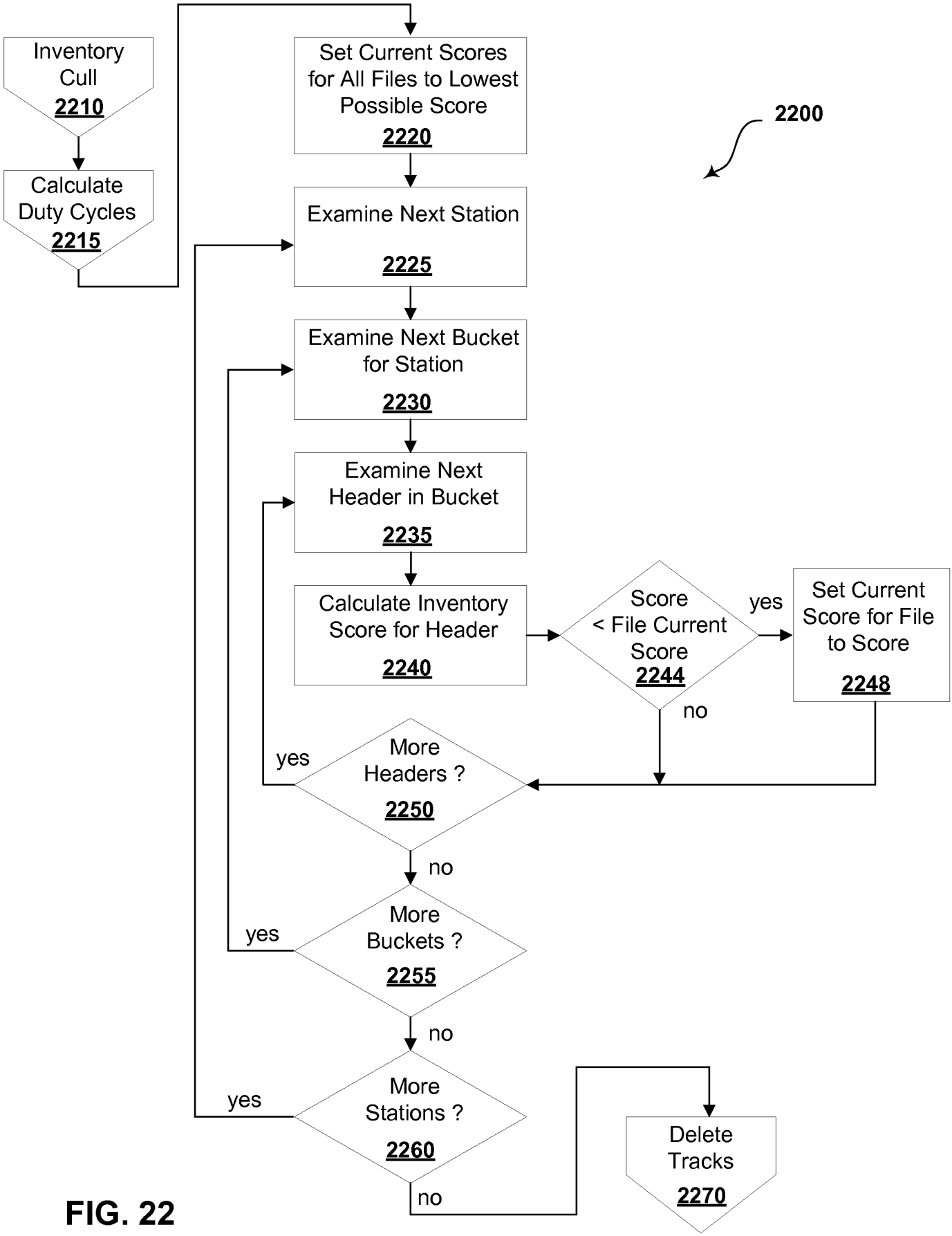
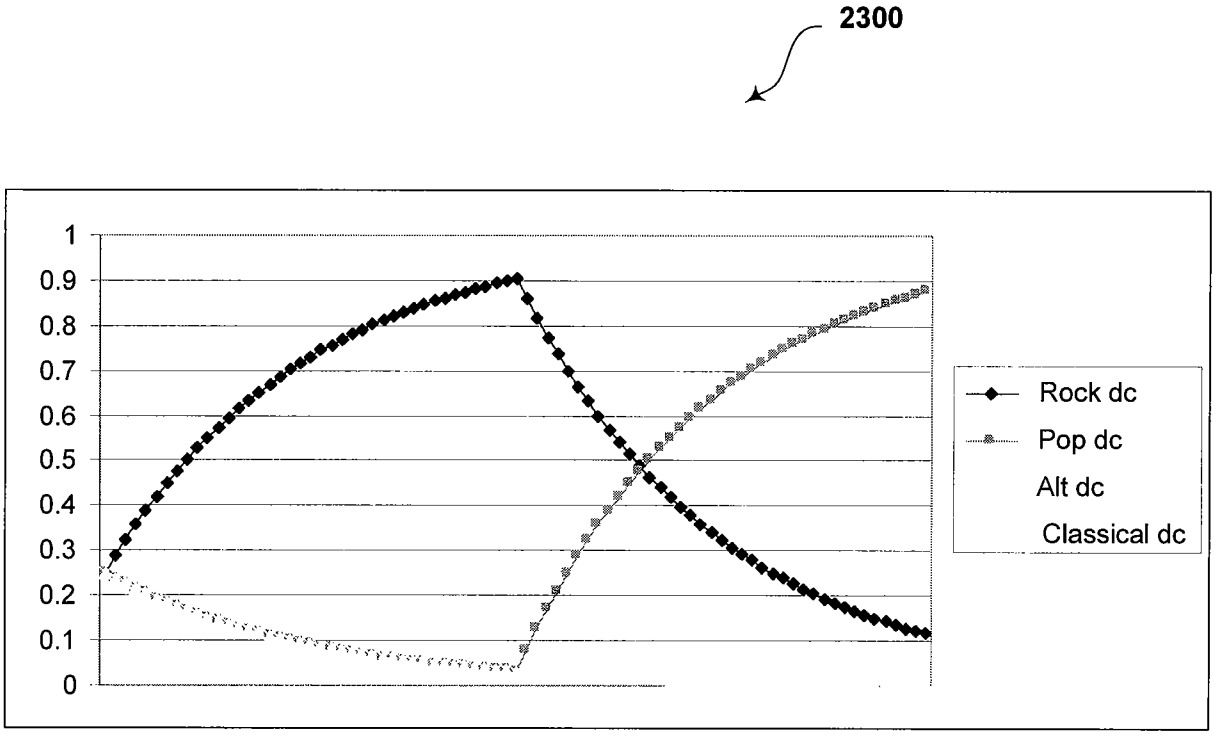

Inventory Cull 2210
Calculate Duty Cycles 2215
Set Current Scores for All Files to Lowest Possible Score 2220
2200
Examine Next Station 2225
Examine Next Bucket for Station 2230
Examine Next Header in Bucket 2235
Calculate Inventory Score for Header 2240
Score < File Current Score 2244
yes
Set Current Score for File to Score 2248
no
More Headers ? 2250
yes
no
More Buckets ? 2255
yes
no
More Stations ? 2260
yes
no
Delete Tracks 2270

FIG. 22



Duty Cycle Adaptation

FIG. 23

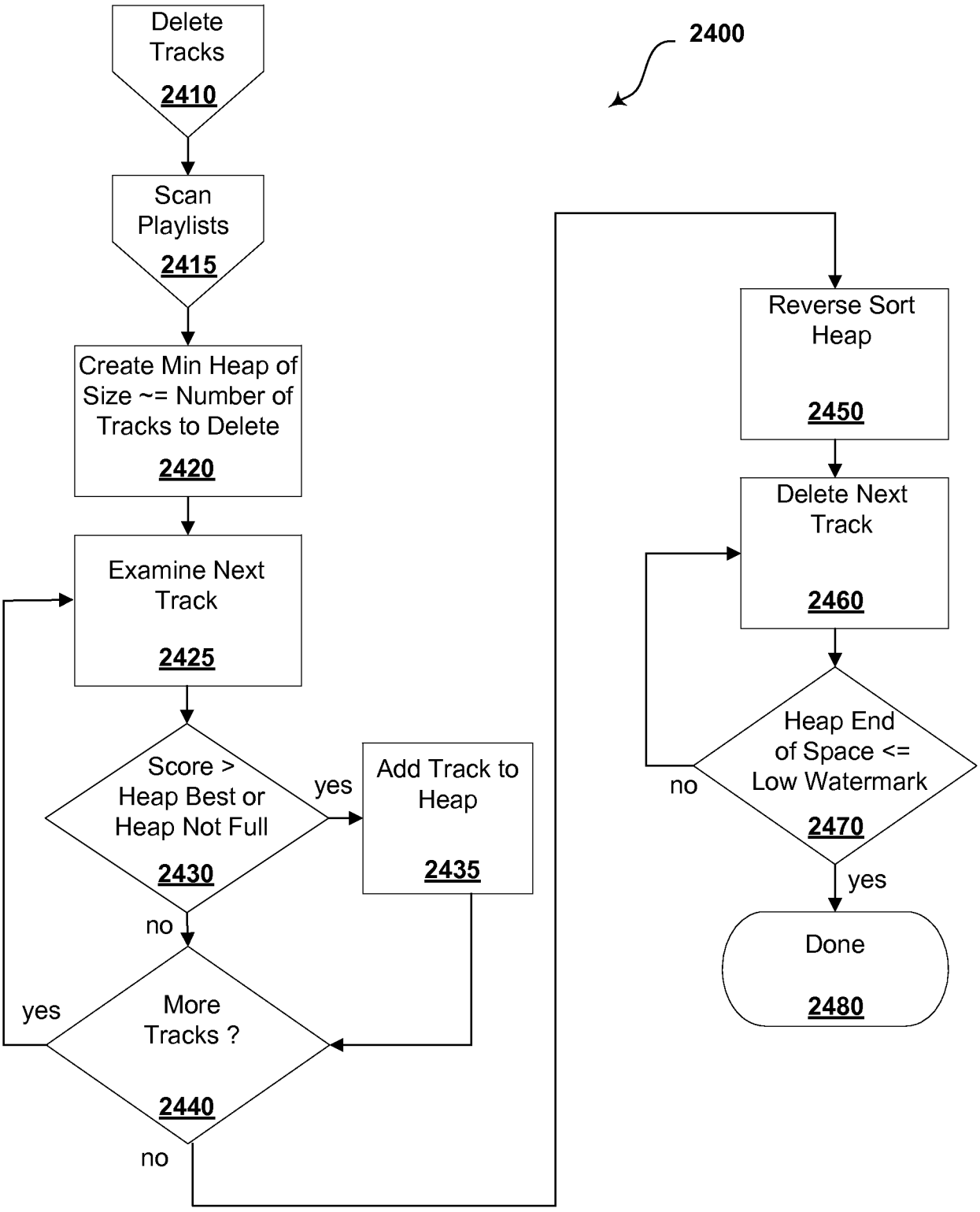

2400
Delete Tracks
2410
Scan Playlists
2415
Create Min Heap of Size ~= Number of Tracks to Delete
2420
Examine Next Track
2425
Score > Heap Best or Heap Not Full
2430
yes
Add Track to Heap
2435
no
More Tracks ?
2440
yes
no
Reverse Sort Heap
2450
Delete Next Track
2460
Heap End of Space <= Low Watermark
2470
no
yes
Done
2480

FIG. 24

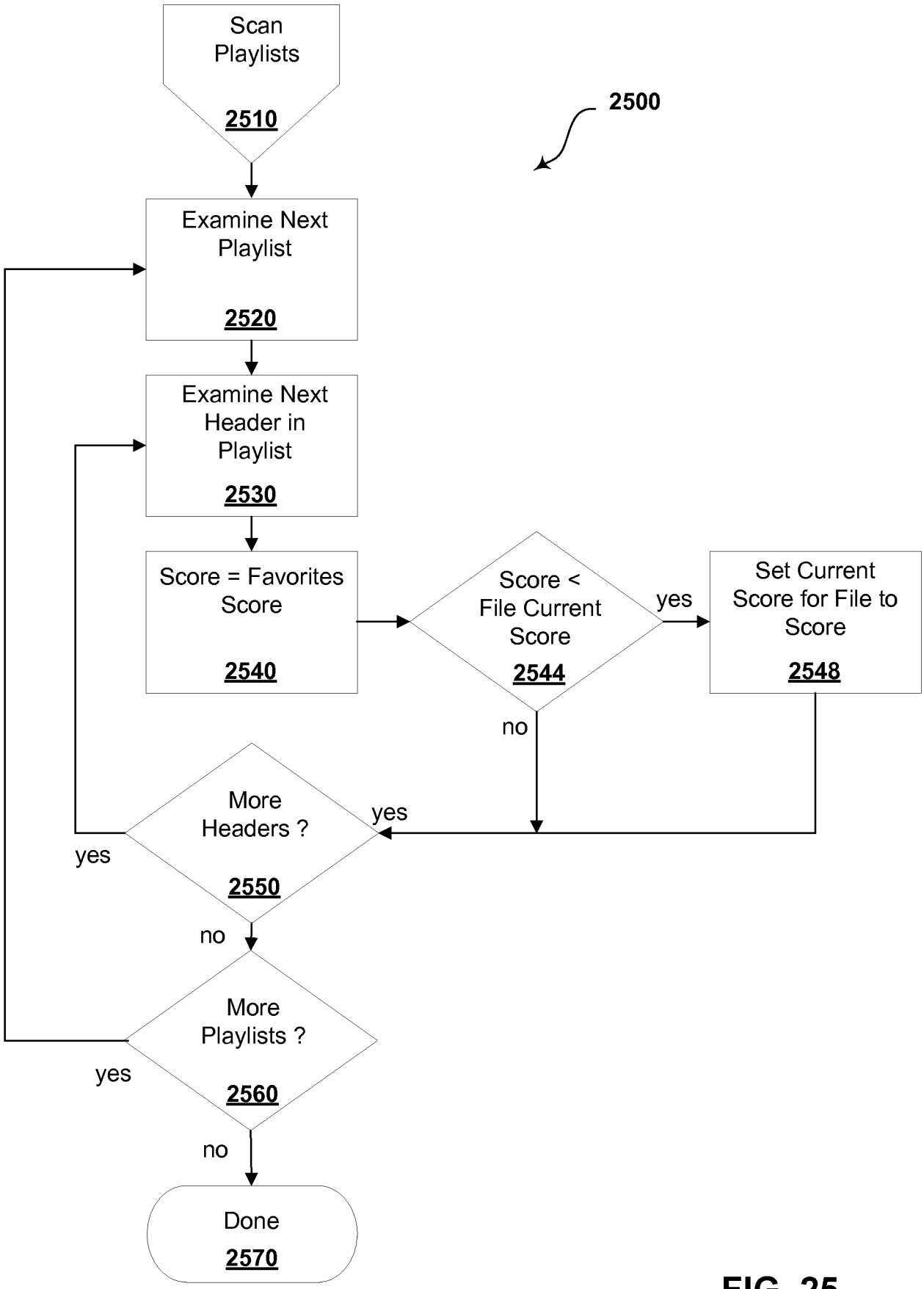

Scan Playlists
2510
2500
Examine Next Playlist
2520
Examine Next Header in Playlist
2530
Score = Favorites Score
2540
Score < File Current Score
2544
yes
Set Current Score for File to Score
2548
no
More Headers ?
2550
yes
yes
no
More Playlists ?
2560
yes
no
Done
2570

FIG. 25

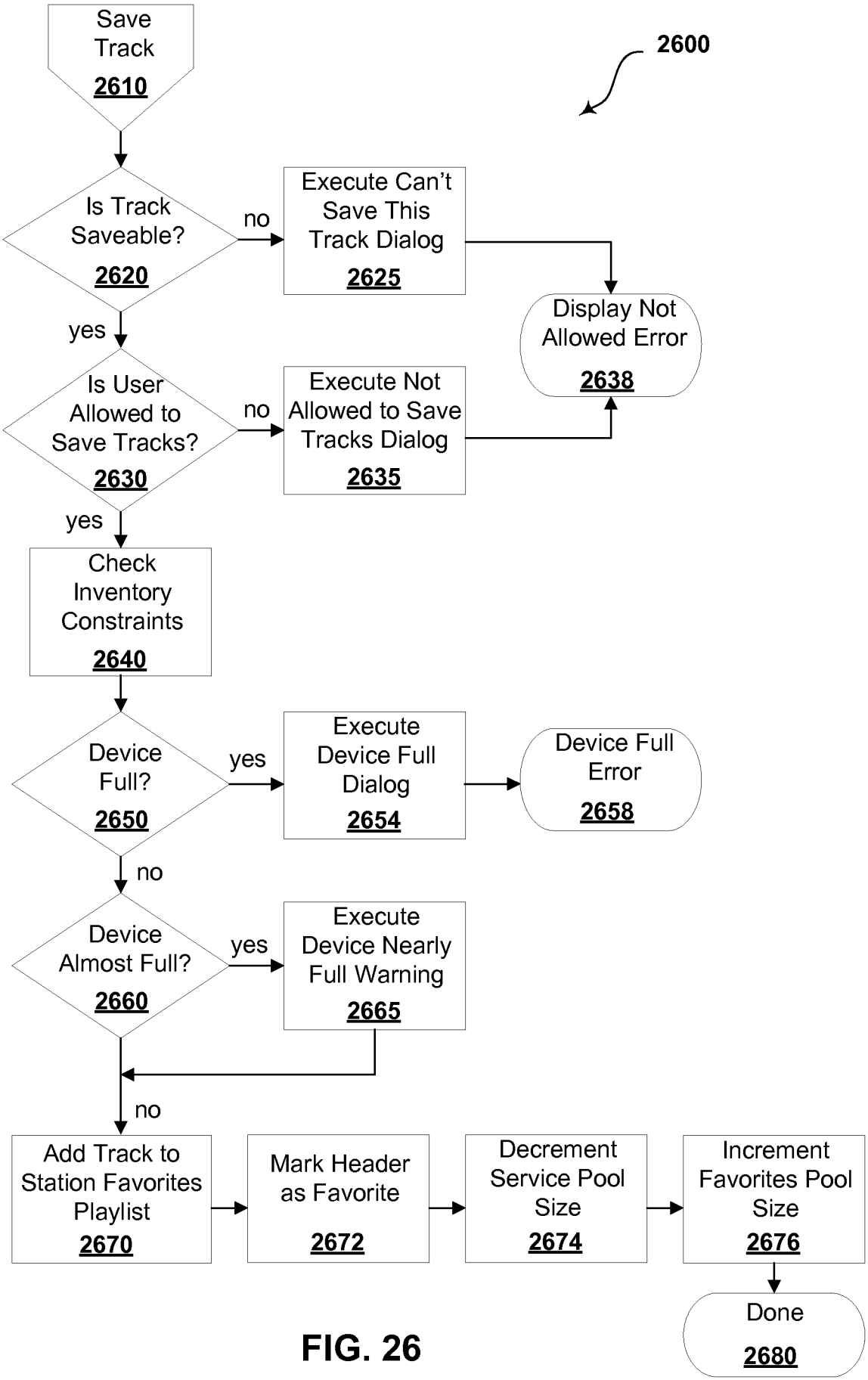

Save Track
2610
2600
Is Track Saveable?
2620
no
Execute Can't Save This Track Dialog
2625
Display Not Allowed Error
2638
yes
Is User Allowed to Save Tracks?
2630
no
Execute Not Allowed to Save Tracks Dialog
2635
yes
Check Inventory Constraints
2640
Device Full?
2650
yes
Execute Device Full Dialog
2654
Device Full Error
2658
no
Device Almost Full?
2660
yes
Execute Device Nearly Full Warning
2665
no
Add Track to Station Favorites Playlist
2670
Mark Header as Favorite
2672
Decrement Service Pool Size
2674
Increment Favorites Pool Size
2676
Done
2680

FIG. 26

Inventory Management Symbols and Descriptions in Accordance with One Embodiment

Name	Symbol	Description	Default Value
Total Service Tracks	T_{total}	Total number of tracks currently allocated to the services pool.	Number of tracks initially loaded onto the device: 1000 – 5000.
Minimum Station Tracks	$T_{s\min}$	Minimum tracks for an inactive station that renders the station reasonable should the user switch to that station	25
Maximum Station Tracks	$T_{s\max}$	Maximum tracks for an active station. Diminishing returns by caching more tracks than this for a single station. Caps the tracks on an active station causing excess tracks to be redistributed to less active or inactive stations.	HUGE (not used)?
Minimum Service Active Pool	$A_{\min}$	Minimum number of tracks to allocate amongst the active stations according to predicted duty cycle. Similar to Tsmax in that a station with 100% duty cycle gets this many tracks. Used to define the minimum pool size for the service: Amax + (Tsmin * inactive stations)	500
Minimum Service Tracks	$T_{\min}$	Minimum number of tracks needed to operate the service. This can be reduced only by deleting inactive stations.	$A_{\min}$ + ($T_{s\min}$ * inactive stations)
Average Track Size (bytes)	t_{size}	Average size of a services track in bytes.	~1MB
High Water	W_{hi}	Threshold of nearing device full that triggers an inventory cull cycle to free space	Initial Services Allocation – (N * t_{size}) N =~ 10?
Low Water	W_{low}	Defines the target amount of space to be free after an inventory management cycle. Setting this farther below W_{hi} results in fewer inventory cull operations, but less efficient use of space.	W_{hi} - W_{delta}
Water Delta	W_{delta}	Fixed constant defining difference between high and low water. A smaller setting results in more frequent Inventory Cull cycles. A larger setting results in fewer usable tracks on average.	20 * t_{size}
Float	T_{float}	Number of tracks currently above W_{low}	
Device full warning threshold	T_{warn}	Number of tracks away from device full to trigger warning messages for device nearly full.	10?

FIG. 27

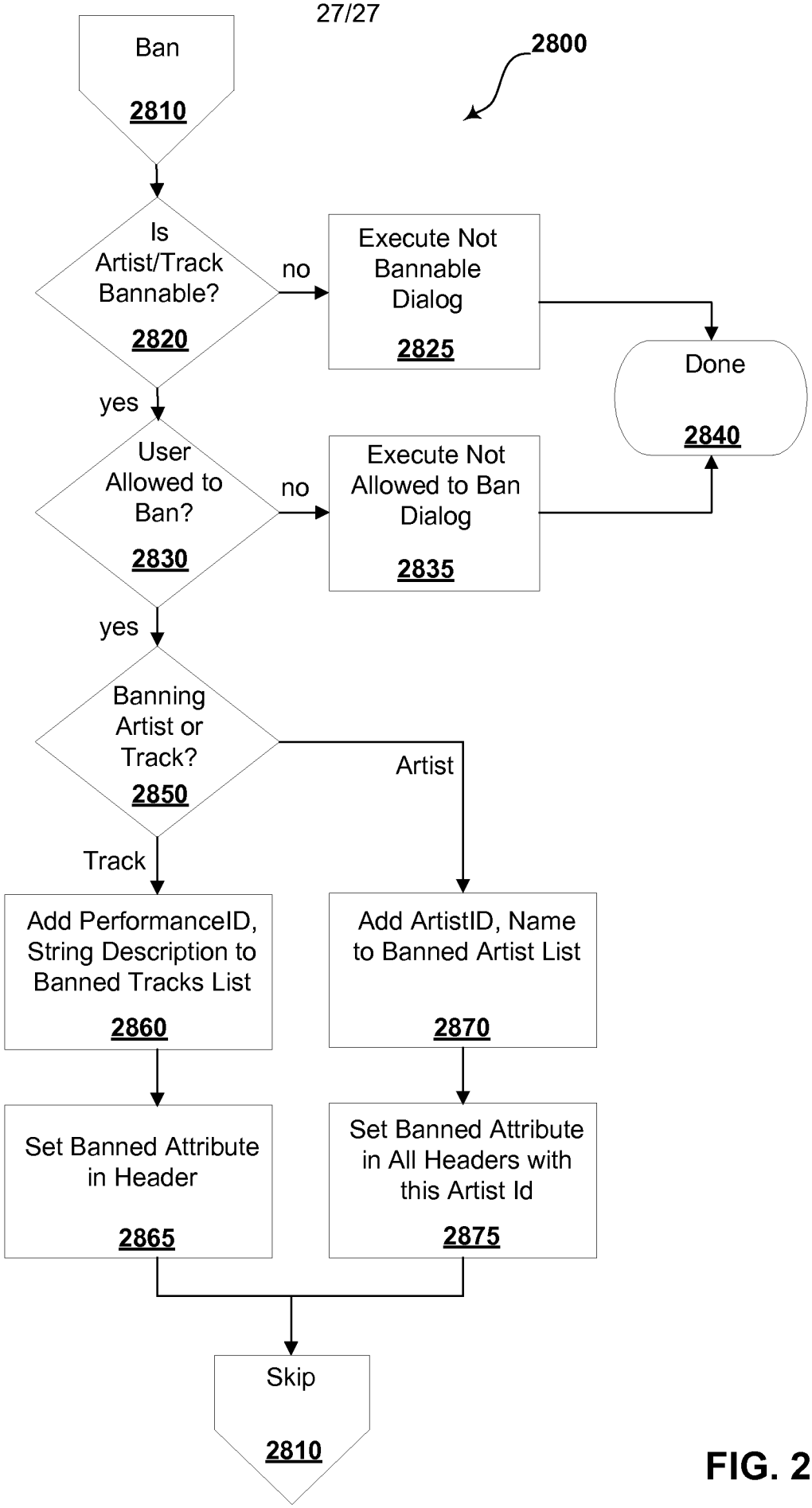

Ban
2810
2800
Is Artist/Track Bannable?
2820
no
Execute Not Bannable Dialog
2825
Done
2840
yes
User Allowed to Ban?
2830
no
Execute Not Allowed to Ban Dialog
2835
yes
Banning Artist or Track?
2850
Artist
Track
Add PerformanceID, String Description to Banned Tracks List
2860
Add ArtistID, Name to Banned Artist List
2870
Set Banned Attribute in Header
2865
Set Banned Attribute in All Headers with this Artist Id
2875
Skip
2810

FIG. 28