Novelty Search Report

System and Method for Conditional Traversal of Virtual Video Containers

Contents

1. Details of the Invention 3
2. Details of Search Coverage 3
   Search Coverage - 1 3
   Search Coverage - 2 3
   Search Coverage - 3 3
3. Key Features 4
4. Key Feature Matrix 6
5. Relevant Citations 8
5.1 Relevant Patent Citations 8
   Patent Citation 1: US20220417619A1 8
   Patent Citation 2: US9236091B2 18
   Patent Citation 3: US20240013810A1 25
   Patent Citation 4: US12316905B2 35
   Patent Citation 5: US11956520B2 43
   Patent Citation 6: US12277238B2 51
5.2 Relevant Non-Patent Citations 64
   Non-patent Citation 1: Eye Tracked Foveated Rendering 64
   Non-patent Citation 2: Virtual Video Files At Scale: Seamlessly Processing Billions Of Videos Per Day 76
6. Additional Citations 88
6.1 Additional Patent Citations 88
6.2 Additional Non-patent Citations 88
7. Search Details 89
7.1 Keywords and Related Terms 89
7.2 Patent Search Strings 90
7.3 Non-patent Search Strings 98
7.4 Relevant Classifications 99
   a. IPC/CPC Classification 99
   b. US Classification 100
   c. F-Term 100
7.5 Key Inventors: 101
7.6 Key Assignee(s): 102
8. Disclaimer 103

1. Details of the Invention

The invention relates to a traversal-governed video assembly system and method for dynamically constructing continuous video streams from virtual video reference containers. The system removes raw media payload data from rendered video files while retaining structural sample timing metadata, enabling containers to be interconnected through traversal references within a recursively nested graph. Runtime branch conditions based on viewer inputs, AI directives, rights policies, and environmental signals determine target containers. Independent cryptographic video tokens control traversal authority, with validation enabling ephemeral key release or signed authorization for media retrieval. Spatial context metrics further establish fidelity classes, including gaze-based resolution allocation. Media blocks may be prefetched into non-executable buffers while authorization remains withheld until token validation. A client cryptographic wallet facilitates secure session initiation. The system may additionally enforce trusted execution, inheritance of restrictive policies, cycle and depth controls, synchronized traversal, audit logging, and automated financial settlement. Video is assembled in volatile memory without writing rendered video files to persistent storage.

2. Details of Search Coverage

Search Coverage - 1

Google Patents
Espacenet
FPO
USPTO
FPO
Jplat-Pat
KIPRIS

Search Coverage - 2

Patseer

Search Coverage - 3

Other Non-Patent Literature Publication Sources

3. Key Features

KF No.	Key Features (Extracted from Disclosure)
1	A system for traversal-governed video assembly.
2	A virtualization engine configured to generate a virtual video reference container by removing the media payload (MDAT) while retaining structural and timing metadata (STBL).
3	The retained metadata comprises rewritten sample table references identifying media sample blocks stored externally from the virtual video reference container.
4	A graph composition mechanism configured to interconnect multiple virtual video reference containers through traversal references to form a traversable video graph, including recursive or nested traversal.
5	A runtime traversal engine configured to evaluate dynamic branch conditions based on real-time signals and select a target video container from multiple candidate containers, wherein the signals include viewer input, AI agent directives, rights policies, or environmental context.
6	A token validation module configured to validate an independent cryptographic video token associated with the selected target video container and grant traversal authority upon successful validation.
7	The cryptographic video token defines or enforces licensing, usage, consent, access, or financial settlement conditions, and successful validation releases an ephemeral decryption key or signed policy authorization.
8	A spatial resolution module configured to determine spatial resolution gates based on viewer spatial context and provide full fidelity for a foveal region, reduced fidelity for a peripheral region, and withhold media for an out-of-view region.
9	A prefetch mechanism configured to retrieve candidate media for possible traversal branches into a decoupled non-executable buffer before branch selection, while withholding the corresponding key or authorization until token validation.
10	A playback assembly mechanism configured to retrieve, decrypt, authorize, and sequence media sample blocks in volatile memory according to traversal authorization and spatial resolution, thereby generating continuous playback without writing a rendered video file to persistent storage.
11	A policy enforcement mechanism configured to apply inherited policies to recursively nested containers, wherein the most restrictive applicable policy governs traversal.
12	The traversal framework further enforces cycle detection, maximum traversal-depth restrictions, and real-time rights verification.
13	A rights management mechanism configured to dynamically compose video content from independently owned content libraries using container-specific authorization and maintain an audit record of traversal events.

14	**The rights management mechanism further generates financial settlement information for respective content owners based on traversed branches.**

4. Key Feature Matrix

Key Features	Relevant Citations							
	Relevant Patent Citations						NPL	
	PC_1	PC_2	PC_3	PC_4	PC_5	PC_6	PC_7	PC_8
1. A system for traversal-governed video assembly.	Y	Y	Y	Y	Y	Y*	Y*	Y*
2. A virtualization engine configured to generate a virtual video reference container by removing the media payload (MDAT) while retaining structural and timing metadata (STBL).	Y	Y	Y	Y	Y	Y	Y	N
3. The retained metadata comprises rewritten sample table references identifying media sample blocks stored externally from the virtual video reference container.	Y*	Y	Y	N	N	N	Y	N
4. A graph composition mechanism configured to interconnect multiple virtual video reference containers through traversal references to form a traversable video graph, including recursive or nested traversal.	Y	Y*	Y	Y	Y	Y	Y	Y*
5. A runtime traversal engine configured to evaluate dynamic branch conditions based on real-time signals and select a target video container from multiple candidate containers, wherein the signals include viewer input, AI agent directives, rights policies, or environmental context.	Y	N	Y	Y	Y	Y*	Y*	Y*
6. A token validation module configured to validate an independent cryptographic video token associated with the selected target video container and grant traversal authority upon successful validation.	Y*	Y	N	N	N	Y	N	Y*
7. The cryptographic video token defines or enforces licensing, usage, consent, access, or financial settlement conditions, and successful validation releases an ephemeral decryption key or signed policy authorization.	Y*	Y	N	N	N	Y	N	Y
8. A spatial resolution module configured to determine spatial resolution gates based on viewer spatial context and provide full fidelity for a foveal region, reduced fidelity for a peripheral region, and withhold media for an out-of-view region.	N	N	N	Y	Y*	N	N	Y
9. A prefetch mechanism configured to retrieve candidate media for possible traversal branches into a decoupled non-executable buffer before branch selection, while withholding the corresponding key or authorization until token validation.	Y	Y	Y*	Y	Y*	Y*	Y*	Y*

10. A playback assembly mechanism configured to retrieve, decrypt, authorize, and sequence media sample blocks in volatile memory according to traversal authorization and spatial resolution, thereby generating continuous playback without writing a rendered video file to persistent storage.	**Y***	**Y***	**Y***	**Y***	**Y***	**Y***	**Y***	**Y***
11. A policy enforcement mechanism configured to apply inherited policies to recursively nested containers, wherein the most restrictive applicable policy governs traversal.	**Y**	**Y***	**Y**	**N**	**N**	**Y***	**Y***	**Y***
12. The traversal framework further enforces cycle detection, maximum traversal-depth restrictions, and real-time rights verification.	**N**	**N**	**N**	**N**	**N**	**Y***	**N**	**N**
13. A rights management mechanism configured to dynamically compose video content from independently owned content libraries using container-specific authorization and maintain an audit record of traversal events.	**Y***	**Y***	**Y***	**Y***	**N**	**Y***	**Y**	**Y***
14. The rights management mechanism further generates financial settlement information for respective content owners based on traversed branches.	**N**	**N**	**Y***	**N**	**N**	**Y***	**Y***	**Y***

Y* - Partial Mapped/Inferred Feature

5. Relevant Citations

5.1 Relevant Patent Citations

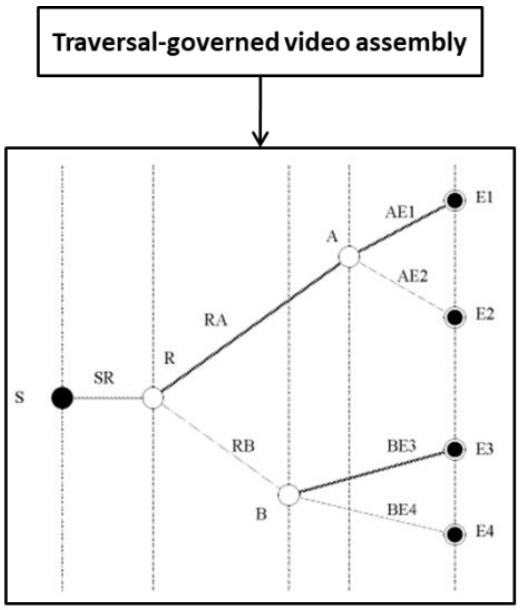
Patent Citation 1: US20220417619A1

Title	Processing and Playing Control Over Interactive Video
Priority Date	24 SEP 2019
Filing Date	22 SEP 2020
Publication Date	29 DEC 2022
Inventors	Junrui Lu
Assignees	GuangZhou Huya Technology Co. Ltd
IPC Classes	H04L43/106; H04N21/2187; H04N21/472; H04N21/8541
CPC Classes	G11B27/031; H04L43/106; H04N21/2187; H04N21/472; H04N21/8541
US Classes	None
Family Members	**WO2021057693A1**

Abstract:

Embodiments of the present disclosure provide an interactive video play control method, apparatus and system. After a branching message sent by a server is received, it is determined whether a difference between a current play time of playing an interactive video by a client and an execution time is greater than or equal to a preset time threshold, and if the difference is greater than or equal to the preset time threshold, branching options are immediately displayed on the client.

Key Features	Relevant Excerpts
1. A system for traversal-governed video assembly.	**[Para 0003]** According to a first aspect of embodiments of the present disclosure, a live broadcast platform is provided. The live broadcast platform includes: a node editor and a video editor. **The node editor is configured to receive a node editing instruction, create a play node according to the node editing instruction and configure a play time offset for each play node, where the play node includes at least one branching node, and at least two child nodes of the at least one branching node.** The video editor is configured to receive multiple video files, and respectively associate the video files with multiple play sub-paths between adjacent play nodes to generate an interactive video, where each play sub-path of a same branching node corresponds to multiple parallel plot branches in the interactive video, and the play time offset is configured to control a play progress of playing the interactive video by each client in a live broadcast room, such that a difference between play times of playing the interactive video by any two clients in the live broadcast room is less than a preset value. **[Para 0036]** FIG. 3 is a schematic diagram illustrating play nodes according to one

	or more embodiments. **In the diagram, 8 play nodes S, R, A, B, E1, E2, E3 and E4 are included. The node S is a start node. The node R is an intermediate node. The nodes A and B are child nodes of the node R. The nodes E1 and E2 are child nodes of the node A. The nodes E3 and E4 are child nodes of the node B. The node S is a non-branching node. The nodes R, A and B are branching nodes. The nodes E1, E2, E3 and E4 are end nodes**.  FIG.3 **Note:** *The mapped citation describes a system for assembling interactive video based on a defined traversal structure. Play nodes, including branching and child nodes, are connected by play sub-paths associated with multiple video files. The resulting hierarchy defines traversal paths through different video portions, supporting traversal-governed video assembly and presentation based on predefined nodes and branching paths.*
2. A virtualization engine configured to generate a virtual video reference container by removing the media payload (MDAT) while retaining structural and timing metadata (STBL).	**[Para 0036]** FIG. 3 is a schematic diagram illustrating play nodes according to one or more embodiments. **In the diagram, 8 play nodes S, R, A, B, E1, E2, E3 and E4 are included. The node S is a start node. The node R is an intermediate node. The nodes A and B are child nodes of the node R. The nodes E1 and E2 are child nodes of the node A. The nodes E3 and E4 are child nodes of the node B. The node S is a non-branching node. The nodes R, A and B are branching nodes. The nodes E1, E2, E3 and E4 are end nodes**. **[Para 0048]** In the embodiment, **the live broadcast platform can directly or indirectly receive instructions sent by operation-maintenance personnel or directors to implement the process of creating, editing, publishing and playing interactive videos.** **[Para 0052] After the play nodes are defined, the video editor 202 can receive video files uploaded by the operation-maintenance personnel or the director, and respectively associate each video file with each corresponding play sub-path of multiple play sub-paths between adjacent play nodes. The video editor 202 can acquire URLs (Uniform Resource Locators) of video files input by the operation-maintenance personnel or the director, and acquire corresponding**

	video files by accessing the URLs. **Note:** *The mapped citation describes a node-based video structure in which play nodes function as virtual video reference containers, defining play logic while associated video files provide the media content. Video files are obtained through URLs and linked to play sub-paths between the virtual containers, supporting the concept of virtual containers retaining structural references to separately stored media.*
3. The retained metadata comprises rewritten sample table references identifying media sample blocks stored externally from the virtual video reference container.	**[Para 0036]** FIG. 3 is a schematic diagram illustrating play nodes according to one or more embodiments. **In the diagram, 8 play nodes S, R, A, B, E1, E2, E3 and E4 are included. The node S is a start node. The node R is an intermediate node. The nodes A and B are child nodes of the node R. The nodes E1 and E2 are child nodes of the node A. The nodes E3 and E4 are child nodes of the node B. The node S is a non-branching node. The nodes R, A and B are branching nodes. The nodes E1, E2, E3 and E4 are end nodes**. **[Para 0048]** In the embodiment, **the live broadcast platform can directly or indirectly receive instructions sent by operation-maintenance personnel or directors to implement the process of creating, editing, publishing and playing interactive videos.** **[Para 0052] After the play nodes are defined, the video editor 202 can receive video files uploaded by the operation-maintenance personnel or the director, and respectively associate each video file with each corresponding play sub-path of multiple play sub-paths between adjacent play nodes. The video editor 202 can acquire URLs (Uniform Resource Locators) of video files input by the operation-maintenance personnel or the director, and acquire corresponding video files by accessing the URLs**. **Note:** *The mapped citation describes video files obtained through URLs provided by authorized personnel, with each file associated with a play sub-path between adjacent play nodes. Thus, the URLs identify and retrieve media stored externally from the play-node structure. However, retained metadata comprising rewritten sample-table references is not expressly disclosed.*
4. A graph composition mechanism configured to interconnect multiple virtual video reference containers through traversal references to form a traversable video graph, including recursive or nested traversal.	**[Para 0003]** According to a first aspect of embodiments of the present disclosure, a live broadcast platform is provided. The live broadcast platform includes: a node editor and a video editor. **The node editor is configured to receive a node editing instruction, create a play node according to the node editing instruction and configure a play time offset for each play node, where the play node includes at least one branching node, and at least two child nodes of the at least one branching node**. The video editor is configured to receive multiple video files, and respectively associate the video files with multiple play sub-paths between adjacent play nodes to generate an interactive video, where each play sub-path of a same branching node corresponds to multiple parallel plot branches in the interactive video, and the play time offset is configured to control a play progress of playing the interactive video by each client in a live broadcast room, such that a difference between play times of playing the interactive video by any two clients in the live broadcast room is less than a preset value.

	[Para 0036] FIG. 3 is a schematic diagram illustrating play nodes according to one or more embodiments. **In the diagram, 8 play nodes S, R, A, B, E1, E2, E3 and E4 are included. The node S is a start node. The node R is an intermediate node. The nodes A and B are child nodes of the node R. The nodes E1 and E2 are child nodes of the node A. The nodes E3 and E4 are child nodes of the node B. The node S is a non-branching node. The nodes R, A and B are branching nodes. The nodes E1, E2, E3 and E4 are end nodes**. **Note:** *The mapped citation describes an interconnected structure of play nodes, branching nodes, child nodes, and play sub-paths forming a traversable interactive video graph. The hierarchical node relationships support multi-level traversal among virtual video reference containers. Thus, the disclosure generally supports graph composition that interconnects virtual video reference containers through branching paths, with recursive traversal reasonably inferable from the node hierarchy.*
5. A runtime traversal engine configured to evaluate dynamic branch conditions based on real-time signals and select a target video container from multiple candidate containers, wherein the signals include viewer input, AI agent directives, rights policies, or environmental context.	**[Para 0036]** FIG. 3 is a schematic diagram illustrating play nodes according to one or more embodiments. **In the diagram, 8 play nodes S, R, A, B, E1, E2, E3 and E4 are included. The node S is a start node. The node R is an intermediate node. The nodes A and B are child nodes of the node R. The nodes E1 and E2 are child nodes of the node A. The nodes E3 and E4 are child nodes of the node B. The node S is a non-branching node. The nodes R, A and B are branching nodes. The nodes E1, E2, E3 and E4 are end nodes**. **[Para 0039]** One interactive video can include multiple video files. Each play sub-path corresponds to one video file of the interactive video. Video files corresponding to all play sub-paths constitute video content of the interactive video. Play nodes and play paths between play nodes constitute play logic of the interactive video. Multiple play sub-paths dependent to the same branching node are parallel play sub-paths. Multiple parallel play sub-paths correspond to multiple parallel branching plots in the interactive video. **A user can select one of the play sub-paths to independently select a plot direction of the interactive video. For example, in the play nodes shown in FIG. 3 , at the node R, a user can select an option corresponding to RA to play a branching plot corresponding to the RA in the interactive video; at the node A, the user can select an option corresponding to AE1 to play a branching plot corresponding to the AE1 in the interactive video**. **Note:** *The mapped citation describes a runtime traversal arrangement in which branching nodes provide multiple video paths, allowing users to select a path during playback. These real-time viewer inputs determine the corresponding target video path. Thus, the disclosure supports runtime evaluation and selection of a target video container from multiple candidates based on viewer input.*
6. A token validation module configured to validate an independent cryptographic video token associated with the selected target video container and grant traversal authority	**[Para 0048]** In the embodiment, **the live broadcast platform can directly or indirectly receive instructions sent by operation-maintenance personnel or directors to implement the process of creating, editing, publishing and playing interactive videos.** **[Para 0059]** In this embodiment, **part of control authority on an interactive video can be opened to the operation-maintenance personnel or the director, and the**

upon successful validation.	**operation-maintenance personnel or the director manually sends an instruction to edit the interactive video. The operation-maintenance personnel or the director can send an interactive video editing instruction to a live broadcast platform.** The interactive video editing instruction can be a play node insertion instruction, a play node deletion instruction, an advertisement insertion instruction or other types of instructions. **After receiving the interactive video editing instruction, the live broadcast platform can correspondingly insert a play node, delete a play node, or insert an advertisement in the interactive video.** **Note:** *The mapped citation describes user-specific authorization in which operation-maintenance personnel or a director is granted control authority over an interactive video. Such authorization can reasonably be understood as validating whether an authorized user has the requisite authority to perform permitted video operations, providing a basis for granting traversal or control authority upon successful authorization. However, an independent cryptographic video token and cryptographic validation of such a token for a selected target video container is not expressly disclosed.*
7. The cryptographic video token defines or enforces licensing, usage, consent, access, or financial settlement conditions, and successful validation releases an ephemeral decryption key or signed policy authorization.	**[Para 0048]** In the embodiment, **the live broadcast platform can directly or indirectly receive instructions sent by operation-maintenance personnel or directors to implement the process of creating, editing, publishing and playing interactive videos.** **[Para 0059]** In this embodiment, **part of control authority on an interactive video can be opened to the operation-maintenance personnel or the director, and the operation-maintenance personnel or the director manually sends an instruction to edit the interactive video. The operation-maintenance personnel or the director can send an interactive video editing instruction to a live broadcast platform.** The interactive video editing instruction can be a play node insertion instruction, a play node deletion instruction, an advertisement insertion instruction or other types of instructions. **After receiving the interactive video editing instruction, the live broadcast platform can correspondingly insert a play node, delete a play node, or insert an advertisement in the interactive video.** **Note:** *The mapped citation describes user-specific control authority for operation-maintenance personnel or directors to perform authorized video operations, supporting access or usage control. However, a cryptographic video token defining licensing, consent, or financial conditions, and release of an ephemeral decryption key or signed policy authorization upon validation, are not expressly disclosed.*
8. A spatial resolution module configured to determine spatial resolution gates based on viewer spatial context and provide full fidelity for a foveal region, reduced fidelity for a peripheral region, and withhold media for an out-of-view region.	N/A

9. A prefetch mechanism configured to retrieve candidate media for possible traversal branches into a decoupled non-executable buffer before branch selection, while withholding the corresponding key or authorization until token validation.	**[Para 0036]** FIG. 3 is a schematic diagram illustrating play nodes according to one or more embodiments. **In the diagram, 8 play nodes S, R, A, B, E1, E2, E3 and E4 are included. The node S is a start node. The node R is an intermediate node. The nodes A and B are child nodes of the node R. The nodes E1 and E2 are child nodes of the node A. The nodes E3 and E4 are child nodes of the node B. The node S is a non-branching node. The nodes R, A and B are branching nodes. The nodes E1, E2, E3 and E4 are end nodes**. **[Para 0039]** One interactive video can include multiple video files. Each play sub-path corresponds to one video file of the interactive video. Video files corresponding to all play sub-paths constitute video content of the interactive video. Play nodes and play paths between play nodes constitute play logic of the interactive video. Multiple play sub-paths dependent to the same branching node are parallel play sub-paths. Multiple parallel play sub-paths correspond to multiple parallel branching plots in the interactive video. **A user can select one of the play sub-paths to independently select a plot direction of the interactive video. For example, in the play nodes shown in FIG. 3 , at the node R, a user can select an option corresponding to RA to play a branching plot corresponding to the RA in the interactive video; at the node A, the user can select an option corresponding to AE1 to play a branching plot corresponding to the AE1 in the interactive video.** **[Para 0082]** For example, according to a preset script time, an interactive video shall start to be played at 18:00:00, and progress to a branching node R at 18:01:00. **If a play path RB is selected, the interactive video progresses to a branching node B via the play path RB at 18:04:00. If a play path RA is selected, the interactive video progresses to a branching node A via the play path RA at 18:06:00.** **[Para 0124]** In an embodiment, **acquiring the video contents starting from the target play time of the interactive video includes: searching for the video contents starting from the target play time of the interactive video in a video buffer area; if having searched out the video contents, acquiring the video contents from the video buffer area; if having not searched out the video contents, obtaining the video contents starting from the target play time of the interactive video from the content delivery network through the interface**. **Note:** *The mapped citation describes a prefetch mechanism in which multiple candidate video files associated with parallel play sub-paths are identified, and a video buffer stores and provides required content before or during playback. Predefined branching paths and script times identify potential subsequent content for retrieval.*
10. A playback assembly mechanism configured to retrieve, decrypt, authorize, and sequence media sample blocks in volatile memory according to traversal authorization and spatial resolution, thereby	**[Para 0036]** FIG. 3 is a schematic diagram illustrating play nodes according to one or more embodiments. **In the diagram, 8 play nodes S, R, A, B, E1, E2, E3 and E4 are included. The node S is a start node. The node R is an intermediate node. The nodes A and B are child nodes of the node R. The nodes E1 and E2 are child nodes of the node A. The nodes E3 and E4 are child nodes of the node B. The node S is a non-branching node. The nodes R, A and B are branching nodes. The nodes E1, E2, E3 and E4 are end nodes**.

generating continuous playback without writing a rendered video file to persistent storage.	**[Para 0039]** One interactive video can include multiple video files. Each play sub-path corresponds to one video file of the interactive video. Video files corresponding to all play sub-paths constitute video content of the interactive video. Play nodes and play paths between play nodes constitute play logic of the interactive video. Multiple play sub-paths dependent to the same branching node are parallel play sub-paths. Multiple parallel play sub-paths correspond to multiple parallel branching plots in the interactive video. **A user can select one of the play sub-paths to independently select a plot direction of the interactive video. For example, in the play nodes shown in FIG. 3 , at the node R, a user can select an option corresponding to RA to play a branching plot corresponding to the RA in the interactive video; at the node A, the user can select an option corresponding to AE1 to play a branching plot corresponding to the AE1 in the interactive video.** **[Para 0082]** For example, according to a preset script time, an interactive video shall start to be played at 18:00:00, and progress to a branching node R at 18:01:00. **If a play path RB is selected, the interactive video progresses to a branching node B via the play path RB at 18:04:00. If a play path RA is selected, the interactive video progresses to a branching node A via the play path RA at 18:06:00.** **[Para 0124]** In an embodiment, **acquiring the video contents starting from the target play time of the interactive video includes: searching for the video contents starting from the target play time of the interactive video in a video buffer area; if having searched out the video contents, acquiring the video contents from the video buffer area; if having not searched out the video contents, obtaining the video contents starting from the target play time of the interactive video from the content delivery network through the interface**. **[Para 0048]** In the embodiment, **the live broadcast platform can directly or indirectly receive instructions sent by operation-maintenance personnel or directors to implement the process of creating, editing, publishing and playing interactive videos.** **[Para 0059]** In this embodiment, **part of control authority on an interactive video can be opened to the operation-maintenance personnel or the director, and the operation-maintenance personnel or the director manually sends an instruction to edit the interactive video. The operation-maintenance personnel or the director can send an interactive video editing instruction to a live broadcast platform.** The interactive video editing instruction can be a play node insertion instruction, a play node deletion instruction, an advertisement insertion instruction or other types of instructions. **After receiving the interactive video editing instruction, the live broadcast platform can correspondingly insert a play node, delete a play node, or insert an advertisement in the interactive video.** **[Para 0157]** …**Computer usable storage media include** permanent **and non-permanent,** removable and non-removable media, and storage of information can be accomplished by any method or technology. Information may be computer readable instructions, data structures, program modules, or other data. Examples of computer storage media include, but are not limited to, a phase change random access memory (PRAM), **a static random access memory (SRAM), a**

	dynamic random access memory (DRAM), other types of random access memory (RAM), a read only memory (ROM), an electrically erasable programmable read only memory (EEPROM), a flash memory or other memory technology, a compact disc read only memory (CD-ROM), a digital versatile disc (DVD) or other optical storage, a magnetic tape cartridge, a magnetic tape/disk storage or other magnetic storage device or any other non-transmission medium that can be used to store information that can be accessed by a computing device. **Note:** *The mapped citation describes runtime playback assembly by retrieving video content from a buffer or content delivery network and sequencing it according to selected play paths, target play times, and play logic. Disclosed RAM can reasonably be understood as volatile memory for temporary in-memory video handling, while control authority supports association of playback operations with authorized users and traversal rights. However, decryption and authorization of individual media sample blocks, spatial-resolution-based processing, and continuous playback without writing a rendered video file to persistent storage are not expressly disclosed.*
11. A policy enforcement mechanism configured to apply inherited policies to recursively nested containers, wherein the most restrictive applicable policy governs traversal.	**[Para 0036]** FIG. 3 is a schematic diagram illustrating play nodes according to one or more embodiments. **In the diagram, 8 play nodes S, R, A, B, E1, E2, E3 and E4 are included. The node S is a start node. The node R is an intermediate node. The nodes A and B are child nodes of the node R. The nodes E1 and E2 are child nodes of the node A. The nodes E3 and E4 are child nodes of the node B. The node S is a non-branching node. The nodes R, A and B are branching nodes. The nodes E1, E2, E3 and E4 are end nodes**. **[Para 0039]** One interactive video can include multiple video files. Each play sub-path corresponds to one video file of the interactive video. Video files corresponding to all play sub-paths constitute video content of the interactive video. Play nodes and play paths between play nodes constitute play logic of the interactive video. Multiple play sub-paths dependent to the same branching node are parallel play sub-paths. Multiple parallel play sub-paths correspond to multiple parallel branching plots in the interactive video. **A user can select one of the play sub-paths to independently select a plot direction of the interactive video. For example, in the play nodes shown in FIG. 3 , at the node R, a user can select an option corresponding to RA to play a branching plot corresponding to the RA in the interactive video; at the node A, the user can select an option corresponding to AE1 to play a branching plot corresponding to the AE1 in the interactive video.** **Note:** *The mapped citation describes a hierarchical branching structure in which the available branching paths determine how traversal proceeds from one node to another and ultimately to the corresponding video content. Accordingly, the branching logic can be understood as providing a traversal policy that governs the available paths through the nested node structure.*
12. The traversal framework further enforces cycle detection, maximum traversal-depth restrictions, and real-time rights	N/A

verification.	
13. A rights management mechanism configured to dynamically compose video content from independently owned content libraries using container-specific authorization and maintain an audit record of traversal events.	**[Para 0134] Assuming that when the user Si LI plays the interactive video, the network is lagging for 8 seconds (lag=8), which happens after the client of Si LI receives a play start instruction and when the interactive video just starts to be played, and assuming that there is no lag in 52 seconds from an absolute time of 2019-07-25 18:00:08 to an absolute time of 2019-07-25 18:01:00, at the absolute time of 2019-07-25 18:01:00, the video watched by the user Si LI has been played to the 52nd second of a "video experience line" of the user Si LI (video_time=52).** **[Para 0135]** At the same time, because an "instruction controller" of a server, after detecting a server time, **determines that a "script timeline/script line" stored by the server has run to the 60th second (script_time=60), the "instruction controller" will send a "branching" information through a long connection (this information records information for a receiver to display an branch selection interface and corresponding options when the video_time reaches (greater than or equal to) the 60th second (a current script time reaches an execution time of executing a plot branching operation on the interactive video execute_time=60))** **[Para 0048]** In the embodiment, **the live broadcast platform can directly or indirectly receive instructions sent by operation-maintenance personnel or directors to implement the process of creating, editing, publishing and playing interactive videos.** **[Para 0059]** In this embodiment, **part of control authority on an interactive video can be opened to the operation-maintenance personnel or the director, and the operation-maintenance personnel or the director manually sends an instruction to edit the interactive video. The operation-maintenance personnel or the director can send an interactive video editing instruction to a live broadcast platform.** The interactive video editing instruction can be a play node insertion instruction, a play node deletion instruction, an advertisement insertion instruction or other types of instructions. **After receiving the interactive video editing instruction, the live broadcast platform can correspondingly insert a play node, delete a play node, or insert an advertisement in the interactive video.** **Note:** *The mapped citation describes maintaining playback and traversal information, including video position, script progression, branching execution times, and playback events, which can be inferred as traversal-event records. It also provides user-specific authorization for controlling and editing interactive video content, supporting authorized content control and event recording. However, independently owned content libraries and container-specific authorization are not expressly disclosed.*
14. The rights management mechanism further generates financial settlement information for respective content owners based on traversed	N/A

branches.	

Patent Citation 2: US9236091B2

Title	Apparatus and method for processing and reading a file having a media data container and a metadata container
Priority Date	02 JUL 2007
Filing Date	01 JUL 2008
Publication Date	12 JAN 2016
Inventors	Stefan Doehla, Stefan Kraegeloh, Nikolaus Faerber, Harald Fuchs
Assignees	Fraunhofer Society for the Promotion of Applied Research eV
IPC Classes	H04N7/167, G11B27/10, H04N21/236, H04N21/2368, H04N21/2381, H04N21/2389, H04N21/43, H04N21/433, H04N21/434, H04N21/4363, H04N21/438, H04N21/6334, H04N21/6437, H04N21/854, G11B20/00
CPC Classes	G11B27/10, H04N21/23614, H04N21/2368, H04N21/2381, H04N21/23895, H04N21/43072, H04N21/4334, H04N21/4341,H04N21/4348, H04N21/4363, H04N21/4381, H04N21/63345, H04N21/6437, H04N21/85406, H04N7/1675, G11B20/00492, G11B20/00731, G11B27/10
US Classes	None
Family Members	**KR101199732B1 | BRPI0811821B1 | WO2009003685A1 | BRPI0811833B1 | RU2459378C2 | CN101731013B | WO2009003684A1 | KR101074585B1 | WO2009003683A1 | EP2149264B1 | DE602008004502D1 | JP4766473B2 | US8462946B2 | JP5334335B2 | RU2492587C2 | ATE495631T1 | EP2160899B1 | TWI446773B | TWI371204B | TWI430665B**

Abstract:

An apparatus for processing stored data packets in a media data container and stored associated meta information in a metadata container, the associated meta information having transport timing information and location information indicating a storage location of the stored data packets in the media data container, the apparatus having a processor for determining, based on the stored data packets and the stored associated meta information, decoding information for a media payload of the stored data packets, wherein the decoding information indicates at which time instant to replay which payload of the stored data packets.

Key Features	Relevant Excerpts
1. A system for traversal-governed video assembly.	**[Column 10; Lines 35-53]** For replaying virtual media tracks, embodiments of **the present invention provide an apparatus for reading a file, the file having stored, in a media data container related to the file, data packets comprising a payload, and having stored, in the media data container, decoding information for the payload of the stored data packets, wherein the decoding information indicates at which time instant to replay which payload of the stored data packets,** the file having stored associated metadata in a media data container, the associated metadata indicating a decoding time and a location of the decoding information in

	the media data container. **The apparatus comprises a processor for determining an output schedule of the payload of the stored data packets by accessing the associated metadata in the media data container and by accessing, based on the metadata, the decoding information in the media data container, and by accessing, based on the decoding information, the payload of the stored data packets. An output controller serves for outputting the payload in accordance with the determined output schedule.** **Note:** *The mapped citation describes an apparatus for replaying virtual media tracks in which a processor uses metadata and decoding information to determine an output schedule for stored media payloads. The controlled retrieval, sequencing, and playback of media based on stored structural and timing information functionally correspond to traversal-governed video assembly, wherein media payloads are accessed according to governing information.*
2. A virtualization engine configured to generate a virtual video reference container by removing the media payload (MDAT) while retaining structural and timing metadata (STBL).	**[Column 10; Lines 06-11] Virtual media tracks may be seen as incomplete media tracks. Therefore, both the recovered timing from RTP and RTCP reception hint tracks and all indexes (typically by using "sample tables") of media tracks can be applied.** Additionally, timed metadata tracks may reference virtual media tracks and extend them. **[Column 10; Lines 54-66]** Compared to an on-the-fly conversion of the received data packets to media tracks during recording, **the virtual media tracks provide media timing and metadata without the necessity to de-packetize the media data (stored data packets) and thus may save storage space. They enable a playback device to search in the file based on media timing.** Virtual media tracks combine the advantages of media tracks and reception hint tracks without doubling a file size, which would be the case if classic media tracks would be added to the file. **With the concept of virtual media tracks all information of the original reception process is kept,** e.g. information about lost packets that helps error concealment and information about reception timing that helps lip-synchronization… **Note:** *The mapped citation describes virtual media tracks containing recovered timing information and sample-table indexes (STBL) without de-packetizing or duplicating the stored media data. Thus, the virtual media tracks correspond to virtual video reference containers that retain structural and timing metadata while keeping the actual media payload separate.*
3. The retained metadata comprises rewritten sample table references identifying media sample blocks stored externally from the virtual video reference container.	**[Abstract]** An apparatus for processing **stored data packets in a media data container and stored associated meta information in a metadata container, the associated meta information having transport timing information and location information indicating a storage location of the stored data packets in the media data container,** the apparatus having a processor for determining, based on the stored data packets and the stored associated meta information, decoding information for a media payload of the stored data packets, wherein the decoding information indicates at which time instant to replay which payload of the stored data packets. **Note:** *The mapped citation describes storing media data packets in a media data container with associated metadata containing transport timing and location*

	information identifying the stored packets. The metadata is used to determine decoding information, functionally corresponding to retained references identifying the locations of media sample blocks.
4. A graph composition mechanism configured to interconnect multiple virtual video reference containers through traversal references to form a traversable video graph, including recursive or nested traversal.	**[Column 11; Lines 11-17] Multiple virtual media tracks may reference transmission units or data packets from the same reception hint track enabling different indexes of the same recorded data packet stream.** E.g., two virtual media tracks with random access, one related to audio, another related to video, point indexes for audio and video of the same MPEG-2 transport stream reception hint track. **[Column 11; Lines 45-56]** For optimized local playback of the recorded streams, a "virtual media track" maps the received TUs to "virtual media samples", using a reversed hinting process. The virtual media samples have the timing of media samples, possibly recovered from the track with CTUs and the reception hint track and need not be complete media samples. Indexing of virtual media tracks is applied if appropriate. **The indexes also apply to the linked samples of the reception hint track. The virtual media track may be used as a reference for other tracks (e.g. "timed metadata tracks") within the file.** Applications may look up the corresponding samples of the reception hint track via the virtual media track. **Note:** *The mapped citation describes multiple virtual media tracks referencing transmission units or data packets, with one track serving as a reference for other tracks and applications accessing corresponding samples through it. This can be functionally inferred as a traversable structure interconnecting virtual video reference containers. However, specific formation of a video graph and recursive or nested traversal among multiple containers are not expressly disclosed.*
5. A runtime traversal engine configured to evaluate dynamic branch conditions based on real-time signals and select a target video container from multiple candidate containers, wherein the signals include viewer input, AI agent directives, rights policies, or environmental context.	N/A
6. A token validation module configured to validate an independent cryptographic video token associated with the selected target video container and grant traversal authority upon successful validation.	**[Column 25-26; Lines 61-67, 01-11] Rights to access data or data packets may be controlled via a rights management system. Receiving data content via a digital communications network may be limited to certain end-users and restricted from other users. E.g., a user may purchase access to a program by paying a fee for the program. If the user pays the fee the user may be granted access to the program for a specified period of time while a user who has not paid the fee may not have access to the program. Access to the program may be regulated by encryption of the transmitted data. The data may be encrypted by any number of encryption standards through the use of an encryption key. At the**

	receiver or user terminal, a key may be used to decrypt the encrypted data such that the content may be viewable at the receiver or user terminal. The key for decrypting the encrypted data packets may also be delivered via the same digital communications network and may also be encrypted. For the delivery of one or more keys also other digital communications networks can be used**. Thus, an end-user wishing to access or view the program or service may need to obtain the rights to the keys.** **Note:** *The mapped citation describes a rights management system that restricts access to authorized end-users and requires rights to cryptographic keys for accessing encrypted content. The encryption key is used to decrypt the protected data at a receiver, functionally supporting validation of authorized cryptographic key access before permitting content access.*
7. The cryptographic video token defines or enforces licensing, usage, consent, access, or financial settlement conditions, and successful validation releases an ephemeral decryption key or signed policy authorization.	**[Column 25-26; Lines 61-67, 01-11] Rights to access data or data packets may be controlled via a rights management system.** Receiving data content via a digital communications network may be limited to certain end-users and restricted from other users. **E.g., a user may purchase access to a program by paying a fee for the program. If the user pays the fee the user may be granted access to the program for a specified period of time while a user who has not paid the fee may not have access to the program. Access to the program may be regulated by encryption of the transmitted data. The data may be encrypted by any number of encryption standards through the use of an encryption key. At the receiver or user terminal, a key may be used to decrypt the encrypted data such that the content may be viewable at the receiver or user terminal.** The key for decrypting the encrypted data packets may also be delivered via the same digital communications network and may also be encrypted. For the delivery of one or more keys also other digital communications networks can be used**. Thus, an end-user wishing to access or view the program or service may need to obtain the rights to the keys.** **Note:** *The mapped citation describes a rights management system that limits content access to authorized users based on licensing conditions, including payment and a specified access period. Access to encrypted data requires rights to the corresponding cryptographic key, supporting authorization through cryptographic key control.*
8. A spatial resolution module configured to determine spatial resolution gates based on viewer spatial context and provide full fidelity for a foveal region, reduced fidelity for a peripheral region, and withhold media for an out-of-view region.	N/A
9. A prefetch mechanism configured to retrieve candidate media for	**[Column 08; Lines 53-62]** The recorded reception hint tracks (first/second data packets) and the recorded associated reception hint tracks (first/second control packets) span an entire recording. **During playback or replay, control data of the**

possible traversal branches into a decoupled non-executable buffer before branch selection, while withholding the corresponding key or authorization until token validation.	**associated reception hint track, e.g. in form of RTCP SRs, may be pre-fetched, e.g. for accurate lip-synchronization by taking multiple future RTCP SRs into account, wherein the future RTCP SRs relate to time instants in the future of a current playback time instant.** **[Column 25; Lines 08-14]** For key stream reception hint tracks, the first mode of operation is that the key stream reception hint track is consumed in parallel to the RTP or MPEG-2 TS reception hint track. **This ensures that key stream data for a particular protected data block of the reception hint tracks is available in a similar fashion as with a real broadcast.** **[Column 26; Lines 18-34]** FIG. 12 shows streamed data packets 1302-1, 1302-2, 1302-3, each having a payload encrypted with cryptographic keys k0, k1, k2. Associated to the data packets 1302 there is a key stream comprising key stream packets 1304-1, 1304-2, 1304-3 and 1304-4. **The key stream packet 1304-1, which is delivered in time before its associated data packet 1302-1 comprises a cryptographic key k0 for the decryption of the encrypted payload of data packet 1302-1. Thereby, the encryption key k0 has a lifetime d0 associated with it assuring that the associated data packets can be encrypted.** The same holds for the second key stream packet 1304-2 and its cryptographic key k1, which may be used for decrypting the payload of the associated data packet 1302-2. **Also here the associated key stream packet 1304-2 is delivered well ahead of time before data packet 1302-2, and the cryptographic key k1 has a lifetime d1 ensuring correct decryption of the payload of data packet 1302-2.** **Note:** *The mapped citation describes prefetching playback control data, including future RTCP sender reports, and providing key-stream data for protected media in advance of the corresponding media packets. This advance availability of playback and decryption information can be functionally inferred to support preparation of media for subsequent playback.*
10. A playback assembly mechanism configured to retrieve, decrypt, authorize, and sequence media sample blocks in volatile memory according to traversal authorization and spatial resolution, thereby generating continuous playback without writing a rendered video file to persistent storage.	**[Column 10; Lines 35-53] For replaying virtual media tracks, embodiments of the present invention provide an apparatus for reading a file, the file having stored, in a media data container related to the file, data packets comprising a payload, and having stored, in the media data container, decoding information for the payload of the stored data packets, wherein the decoding information indicates at which time instant to replay which payload of the stored data packets**, the file having stored associated metadata in a media data container, the associated metadata indicating a decoding time and a location of the decoding information in the media data container. **The apparatus comprises a processor for determining an output schedule of the payload of the stored data packets by accessing the associated metadata in the media data container and by accessing, based on the metadata, the decoding information in the media data container, and by accessing, based on the decoding information, the payload of the stored data packets.** An output controller serves for outputting the payload in accordance with the determined output schedule. **[Column 28; Lines 08-17] The processor for assigning decryption information may be used for a player of the payload of the encrypted data packets 1110. For this reason, decrypted data packets may be output to a decoder based on the**

	assigned decryption information. The processor for assigning the decryption information may also be used to generate virtual decryption information to be stored partly in the media data container 1104 as virtual key samples. Associated meta information is stored in metadata track 1128. This corresponds to the concept of the virtual media tracks described above. **Note:** *The mapped citation describes retrieving media payloads using metadata and decoding information, determining an output schedule, and outputting decrypted data packets to a decoder according to the schedule. This supports controlled retrieval, decryption, sequencing, and playback of protected media. However, volatile-memory-only playback, traversal authorization, spatial-resolution-based playback, and continuous playback without writing a rendered video file to persistent storage are not expressly disclosed.*
11. A policy enforcement mechanism configured to apply inherited policies to recursively nested containers, wherein the most restrictive applicable policy governs traversal.	**[Column 25-26; Lines 61-67, 01-11]** Rights to access data or data packets may be controlled via a rights management system. **Receiving data content via a digital communications network may be limited to certain end-users and restricted from other users.** E.g., a user may purchase access to a program by paying a fee for the program. **If the user pays the fee the user may be granted access to the program for a specified period of time while a user who has not paid the fee may not have access to the program. Access to the program may be regulated by encryption of the transmitted data. The data may be encrypted by any number of encryption standards through the use of an encryption key. At the receiver or user terminal, a key may be used to decrypt the encrypted data such that the content may be viewable at the receiver or user terminal.** The key for decrypting the encrypted data packets may also be delivered via the same digital communications network and may also be encrypted. For the delivery of one or more keys also other digital communications networks can be used. Thus, **an end-user wishing to access or view the program or service may need to obtain the rights to the keys.** **Note:** *The mapped citation describes a rights management system that restricts access to protected content based on payment, access periods, encryption, and rights to cryptographic keys, supporting policy-based media access control. However, applying inherited policies to recursively nested containers is not expressly disclosed.*
12. The traversal framework further enforces cycle detection, maximum traversal-depth restrictions, and real-time rights verification.	N/A
13. A rights management mechanism configured to dynamically compose video content from independently owned content libraries using container-specific authorization and maintain	**[Column 25-26; Lines 61-67, 01-11] Rights to access data or data packets may be controlled via a rights management system.** Receiving data content via a digital communications network may be limited to certain end-users and restricted from other users. E.g., a user may purchase access to a program by paying a fee for the program. If the user pays the fee the user may be granted access to the program for a specified period of time while a user who has not paid the fee may not have

an audit record of traversal events.	access to the program. **Access to the program may be regulated by encryption of the transmitted data. The data may be encrypted by any number of encryption standards through the use of an encryption key. At the receiver or user terminal, a key may be used to decrypt the encrypted data such that the content may be viewable at the receiver or user terminal.** The key for decrypting the encrypted data packets may also be delivered via the same digital communications network and may also be encrypted. For the delivery of one or more keys also other digital communications networks can be used. Thus, **an end-user wishing to access or view the program or service may need to obtain the rights to the keys.** **[Column 26; Lines 18-34]** FIG. 12 shows **streamed data packets 1302-1, 1302-2, 1302-3, each having a payload encrypted with cryptographic keys k0, k1, k2. Associated to the data packets 1302 there is a key stream comprising key stream packets 1304-1, 1304-2, 1304-3 and 1304-4. The key stream packet 1304-1, which is delivered in time before its associated data packet 1302-1 comprises a cryptographic key k0 for the decryption of the encrypted payload of data packet 1302-1.** Thereby, the encryption key k0 has a lifetime d0 associated with it assuring that the associated data packets can be encrypted. **The same holds for the second key stream packet 1304-2 and its cryptographic key k1, which may be used for decrypting the payload of the associated data packet 1302-2.** Also here **the associated key stream packet 1304-2 is delivered well ahead of time before data packet 1302-2, and the cryptographic key k1 has a lifetime d1 ensuring correct decryption of the payload of data packet 1302-2.** **Note:** *The mapped citation describes rights management that restricts content access based on authorization, payment, access periods, and rights to cryptographic keys, including keys associated with specific encrypted media packets and their lifetimes. This supports controlled access to media content. However, dynamically composing video content from independently owned libraries and maintaining an audit record of traversal events are not expressly disclosed.*
14. The rights management mechanism further generates financial settlement information for respective content owners based on traversed branches.	N/A

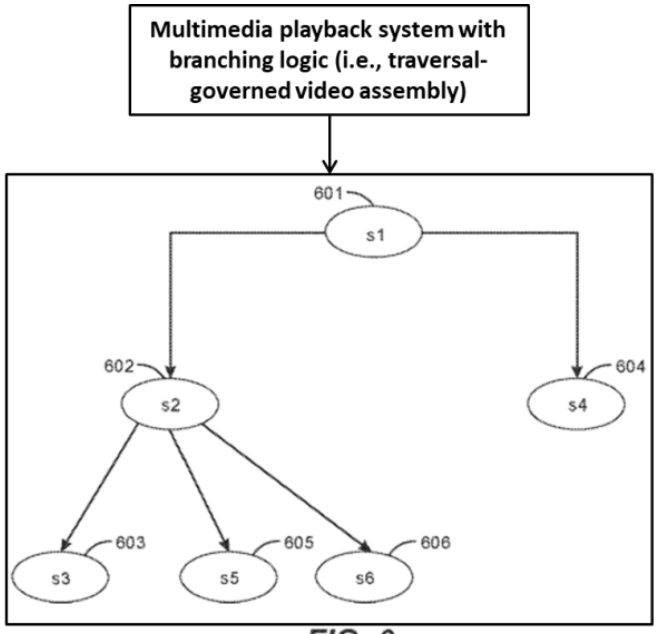
Patent Citation 3: US20240013810A1

Title	Branching logic in a playback environment
Priority Date	30 JAN 2019
Filing Date	19 SEP 2023
Publication Date	11 JAN 2024
Inventors	Lee Kevin Pei-Wen; Jones Lawrence E.; Lea IV Joseph Peyton
Assignees	Netflix Inc.
IPC Classes	G06F16/44; G11B27/10; H04N21/472
CPC Classes	G06F16/44; G11B27/031; G11B27/10; G11B27/102; H04N21/47205; H04N21/47217; H04N21/8456; H04N21/85406; H04N21/8541
US Classes	**None**
Family Members	**MX2021001993A | AU2020216187B2 | US11217280B2 | EP3918810A1 | BR112021015042A2 | WO2020160341A1 | US11837261B2 | US11763852B2**

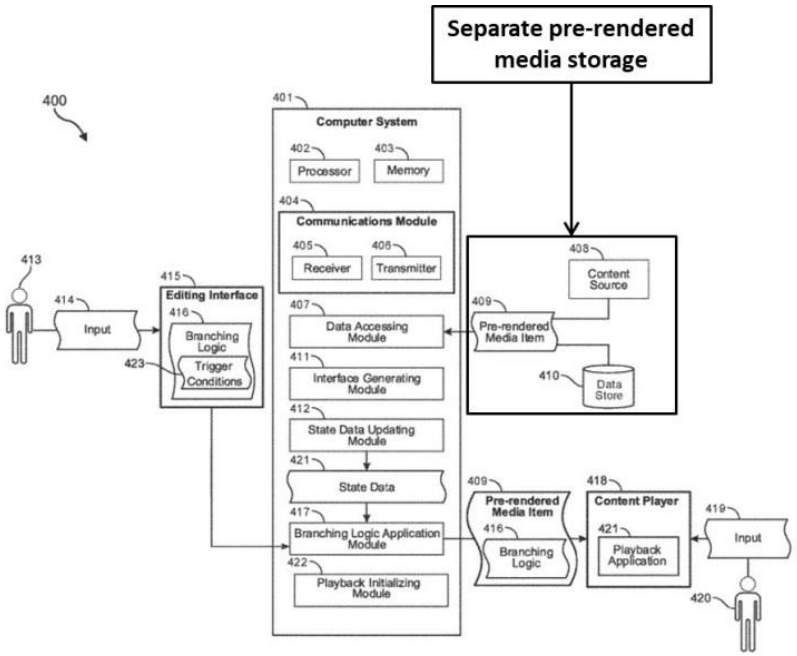
Abstract:

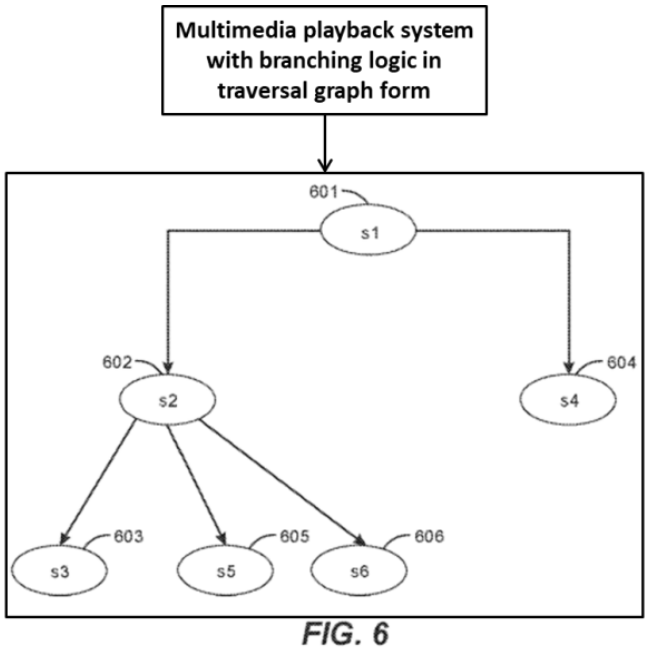
The disclosed computer-implemented method may include accessing a pre-rendered multimedia item. The pre-rendered multimedia item may have branching logic associated with it, where the branching logic includes branching points that direct non-sequential playback of the pre-rendered multimedia item. The method may also include initializing playback of the pre-rendered multimedia item and accessing, at the branching points, various trigger conditions that direct playback order of different segments of the pre-rendered multimedia item. The method may then include updating, based on the trigger conditions, at least some portion of custom state data. The method may further include playing back the segments of the pre-rendered multimedia item according to the updated state data. Various other methods, systems, and computer-readable media are also disclosed.

Key Features	Relevant Excerpts
1. A system for traversal-governed video assembly.	**[Para 0005]** In one example, **a computer-implemented method for playing back a pre-rendered multimedia item according to branching logic may include accessing a pre-rendered multimedia item. The pre-rendered multimedia item may have branching logic associated with it, where the branching logic includes branching points that direct non-sequential playback of the pre-rendered multimedia item. The method may also include initializing playback of the pre-rendered multimedia item and accessing, at the branching points, various trigger conditions that direct playback order of different segments of the pre-rendered multimedia item. The method may then include updating, based on the trigger conditions, at least some portion of custom state data. The method may further include playing back the segments of the pre-rendered multimedia item according to the updated state data**. **[Para 0066]** …Each segment, as shown in the branching narrative graph of FIG. 6 ,

	may have zero or more branching points associated with it. **Segment s1 (601) may have a branching point associated with it, for example, that branches to segment s2 (602) or segment s4 (604). Segment s4 may have no branching points, while segment s2 may have a branching point with three different options that lead to segment s3 (603), segment s5 (605), or to segment s6 (606). Segments s3, s5, and s6 may also have branching points associated therewith that, although not shown, take the viewer to other segments**.  *FIG. 6* **Note:** *The mapped citation describes a computer-implemented multimedia playback system using branching logic for non-sequential playback. Branching points and trigger conditions determine playback paths among interconnected multimedia segments, forming a branching narrative structure. Accordingly, the disclosure supports traversal-governed assembly and playback through selection and sequencing of interconnected segments.*
2. A virtualization engine configured to generate a virtual video reference container by removing the media payload (MDAT) while retaining structural and timing metadata (STBL).	**[Para 0062] As the viewer 420 progresses through the pre-rendered media item 409 (e.g., while watching the media item and making choices at branching points), the state data generating module 412 of computer system 401 may generate custom state data 421 which may be stored with or separate from the branching logic. The state data 421, as will be explained further below, allows the content player 418 to track which decisions the user 420 has made at each branching point, and also stores items associated with the user, preferences associated with the user, or other information used in directing the non-sequential playback of the pre-rendered media item 409.** This custom state data may be applied to the pre-rendered media item 409 along with the branching logic 416 by the branching logic application module 417. **[Para 0095]** …An automatic branch decision may be executed as follows: 1) The application enters a new logical segment (e.g., a movie scene) for a branching narrative video. 2) **The UI checks the segment group's map or registry and checks for a segment group identifier that matches the identifier of the current logical segment.** 2a) If an entry does not exist, then no additional steps are executed.

	Otherwise, execution continues to Step 3. 3) The UI takes the segment group definition and runs it through the rules engine. If the precondition for a rule resolves to true, then the UI queues up the associated segment identifier as the next logical segment that should be played after the current segment is played. **In a user-initiated branch decision: branching point video moments may be defined using a special choices array. Each entry in the choices array maps to specified metadata that will include either a segment group identifier or a segment identifier. If a segment identifier is provided, then the UI may take the user to that particular segment if the associated choice is selected.** However, if a segment group identifier is provided, then the UI will look up the segment group from the segment group map/registry and execute the segment group via the rules engine to determine which segment to go to next. **[Para 0123]** 17. The system of claim 12, further comprising **storing at least a portion of state data at a playback location of the pre-rendered multimedia item that is different than at a branching point.** **Note:** *The mapped citation describes a system that separates pre-rendered media content from generated state data and segment-related metadata, including identifiers used to control playback. This conceptually corresponds to retaining structural and timing/selection information while separately maintaining the underlying media payload, similar to retaining STBL metadata while removing or externally maintaining the MDAT payload.*
3. The retained metadata comprises rewritten sample table references identifying media sample blocks stored externally from the virtual video reference container.	**[Para 0060]** …**In some cases, the pre-rendered media item 409 may be stored in a data store 410 which may be a local or distributed data store (e.g., a cloud data store).** In other cases, **the pre-rendered media item 409 may be streamed directly from a content source 408 to a user (e.g., 420) or from a server owned or managed by the content source 408.** In still other cases, the pre-rendered media item 409 may be provided to computer system 401 where branching logic may be applied prior to or during streaming. **[Para 0062] As the viewer 420 progresses through the pre-rendered media item 409 (e.g., while watching the media item and making choices at branching points), the state data generating module 412 of computer system 401 may generate custom state data 421 which may be stored with or separate from the branching logic. The state data 421, as will be explained further below, allows the content player 418 to track which decisions the user 420 has made at each branching point, and also stores items associated with the user, preferences associated with the user, or other information used in directing the non-sequential playback of the pre-rendered media item 409.** This custom state data may be applied to the pre-rendered media item 409 along with the branching logic 416 by the branching logic application module 417. **[Para 0095]** …An automatic branch decision may be executed as follows: 1) The application enters a new logical segment (e.g., a movie scene) for a branching narrative video. 2) **The UI checks the segment group's map or registry and checks for a segment group identifier that matches the identifier of the current logical segment.** 2a) If an entry does not exist, then no additional steps are executed. Otherwise, execution continues to Step 3. 3) The UI takes the segment group

	definition and runs it through the rules engine. If the precondition for a rule resolves to true, then the UI queues up the associated segment identifier as the next logical segment that should be played after the current segment is played. **In a user-initiated branch decision: branching point video moments may be defined using a special choices array. Each entry in the choices array maps to specified metadata that will include either a segment group identifier or a segment identifier. If a segment identifier is provided, then the UI may take the user to that particular segment if the associated choice is selected.** However, if a segment group identifier is provided, then the UI will look up the segment group from the segment group map/registry and execute the segment group via the rules engine to determine which segment to go to next.  **Note:** *The mapped citation describes pre-rendered media maintained in a separate local, distributed, or cloud data store, while branching logic and state data retain segment identifiers for selecting particular media segments. The associated metadata thus provides references for locating separately stored media content, corresponding to retained sample-reference metadata identifying externally maintained media blocks..*
4. A graph composition mechanism configured to interconnect multiple virtual video reference containers through traversal references to form a traversable video graph, including recursive or nested traversal.	**[Para 0005]** In one example, **a computer-implemented method for playing back a pre-rendered multimedia item according to branching logic may include accessing a pre-rendered multimedia item. The pre-rendered multimedia item may have branching logic associated with it, where the branching logic includes branching points that direct non-sequential playback of the pre-rendered multimedia item. The method may also include initializing playback of the pre-rendered multimedia item and accessing, at the branching points, various trigger conditions that direct playback order of different segments of the pre-rendered multimedia item. The method may then include updating, based on the trigger conditions, at least some portion of custom state data. The method may further include playing back the segments of the pre-rendered multimedia item according to the updated state data**. **[Para 0066]** …**Each segment, as shown in the branching narrative graph of FIG. 6**

	, may have zero or more branching points associated with it. Segment s1 (601) may have a branching point associated with it, for example, that branches to segment s2 (602) or segment s4 (604). Segment s4 may have no branching points, while segment s2 may have a branching point with three different options that lead to segment s3 (603), segment s5 (605), or to segment s6 (606). Segments s3, s5, and s6 may also have branching points associated therewith that, although not shown, take the viewer to other segments. **[Para 0071] As noted above, FIG. 6 illustrates an embodiment of a branching narrative graph. Unlike traditional forms of branching narratives (e.g., video games) where scenes may be rendered dynamically on the fly, the branching structure and the comprehensive set of segments or scenes that make up the narrative graph (e.g., s1-s6) are compiled together at encode-time for branching narrative videos.** Since a video or similar media item is a linear structure, the video frames may be initially laid out from s1-s6 (or in some other order). **In such cases, the entire video is composed of logical segments that map 1-to-1 to the nodes in the branching narrative graph.**  FIG. 6 **Note:** *The mapped citation describes a branching narrative graph comprising multiple logical video segments (e.g., "s1–s6") interconnected through branching points that direct playback among segments. The segments map to graph nodes, forming a traversable video graph, while nested branching points enable recursive traversal to additional segments.*
5. A runtime traversal engine configured to evaluate dynamic branch conditions based on real-time signals and select a target video container from multiple candidate containers, wherein the signals include viewer input, AI agent directives, rights	**[Para 0006]** In some examples, the custom state data includes various primitives or complex objects. **In some examples, at least one of the primitives or complex objects includes a precondition definition that specifies a precondition that is to occur before at least one of the branching points is triggered. In some examples, the precondition is executed at runtime to determine whether a branching point selection associated with the precondition is to be displayed during playback.** **[Para 0067] At a branching point, for example, at s1 (601), a user (e.g., 420 of FIG. 4) may make a selection via input 419**. This input may be interpreted by the

policies, or environmental context.	content player 418 using the branching logic 416 that is streamed with the pre-rendered media item 409 or is accessed separate from the pre-rendered media item. The branching logic 416 may indicate that input "A" leads to segment s2 (602) and input "B" leads to segment s4 (604). **Because the pre-rendered media item 409 has already been rendered sequentially, with segments s1-s6 being rendered and stored one consecutively after the other, the content player 418 skips between segments (e.g., skipping from s1 to s4 if input "B" is provided). In other embodiments, where the pre-rendered media item 409 is streamed to the content player, the content source 408 or computer system 401 (which may belong to or may be controlled by the content source) may stream those segments that are to be presented on the content player 418 based on the user's inputs 419.** Thus, while the data store 410 or content source 408 store the pre-rendered media item 409 in a single, pre-rendered media file, the data store or content source may provision those segments from the pre-rendered media item 409 that are selected by the user according to branching logic 416. **[Para 0075]** In other cases, a trigger condition 423 may related to time. **For instance, a time-based trigger condition may trigger execution of certain branching point logic that (re)directs playback order of certain media segments after a certain amount of time has elapsed, or when a certain time in the duration of the media item is reached, or when a certain real-world time has been reached.** Many other time-based variants are also possible. Still further, the trigger condition 423 may be based on an evaluation of the current condition or state of the custom state data 421. **The custom state data at one branching point may be set to "True" or "False" or "Sword" or "Pistol" or "Character A" or "Character B" or some other value.** Based on the current state of this custom state data 421, the trigger condition 423 may be met or not met, or may be met in a specific way that causes a specific action. Accordingly, the trigger condition 423 may cause execution of certain branching point logic based on the current condition or state of that state data. Other types of trigger conditions 423 may also be used, alone or in conjunction with the trigger conditions described above. **Note:** *The mapped citation describes a runtime content player that evaluates preconditions, trigger conditions, user input, and state data at branching points to determine playback redirection. Based on these conditions, the player selects and streams a target segment from multiple alternatives. Accordingly, this corresponds to a runtime traversal engine evaluating dynamic branch conditions and selecting a target video container.*
6. A token validation module configured to validate an independent cryptographic video token associated with the selected target video container and grant traversal authority upon successful validation.	N/A
7. The cryptographic video token defines or enforces licensing, usage, consent,	N/A

access, or financial settlement conditions, and successful validation releases an ephemeral decryption key or signed policy authorization.	
8. A spatial resolution module configured to determine spatial resolution gates based on viewer spatial context and provide full fidelity for a foveal region, reduced fidelity for a peripheral region, and withhold media for an out-of-view region.	N/A
9. A prefetch mechanism configured to retrieve candidate media for possible traversal branches into a decoupled non-executable buffer before branch selection, while withholding the corresponding key or authorization until token validation.	**[Para 0048]** In one embodiment, playback application 310 begins downloading the content associated with the requested title by downloading sequence data encoded to the lowest audio and/or video playback bitrates to minimize startup time for playback. **The requested digital content file is then downloaded into content buffer 312, which is configured to serve as a first-in, first-out queue.** In one embodiment, each unit of downloaded data includes a unit of video data or a unit of audio data. **As units of video data associated with the requested digital content file are downloaded to the content player 120, the units of video data are pushed into the content buffer 312.** Similarly, as units of audio data associated with the requested digital content file are downloaded to the content player 120, the units of audio data are pushed into the content buffer 312. In one embodiment, the units of video data are stored in video buffer 316 within content buffer 312 and the units of audio data are stored in audio buffer 314 of content buffer 312. **[Para 0049] A video decoder 320 reads units of video data from video buffer 316 and outputs the units of video data in a sequence of video frames corresponding in duration to the fixed span of playback time. Reading a unit of video data from video buffer 316 effectively de-queues the unit of video data from video buffer 316. The sequence of video frames is then rendered by graphics interface 326 and transmitted to graphics device 328 to be displayed to a user.** **[Para 0061]** For example, the data accessing module 407 may access pre-rendered media item 409 from content source 408 or from data store 410 after which the interface generating module 411 may generate an editing interface (e.g., 415). **The editing interface may be used to define and/or apply branching logic 416 to the pre-rendered media item 409.** In some cases, the branching logic 416 is applied by a content manager 413 or other user via input 414. In other cases, the branching logic may potentially be applied by a software application or software algorithm designed to automatically generate branching logic for a portion of multimedia content. **The "branching logic," as the term is used herein, may include data that defines how a user may interact with the pre-rendered media item 409. For example, the branching logic 416 may specify branching points at**

	which a user may take one of a variety of different routes through the pre-rendered media item 409. For instance, while viewing the pre-rendered media item 409, the viewer 420 may be presented with a choice of X, Y, or Z. Each of these options may direct playback through a different series of media segments within the pre-rendered media item 409. Other branching logic 416 may specify items that the user has collected (e.g., a virtual sword, or virtual currency), **or may include preferences associated with the user, or may provide other decisions that the user can make to modify playback of the media content in some form.** **[Para 0096]** In some cases, the rules engine may not have enough information to determine which logic segment should be played next. In some cases, this may occur because the system is waiting on user input. However, in such cases, **the system may have enough information to narrow down the possible set of next logical segments. This may be helpful for video playback since it helps optimize the content-prefetching logic for the video.** Indeed, the system may make on-the-fly modifications to branch weights based on which are more or less likely to be taken by the user. Those more likely to be selected may have their associated branch weights increased, and those less likely to be selected may have their associated branch weights decreased. **Those branches having higher weights may be prefetched for faster provisioning to the client device**. **Note:** *The mapped citation describes downloading video and audio data into a content buffer for decoding and playback. When a branch decision is pending, multiple candidate segments may be identified, weighted by likelihood, and prefetched for faster provisioning. Thus, the disclosure supports buffering candidate media for potential traversal branches before selection. However, withholding a corresponding cryptographic key or authorization until token validation is not expressly disclosed.*
10. A playback assembly mechanism configured to retrieve, decrypt, authorize, and sequence media sample blocks in volatile memory according to traversal authorization and spatial resolution, thereby generating continuous playback without writing a rendered video file to persistent storage.	**[Para 0037]** **Memory 114 and 124 generally represent any type or form of volatile** or non-volatile **storage device or medium capable of storing data and/or computer-readable instructions.** In one example, memory 114 and/or 124 stores, loads, and/or maintains one or more of modules 116 and 126. Examples of memory 114 and/or 124 include, without limitation, random access memory (RAM), read only memory (ROM), flash memory, hard disk drives (HDDs), solid-state drives (SSDs), optical disk drives, caches, variations or combinations of one or more of the same, and/or any other suitable memory device or system. **[Para 0067]** At a branching point, for example, at s1 (601), a user (e.g., 420 of FIG. 4) may make a selection via input 419. **This input may be interpreted by the content player 418 using the branching logic 416 that is streamed with the pre-rendered media item 409 or is accessed separate from the pre-rendered media item. The branching logic 416 may indicate that input "A" leads to segment s2 (602) and input "B" leads to segment s4 (604). Because the pre-rendered media item 409 has already been rendered sequentially, with segments s1-s6 being rendered and stored one consecutively after the other, the content player 418 skips between segments (e.g., skipping from s1 to s4 if input "B" is provided)**… **[Para 0096]** …However, in such cases, **the system may have enough information to narrow down the possible set of next logical segments. This may be helpful**

	for video playback since it helps optimize the content-prefetching logic for the video. Indeed, the system may make on-the-fly modifications to branch weights based on which are more or less likely to be taken by the user. Those more likely to be selected may have their associated branch weights increased, and those less likely to be selected may have their associated branch weights decreased. **Those branches having higher weights may be prefetched for faster provisioning to the client device.** **Note:** *The mapped citation describes a playback system that stores media segments in memory, including RAM, and selects or skips between pre-rendered segments according to branching logic. Likely next segments may be weighted and prefetched when a branch decision is pending, supporting temporary storage and sequencing of candidate media during playback. However, specific decryption and authorization of media blocks, spatial-resolution-based playback, and continuous playback without writing a rendered video file to persistent storage are not expressly disclosed.*
11. A policy enforcement mechanism configured to apply inherited policies to recursively nested containers, wherein the most restrictive applicable policy governs traversal.	**[Para 0011]** In some examples, **the branching logic is interpreted by a rules engine that applies branching logic to multiple different pre-rendered multimedia items.** In some examples, **the system further stores at least a portion of state data at a playback location of the pre-rendered multimedia item that is different than at a branching point.** In some examples, the system further stores an accumulation of user selections and/or potentially previously accumulated custom user state as a snapshot in time, referred to herein as a save slot or a state snapshot. In some examples, **the system resumes playback in a subsequent playback session of the pre-rendered multimedia item and restores the user's selections from the stored state snapshot.** **[Para 0095]** …**An automatic branch decision may be executed as follows: 1) The application enters a new logical segment (e.g., a movie scene) for a branching narrative video. 2) The UI checks the segment group's map or registry and checks for a segment group identifier that matches the identifier of the current logical segment. 2a) If an entry does not exist, then no additional steps are executed. Otherwise, execution continues to Step 3. 3) The UI takes the segment group definition and runs it through the rules engine. If the precondition for a rule resolves to true, then the UI queues up the associated segment identifier as the next logical segment that should be played after the current segment is played. In a user-initiated branch decision: branching point video moments may be defined using a special choices array. Each entry in the choices array maps to specified metadata that will include either a segment group identifier or a segment identifier.** If a segment identifier is provided, then the UI may take the user to that particular segment if the associated choice is selected. However, **if a segment group identifier is provided, then the UI will look up the segment group from the segment group map/registry and execute the segment group via the rules engine to determine which segment to go to next.** **Note:** *The mapped citation describes a rules-based traversal mechanism in which branching logic is applied across multiple pre-rendered multimedia items. Segment groups, nested lookups, and rule preconditions can be inferred to allow higher-level policies to govern traversal into associated lower-level segments or groups.*

	Accordingly, the disclosure supports inherited policy enforcement for recursively-nested containers, with rules determining whether traversal to a nested segment is permitted.
12. The traversal framework further enforces cycle detection, maximum traversal-depth restrictions, and real-time rights verification.	N/A
13. A rights management mechanism configured to dynamically compose video content from independently owned content libraries using container-specific authorization and maintain an audit record of traversal events.	**[Para 0039]** As shown, storage 210 may store a variety of different items including content 212, user data 214, and/or log data 216. Content 212 includes television shows, movies, video games, user-generated content, and/or any other suitable type or form of content. **User data 214 includes personally identifiable information (PII),** payment information, preference settings, language and accessibility settings, **and/or any other information associated with a particular user or content player. Log data 216 includes viewing history information, network throughput information, and/or any other metrics associated with a user's connection to or interactions with distribution infrastructure 110.** **Note:** *The mapped citation describes dynamically managing content and user-related information, including logs of viewing history and user interactions, supporting auditing of video traversal. User-specific information can be inferred to support rights management and determine authorization for accessing content from different sources or libraries. However, independently owned content libraries with container-specific authorization are not expressly disclosed.*
14. The rights management mechanism further generates financial settlement information for respective content owners based on traversed branches.	**[Para 0039]** As shown, storage 210 may store a variety of different items including content 212, user data 214, and/or log data 216. Content 212 includes television shows, movies, video games, user-generated content, and/or any other suitable type or form of content. **User data 214 includes personally identifiable information (PII), payment information, preference settings, language and accessibility settings, and/or any other information associated with a particular user or content player.** Log data 216 includes viewing history information, network throughput information, and/or any other metrics associated with a user's connection to or interactions with distribution infrastructure 110. **Note:** *The mapped citation describes user-associated payment information, stored content, and viewing history that can support tracking content usage and related financial accounting. Thus, traversed content branches may be associated with payment and usage records for respective content providers or owners. However, generating financial settlement information for individual content owners based on specific traversed branches is not expressly disclosed.*

Patent Citation 4: US12316905B2

Title	Systems and methods for seamless audio and video endpoint transitions
Priority Date	07 JUL 2020
Filing Date	20 JUN 2024
Publication Date	27 MAY 2025
Inventors	Tal Zubalsky, Jonathan Bloch, Yuval Hofshy, Barak Feldman
Assignees	JBF Interlude 2009 Ltd
IPC Classes	G06V20/40, H04N21/44, H04N21/845
CPC Classes	G06V20/49, H04N21/44, H04N21/44016, H04N21/4532, H04N21/458, H04N21/4758, H04N21/8456, H04N21/8541, G06V20/49
US Classes	None
Family Members	**US12047637B2 | US20250280169A1**

Abstract:

To provide for seamless audio, video, and contextual transitions in interactive media presentations, endpoints within the presentation are defined. The endpoints are disposed at different times within the presentation and form respective links between points in segments in the presentation. While the presentation is being played, a user interaction is received and an appropriate endpoint is selected based on the interaction. When the selected endpoint is reached, the presentation is seamlessly transitioned to the point in the segment linked to by the endpoint.

Key Features	Relevant Excerpts
1. A system for traversal-governed video assembly.	**[Column 09; Lines 19-34] Project Configuration File 230 can include information defining the media presentation, such as the video tree or other structure, and how video segments can be linked together in various manners to form one or more paths. Project Configuration File 230 can further specify which audio, video, and/or other media files correspond to each segment (e.g., node in a video tree), that is, which audio, video, and/or other media should be retrieved when application 112 determines that a particular segment should be played.** Additionally, Project Configuration File 230 can indicate interface elements that should be displayed or otherwise presented to users, as well as when the elements should be displayed, such that the audio, video, and interactive elements of the media presentation are synchronized. Project Configuration File 230 can be stored on user device 110 or can be remotely accessed by Choice Manager 216. **[Column 05; Lines 18-34] The media presentations referred to herein can be structured in various forms. For example, a particular media presentation can be an online streaming video having multiple tracks or streams that a user can**

	interactively switch among in real-time or near real-time. For example, a media presentation can be structured using parallel audio and/or video tracks as described in U.S. Patent Application Pub. No. 2016/0105724, published on Apr. 14, 2016, and entitled "Systems and Methods for Parallel Track Transitions," the entirety of which is incorporated by reference herein. More specifically, **a playing video file or stream can have one or more parallel tracks that can be switched among in real-time automatically and/or based on user interactions.** In some implementations, such switches are made seamlessly and substantially instantaneously (within milliseconds), such that the audio and/or video of the playing content can continue without any perceptible delays, gaps, or buffering. **[Column 05; Lines 42-55]** **Such media presentations can also include interactive video structured in a video tree, hierarchy, or other form or structure. A video tree can be formed by nodes that are connected in a branching, hierarchical, or other linked form. Nodes can each have an associated video segment, audio segment, graphical user interface (GUI) elements, and for other associated media. Users (e.g., viewers) can watch a video that begins from a starting node in the tree and proceeds along connected nodes in a branch or path.** Upon reaching a point during playback of the video where multiple video segments (child nodes) branch off from a segment (parent node), the user can interactively select the branch or path to traverse and, thus, the next video segment to watch. **Note:** *The mapped citation describes a media presentation system with interconnected nodes and video segments forming traversable paths, where user selection of a branch causes the corresponding media segment to be retrieved and played. Real-time or near-real-time switching among multiple audio/video tracks based on user interaction supports seamless assembly and playback of selected media, functionally corresponding to traversal-governed video assembly.*
2. A virtualization engine configured to generate a virtual video reference container by removing the media payload (MDAT) while retaining structural and timing metadata (STBL).	**[Column 05; Lines 42-55]** Such media presentations can also include interactive video structured in a video tree, hierarchy, or other form or structure. **A video tree can be formed by nodes that are connected in a branching, hierarchical, or other linked form. Nodes can each have an associated video segment, audio segment, graphical user interface (GUI) elements, and/or other associated media.** Users (e.g., viewers) can watch a video that begins from a starting node in the tree and proceeds along connected nodes in a branch or path. Upon reaching a point during playback of the video where multiple video segments (child nodes) branch off from a segment (parent node), the user can interactively select the branch or path to traverse and, thus, the next video segment to watch. **Note:** *The mapped citation describes an interactive video structure comprising nodes connected in a branching, hierarchical, or linked form, wherein each node is associated with a video segment, audio segment, GUI elements, and/or other media. The node-based structure therefore provides a structural representation of the video presentation and identifies the media content associated with respective nodes.*
3. The retained metadata comprises rewritten sample table references	N/A

identifying media sample blocks stored externally from the virtual video reference container.	
4. A graph composition mechanism configured to interconnect multiple virtual video reference containers through traversal references to form a traversable video graph, including recursive or nested traversal.	**[Column 05; Lines 42-55]** Such media presentations can also include interactive video structured in a video tree, hierarchy, or other form or structure. **A video tree can be formed by nodes that are connected in a branching, hierarchical, or other linked form. Nodes can each have an associated video segment, audio segment, graphical user interface (GUI) elements, and/or other associated media. Users (e.g., viewers) can watch a video that begins from a starting node in the tree and proceeds along connected nodes in a branch or path.** Upon reaching a point during playback of the video where multiple video segments (child nodes) branch off from a segment (parent node), the user can interactively select the branch or path to traverse and, thus, the next video segment to watch. **[Column 05-06; Lines 56-67, 01-03]** As referred to herein, **a particular branch or path in an interactive media structure, such as a video tree, can refer to a set of consecutively linked nodes between a starting node and ending node, inclusively, or can refer to some or all possible linked nodes that are connected subsequent to (e.g., sub-branches) or that include a particular node.** Branched video can include seamlessly assembled and selectably presentable multimedia content such as that described in U.S. Patent Application Pub. No. 2011/0200116, published on Aug. 18, 2011, and entitled "System and Method for Seamless Multimedia Assembly", and U.S. Patent Application Pub. No. 2015/0067723, published on Mar. 5, 2015, and entitled "Methods and Systems for Unfolding Video Pre-Roll," the entireties of which are hereby incorporated by reference. **Note:** *The mapped citation describes an interactive video structure comprising interconnected nodes associated with respective video segments, where the nodes can be functionally regarded as virtual video reference containers. Connected nodes define traversal paths, with subsequent nodes including sub-branches that correspond to traversal references and recursive or nested traversal. User selection of a branch or path directs traversal to a corresponding subsequent video segment, supporting traversal-based target container or node selection.*
5. A runtime traversal engine configured to evaluate dynamic branch conditions based on real-time signals and select a target video container from multiple candidate containers, wherein the signals include viewer input, AI agent directives, rights policies, or environmental context.	**[Column 05; Lines 42-55]** Such media presentations can also include interactive video structured in a video tree, hierarchy, or other form or structure. A video tree can be formed by nodes that are connected in a branching, hierarchical, or other linked form. Nodes can each have an associated video segment, audio segment, graphical user interface (GUI) elements, and/or other associated media. Users (e.g., viewers) can watch a video that begins from a starting node in the tree and proceeds along connected nodes in a branch or path. **Upon reaching a point during playback of the video where multiple video segments (child nodes) branch off from a segment (parent node), the user can interactively select the branch or path to traverse and, thus, the next video segment to watch.** **[Column 13; Lines 21-33]** FIG. 3A depicts an example branching multimedia presentation 300 with multiple nodes and branching paths. For purposes of simplifying visualization, the presentation 300 includes a small number of nodes

	and paths; however, it is to be appreciated that significantly more complex presentations are contemplated. **The presentation includes seven nodes (A, B1, B2, B3, C1, C2, and D) and four distinct paths (A→B1→C1; A→B1→C2; A→B2→D; and A→B3→D). Node A can include a decision period during which a user can make choices that determine whether the presentation will then proceed from Node A to Node B1, B2, or B3. Similarly, Node B1 can include a decision period the result of which guides the presentation into Node C1 or Node C2.** **Note:** *The mapped citation describes an interactive video presentation having multiple nodes and branching paths, wherein, upon reaching a branching point, a user can interactively select a branch or path to determine the next video segment to be watched (i.e., selecting a target video node or container from multiple candidate nodes based on a viewer or user input).*
6. A token validation module configuredto validate an independent cryptographic video token associated with the selected target video container and grant traversal authority upon successful validation.	N/A
7. The cryptographic video token defines or enforces licensing, usage, consent, access, or financial settlement conditions, and successful validation releases an ephemeral decryption key or signed policy authorization.	N/A
8. A spatial resolution module configured to determine spatial resolution gates based on viewer spatial context and provide full fidelity for a foveal region, reduced fidelity for a peripheral region, and withhold media for an out-of-view region.	**[Column 08-09; Lines 66-67, 01-18] Inputs Collector 244 receives user inputs 240 from input components such as a** device display screen 272, keyboard, mouse, microphone, virtual reality headset, and the like. **Such inputs 240 can include,** for example, mouse clicks, keyboard presses, touchpad presses, **eye movement, head movement, voice input, etc. Inputs Collector 244 provides input information based on the inputs 240 to Profile Manager 212 and Choice Manager 216, the latter of which also receives information from Profile Manager 212 as well as a Project Configuration File 230 to determine which video segment should be currently played and which video segments may be played or presented as options to be played at a later time (e.g., influenced by information from the User Profile 210).** Choice Manager 216 notifies Video Appender 270 of the video segment to be currently played, and Video Appender 270 seamlessly connects that video segment to the video stream being played in real time. **Choice Manager 216 notifies Loading Manager 262 of the video segments that may be played or presented as options to be played at a later time.**

	[Column 10; Lines 49-64] Choice Manager 216 and Loading Manager 262 can function to manage the downloading of hosted streaming media according to a loading logic. In one implementation, **Choice Manager 216 receives information defining the media presentation structure from Project Configuration File 230 and, using information from Inputs Collector 244 and Profile Manager 212, determines which media segments to download and/or buffer (e.g., if the segments are remotely stored).** For example, if Choice Manager 216 informs Loading Manager 262 that a particular segment A will or is likely to be played at an upcoming point in the presentation timeline, Loading Manager 262 can intelligently request the segment for download, as well as additional media segments X, Y and Z that can be played following segment A, in advance of playback or notification of potential playback thereof… **[Column 17; Lines 12-22]** In another implementation, **endpoints are used to facilitate seamless transitions among parallel tracks or nodes. Typically, with parallel tracks, the user can switch among the tracks in substantially real-time. The track that the user has selected is presented to the user, while the other tracks continue playing either with reduced focus or completely invisible to the user.** Similarly to television channels, the content on one track does not pause if the user switches to view content on another track. However, an instant switch from one track to another can result in a transition that is not seamless. **Note:** *The mapped citation describes receiving viewer-related inputs, including eye and head movements, to determine the video segment to be played and available alternatives, supporting context-based media presentation. The Input Collector and Profile Manager further use viewer input to determine which media segments to download or buffer. For parallel tracks, the selected track is presented with higher focus while other tracks are reduced or suppressed, supporting context-based presentation according to viewer input.*
9. A prefetch mechanism configured to retrieve candidatemedia for possible traversal branches into a decoupled non-executable buffer before branch selection, while withholding the corresponding key or authorization until token validation.	**[Column 09; Lines 35-62]** In some implementations, **Project Configuration File 230 is also used in determining which media files should be loaded or buffered prior to being played (or potentially played). Because decision points can occur near the end of a segment, it may be necessary to begin transferring one or more of the potential next segments to viewers prior to a selection being made.** For example, **if a viewer is approaching a decision point with three possible branches, all three potential next segments can be preloaded partially or fully to ensure a smooth transition upon conclusion of the current segment.** Intelligent buffering and progressive downloading of the video, audio, and/or other media content can be performed as described in U.S. Patent Application Pub. No. 2013/0259442, published Oct. 3, 2013, and entitled "Systems and Methods for Loading More Than One Video Content at a Time," the entirety of which is incorporated by reference herein. Similarly, where **endpoints (as described below) are used in a branching video presentation, parallel track presentation, or other media presentation, buffering or pre-loading of content associated with one or more endpoint destinations can occur so that seamless transitions can be made between points in the presentation linked by endpoints.** For example, if a decision period in a video segment has three defined endpoints that link to three different points in one or more other video segments,

	the video content associated with all three endpoint destinations can be downloaded and buffered at an appropriate time prior to reaching any of the endpoints. **Note:** *The mapped citation describes loading or buffering media files in advance and transferring potential next segments before a selection is made, supporting prefetching of candidate media for possible traversal branches. At a decision point with three possible branches, all potential next segments may be partially or fully preloaded, with content for multiple endpoint destinations downloaded and buffered before the corresponding endpoint is reached.*
10. A playback assembly mechanism configured to retrieve, decrypt, authorize, and sequence media sample blocks in volatile memory according to traversal authorization and spatial resolution, thereby generating continuous playback without writing a rendered video file to persistent storage.	**[Column 10; Lines 25-43]** **In addition to reading information from Project Configuration File 230, Choice Manager 216 is notified of user interactions (e.g., mouse clicks, keyboard presses, touchpad presses, eye movements, etc.) from Inputs Collector 244, which interactions can be translated into actions associated with the playback of a media presentation (e.g., segment selections, playback controls, etc.). Based thereon, Choice Manager 216 notifies Loading Manager 262, which can process the actions as further described below. Choice Manager 216 can also interface with Loading Manager 262 and Video Appender 270.** For example, Choice Manager 216 can listen for user interaction information from Inputs Collector 244 and notify Loading Manager 262 when an interaction by the viewer (e.g., a selection of an option displayed during the video) has occurred. In some implementations, based on its analysis of received events, Choice Manager 216 causes the presentation of various forms of sensory output, such as visual, aural, tactile, olfactory, and the like. **[Column 10; Lines 44-67]** As earlier noted, **Choice Manager 216 can also notify Loading Manager 262 of video segments that may be played at a later time, and Loading Manger 262 can retrieve the corresponding videos 225 (whether stored locally or on, e.g., content server 102) to have them prepared for potential playback through Video Appender 270.** Choice Manager 216 and Loading Manager 262 can function to manage the downloading of hosted streaming media according to a loading logic. In one implementation, **Choice Manager 216 receives information defining the media presentation structure from Project Configuration File 230 and, using information from Inputs Collector 244 and Profile Manager 212, determines which media segments to download and/or buffer (e.g., if the segments are remotely stored).** For example, if Choice Manager 216 informs Loading Manager 262 that a particular segment A will or is likely to be played at an upcoming point in the presentation timeline, Loading Manager 262 can intelligently request the segment for download, as well as additional media segments X, Y and Z that can be played following segment A, in advance of playback or notification of potential playback thereof. The downloading can occur even if fewer than all of X, Y, Z will be played (e.g., if X, Y and Z are potential segment choices branching off segment A and only one will be selected for playback). **[Column 11; Lines 14-27]** **Video Appender 270 receives media content from Loading Manager 262 and instructions from Choice Manager 216 on which media segments to include in a media presentation. Video Appender 270 can analyze and/or modify raw video or other media content, for example, to**

	concatenate two separate media streams into a single timeline. Video Appender 270 can also insert cue points and other event markers, such as junction events, into media streams. Further, **Video Appender 270 can form one or more streams of bytes from multiple video, audio or other media streams, and feed the formed streams to a video playback function such that there is seamless playback of the combined media content on display screen 272 (as well as through speakers for audio, for example).** **Note:** *The mapped citation describes a Video Appender that receives playback-related user interactions, identifies potential future video segments, retrieves corresponding media from local or remote storage, and determines content to download or buffer for playback. Based on user input and instructions from the Loading and Choice Managers, it concatenates multiple video, audio, or other media streams into a single timeline and byte stream for seamless playback, supporting sequencing and assembly of selected media segments. However, decryption and authorization of media sample blocks based on traversal authorization, volatile-memory processing, spatial-resolution controls, and playback without writing a rendered video file to persistent storage are not expressly disclosed*
11. A policy enforcement mechanism configured to apply inherited policies to recursively nested containers, wherein the most restrictive applicable policy governs traversal.	N/A
12. The traversal framework further enforces cycle detection, maximum traversal-depth restrictions, and real-time rights verification.	N/A
13. A rights management mechanism configured to dynamically compose video content from independently owned content libraries using container-specific authorization and maintain an audit record of traversal events.	**[Column 11; Lines 28-57]** In some implementations, **application 112 tracks information regarding user interactions, users, and/or player devices, and stores the information in a database. Collected analytics can include,** but are not limited to: (**1) device information,** such as number, type, brand, model, device location, operating system, installed software, browser, browser parameters, user agent string, screen size, bandwidth, and network connection parameters; (**2) user tracking and demographic data,** such as login information, name, address, age, sex, referrer, uniform resource locator (URL), urchin tracking module (UTM) parameters; **(3) user or automated action information**, such as button/touchpad presses, mouse clicks, mouse/touchpad movements, action timings, media player controls (play, pause, volume up/down, mute, full screen, minimize, seek forward, seek backward, etc.), link outs, shares, rates, comments; **(4) information associated with interactions with interactive media content, such as decisions made by users or made automatically (e.g., content segment user choices or default selections), starting a node, ending a node, and content paths followed in the presentation content structure; and (5) information associated with**

	media playback events and timing, such as loading events, buffering events, play and pause events. The analytics can include those described in U.S. Patent Application Pub. No. 2011/0202562, entitled "System and Method for Data Mining within Interactive Multimedia," and published Aug. 18, 2011, the entirety of which is incorporated by reference herein. In one instance, some or all of these analytics are included in or used to inform attributes in User Profile 210. **Note:** *The mapped citation describes storing user and player-device interaction data in a database, including node starts and ends, user or automatic decisions, content paths followed, and media playback events and timing. This supports maintaining audit records of traversal events, including node traversal, branch/path selection, and playback events. However, composing video content from independently owned content libraries and using container-specific authorization to access such content are not expressly disclosed.*
14. The rights management mechanism further generates financial settlement information for respective content owners based on traversed branches.	N/A

Patent Citation 5: US11956520B2

Title	Methods and systems for providing dynamically composed personalized media assets
Priority Date	02 MAR 2021
Filing Date	02 MAR 2021
Publication Date	09 APR 2024
Inventors	Christine Doig-Cardet, Bruce Wobbe, Sanford Holsapple, Xander Lott, Sonali Sharma, Clay Gimenez, Jeremy Kelly, Jeff Kirchner, Leena Janardanan
Assignees	Netflix, Inc.
IPC Classes	H04N21/234; H04N21/258; H04N21/262; H04N21/2668; H04N21/845; H04N21/8549
CPC Classes	H04N21/23424 ; H04N21/25891; H04N21/26258; H04N21/2668; H04N21/44222; H04N21/6582; H04N21/8456; H04N21/8549
US Classes	None
Family Members	**EP4302482A1| BR112023017747A2 | WO2022187269A1 | US11871095B2 | US12177542B2 | US20250080812A1**

Abstract:

The disclosed computer-implemented method may include accessing a media item that includes multiple media item segments that are to be played back in a specific manner. The method may also include generating playgraphs for the media item, where the playgraphs define different playback paths between media item segments. The method may next include selecting a specific playgraph from the generated playgraphs, and then providing the selected playgraph to a playback device. Playback of the media item according to the selected playgraph may thereby provide a customized presentation of the media item. Various other methods, systems, and computer-readable media are also disclosed.

Key Features	Relevant Excerpts
1. A system for traversal-governed video assembly.	**[Claim 18]** A computer-implemented method comprising: receiving an indication that a media item is to be played back in a specific manner, wherein the media item, including a plurality of corresponding media item segments, comprises a single data structure, **the plurality of media item segments being encoded in a specified order;** accessing at least one of a plurality of playgraphs for the media item, the playgraphs defining various different playback paths between the media item segments; and **playing back the media item according to the accessed playgraph to provides a customized presentation of the media item, wherein playback of the media item according to the accessed playgraph includes playback of at least one non-**

	sequential media item segment that is out of order relative to the order in which the plurality of media item segments were initially encoded. **[Column 8; Lines 4-21]** Turning now to FIG. 3, a media item 301 is presented that has multiple different virtual assets 302a and 302b. The virtual asset generating module 119 of FIG. 1 may be used to generate a virtual asset 125. As the term is used herein, **a "virtual asset" may refer to a data structure that includes a unique identifier 126 that ties the virtual asset 125 to the media item 122, as well as a specific selected playgraph 112 that provides a defined playback path 110 through the media item segments 123.** Thus, in FIG. 3, virtual asset 302a may include a unique identifier 303a that uniquely identifies that virtual asset 302a, as well as a selected playgraph 304a that defines a specific playback pathway 305a. Similarly, virtual asset 302b may include a unique identifier 303b that uniquely identifies virtual asset 302b, as well as a selected playgraph 304b that defines a specific playback pathway 305b. While the media item 301 is shown as having two virtual assets, it will be understood that each media item may have substantially any number of virtual assets. **[Column 7,8; lines 46-67,1-3]** LinesAt step 230, the playgraph selecting module 111 may select a specific playgraph 112 from the various generated playgraphs 109. Specific playgraphs may have specific goals. For example, the goal of a specific trailer may be to appeal to a given user. Based on information that is known 50 about that user (e.g., user 116), the playgraph selecting module 111 may select a playgraph that will provide a playback path 110 that chooses clips (and potentially the ordering of those clips) that will appeal to that user. The playgraph selecting module 111 may select other playgraphs 55 for other users to appeal to the personal preferences ofthose users. Moreover, instead of the data store 120 having to store multiple different versions of a trailer to appeal to the tastes of different users, the embodiments herein allow storage of a single media file that includes all of the clips that may be 60 used in a given trailer. Then, the computer system 101 determines which ordering of clips is to be used to create a customized trailer for each user. **Once the playgraph selecting module 111 has selected the playgraph 112 for that user, the provisioning module 113, at step 240 of method 200, 65 provides the selected playgraph 112 to the user's playback device 117. The user's device then plays back the media item 122 according to the selected playgraph, thereby providing a customized presentation of the media item that is tailored to the user 116.** **Note:** *The mapped citation describes playgraphs defining multiple playback and traversal paths through media segments, including nonlinear and out-of-order paths. A playgraph is selected for a particular playback session, and the playback device assembles and presents media segments according to the selected path, with a virtual asset packaging the selected playgraph.*
2. A virtualization engine configured to generate a virtual video reference container by removing the media payload (MDAT) while retaining structural and timing metadata	**[Claim18]** A computer-implemented method comprising: receiving an indication that a media item is to be played back in a specific manner, wherein **the media item, including a plurality of corresponding media item segments, comprises a single data structure, the plurality of media item segments being encoded in a specified order;**

(STBL).	accessing at least one of a plurality of playgraphs for the media item, **the playgraphs defining various different playback paths between the media item segments;** and playing back the media item according to the accessed playgraph to provides a customized presentation of the media item, wherein playback of the media item according to the accessed playgraph includes playback of at least one non-sequential media item segment that is out of order relative to the order in which the plurality of media item segments were initially encoded. **[Column 8; Lines 4-21]** Turning now to FIG. 3, a media item 301 is presented that has multiple different virtual assets 302a and 302b. The virtual asset generating module 119 of FIG. 1 may be used to generate a virtual asset 125. As the term is used herein, **a "virtual asset" may refer to a data structure that includes a unique identifier 126 that ties the virtual asset 125 to the media item 122, as well as a specific selected playgraph 112 that provides a defined playback path 110 through the media item segments 123.** Thus, in FIG. 3, virtual asset 302a may include a unique identifier 303a that uniquely identifies that virtual asset 302a, as well as a selected playgraph 304a that defines a specific playback pathway 305a. Similarly, virtual asset 302b may include a unique identifier 303b that uniquely identifies virtual asset 302b, as well as a selected playgraph 304b that defines a specific playback pathway 305b. While the media item 301 is shown as having two virtual assets, it will be understood that each media item may have substantially any number of virtual assets. **Note:** *The mapped citation describes a media item comprising multiple media segments within a single data structure, with the segments encoded in a specified order and associated with playgraphs defining playback paths.*
3. The retained metadata comprises rewritten sample table references identifying media sample blocks stored externally from the virtual video reference container.	N/A
4. A graph composition mechanism configured to interconnect multiple virtual video reference containers through traversal references to form a traversable video graph, including recursive or nested traversal.	**[Claim18]** A computer-implemented method comprising: receiving an indication that a media item is to be played back in a specific manner, wherein the media item, including a plurality of corresponding media item segments, comprises a single data structure, the plurality of media item segments being encoded in a specified order; accessing at least one of a plurality of playgraphs for the media item, the playgraphs defining various different playback paths between the media item segments; and **playing back the media item according to the accessed playgraph to provides** a customized presentation of the media item, **wherein playback of the media item according to the accessed playgraph includes playback of at least one non-sequential media item segment that is out of order relative to the order in**

	which the plurality of media item segments were initially encoded. **[Column 11; Lines 51-60]** In some cases, a virtual asset may be generated based on multiple different media items. **While the embodiments described above generally refer to scenarios where a playgraph defines a playback path through a single mega asset, it will be recognized that a playgraph may specify a playback path through a plurality of different media items and/or media clips.** Thus, for example, the computer system 101 of FIG. 1 may be configured to access alternative media items in addition to or in place of media item 122... **Note:** *The mapped citation describes multiple playgraphs defining different playback paths between media segments, which can reasonably be understood as traversal relationships forming a traversable graph structure. Non-sequential playback further supports traversal through paths different from the original encoding sequence, with playgraph paths functioning as traversal references between media components.*
5. A runtime traversal engine configured to evaluate dynamic branch conditions based on real-time signals and select a target video container from multiple candidate containers, wherein the signals include viewer input, AI agent directives, rights policies, or environmental context.	**[Calim 18]** A computer-implemented method comprising: **receiving an indication that a media item is to be played back in a specific manner,** wherein the media item, including a plurality of corresponding media item segments, comprises a single data structure, the plurality of media item segments being encoded in a specified order; **accessing at least one of a plurality of playgraphs for the media item, the playgraphs defining various different playback paths between the media item segments;** and **playing back the media item according to the accessed playgraph to provides a customized presentation of the media item,** wherein playback of the media item according to the accessed **playgraph includes playback of at least one non-sequential media item segment that is out of order relative to the order** in which the plurality of media item segments were initially encoded. **[Column 6; lines 4-24]** Once a specific media item 122 has been accessed, the playgraph generating module 108 may generate one or more playgraphs 109 for the media item. The playgraphs 109 specify one or more different playback paths through the media item segments 123. For example, **in contrast to traditional playback systems that play back a media item sequentially from a starting position to an ending position,** the embodiments described herein may be configured to begin playback at substantially any position within a media item 122 or may begin playback with substantially any media item segment 123. **Then, from that specified starting point, the playgraph may dictate a flow through the media item 122 or the media item segments 123 that may be nonlinear, flowing between different out-of-order media item segments as dictated by the playgraph and/or as dictated by inputs 118 from a viewing user (e.g., 116).** The playgraph may specify that, for instance, playback is to start with media item segment B, then go to media item segment D, then play media item segment A, omitting media item segment entirely, in a case where media item segments A, B, C, and D are all encoded into a single data structure. **[Column 6; lines 39-60] The playgraph selecting module 111 may then select,**

	from among potentially many different playgraphs 109 that were generated for the media item 122, a selected playgraph 112 that is to be used for this specific playback (e.g., for this specific user 116 or playback device 117). The selected playgraph 112 may be personalized for the user 116, providing the user with a trailer that is played back in a customized manner to provide a specific feel or a specific appeal to that user. For instance, the trailer may highlight movie clips that showcase an actor or actress that the user 116 has shown an interest in, or may highlight certain segments that show the genre or some of the comedic moments of the movie. Other personalizations may also be made such that the trailer playback experience is truly customized to the user 116… **Note:** *The mapped citation describes a runtime traversal mechanism that receives an indication of how a media item is to be played and selects a corresponding playgraph from multiple playback paths. This indication can reasonably be interpreted as viewer input or a playback instruction influencing playgraph selection, which then governs traversal through media segments, including non-sequential playback.*
6. A token validation module configuredto validate an independent cryptographic video token associated with the selected target video container and grant traversal authority upon successful validation.	N/A
7. The cryptographic video token defines or enforces licensing, usage, consent, access, or financial settlement conditions, and successful validation releases an ephemeral decryption key or signed policy authorization.	N/A
8. A spatial resolution module configured to determine spatial resolution gates based on viewer spatial context and provide full fidelity for a foveal region, reduced fidelity for a peripheral region, and withhold media for an out-of-view region.	**[Column 18; Lines 12-22] In situations where the bandwidth of distribution infrastructure 1110 is limited and/or variable, playback application 1310 downloads and buffers consecutive portions of video data and/or audio data from video encodings with different bit rates based on a variety of factors (e.g., scene complexity, audio complexity, network bandwidth, device capabilities, etc.).** In some embodiments, **video playback quality is prioritized over audio playback quality. Audio playback and video playback quality are also balanced with each other, and in some embodiments audio playback quality is prioritized over video playback quality.** **Note:** *The mapped citation describes adapting playback quality based on factors such as scene complexity, network bandwidth, and device capabilities. However, provide full fidelity for a foveal region, reduced fidelity for a peripheral region, and withhold media for an out-of-view region is not disclosed.*

9. A prefetch mechanism configured to retrieve candidatemedia for possible traversal branches into a decoupled non-executable buffer before branch selection, while withholding the corresponding key or authorization until token validation.	**[Column 18; Lines 12-22]** In situations where the bandwidth of distribution infrastructure 1110 is limited and/or variable, **playback application 1310 downloads and buffers consecutive portions of video data and/or audio data from video encodings with different bit rates based on a variety of factors (e.g., scene complexity, audio complexity, network bandwidth, device capabilities, etc.).** In some embodiments, video playback quality is prioritized over audio playback quality. Audio playback and video playback quality are also balanced with each other, and in some embodiments audio playback quality is prioritized over video playback quality **[Column 17; Lines 44-61]** In one embodiment, playback application 1310 begins downloading the content associated with the requested title by downloading sequence data encoded to the lowest audio and/or video playback bit rates to minimize startup time for playback. **The requested digital content file is then downloaded into content buffer 1312, which is configured to serve as a first-in, first-out queue.** In one embodiment, each unit of downloaded data includes a unit of video data or a unit of audio data. As units of video data associated with the requested digital content file are downloaded to the content player 1120, the units of video data are pushed into the content buffer 1312. Similarly, as units of audio data associated with the requested digital content file are downloaded to the content player 1120, **the units of audio data are pushed into the content buffer 1312.** In one embodiment, the units of video data are stored in video buffer 1316 within content buffer 1312 and the units of audio data are stored in audio buffer 1314 of content buffer 1312. **Note:** *The mapped citation describes a playback application that downloads and buffers video and audio data in advance in separate buffers. However, retrieving candidate media for potential traversal branches before branch selection, using a decoupled non-executable buffer, and withholding a key or authorization until token validation are not expressly disclosed.*
10. A playback assembly mechanism configured to retrieve, decrypt, authorize, and sequence media sample blocks in volatile memory according to traversal authorization and spatial resolution, thereby generating continuous playback without writing a rendered video file to persistent storage.	**[Column 17; Lines 28-43] Playback application 1310 is configured to retrieve digital content via communication interface 1322 and to play the digital content through graphics interface 1326.** Graphics interface 1326 is configured to transmit a rendered video signal to graphics device 1328. In normal operation, playback application 310 receives a request from a user to play a specific title or specific content. Playback application 310 then identifies one or more encoded video and audio streams associated with the requested title. After playback application 1310 has located the encoded streams associated with the requested title, playback application 1310 downloads sequence header indices associated with each encoded stream associated with the requested title from distribution infrastructure 1110. A sequence header index associated with encoded content includes information related to the encoded sequence of data included in the encoded content. **[Column 17; Lines 44-61]** In one embodiment, **playback application 1310 begins downloading the content associated with the requested title by downloading sequence data encoded to the lowest audio and/or video playback bit rates to minimize startup time for playback.** The requested digital content file is then

	downloaded into content buffer 1312, which is configured to serve as a first-in, first-out queue. In one embodiment, each unit of downloaded data includes a unit of video data or a unit of audio data. As units of video data associated with the requested digital content file are downloaded to the content player 1120, the units of video data are pushed into the content buffer 1312. Similarly, as units of audio data associated with the requested digital content file are downloaded to the content player 1120, the units of audio data are pushed into the content buffer 1312. In one embodiment, the units of video data are stored in video buffer 1316 within content buffer 1312 and the units of audio data are stored in audio buffer 1314 of content buffer 1312. **[Column 18; Lines 12-22]** In situations where the bandwidth of distribution infrastructure 1110 is limited and/or variable, **playback application 1310 downloads and buffers consecutive portions of video data and/or audio data from video encodings with different bit rates based on a variety of factors (e.g., scene complexity, audio complexity, network bandwidth, device capabilities, etc.).** In some embodiments, video playback quality is prioritized over audio playback quality. Audio playback and video playback quality are also balanced with each other, and in some embodiments audio playback quality is prioritized over video playback quality **[Column 17; Lines 8-18]** As noted, **memory 1124 generally represents any type or form of volatile or non-volatile storage device or medium capable of storing data and/or other computer-readable instructions.** In some examples, memory 1124 stores and/or loads an operating system 1308 for execution by processor 1122. In one example, operating system 1308 includes and/or represents software that manages computer hardware and software resources and/or provides common services to computer programs and/or applications on content player 1120. **Note:** *The mapped citation describes retrieving and sequencing media data in volatile memory, because the playback application downloads video/audio units into the content buffer and separate video/audio buffers for playback, while memory stores the operating system and application instructions. However, it does not expressly disclose decrypting or authorizing media sample blocks according to traversal authorization or spatial resolution and generating continuous playback without writing a rendered video file to persistent storage.*
11. A policy enforcement mechanism configured to apply inherited policies to recursively nested containers, wherein the most restrictive applicable policy governs traversal.	N/A
12. The traversal framework further enforces cycle detection, maximum traversal-depth restrictions, and real-time	N/A

rights verification.	
13. A rights management mechanism configured to dynamically compose video content from independently owned content libraries using container-specific authorization and maintain an audit record of traversal events.	N/A
14. The rights management mechanism further generates financial settlement information for respective content owners based on traversed branches.	N/A

Patent Citation 6: US12277238B2

Title	System and method for controlling data using containers
Priority Date	17 DEC 2020
Filing Date	20 JUN 2022
Publication Date	15 APR 2025
Inventors	Alan Rodriguez
Assignees	Alan Rodriguez
IPC Classes	G06F21/62, G06F21/78, H04L9/30
CPC Classes	G06F21/62, G06F21/6245, G06F21/78, H04L9/008, H04L9/30, H04L9/50, G06F2221/2101, H04L2209/16, H04L2209/30
US Classes	None
Family Members	**US11372987B1 | EP4264445A1 | WO2022133160A1**

Abstract:

An electronic device for managing secured data containers, the electronic device comprising at least one network interface, at least one memory storing executable instructions, and at least one processor coupled to the at least one network interface and the at least one memory. Execution of the executable instructions by the at least one processor causes the electronic device to receive a request for data container creation, retrieve data related to the request for data container creation, retrieve one or more parameters constraining use of the data, encrypt the data using a public encryption key, encode the encrypted data into a data storage area of a data container, encode the one or more parameters constraining use of the data into a machine readable parameter storage area of the data container, and assign a UUID to the data container.

Key Features	Relevant Excerpts
1. A system for traversal-governed video assembly.	**[Column 49; Lines 32-46]** **Containers owned by different users and organizations can also be configured as components of an application, such as using a SQLITE data file format for a containers where the SQLITE data file is configured to be used as an application file format. Various programmable and executable application components can be implemented in containers in this way, and such containers can be influenced by or influence other containers.** For example, one application file format container can include graphical user interface (GUI) elements for presenting data, requests, options, or other content to other users or organizations, and these containers can be influenced by changes to other containers, either temporarily or permanently. As shown in FIG. 19 , container 1240 owned by Organization 3 and container 1233 owned by User 1 influence each other. **[Column 53; Lines 1-19]** FIG. 21 illustrates an example composable application diagram 2100 in accordance with various embodiments of this disclosure. As

	described hereinabove, **data containers of the present application can be used as building blocks for applications, enabling users or organizations to create intelligent composable applications that update or respond to inputs and outputs to facilitate interactions with other users and organizations.** Composable applications can allow for rapidly redesigned digital business solutions, new business models, automation and new products, services and channels via modular data and application architectures. As described herein, relationships between containers can be established such as by linking containers by graph database reference or by nesting a container within another container. These relationships can be temporary to facilitate the exchange of information or the updating of application states, or can be permanent, such as permanently storing a new data item in a container after it interacts with an application data container. **[Column 52; Lines 1-10]** In some embodiments, **containers can include one or more archive file container formats such as ZIP format, RAR format, etc., or multimedia playback container formats such as MASTROSKA, MP4, AVI, etc.** Other containers types can also be implemented. For example, containers that interleave disparate file or data types into one container for enhanced security and convenience can be used, such as CHM or CHIF files as disclosed in U.S. Pat. No. 10,187,443 and U.S. Patent Application Publication No. 2020/0351323, which are each incorporated by reference herein. **Note:** *The mapped citation describes about containers that can contain multimedia including MP4/AVI/Matroska, composable applications, relationships between containers, and smart-contract-controlled access. However, information about system for traversal-governed video assembly is not disclosed.*
2. A virtualization engine configured to generate a virtual video reference container by removing the media payload (MDAT) while retaining structural and timing metadata (STBL).	**[Column 50; Lines 46-57]** FIG. 20 illustrates an example container nesting process 2000 in accordance with various embodiments of this disclosure. As described herein, containers can be related to one another in various ways, such as by establishing relationship between two containers in a database. Another method of creating either temporary or permanent relationships between containers is to nest a container within another container. Containers as disclosed herein can be in different formats. **A container belongs to a class of objects that can allow for multiple data streams to be encoded into a single file, typically with metadata identifying and further defining the data streams.** **[Column 52; Lines 1-10]** In some embodiments, **containers can include one or more archive file container formats such as ZIP format, RAR format, etc., or multimedia playback container formats such as MASTROSKA, MP4, AVI, etc.** Other containers types can also be implemented. For example, containers that interleave disparate file or data types into one container for enhanced security and convenience can be used, such as CHM or CHIF files as disclosed in U.S. Pat. No. 10,187,443 and U.S. Patent Application Publication No. 2020/0351323, which are each incorporated by reference herein. **Note:** *The mapped citation describes about multimedia containers and metadata, including MP4.*

3. <u>The retained metadata comprises rewritten sample table references identifying media sample blocks stored externally from the virtual video reference container.</u>	N/A
4. <u>A graph composition mechanism</u> configured <u>to interconnect multiple virtual video reference containers through traversal references to form a traversable video graph</u>, <u>including recursive or nested traversal.</u>	**[Column 50; Lines 26-45]** As also shown in FIG. 19 , **<u>the graph database can also include relationships between containers and their copies.</u>** For example, User 2 is shown as owning two containers, container 1236 and container 1236 . 1 , container 1236 . 1 being a copy of container 1236 . In some embodiments, container copies can be designated as a copy of another container in the database, and also be indicated as a copy by using a floating point number for the UUID, where the UUID is the same as the original container except for including incremental decimal point values in the UUID. It will be understood that other ways of indicating a container is a copy can also be used. For example, to save memory, instead of using a floating point value, other formats can be used such as using a combination of numbers and letters, 1236 A, for example, such as by using ASCII or string data structures for the UUID. As shown in FIG. 19 , container 1236 is accessible by Organization 2 , while the copy is accessible to User 1 . As described in the various embodiments of this disclosure, containers and their copies, including their storage locations and lineage, can also be tracked by the container registry. **[Column 50; Lines 46-57]** FIG. 20 illustrates an example container nesting process 2000 in accordance with various embodiments of this disclosure. As described herein, **<u>containers can be related to one another in various ways, such as by establishing relationship between two containers in a database. Another method of creating either temporary or permanent relationships between containers is to nest a container within another container</u>**. Containers as disclosed herein can be in different formats. A container belongs to a class of objects that can allow for multiple data streams to be encoded into a single file, typically with metadata identifying and further defining the data streams. **[Column 52; Lines 38-47]** As illustrated in FIG. 20 , the processor executing the master data controller 710 retrieves two containers from the container storage 708 . In some embodiments, the containers can be transmitted to the master data controller 710 from one or more users or organizations. In this example, **<u>the two data containers to be nested include the SQLITE data container 2002 and another data container 2016 of another type, such as those described above.</u>** It will be understood that **<u>data containers of the same or differing types can be nested, and more than two containers can all be nested within each other.</u>** **Note:** *The mapped citation describes a graph database containing relationships between containers, and containers can be nested, including more than two containers nested within each other.*
5. <u>A runtime traversal engine</u> configured <u>to</u>	**[Column 53; Lines 32-45]** The model 2102 is programmed to provide a view 2106 to a user and update that view based on inputs from the user or other sources.

evaluate dynamic branch conditions based on real-time signals and select a target video container from multiple candidate containers, wherein the signals include viewer input, AI agent directives, rights policies, or environmental context.	For example, the view 2106 can present an online shopping GUI that includes a button to add an item to the user's cart. Upon interacting with the "add to cart" button, the view 2106 sends the user input to a controller 2108 . **The controller 2108 can be the control logic for the application that handles events for the application, such as user inputs.** In some cases, the controller 2108 can update the view 2106 directly. In other cases, the controller 2108 manipulates or updates the model 2102 to update the data stored therein to reflect the added cart item. The model 2102 then updates the view 2106 to show the item in the cart. **[Column 29; Lines 10-28]** At decision block 810 , the processor determines if the electronic device provides location parameters for the data to be controlled by the smart data contract. **Location parameters can include parameters related to where data can be collected.** For example, for the traveler subscription service, location parameters could include parameters designating that the subscription service only collect information on the traveler at when the traveler is in certain locations, such as only collecting data on the traveler when the traveler is in an airport, such as defined by global positioning system (GPS) information received from the traveler's electronic device. If, at decision block 810 , the processor determines that the electronic device does not provide location parameters, the process 800 moves to decision block 814 . If, at decision block 810 , the processor determines that the electronic device provides location parameters for the data to be controlled by the smart data contract, at block 812 the processor receives and sets the location parameters and the process 800 moves to decision block 814 . **[Column 32; Lines 41-65]** It will be understood that the blocks 808 , 812 , 816 , 820 , 824 , 828 , 832 , 836 , 840 , and 844 can be performed together after a data owner provides answers for each of the different parameter options. In various embodiments, the different parameters can be updated after initial configuration by updating the smart data contract and any related containers with the new parameters. The smart data contract can also designate the entity requesting permission to use the data, only allowing the particular entity to use the data in the container and protected by the smart data contract, unless specific proxy settings. In various embodiments, the parameters can be set in the smart data contract to be in human readable simple text and legal code or text. **The parameters can also be set in the smart data contract as machine readable parameters that are used to determine when and how the data can be shared or accessed. For example, the parameters can each be set as a binary or Boolean variable that are processed along with other data such as GPS data, timestamps, or other data to determine if the variable evaluates to true or false.** For example, if a temporal parameter compared to a current timestamp evaluates to true, the data may be shared. It will also be understood that the traveler subscription service described herein is but one example and the various smart data contract parameters can be used with various other use cases. **Note:** *The mapped citation describes the reference allows application behavior to respond to user inputs, and smart-contract parameters can be evaluated using information such as GPS data and timestamps. However, it does not specifically disclose selecting a video branch/container based on viewer input, AI directives,*

	rights policies and environmental context.
6. A token validation module configured to validate an independent cryptographic video token associated with the selected target video container and grant traversal authority upon successful validation.	**[Column 27; Lines 18-41]** In some embodiments, data containers 716 can also be used to enforce non-fungible token (NFT) usage and transactions. An NFT can be considered a unique digital proof of ownership or access to an associated digital object or asset, such as text data, images, drawings, music, or other data. For example, **a user or organization creating digital artwork can encrypt and encode the digital artwork into a data container 716 , preventing unauthorized users or organizations from accessing and copying or distributing the artwork. Users or organizations that store digital assets in a data container 716 can use the constraining parameters of the data container 716 to provide the digital asset for sale, and allow only authorized purchasers to access the digital asset using an NFT. If any unauthorized user somehow has the data container 716 in their possession, the system and methods of the various embodiments herein would prevent that unauthorized user from breaking into the encrypted data of the container and accessing the digital assets.** In some embodiments, NFTs are used as part of a blockchain architecture that holds a reference to the NFT in the blockchain. Digital assets can be stored and constrained by the data containers 716 within that architecture and the use thereof controlled as provided by the various embodiments of this disclosure. **[Column 66; Lines 8-25]** The process 2900 beings at block 2902 , in which the processor encodes media into a data container and assigns a UUID to the new data container. The media can include text, audio, video, one or more images, a GIF, or any other type of data. At block 2904 , the processor provides and/or actively records in a block on the blockchain an identification of the data container that includes the UUID and the owner ID, such as described with respect to FIG. 28 , minting an NFT in some embodiments for the media stored in the data container. **The blockchain holds a reference to the NFT, e.g., the UUID paired with the owner ID, with the blockchain recording any subsequent licensing, sale, or transfer of the data container**. In some embodiments, the UUID paired with the owner ID represents the NFT. In some embodiments, a hash of these two values, the UUID and the owner ID, is the NFT. In some embodiments, a hash of the UUID is the NFT, with the owner ID/public key being used for verification and security using a private-public key scheme. **[Column 66; Lines 26-48]** At block 2806 , the processor provides or receives an offer to transfer or access the data container. For example, an owner of the media in the data container can offer the media for license, transfer, or sale to a third party, or an offer can be sent from a third party to the owner. **The offer and subsequent steps in the transaction between the owner and third party can be secured using the public key associated with the data container generated from the owner's private key.** Upon providing or receiving an offer, at decision block 2908 , the processor determines if the offer is compliant with the smart contract governing the data container, such as sending a notification of the offer to the edge data controller and/or the master data controller, as described in the various embodiments of this disclosure. The processor can then be notified of compliance or noncompliance. If the offer is not compliant, the process 2900 moves to block 2914 and the failed offer is recorded in a block of the blockchain.

	In some embodiments, failed offers may not be recorded, but only successful offers are recorded on the blockchain. If, at decision block 2908 , the processor determines that the offer is compliant with the smart contract, the process 2900 moves to block 2910 . **Note:** *The mapped citation describes about cryptographic authorization; public/private keys, NFTs, UUIDs, blockchain records and authorization to access protected digital assets.*
7. <u>The cryptographic video token defines or enforces licensing, usage, consent, access, or financial settlement conditions, and successful validation releases an ephemeral decryption key or signed policy authorization.</u>	**[Column 66; Lines 26-48]** At block 2806 , the processor provides or receives an offer to transfer or access the data container. For example, an owner of the media in the data container can offer the media for license, transfer, or sale to a third party, or an offer can be sent from a third party to the owner. The offer and subsequent steps in the transaction between the owner and third party can be secured using the public key associated with the data container generated from the owner's private key. Upon providing or receiving an offer, at decision block 2908 , **<u>the processor determines if the offer is compliant with the smart contract governing the data container</u>,** such as sending a notification of the offer to the edge data controller and/or the master data controller, as described in the various embodiments of this disclosure. The processor can then be notified of compliance or noncompliance. If the offer is not compliant, the process 2900 moves to block 2914 and the failed offer is recorded in a block of the blockchain. In some embodiments, failed offers may not be recorded, but only successful offers are recorded on the blockchain. If, at decision block 2908 , the processor determines that the offer is compliant with the smart contract, the process 2900 moves to block 2910 . **[Column 66-67; Lines 49-67 1-9]** At block 2910 , the processor can receive a transfer of consideration for the data container from the third party. For example, in embodiments in which media stored in the data container is intended to be sold or licensed to a third party, **<u>the owner of the media stored in the data container can request payment as a condition of the offer.</u>** The payment can be handled off the blockchain in some embodiments. In embodiments in which payment is provided via cryptocurrency, the payment can be handled on the blockchain, or on a separate blockchain. At block 2912 , the processor transfers or provides access to the data container, such as described in the various embodiments of this disclosure, **<u>including providing information to decrypt the data container to the third party.</u>** At block 2914 , the successful offer and license, transfer, or sale of the data container is recorded in a block on the blockchain. At block 2916 , the processor and/or the edge data controller/master data controller continue to monitor compliance with the smart contract. For example, the data container as described in the various embodiments of this disclosure is subject to a smart contract that defines the accepted use of the data container. As part of the process 2900 , **<u>the third party may only be given a license or other conditional use of the data container and its media as governed by the smart contract.</u>** If, for example, the third party attempts to share the data container with another unauthorized party, such an action is detected and remedial actions can be taken, such as revoking access to the data container, as described in the various embodiments of this disclosure. The process 2900 ends at block 2918 .

	Note: *The mapped citation describes about NFTs and access authorization, smart contracts controlling use/access, licensing/sale conditions, payment, and provision of information to decrypt the data container after the compliant transaction.*
8. A spatial resolution module configured to determine spatial resolution gates based on viewer spatial context and provide full fidelity for a foveal region, reduced fidelity for a peripheral region, and withhold media for an out-of-view region.	N/A
9. A prefetch mechanism configured to retrieve candidate media for possible traversal branches into a decoupled non-executable buffer before branch selection, while withholding the corresponding key or authorization until token validation.	**[Column 49; Lines 32-46]** In some embodiments, the edge data controller 718 is situated at an edge of a network to shift data processing as close to electronic devices 715 as possible to enhance performance, security and privacy. This enables data intensive workloads at the network edge closest to end users where security and privacy are best enforced and protected. In some embodiments, the edge data controller 718 is situated at an edge node and connects to the electronic devices 715 to manage the creation of or access to data containers for the electronic devices 715 using the established access parameters or smart contracts. The electronic devices 715 can in some embodiments be referred to as edge devices that can desktop or laptop computers, servers, mobile devices, Internet of Things (IoT) devices, sensors, or other devices. In some embodiments, **the edge data controller 718 has access to a container cache 712 that stores copies of containers from container storage 708 for more efficient retrieval of containers for use by or viewing of the containers by the electronic devices 715 .** The container storage 708 and/or the container cache 712 can secure thousands of different tables, with thousands of fields, each with different datatypes and constraints, all automatically indexed for rapid retrieval, and all stored efficiently and compactly in a single disk file, such as a SQLITE file. In some embodiments, providing the edge data controller 718 at the network edge also allows for the interoperability between smart data contracts with blockchain smart contracts enabling hybrid solutions. Blockchains provide global marketplace scale data transactions while data containers provide private and secure multiparty data transactions, allowing entities to build decentralized applications (Dapps) with hybrid capabilities. **[Column 26-27; Lines 52-67 1-16]** In some embodiments, the edge data controller 718 notifies the master data controller 710 of access attempts or other events related to data containers 716 , and the master data controller 710 updates a log accordingly. In some embodiments, the edge data controller 718 can keep a log of access attempts or other events related to data containers 716 . In some embodiments, the edge data controller 718 periodically provides its container usage log to the master data controller 710 to keep a log of the master data controller 710 synced with the log of the edge data controller 718 . When a

	container is requested to be accessed by an electronic device 715 , one or more processors executing the edge data controller retrieve the container from the container cache 712 , if present, for use by the electronic device 715 . In some embodiments, **the edge data controller 718 transmits the container to the requesting electronic device 715 along with decryption information. In some embodiments, the edge data controller 718 transmits the container to the requesting electronic device 715 and the decryption key is separately provided by the owner of the data container.** In some embodiments, the edge data controller 718 is provided the decryption key either at an earlier time or upon request of the data container, decrypts the data in the data container and/or the data container, and then presents the contents of the data container to the requesting electronic device 715 , such as in container and developer defined user interface or presentation such as with respect to application 614 , without providing the container itself to the requesting electronic device 715 . **[Column 52; Lines 48-67]** In this example, the processor executing the master data controller 710 nests the containers 2002 and 2016 by storing the SQLITE data container 2002 within the data container 2016 subsequent to all the data in data container 2016 , creating a nested data container 2017 . This may be performed by the processor executing the master data controller 710 decrypting the containers 2002 and 2016 , encoding the data from one into the other, and then re-encrypting the nested data container 2017 . In some embodiments, the data from the data container 2002 can precede the data from data container 2016 . In some embodiments, the data from containers 2002 and 2016 can be intermingled or interleaved into the nested data container 2017 , and then reversing the process when decoding the nested data container 2017 . Once the nested data container 2017 is created, the processor executing the master data controller 710 can store the newly created nested data container 2017 in the container storage 708 . **The processor executing the master data controller 710 can also transmit the nested data container 2017 to the edge data controller 718 for storage in the container cache 712 .** **Note:** *The mapped citation describes about container caching at the edge and separately provided decryption information/keys. This provides functional support for pre-retrieval/caching and separately controlled decryption. However, it does not specifically disclose prefetching candidate traversal branches before branch selection into a non-executable buffer while withholding authorization until token validation.*
10. A playback assembly mechanism configured to retrieve, decrypt, authorize, and sequence media sample blocks in volatile memory according to traversal authorization and spatial resolution, thereby generating continuous playback	**[Column 27; Lines 42s-66]** In some embodiments, the edge data controller 718 notifies the master data controller 710 of access attempts or other events related to data containers 716 , and the master data controller 710 updates a log accordingly. In some embodiments, the edge data controller 718 can keep a log of access attempts or other events related to data containers 716 . In some embodiments, the edge data controller 718 periodically provides its container usage log to the master data controller 710 to keep a log of the master data controller 710 synced with the log of the edge data controller 718 . When a container is requested to be accessed by an electronic device 715 , one or more processors executing the edge data controller retrieve the container from the

without writing a rendered video file to persistent storage.	container cache 712 , if present, for use by the electronic device 715 . In some embodiments, the edge data controller 718 transmits the container to the requesting electronic device 715 along with decryption information. In some embodiments, the edge data controller 718 transmits the container to the requesting electronic device 715 and the decryption key is separately provided by the owner of the data container. In some embodiments, **the edge data controller 718 is provided the decryption key either at an earlier time or upon request of the data container, decrypts the data in the data container and/or the data container, and then presents the contents of the data container to the requesting electronic device 715..**. **[Column 53; Lines 46-54]** It will be understood that the updating of the model 2102 can be temporary, such as if an instance of the data container is executed and is updated multiple times as shown in FIG. 21 until a user ceases using the application, at which point the application is terminated and the original data container remains unchanged. In some embodiments, **the instance can be a temporary copy of the data container. In some embodiments, multiple data containers can be used to execute the composable application.** **Note:** *The mapped citation describes about multimedia containers, retrieval, decryption, temporary instances/copies, and presentation of container contents without necessarily providing the container itself. However, it does not disclose sample-block sequencing in volatile memory to produce continuous traversal playback without writing a rendered video file*
11. A policy enforcement mechanism configured to apply inherited policies to recursively nested containers, wherein the most restrictive applicable policy governs traversal.	**[Column 53; Lines 20-32]** The example illustrated in FIG. 21 shows a customized model-view-controller (MVC) object model utilizing containers and smart contracts that can be used as a model for creating GUIs, web applications, or other programs. As illustrated in FIG. 21 , a model 2102 can be a defined model or data structure, such as a container of the various types and formats disclosed herein. **Usage, execution, or access to the model 2102 is controller by a smart contract 2104 as described in the various embodiments herein.** For example, **the model 2102 as constrained by the smart contract 2104 may only present the application to authorized individuals and based on various constraints** such as those described with respect to FIG. 8 . **[Column 66; Lines 48-67]** At block 2910 , the processor can receive a transfer of consideration for the data container from the third party. For example, in embodiments in which media stored in the data container is intended to be sold or licensed to a third party, the owner of the media stored in the data container can request payment as a condition of the offer. The payment can be handled off the blockchain in some embodiments. In embodiments in which payment is provided via cryptocurrency, the payment can be handled on the blockchain, or on a separate blockchain. At block 2912 , the processor transfers or provides access to the data container, such as described in the various embodiments of this disclosure, including providing information to decrypt the data container to the third party. At block 2914 , the successful offer and license, transfer, or sale of the data container is recorded in a block on the blockchain. At block 2916 , the processor and/or the edge data controller/master data controller continue to monitor compliance with the smart contract. For example, **the data container as**

	described in the various embodiments of this disclosure is subject to a smart contract that defines the accepted use of the data container… **[Column 53; Lines 1-19]** FIG. 21 illustrates an example composable application diagram 2100 in accordance with various embodiments of this disclosure. As described hereinabove, data containers of the present application can be used as building blocks for applications, enabling users or organizations to create intelligent composable applications that update or respond to inputs and outputs to facilitate interactions with other users and organizations. Composable applications can allow for rapidly redesigned digital business solutions, new business models, automation and new products, services and channels via modular data and application architectures. As described herein, **relationships between containers can be established such as by linking containers by graph database reference or by nesting a container within another container.** These relationships can be temporary to facilitate the exchange of information or the updating of application states, or can be permanent, such as permanently storing a new data item in a container after it interacts with an application data container. **Note:** *The mapped citation describes Smart contracts govern access/use of containers, and containers may be recursively nested. It also describes accumulated contract rights and limitations. However, it does not expressly disclose that inherited policies apply recursively and the most restrictive applicable policy always governs.*
12. The traversal framework further enforces cycle detection, maximum traversal-depth restrictions, and real-time rights verification.	**[Column 50; Lines 26-45]** As also shown in FIG. 19 , the graph database can also include relationships between containers and their copies. For example, User 2 is shown as owning two containers, container 1236 and container 1236 . 1 , container 1236 . 1 being a copy of container 1236 . In some embodiments, container copies can be designated as a copy of another container in the database, and also be indicated as a copy by using a floating point number for the UUID, where the UUID is the same as the original container except for including incremental decimal point values in the UUID. It will be understood that other ways of indicating a container is a copy can also be used. For example, to save memory, instead of using a floating point value, other formats can be used such as using a combination of numbers and letters, 1236 A, for example, such as by using ASCII or string data structures for the UUID. As shown in FIG. 19 , container 1236 is accessible by Organization 2 , while the copy is accessible to User 1 . As described in the various embodiments of this disclosure, **containers and their copies, including their storage locations and lineage, can also be tracked by the container registry.** **[Column 66-67; Lines 48-67 1-9]** At block 2910 , the processor can receive a transfer of consideration for the data container from the third party. For example, in embodiments in which media stored in the data container is intended to be sold or licensed to a third party, the owner of the media stored in the data container can request payment as a condition of the offer. The payment can be handled off the blockchain in some embodiments. In embodiments in which payment is provided via cryptocurrency, the payment can be handled on the blockchain, or on a separate blockchain. At block 2912 , the processor transfers or

	provides access to the data container, such as described in the various embodiments of this disclosure, including providing information to decrypt the data container to the third party. At block 2914 , the successful offer and license, transfer, or sale of the data container is recorded in a block on the blockchain. At block 2916 , **the processor and/or the edge data controller/master data controller continue to monitor compliance with the smart contract.** For example, the data container as described in the various embodiments of this disclosure is subject to a smart contract that defines the accepted use of the data container. As part of the process 2900 , the third party may only be given a license or other conditional use of the data container and its media as governed by the smart contract. If, for example, the third party attempts to share the data container with another unauthorized party, **such an action is detected and remedial actions can be taken, such as revoking access to the data container,** as described in the various embodiments of this disclosure. The process 2900 ends at block 2918 . **Note:** *The mapped citation describes about nested containers, container lineage, continued monitoring of smart-contract compliance and revocation of access. However, cycle detection and maximum traversal-depth restrictions are not disclosed.*
13. A rights management mechanism configured to dynamically compose video content from independently owned content libraries using container-specific authorization and maintain an audit record of traversal events.	**[Column 49; Lines 32-46] Containers owned by different users and organizations can also be configured as components of an application, such as using a SQLITE data file format for a containers where the SQLITE data file is configured to be used as an application file format.** Various programmable and executable application components can be implemented in containers in this way, and such containers can be influenced by or influence other containers. For example, one application file format container can include graphical user interface (GUI) elements for presenting data, requests, options, or other content to other users or organizations, and these containers can be influenced by changes to other containers, either temporarily or permanently. As shown in FIG. 19 , container 1240 owned by Organization 3 and container 1233 owned by User 1 influence each other. **[Column 53; Lines 1-19]** FIG. 21 illustrates an example composable application diagram 2100 in accordance with various embodiments of this disclosure. As described hereinabove, **data containers of the present application can be used as building blocks for applications,** enabling users or organizations to create intelligent composable applications that update or respond to inputs and outputs to facilitate interactions with other users and organizations. Composable applications can allow for rapidly redesigned digital business solutions, new business models, automation and new products, services and channels via modular data and application architectures. As described herein, **relationships between containers can be established such as by linking containers by graph database reference or by nesting a container within another container.** These relationships can be temporary to facilitate the exchange of information or the updating of application states, or can be permanent, such as permanently storing a new data item in a container after it interacts with an application data container.

	[Column 26; Lines 31-51] The architecture 700 also includes an edge data controller 718 . It will be understood that the edge data controller 718 can be one or more software applications stored in a memory and executed by one or more processors that manages use of data containers for users and other entities. In various embodiments of this disclosure the edge data controller 718 can be the container controller 618 described with respect to FIG. 6 A. In various embodiments of this disclosure, the container controller 618 and/or the edge data controller 718 gate access to and use or viewing of data in the data containers 716 , providing remote control, decryption, access and audit logging, and smart data contract and compliance enforcement. In various embodiments, the data containers 716 are read only and secured by encryption, such as using a public-private key scheme, and/or a BYOK key scheme with data owner's managing decryption keys. The data containers 716 , as described in the various embodiments of this disclosure, are programmable data objects that can be copied and distributed, with **their distribution and access logged by the master data controller 710 and/or the edge data controller 718.** **Note:** *The mapped citation describes about containers owned by different users/organizations, composable applications, container relationships, container-specific access control and logging/audit. However, the reference does not specifically disclose dynamic composition of video content from independently owned video libraries based on traversal.*
14. The rights management mechanism further generates financial settlement information for respective content owners based on traversed branches.	**[Column 66; Lines 8-25]** The process 2900 beings at block 2902 , in which the processor encodes media into a data container and assigns a UUID to the new data container. The media can include text, audio, video, one or more images, a GIF, or any other type of data. At block 2904 , the processor provides and/or actively records in a block on the blockchain an identification of the data container that includes the UUID and the owner ID, such as described with respect to FIG. 28 , minting an NFT in some embodiments for the media stored in the data container. The blockchain holds a reference to the NFT, e.g., the UUID paired with the owner ID, with **the blockchain recording any subsequent licensing, sale, or transfer of the data container.** In some embodiments, the UUID paired with the owner ID represents the NFT. In some embodiments, a hash of these two values, the UUID and the owner ID, is the NFT. In some embodiments, a hash of the UUID is the NFT, with the owner ID/public key being used for verification and security using a private-public key scheme. **[Column 66; Lines 48-63]** At block 2910 , the processor can receive a transfer of consideration for the data container from the third party. For example, in embodiments in which media stored in the data container is intended to be sold or licensed to a third party**, the owner of the media stored in the data container can request payment as a condition of the offer. The payment can be handled off the blockchain in some embodiments. In embodiments in which payment is provided via cryptocurrency, the payment can be handled on the blockchain, or on a separate blockchain.** At block 2912 , the processor transfers or provides access to the data container, such as described in the various embodiments of this disclosure, including providing information to decrypt the data container to the third party. **At block 2914 , the successful offer and license, transfer, or sale**

	<u>of the data container is recorded in a block on the blockchain.</u>.. **Note:** *The mapped citation describes about licensing, sale, transfer, payment/consideration and blockchain recording of transactions. However, the payment is not specifically calculated based on traversed video branches or branch-specific traversal events.*

5.2 Relevant Non-Patent Citations

Non-patent Citation 1: Eye Tracked Foveated Rendering

Title	Eye Tracked Foveated Rendering
Author(s)	None
Affiliation(s)	None
Publisher	None
Publication Date	None
DOI	None
Source Link	https://developers.meta.com/vr/documentation/unity/unity-eye-tracked-foveated-rendering/

Abstract: NA

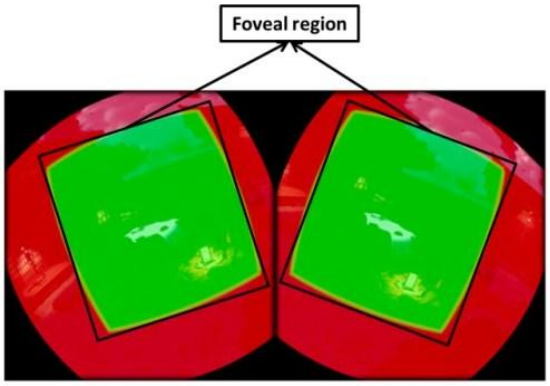
Key Features	Relevant Excerpts
1. A system for traversal-governed video assembly.	The PremiumMediaSample showcases **advanced media streaming and spatial integration techniques** for Meta Quest headsets. It demonstrates DRM-protected **video playback,** custom shaders that cast screen light onto room surfaces, multi-process Jetpack Compose panels, and MRUK-based wall snapping. ...**Stream Widevine DRM-protected content using ExoPlayer with direct-to-surface rendering** Explore the sample The sample uses a runtime-constructed scene with no GLXF files**. All entities, panels, and lighting meshes are created programmatically:** ImmersiveViewModel.kt Orchestrates ExoPlayer, panels, cinema states **Central state management, entity lifecycle coordination** Runtime behavior When you run this sample, you see a home screen panel displaying a carousel of media thumbnails. **Select an item to start playback.** For rectilinear content (standard flat video), a control panel appears below **the video with play/pause,** seek controls, passthrough and lighting sliders, and Cinema/TV mode buttons. Cinema mode creates a large virtual theater screen with dimmed passthrough, while TV mode displays a smaller screen with visible room lighting effects. The video screen casts ambient light onto nearby walls, floors, and ceilings, creating soft reflections on floor surfaces. For 180-degree content, the video wraps around you in a hemispherical view with detached controls. **Source: [Link 1]**

	Overview This sample demonstrates how to build an **immersive stereo video player** with directional spatial audio using the Meta Spatial SDK. It covers VR and mixed reality mode switching, multiple panel registration approaches, and custom **mesh rendering** with ECS-based audio spatialization. **Source: [Link 2]** **Note:** *The mapped citation describes a Meta Quest system for runtime video assembly and playback, including a carousel of media thumbnails where selecting an item starts playback, together with central state management and orchestration of ExoPlayer, panels, and cinema states. However, it does not expressly describe traversal-governed assembly.*
2. A virtualization engine configured to generate a virtual video reference container by removing the media payload (MDAT) while retaining structural andtiming metadata (STBL).	N/A
3. The retained metadata comprises rewritten sample table references identifying media sample blocks stored externally from the virtual video reference container.	N/A
4. A graph composition mechanism configured to interconnect multiple virtual video reference containers through traversalreferences to form a traversable video graph, including recursive or nested traversal.	Runtime behavior When you run this sample, you see a custom 3D room environment called MediaRoom with a large flat video panel mounted on the wall. To the left, **a catalog panel displays 10 video thumbnails in a grid (9 YouTube videos and 1 local 360 video).** Below the catalog, a small panel shows a passthrough toggle switch. When you select a YouTube video from the catalog, it loads in the flat panel via WebView, and the video content projects onto the room walls as emissive lighting. When you select the “Soloist 3D” video, the room animates out with a transition shader effect and a 360-degree stereoscopic sphere appears around you. **ExoPlayer plays the immersive video in top/bottom stereoscopic format.** Toggle the passthrough switch to hide the room and skybox, enable camera passthrough with environment depth, and make the video panel grabbable so you can reposition it in your physical space. Key concepts Four panel registration types All four Spatial SDK panel registration types appear in a single registerPanels() method. The video panel uses LayoutXMLPanelRegistration with a WebView defined in XML. The **catalog panel uses ActivityPanelRegistration, launching a separate ComponentActivity** with Jetpack Compose UI. The passthrough toggle

	uses ComposeViewPanelRegistration, rendering inline Compose content without a separate activity. The 360 video sphere uses VideoSurfacePanelRegistration with Equirect360ShapeOptions and StereoMode.UpDown for top/bottom **stereoscopic rendering.** See MediaPlayerSampleActivity.kt method registerPanels(). **Dual video playback pathways** **Video routes to different players based on content type.** YouTube URLs load in a WebView panel as embedded iframes, suitable for flat web video. **Local video URIs load in ExoPlayer and render to a VideoSurfacePanelRegistration configured with Equirect360ShapeOptions for 360-degree playback. The playVideo() method inspects the URL string and sets movieState accordingly.** See MediaPlayerSampleActivity.kt method playVideo() and the movieState setter. **Source: [Link 3]** **Note:** *The mapped citation describes a catalog of multiple video thumbnails, including YouTube and local 360° videos, where selecting a video loads it through a corresponding playback path. It further discloses different player types based on content type, with playVideo() inspecting the URL to determine playback behavior. However, traversal references, a traversable video graph, and recursive or nested traversal are not expressly disclosed.*
5. A runtime traversal engine configured to evaluate dynamic branch conditions based on real-time signals and select a target video container from multiple candidate containers, wherein the signals includeviewer input, AI agent directives, rights policies, or environmental context.	Overview **Gaze interactions let users target virtual objects with their eyes. The user looks at an object, then confirms the action with a pinch or a controller button. You can combine gaze with hand tracking and controllers.** Gaze suits menus, buttons, and other user interface work. Architecture of a gaze interaction A gaze interaction starts with the user's gaze ray. Interaction SDK derives that ray from eye tracking and head tracking. A hit-test mechanism casts the ray against the gaze targets in the scene. **The target that wins the hit test then receives interaction events such as hover and select.** **Source: [Link 4]** Gaze availability is not guaranteed at runtime**. Applications that receive gaze data directly request a runtime permission, and the user can revoke it at any time.** **On Meta VR devices with eye tracking**, 2D panel surfaces do not receive **gaze data,** because eye tracking data is sensitive. This covers 2D Android applications and **web content in Meta Quest Browser. The system renders hover on their behalf and delivers touch events only.** **Source: [Link 5]** **Note:** *The mapped citation describes runtime gaze interaction in which eye- and head-tracking data are used to evaluate multiple gaze targets and select a target based on the real-time viewer signal. However, video containers, traversal between video branches, and AI-agent directives or rights policies as branch conditions are*

	not expressly disclosed.
6. A token validation module configured to validate an independent cryptographic video token associated with the selected target video container and grant traversal authority upon successful validation.	**Token Generation - Based on the received security signals, the Attestation Server generates an attestation token. This token is cryptographically signed by the Attestation Server, reinforcing the security and reliability of the process.** **Token Receipt - The Attestation API receives the attestation token and sends it to the application client.** **Token Verification Request - Upon receiving the service request, the Application Server sends a token verification request to the Attestation Server.** **Token Verification - The Attestation Server validates the token and sends back a success or failure message along with token claims to the Application Server. If the token is invalid, an error message is sent. However, if the token is valid, token claims are sent.** **Source: [Link 6]** How authentication works When your app requests user identity, Horizon OS handles the authentication flow through the platform. The user doesn't need to enter credentials inside your app — they're already signed into their Meta account on the device. Your app **receives a token that identifies the user and grants access** to the platform features you've requested. Key capabilities User identity — Get a unique, stable user ID for your app without requiring a separate login **Access tokens — OAuth 2.0 tokens for calling Meta Platform APIs** **Source: [Link 7]** **Note:** *The mapped citation describes a cryptographically signed attestation token that is validated by an Attestation Server, which returns token claims and a success or failure result. The validated token identifies the user and grants access to requested platform features, supporting token-based authorization. However, traversal authority is not expressly disclosed.*
7. The cryptographic video token defines or enforces licensing, usage, consent, access, or financial settlement conditions, and successful validation releases an ephemeral decryption key or signed policy authorization.	**Token Generation - Based on the received security signals, the Attestation Server generates an attestation token. This token is cryptographically signed by the Attestation Server, reinforcing the security and reliability of the process.** **Token Receipt - The Attestation API receives the attestation token and sends it to the application client.** **Token Verification Request - Upon receiving the service request, the Application Server sends a token verification request to the Attestation Server.** **Source: [Link 6]** How authentication works When your app requests user identity, Horizon OS handles the authentication flow through the platform. The user doesn't need to enter credentials inside your app —

	they're already signed into their Meta account on the device. Your app **receives a token that identifies the user and grants access** to the platform features you've requested. Key capabilities User identity — Get a unique, stable user ID for your app without requiring a separate login **Access tokens — OAuth 2.0 tokens for calling Meta Platform APIs** Account linking — Connect Meta accounts to your own backend identity system Federated authentication — Let users sign into third-party services through Meta's authentication flow **Permissions — Request specific data access** (friends list, avatar, user profile) with **user consent** **Entitlement checks — Verify that the user owns your app before granting access to content** **Entitlement verification** Every app distributed on the **Meta Horizon Store should verify that the user has a valid entitlement (purchase or free-to-download claim) before granting access.** This prevents unauthorized copies from running **Source: [Link 7]** **Note:** *The mapped citation describes a cryptographically signed attestation token verified by an Application Server to identify the user and enforce consent, access permissions, and entitlement conditions. Entitlement checks, including valid purchase or free-download claims, are performed before granting access to content, supporting cryptographic token-based authorization.*
8. A spatial resolution module configured to determine spatial resolution gates based on viewer spatial context and provide full fidelity for a foveal region, reduced fidelity for a peripheral region, and withhold media for an out-of-view region.	Overview **Gaze interactions let users target virtual objects with their eyes. The user looks at an object, then confirms the action with a pinch or a controller button. You can combine gaze with hand tracking and controllers**. Gaze suits menus, buttons, and other user interface work. Architecture of a gaze interaction A gaze interaction starts with the user's gaze ray. Interaction SDK derives that ray from eye tracking and head tracking. A hit-test mechanism casts the ray against the gaze targets in the scene. **The target that wins the hit test then receives interaction events such as hover and select.** **Source: [Link 4]** Gaze availability is not guaranteed at runtime**. Applications that receive gaze data directly request a runtime permission, and the user can revoke it at any time.** **On Meta VR devices with eye tracking**, 2D panel surfaces do not receive **gaze data,** because eye tracking data is sensitive. This covers 2D Android applications and **web content in Meta Quest Browser. The system renders hover on their behalf and**

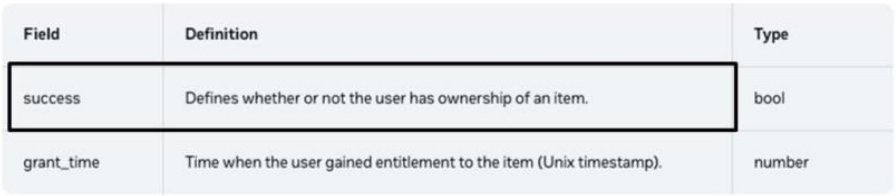
	delivers touch events only. **Source: [Link 5]** **Eye Tracked Foveated Rendering** **Eye Tracked Foveated Rendering (ETFR) utilizes gaze direction to render full resolution where you are looking (the foveal region), and low pixel density in the periphery, taking advantage of lower peripheral acuity in human vision.** Compared to using Fixed Foveated Rendering, ETFR can reduce the perceptible difference in image quality for a given amount of foveation, leading to substantial GPU savings for Meta Quest applications without compromising the visual experience. **Source: [Link 8]** Stereo with foveated inset (quad views) XR_VIEW_CONFIGURATION_TYPE_PRIMARY_STEREO_WITH_FOVEATED_INSET is a view configuration that renders four views per frame instead of the usual two: a wide, **low-resolution outer view per eye, plus a narrow, high-resolution inset view per eye. The inset views cover a small portion of the field of view at higher pixels per degree, while the outer views cover the full field of view at lower pixels per degree. On devices with eye tracking, the runtime steers the inset to follow the user's gaze each frame.** **Because the human visual system only resolves high detail in the foveal region, this configuration can render a significantly smaller total pixel count while preserving perceived sharpness where the user is actually looking.** Splitting the per-eye image into a high-density inset and a lower-density outer view typically cuts the total pixels rendered by roughly 50% compared with a uniform-density stereo target at matching foveal resolution.  **Source: [Link 9]** **Note:** *The mapped citation describes gaze direction being used to render full resolution where the user is looking (the foveal region) and low pixel density in the periphery. It further describes a wide, low-resolution outer view and a narrow, high-resolution inset view, with the runtime steering the inset to follow the user's gaze each frame.*
9. A prefetch mechanism configured to retrieve candidate media for	… **Stream Widevine DRM-protected content using ExoPlayer with direct-to-surface rendering**

possible traversal branches into a decoupled non-executable buffer before branch selection, while withholding the corresponding key or authorization until token validation.	Create custom GLSL shaders that sample panel textures and project ambient lighting onto MRUK surfaces Snap entities to walls, floors, and ceilings detected by MRUK **…** … ExoPlayerExt.kt ExoPlayer factory with DRM support **OculusMediaCodecVideoRenderer, Widevine DRM session manager,** DASH sources… Key concepts DRM-protected streaming The sample uses a custom OculusMediaCodecVideoRenderer to enable Widevine DRM playback. **When a DRM license URL is provided, or when using non-rectilinear video shapes like 180° equirect, the sample forces direct-to-surface rendering**:.. **Source: [Link 1]** More samples **The Widevine Video sample shows an example of using the MoviePlayer Game Object with content-protected video hosted online. This sample can be found in the sample folder as Stereo180Video.** Widevine is a proprietary digital rights management system developed by Google. It can easily be used when targeting the Mobile platforms. There are only a few differences to the **Stereo180Video sample.** Setup: The Widevine Video sample requires the same setup as Stereo180Video to enable the Native Android Video Player. **There is no need to import a local .mp4 video file directly into your Unity project.** Movie Player Sample: **instead of providing a local file path to a video, you provide a URL in the Movie Name property. You also populate the Drm License Url property**. **Source: [Link 10]** **Note:** *The mapped citation describes DRM-protected streaming using ExoPlayer, Widevine DRM, DASH sources, and a separate DRM license URL, supporting separation of the video content source from DRM authorization. However, prefetching candidate media before branch selection and use of a decoupled non-executable buffer are not expressly disclosed.*
10. A playback assembly mechanism configured to retrieve, decrypt, authorize, and sequence media sample blocks in volatile memory according to traversal authorization and spatial resolution, thereby	Media panel registration overview Spatial SDK offers two specialized panel registration classes for media content: **VideoSurfacePanelRegistration: Direct-to-surface rendering for maximum performance and DRM support** **ReadableVideoSurfacePanelRegistration: Surface rendering with post-processing capabilities for custom effects** **These classes handle surface management, media source integration, and**

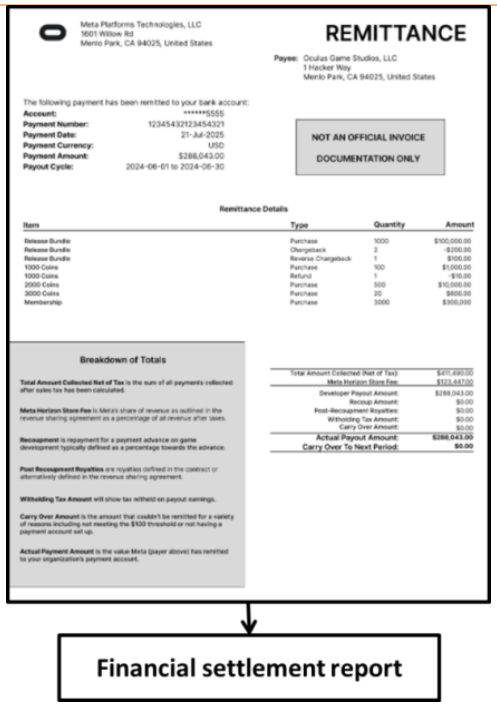
generating continuous playback without writing a rendered video file to persistent storage.	**performance tuning. They provide APIs for connecting media players like ExoPlayer.** **Direct-to-surface rendering** **Direct-to-surface rendering writes directly to a panel surface, improving performance by skipping Android UI setup.** Direct-to-compositor is a prerequisite. If you enable direct-to-surface without direct-to-compositor, your app will crash. **VideoSurfacePanelRegistration** sets this up automatically. The following code sample shows how to set it up manually if you need more direct control. …This approach bypasses **the Android UI rendering pipeline. It improves performance by allowing ExoPlayer to render video frames directly to the panel surface.** **Source: [Link 11]** … **Stream Widevine DRM-protected content using ExoPlayer with direct-to-surface rendering** Create custom GLSL shaders that sample panel textures and project ambient lighting onto MRUK surfaces Snap entities to walls, floors, and ceilings detected by MRUK **…** … ExoPlayerExt.kt ExoPlayer factory with DRM support **OculusMediaCodecVideoRenderer, Widevine DRM session manager,** DASH sources… Key concepts DRM-protected streaming The sample uses a custom **OculusMediaCodecVideoRenderer** to enable Widevine DRM playback. **When a DRM license URL is provided, or when using non-rectilinear video shapes like 180° equirect, the sample forces direct-to-surface rendering**:.. **Source: [Link 1]** **Video Player – Used for Rift video playback.** OVR Overlay – Used as a surface on which **ExoPlayer plays the video on Mobile platforms.** Of note is the Android-only Is External Surface check box. When checked, this allows for an Android external surface to be controlled by the Asynchronous TimeWarp (ATW) layer. In this sample, this allows the Android ExoPlayer video player surface to be run as an ATW layer, **avoiding redundant texture copies to save memory and cycles.** **Source: [Link 12]** **Note:** *The mapped citation describes Widevine DRM-protected streaming with ExoPlayer and direct-to-surface rendering, avoiding redundant texture copies and supporting memory-efficient protected video playback without requiring a rendered*

	video file. However, traversal authorization and sequencing media sample blocks in volatile memory are not expressly disclosed.
11. A policy enforcement mechanism configured to apply inherited policies to recursively nested containers, wherein the most restrictive applicable policy governs traversal.	Key capabilities User identity — Get a unique, stable user ID for your app without requiring a separate login Access tokens — OAuth 2.0 tokens for calling Meta Platform APIs Account linking — Connect Meta accounts to your own backend identity system Federated authentication — Let users sign into third-party services through Meta's authentication flow Permissions — Request specific data access (friends list, avatar, user profile) with user consent **Entitlement checks — Verify that the user owns your app before granting access to content** Common patterns Entitlement verification **Every app distributed on the Meta Horizon Store should verify that the user has a valid entitlement (purchase or free-to-download claim) before granting access. This prevents unauthorized copies from running.** **Source: [Link 7]** Data Use Checkup for user-data features If your app uses a feature in Table 1 of the **Data Use Checkup page, an admin from your team must complete a Data Use Checkup (DUC) to certify that your API access to those features complies with Developer Policies. The app's access to those features is contingent on a successful review of its DUC by the app review team.** During development, a submitted DUC gives your app automatic provisional access to the features it requests until you submit the app for review; see When to submit a DUC. Test users can also call these features before DUC approval, because test user accounts are exempt from DUC requirement **Source: [Link 13]** Entitlement Check for Meta Horizon Store Apps Apps being sold in the Meta Horizon Store that wish to pass VRC.Quest.Security.1, or wish to implement anti-piracy measures, can perform **a platform-level check to verify the user purchased or obtained your app legitimately.** This check is called the entitlement check. You should make the entitlement check within 10 seconds of the user launching your app. The entitlement check does not require the user to be connected to the Internet. Also, you must handle **a failed entitlement check in your app code**. A failed entitlement check won't result in any action on its own. For example, if the check fails, you could show the user an error message and quit the app, or go into a

	limited demo mode. Initialize the Platform and Perform the Entitlement Check… **Source: [Link 14]** **Note:** *The mapped citation describes user-consent permissions, entitlement checks, and policy compliance reviews that govern access to content and API functionality. Failed entitlement checks may restrict behavior, such as exiting the application or entering a limited demo mode. However, recursively nested containers and policies governing traversal are not expressly disclosed.*
12. The traversal framework further enforces cycle detection, maximum traversal-depth restrictions, and real-time rights verification.	N/A
13. A rights management mechanism configured to dynamically compose video content from independently owned content libraries using container-specific authorization and maintain an audit record of traversal events.	Entitlement Check for Meta Horizon Store Apps Apps being sold in the Meta Horizon Store that wish to pass VRC.Quest.Security.1, or wish to implement anti-piracy measures, can perform **a platform-level check to verify the user purchased or obtained your app legitimately.** This check is called the entitlement check. You should make the entitlement check within 10 seconds of the user launching your app. Entitlement Check Server APIs **Entitlement checks can also be performed via S2S REST APIs. This prevents any client-side tampering, and provides additional information about user entitlements**. Using the S2S APIs are not required. See the Server-to-Server API Basics page for information about interacting with our server APIs. **Source: [Link 14]** These features can all be managed in the Developer Dashboard at Monetization > Add-ons. **Quest monetization features allow you to strategize engagement methods and additional revenue sources** for your app after launching it to the Quest store. Decisions around monetization can help improve the discoverability and longevity of your app. Downloadable content (DLC) and in-app purchases (IAP) Both DLC and IAP can be used to add content to your app or offer services, passes, or **rewards to users** interested in purchasing additional content. **Source: [Link 15]** Set up Platform SDK entitlements Set up Platform SDK entitlements for your Unity app to unlock the following benefits:

	User identity and authentication: Verifies the user is entitled to run your app. Avatars and social features: Gives users access to social features like matchmaking and voice chat. **Monetization: Enables in-app purchases, subscriptions, and downloadable content (DLC).** Engagement and retention services: Provides user notifications, leaderboards, and more. **Source: [Link 16]**  **Source: [Link 14]** **Note:** *The mapped citation describes entitlement checks verifying legitimate app purchases, including S2S REST APIs for obtaining entitlement information. It further discloses DLC and IAP for adding separately purchased content, services, subscriptions, and rewards, supporting rights management and entitlement-based authorization. However, dynamically composing video content from independently owned libraries and maintaining traversal-event audit records are not expressly disclosed.*
14. The rights management mechanism further generates financial settlement information for respective content owners based on traversed branches.	Payments This topic describes how to view and manage your payment information for apps and digital products sold on the Meta Horizon Store. **As a VR app developer, you can use the Payment dashboard to track earnings, view transactions, configure payment details, and download financial reports.** All steps in this topic are performed using the Developer Dashboard. Transactions tab **The Transactions tab displays a detailed list of all your payment transactions, including earnings from app sales and payouts to your account.** Use the following steps to view your transaction history from the Payment dashboard: Click the Transactions tab… **Source: [Link 17]** **Get financial reports** After configuring your team's banking and financial settings, **you can access financial information about your team and its apps. This includes metrics such as payments from Meta and revenue reports for your apps and team.** This topic will guide you on the available metrics and report types, and how to access them by way of the Meta Quest Developer Dashboard

Field	Definition	Type
success	Defines whether or not the user has ownership of an item.	bool
grant_time	Time when the user gained entitlement to the item (Unix timestamp).	number



Source: [Link 18]

Note: *The mapped citation describes a Payment dashboard for apps and digital products, allowing developers to track earnings, transactions, payouts, and financial reports. Such transaction and revenue information associated with particular apps or digital products can reasonably support attribution of financial data to accessed content, including traversed branches. However, branch-specific financial settlement is not expressly disclosed.*

Non-patent Citation 2: Virtual Video Files At Scale: Seamlessly Processing Billions Of Videos Per Day

Title	Virtual Video Files At Scale: Seamlessly Processing Billions Of Videos Per Day
Author(s)	None
Affiliation(s)	None
Publisher	None
Publication Date	None
DOI	None
Source Link	https://atscaleconference.com/virtual-video-files-at-scale-seamlessly-processing-billions-of-videos-per-day/?

Abstract: At Meta, we handle billions of video uploads per day across Facebook, Instagram, Messenger, Whatsapp, and Oculus. These videos can range from short clips, such as Stories and Reels, to multi-hour live broadcasts or high-resolution 360 content for Oculus headsets.

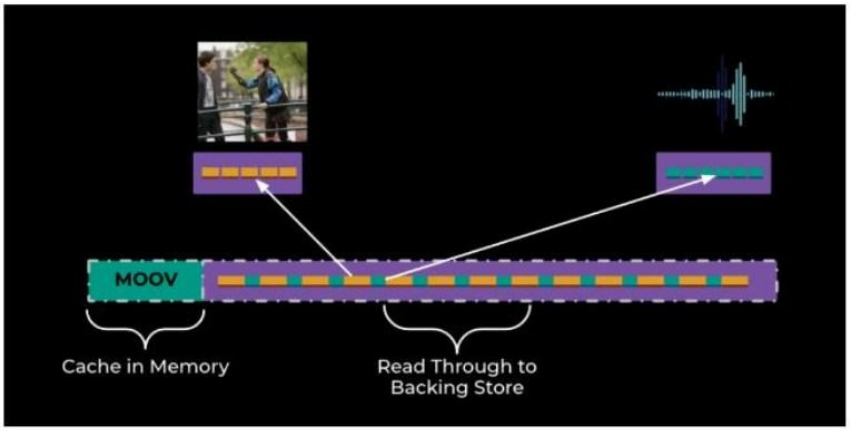
To more seamlessly process this wide range of content, we have built several technologies to simplify the process: virtual video files, FFUSE, and FUSE file systems.

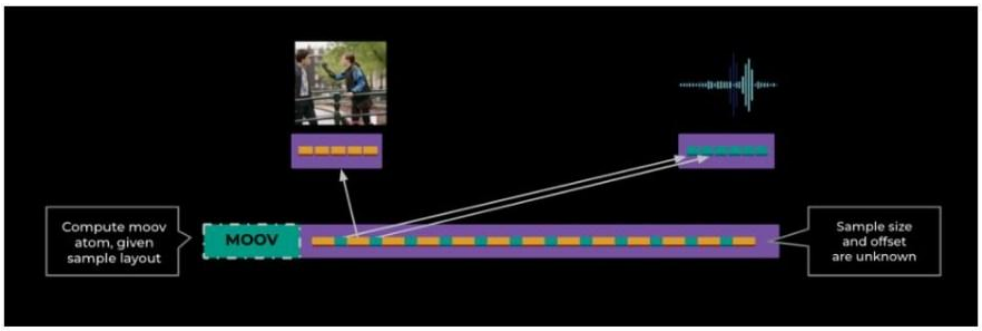
With these tools, we can instantly and efficiently transfer video bytes around our data centers to provide the best possible experience for people using our apps.

In this blog post, we'll cover the challenges of processing and storing videos at Meta Scale, how we leverage the MP4 file container format to create a virtual video abstraction to more easily manipulate video data, and lastly how we connect this all to FFmpeg—the encoding "Swiss

Army Knife."

Key Features	Relevant Excerpts
1. A system for traversal-governed video assembly.	With these tools, we can instantly and **efficiently transfer video bytes around our data centers to provide the best possible experience for people using our apps.** WHAT ARE VIRTUAL VIDEO FILES? The core idea behind **virtual video files is to have the ability to take any set of frames or samples, from any set of source files, and instantly produce not a new file, per se, but a representation of the file:** We know what the file size is, and we can read the file at any offset; the data itself, however, is only produced as requested, lazily. Virtual video files solve this problem. The goal is to implement a virtual-file interface—**what the file would be if we did the full remix—without having to materialize the data up front. In doing so, we can serve the file (e.g., over an HTTP server) as if it exists, but only materialize bytes as needed.** Once this file is created, both reads() and stat() can be served in constant time. …At a high level, an MP4 file is a structure that contains a list of tracks. Each track contains some metadata and a list of samples. And finally, the sample table shows information contained within the list of samples: their position, size, timestamps, and flags.

	One important thing we'll also track is the file from which each sample comes. Because we may be **combining samples from multiple different source files, we need to know which file to read later on when referring to this sample**… **Source : [Link 1]** **Note:** *The mapped citation describes taking any set of frames or samples from any set of source files and producing a representation of the file, with the file size and ability to read the file at any offset, and tracking the file from which each sample comes when combining samples from multiple different source files. However, it does not describe traversal being governed.*
2. A virtualization engine configured to generate a virtual video reference container by removing the media payload (MDAT) while retaining structural and timing metadata (STBL).	WHAT ARE VIRTUAL VIDEO FILES? The core idea behind **virtual video files is to have the ability to take any set of frames or samples, from any set of source files, and instantly produce not a new file, per se, but a representation of the file: We know what the file size is, and we can read the file at any offset; the data itself, however, is only produced as requested, lazily**… …The purpose of **a video file format (or container) is to provide organization to the samples stored within,** and to the instructions needed to interpret them correctly. An important data structure **within a container is the sample table. This is a table containing the file offset, size, PTS, DTS, and flags (e.g., if a frame is a key frame) for every sample in a given track.** **With the track metadata and sample tables, a video player has all the information needed for playback.** …For example, the ftyp atom, or "File Type" atom, indicates the file type. The moov (Movie) atom contains the most crucial information about the file, including information about tracks and their sample tables. **The mdat (Move Data) atom contains an arbitrary blob of data, which would typically contain the raw audio/video samples.** **…The mdat atom is never directly constructed. Instead, for any read overlapping with this atom, we'll determine which portion of which samples are stored there,** and then read that sample data directly from the original sources.  Figure 16: Handling virtual read requests **Source : [Link 1]** **Note:** *The mapped citation describes a virtual video file/container with structural and timing metadata in the sample table, including file offset, size, PTS, DTS, and*

	flags, while the mdat media payload is not directly constructed and sample data is read from the original sources as needed. Therefore, it define a generating a virtual video reference container by removing the media payload while retaining structural and timing metadata.
3. <u>The retained metadata comprises rewritten sample table references identifying media sample blocks stored externally from the virtual video reference container.</u>	First, we must parse the MP4 atom structure to build an in-memory representation of both the sample tables, as well as all other key **metadata contained within the file. This can be represented in memory as a set of structures that can then be used to construct a new MP4 file.** At a high level, an MP4 file is a structure that contains a list of tracks. Each track contains some metadata and a list of samples. And finally, **the sample table shows information contained within the list of samples: their position, size, timestamps, and flags.** One important thing we'll also **<u>track is the file from which each sample comes. Because we may be combining samples from multiple different source files, we need to know which file to read later on when referring to this sample.</u>** STEP 2: COMPUTE SAMPLE LAYOUT Next, we'll compute **the new sample table for our virtual file.** For a progressive MP4, we do this by interleaving samples in ascending decode-timestamp order. This allows a video player to fetch both audio and video samples in a single contiguous range of data. **<u>As we lay these samples out, the result is a new sample table, with references to the original samples. We know how large the new sample layout is (the mdat atom size), and the relative offset of each sample within.</u>** …**<u>The mdat atom is never directly constructed. Instead, for any read overlapping with this atom, we'll determine which portion of which samples are stored there, and then read that sample data directly from the original sources.</u>** …**<u>With some careful accounting, we now know the new moov atom size. We can then adjust the sample offsets accordingly, from their relative position within the mdat atom to their final absolute position in our virtual video file.</u>**  Figure 15: Virtual moov atom **Source : [Link 1]** **Note:** *The mapped citation describes a computing a new sample table with references to the original samples, including the position, size, timestamps, and flags of the samples, and tracking the file from which each sample comes so that the sample can be read later. It also define that the mdat atom is never directly constructed and that sample data is read directly from the original sources, with sample offsets adjusted to their final absolute positions in the virtual video file.*

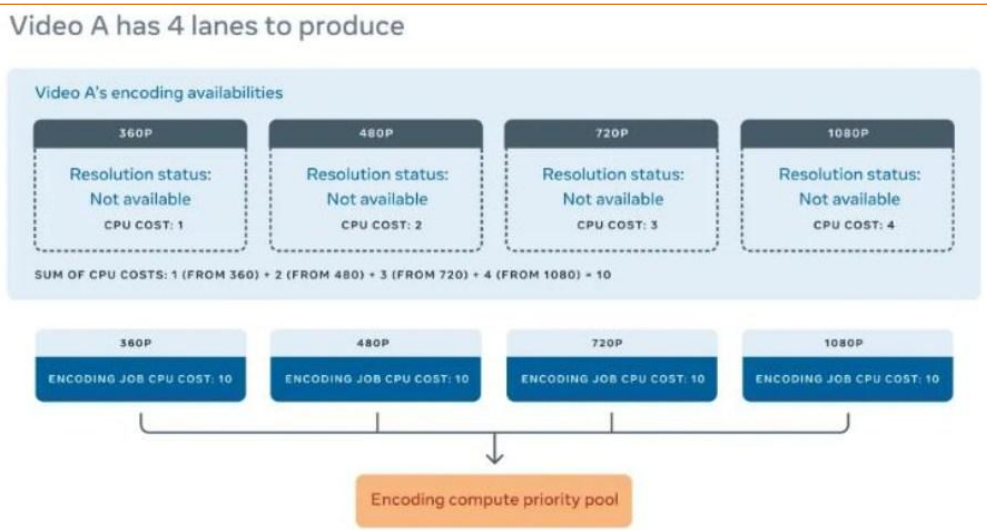
4. A graph composition mechanism configured to interconnect multiple virtual video reference containers through traversal references to form a traversable video graph, including recursive or nested traversal.	MP4 FILE LAYOUT **MP4 files use a tree-like data structure to organize information. The file itself sits at the root of this tree, and then nested nodes called "boxes" or "atoms" contain a hierarchy of information.** **Each atom contains a 4-byte type, a size, and may also contain children. Generally the data is stored in the leaf nodes, whereas the interior nodes simply provide organizational structure.** USE CASE: DISTRIBUTED ENCODING While virtual video files were initially built to remux DASH to Progressive instantly, we have found many other uses for them. Distributed encoding is one example. **With distributed encoding, a large video is broken into smaller chunks that can be processed in parallel. The outputs for each chunk are then combined together to produce one logical output.** In doing so, even long videos (e.g., multi-hour live broadcasts) or high-resolution content (e.g., 8k stereoscopic Oculus) can be processed much faster than in serial encoding. With virtual video files, this process can be done seamlessly, without any single host having to materialize the entire output file. First, we analyze the source file to be processed and identify the key frames to be used as chunking points. **For each chunk, we construct a virtual video file to refer to just those samples contained within the chunk.** **Second, we can process each chunk in parallel. Because the virtual chunk refers only to the samples we want to process, we download the exact set of bytes we need from the original source.** **Lastly, we can construct one more virtual video file to combine the results of each chunk.** This video file can make the disparate chunks, stored in different locations, available as one logical, final output. **Source : [Link 1]** Objects and associations In 2007, a few Facebook engineers set out to define new data storage abstractions that would fit the needs of all but the most demanding features of the site while hiding most of the complexity of the underlying distributed data store from product engineers. **The Objects and Associations API that they created was based on the graph data model** and was initially implemented in PHP and ran on Facebook's web servers. **It represented data items as nodes (objects), and relationships between them as edges (associations).** The API was an immediate success, with several high-profile features, such as likes, pages, and events implemented entirely on objects and associations, with no direct memcache or MySQL calls. This simple example shows **a subgraph of objects** and associations that is created in TAO after Alice checks in at the Golden Gate Bridge and tags Bob there, while Cathy comments on the check-in and David likes it. **Every data item, such as a user, check-in, or comment, is represented by a typed object containing a dictionary of named fields. Relationships between objects, such as "liked by" or "friend of," are represented by typed edges (associations) grouped in association lists by their origin. Multiple associations may connect the same pair of objects as long as the types of all those associations are distinct. Together objects and associations form a labeled directed multigraph.** **For every association type a so-called inverse type can be specified. Whenever an**

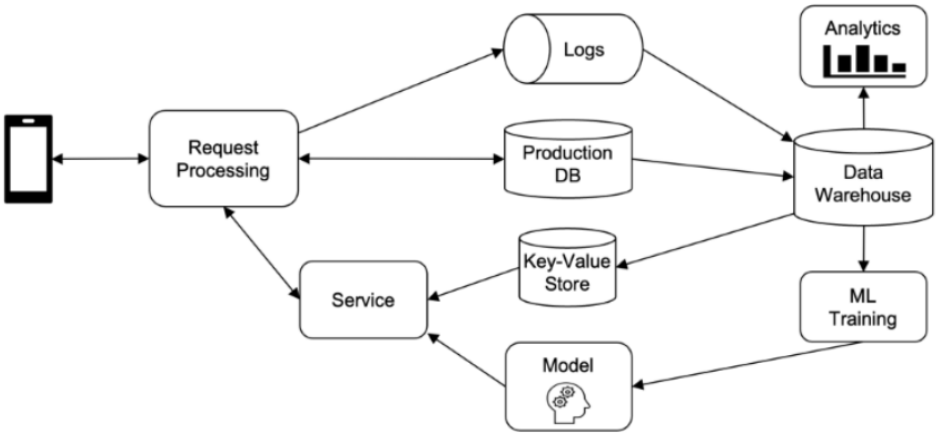
	edge of the direct type is created or deleted between objects with unique IDs id1 and id2, TAO will automatically create or delete an edge of the corresponding inverse type in the opposite direction (id2 to id1). The intent is to help the application programmer maintain referential integrity for relationships that are naturally mutual, like friendship, or where support for **graph traversal in both directions is performance critical, as for example in "likes" and "liked by."** **Source : [Link 2]** Essentially our ranking stack is **a graph-based execution service that orchestrates the entire serving workflow to eventually generate a number of video stories and return them to the web server.** The recommendation-serving workflow typically includes **multiple stages such as candidate retrieval from multiple types of retrieval ML models, various filtering, point-wise ranking, list-wise ranking, and heuristic-diversity control.** A candidate will have to go through all these stages to survive and be delivered**.** **Source : [Link 3]** Figure 19: **Source : [Link 1]** **Note:** *The mapped citation describes a constructing virtual video files for individual chunks and then constructing one more virtual video file to combine the results of each chunk, making disparate chunks stored in different locations available as one logical, final output. It further describes a tree-like data structure with nested nodes and a graph data model in which relationships between objects are represented by associations, supporting graph traversal in both directions.*
5. A runtime traversal engine configured to evaluate dynamic branch conditions based on real-time signals and select a target video container from multiple candidate containers, wherein the signals include viewer input, AI agent directives, rights policies, or environmental context.	…The recommendation-serving workflow typically includes **multiple stages such as candidate retrieval from multiple types of retrieval ML models, various filtering, point-wise ranking, list-wise ranking, and heuristic-diversity control.** A candidate will have to go through all these stages to survive and be delivered. Real-time ranking **adjusts video content ranking based on user interactions and engagement signals, delivering more relevant content as people interact with the platform** **Collecting real-time signals such as video view time, likes, and other interactions using asynchronous data pipelines enables accurate ranking, or it enables using synchronous data pipelines such as piggybacking a batch of these signals into the next tail load request. This process relies on system latency and signal**

	completeness. It's important to note that if the snapshot of real-time signals between two distinct ranking requests is similar, there is little to no adjustment that ranking can perform to react to the person's current interest. Ranking videos in real time ensures prominent display of relevant and engaging content, while eliminating duplicates for content diversification as well as new topic exploration. This approach enhances **user engagement by providing a personalized and responsive viewing experience,** adapting to people's preferences and behaviors in real time during app sessions. Think of responsiveness as how well and consistently our end-to-end infrastructure delivers fresh content. **Source : [Link 3]** … To address this gap, we **integrated friend-bubble interaction signals as features and added new tasks into both early-stage and late-stage ranking multi-task, multi-label (MTML) models to incorporate viewer-friend relationship strength and to learn downstream engagement on videos** with social bubbles. With these signals added across the ranking funnel, the models can better recognize the value of friend-interacted content, learn the relationship between closeness and viewer interest, and rank high-quality friend content higher when it is most relevant… **Source : [Link 4]** **Note:** *The mapped citation describes a recommendation-serving workflow that retrieves multiple video candidates, decision condition (such as applies filtering, point-wise ranking, list-wise ranking, and heuristic-diversity control), and delivers a candidate after it survives these stages. It further describes real-time ranking that adjusts video content based on user interactions (i.e., user input) and engagement signals. However, it does not describe a runtime traversal engine or dynamic branch conditions, nor does it identify AI agent directives, rights policies, or environmental context.*
6. A token validation module configured to validate an independent cryptographic video token associated with the selected target video container and grant traversal authority upon successful validation.	N/A
7. The cryptographic video token defines or enforces licensing, usage, consent, access, or financial settlement conditions, and successful validation releases an ephemeral decryption key or signed policy authorization.	N/A
8. A spatial resolution module configured to	N/A

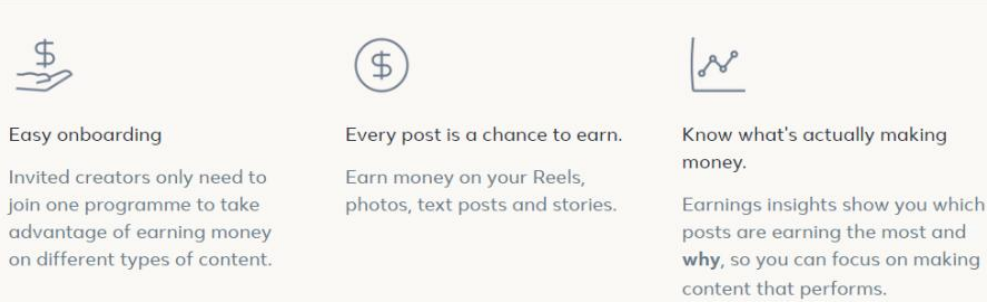
determine spatial resolution gates based on viewer spatial context and provide full fidelity for a foveal region, reduced fidelity for a peripheral region, and withhold media for an out-of-view region.	
9. A prefetch mechanism configured to retrieve candidate media for possible traversal branches into a decoupled non-executable buffer before branch selection, while withholding the corresponding key or authorization until token validation.	The mobile client sends a request The mobile client generally has a few mechanisms in place that can trigger a network request. Each request type has its own trigger. **A prefetch request (one issued before the surface is visible) is triggered a short time after app startup. Prefetch is available only for our tab surface.** A head load (the initial network request) is triggered when the user navigates to the surface. A tail load request (which includes all the subsequent requests) is triggered every time the user scrolls, with some caveats. **A prefetch and a head load can both be in flight at once**. We will vend content for whichever one returns first. Or if they both take too long, we will vend cache content… The mobile client receives a response from the server **When the client receives the network response, the video stories are added to the in-memory pool.** We prioritize the stories based on the server sort key, which is provided by the ranking layer, as well as on whether it has been viewed by the person. In this way, we defer content prioritization to the ranking layer, which has much more complex mechanisms for content recommendation compared to the client. …**When stories are added from the in-memory pool,** we also perform media prefetching; **this ensures that swiping from one video to another is a seamless experience.** **Source : [Link 3]** **Note:** *The mapped citation describes a prefetch request issued before the surface is visible, with the mobile client receiving video stories from the server and adding them to an in-memory pool, and performing media prefetching when stories are added from the in-memory pool to provide a seamless experience when swiping from one video to another. However, it does not disclose about a traversal branches into a decoupled non-executable buffer or withholding the corresponding key or authorization until token validation.*
10. A playback assembly mechanism configured to retrieve, decrypt, authorize, and sequence media sample blocks in volatile memory according to traversal authorization and spatial	…A tail load will be attempted every time the user scrolls, though we will **issue a request only if the in-memory pool (the memory store for video stories that are ready for viewing) has three or fewer video stories from the network.** So, if we have four stories from the network, we won't issue the request. If we have four stories from the cache, we will issue the network request. Alongside the tail load request, we will send user signals to ranking to help **generate good video candidates for that request.**

resolution, thereby generating continuous playback without writing a rendered video file to persistent storage.	The mobile client receives a response from the server **When the client receives the network response, the video stories are added to the in-memory pool.** We prioritize the stories based on the server sort key, which is provided by the ranking layer, as well as on whether it has been viewed by the person. In this way, we defer content prioritization to the ranking layer, which has much more complex mechanisms for content recommendation compared to the client… The server receives a response from ranking At the most basic level, ranking gives the server a list of video IDs and ranking metadata for each one. **The server then needs to load the video entities from our TAO database and execute privacy checks to ensure that the viewer can see these videos.** The logic of these privacy checks is defined on the server, so ranking can't execute these privacy checks, although ranking has some heuristics to reduce the prevalence of recommending videos that the viewer won't be able to see anyway. Then the video is passed back to the GraphQL framework, which materializes the fields that the client's query originally asked for. These two steps, **privacy checking** and materialization, together and in aggregate constitute a meaningful portion of global CPU usage in our data centers, **so optimizing here is a significant focus area to alleviate our data center demand and power consumption.** **Source : [Link 3]** …help with this, we employ a variety of codecs as well as adaptive bitrate streaming (ABR), **which improves the viewing experience and reduces buffering by choosing the best quality based on a viewer's network bandwidth**. But while more advanced codecs like VP9 provide better compression performance over older codecs, like H264, they also consume more computing power. From a pure computing perspective, applying the most advanced codecs to every video uploaded to Facebook would be prohibitively inefficient. Which means there needs to be a way **to prioritize which videos need to be encoded using more advanced codecs.** Today, Facebook deals with its **high demand for encoding high-quality video content by combining a benefit-cost model with a machine learning (ML) model that lets us prioritize advanced encoding for highly watched videos. By predicting which videos will be highly watched and encoding them first,** we can reduce buffering, improve overall visual quality, and allow people on Facebook who may be limited by their data plans to watch more videos. The impact of the new video encoding model In addition to improving viewer experience with **newly uploaded videos, the new model can identify older videos on Facebook that should have been encoded with more advanced encodings and route more computing resources to them. Doing this has shifted a large portion of watch time to advanced encodings, resulting in less buffering without requiring additional computing resources.** The improved compression has also allowed people on Facebook with limited data plans, such as those in emerging markets, to watch more videos at better quality…

	 **Source : [Link 5]** **Note:** *The mapped citation describe a retrieving video stories in response to tail load requests, adding the video stories to an in-memory pool, prioritizing the stories based on the server sort key and whether they have been viewed, and executing privacy checks to ensure that the viewer can see the videos. An adaptive bitrate streaming that chooses the best quality based on the viewer's network bandwidth. However, it does not disclose about decrypting and sequencing media sample blocks according to traversal authorization and spatial resolution, or generating continuous playback.*
11. A policy enforcement mechanism configured to apply inherited policies to recursively nested containers, wherein the most restrictive applicable policy governs traversal.	The server receives a response from ranking At the most basic level, ranking gives the server a list of video IDs and ranking metadata for each one**. The server then needs to load the video entities from our TAO database and execute privacy checks to ensure that the viewer can see these videos. The logic of these privacy checks is defined on the server, so ranking can't execute these privacy checks, although ranking has some heuristics to reduce the prevalence of recommending videos that the viewer won't be able to see anyway. Then the video is passed back to the GraphQL framework, which materializes the fields that the client's query originally asked for.** These two steps, privacy checking and materialization, together and in aggregate constitute a meaningful portion of global CPU usage in our data centers, so optimizing here is a significant focus area to alleviate our data center demand and power consumption. **Source: [Link 3]** At Meta, we've been diligently working to incorporate privacy into different systems of our software stack over the past few years. Today, we're excited to share some cutting-edge technologies that are part of **our Privacy Aware Infrastructure (PAI) initiative.** These innovations mark a major milestone in our ongoing commitment to honoring user privacy. PAI offers efficient and reliable first-class privacy constructs embedded in Meta infrastructure to address complex privacy issues. For example, **we built Policy Zones that apply across our infrastructure to address restrictions on data, such as using it only for allowed purposes, providing strong guarantees for limiting the purposes of its processing.**

	…**PAI is designed to check multiple requirements involved in data flows and is highly flexible to adapt to changing requirements.** …Building on these insights, in our latest iteration, we found that the information flow control (IFC) model offers a more durable and sustainable approach by **controlling not only data access but also how data is processed and transferred in real-time, rather than relying on point checking or out-of-band audits.** Thus, we **developed Policy Zones as our IFC-based technology and integrated it across major Meta systems to enhance our purpose limitation capabilities at scale.** This effort was later expanded into the Privacy Aware Infrastructure (PAI) initiative, a transformative investment that integrates first-class privacy support into Meta's infrastructure systems. We believe PAI is the right investment to protect people's privacy at scale **and can effectively enforce purpose limitation requirements.**  **Source : [Link 6]** **Note:** *The mapped citation describe an executing privacy checks to ensure that the viewer can see the video entities and applying Policy Zones across infrastructure to address restrictions on data and limit the purposes of its processing. It further define checking multiple requirements involved in data flows and controlling data access and how data is processed and transferred in real time. However, it does not disclose about recursively nested containers or determining traversal based on the most restrictive applicable policy.*
12. The traversal framework further enforces cycle detection, maximum traversal-depth restrictions, and real-time rights verification.	N/A
13. A rights management mechanism configured to dynamically compose video content from independently owned content libraries using container-specific	Recommends content that fulfills people's interests (i.e., short-term, long-term, and real-time) while also allowing for novel content and discovery of content that is outside the person' historical engagement. The architecture supports flexibility for various optimization functions and value modeling, and it builds on delivery **systems that allow for tight latency budgets, rapid modeling deployments, and bias towards fresh content. (We classify freshness as how long ago ranking**

authorization and maintain an audit record of traversal events.	**generated this video candidate.** We generally consider fresher content to be better, since it operates on more recent and relevant signals. … In addition to UI unification, another significant challenge is building a ranking system that can recommend a mixed inventory of Reels and **VOD content while catering to both short-form video-heavy and long-form video-heavy users.** The ranking team has made tremendous progress in this regard, starting with the unification of our data, infrastructure, and algorithmic foundations across the Reels and Watch stacks. **They followed with the creation of a unified content pool that includes both short-form and long-form videos, enabling greater content liquidity. The team then optimized the recommendation machine learning (ML) models to surface the most relevant content without video length bias, ensuring a seamless transition for users with different product affinities (e.g., Reels heavy versus Watch heavy) to a unified content-recommendation experience.** At the most basic level, ranking gives the server a list of video IDs and ranking metadata for each one. **The server then needs to load the video entities from our TAO database and execute privacy checks to ensure that the viewer can see these videos.** The logic of these privacy checks is defined on the server, so ranking can't execute these privacy checks, although ranking has some heuristics to reduce the prevalence of recommending videos that the viewer won't be able to see anyway. **Then the video is passed back to the GraphQL framework, which materializes the fields that the client's query originally asked for. These two steps, privacy checking and materialization, together and in aggregate constitute a meaningful portion of global CPU usage in our data centers, so optimizing here is a significant focus area to alleviate our data center demand and power consumption.** **Source : [Link 3]** **Note:** *The mapped citation describe an unified content pool that includes both short-form and long-form videos, enabling greater content liquidity, and ranking that provides a list of video IDs and ranking metadata, followed by loading video entities and executing privacy checks to ensure that the viewer can see the videos.*
14. The rights management mechanism further generates financial settlement information for respective content owners based on traversed branches.	Facebook **content monetisation** **Simplify your earnings, monetise more content.** It is easier than ever to **earn money across the range of content formats** on Facebook. You can **earn money on Reels, photos, stories and text posts.** Try different formats to see what resonates with your audience and drives earnings… **Creators receive the greatest distribution when they primarily post content that is filmed or created by them.** Original content contains your unique voice and demonstrates the unique perspective you bring to Facebook. Original content is preferred because it is distinctive and engaging…



Source : [Link 7]

Note: *The mapped citation describe a monetizing content across Reels, photos, stories, and text posts, with creators earning money from their content, and states that creators receive the greatest distribution when they primarily post content that is filmed or created by them. However, it does not disclose generating financial settlement information for respective content owners based on traversed branches.*

6. Additional Citations

6.1 Additional Patent Citations

S No.	Publication no.	Title	Filing Date	Publication Date	Assignee(s)
1	**US8860882B2**	Systems and methods for constructing multimedia content modules	19 SEP 2012	14 OCT 2014	JBF Interlude 2009 Ltd
2	**US9591383B2**	Method, device, and computer program for encapsulating partitioned timed media data	17 JAN 2014	07 MAR 2017	Canon Inc
3	**US8488943B1**	Trimming media content without transcoding	31 JAN 2012	16 JUL 2013	Google LLC
4	**US20180242028A1**	Forming A Tiled Video On The Basis Of Media Streams	19 AUG 2016	23 AUG 2018	Royal KPN NV
5	**US20080036917A1**	Methods and systems for generating and delivering navigatable composite videos	09 APR 2007	14 FEB 2008	Adeia Media Solutions Inc.
6	**US11509878B2**	Methods and apparatus for using track derivations for network based media processing	11 SEP 2019	22 NOV 2022	MediaTek Singapore Pte Ltd
7	**WO2016155800A1**	Accessing content	31 MAR 2015	06 OCT 2016	Irdeto BV

6.2 Additional Non-patent Citations

S No.	Title	Authors	Publication Date	Affiliations	Source Link
1	Unified Streaming	NA	07 MAR 2014	Unified Streaming	**Link**
2	AWS Elemental MediaTailor Documentation	NA	11 AUG 2020	Amazon Web Services, Inc.	**Link**
3	Dynamic Delivery	NA	09 DEC 2022	Brightcove Inc.	**Link**

7. Search Details

7.1 Keywords and Related Terms

Video, Media, Multimedia, Audio, Audiovisual, Videotape, Film, Movie, Song, Clip
Cryptographic, Encrypted, Signed, Hashed, Authenticated, Coding, Cryptograph
Verification, Validation, Checking, Testing
Authentication Rules, Authentication Policy, Policy-Based Authentication, Security Policy, Trust Policy, Verification Policy, Authorization Policy, Policy Definition, Policy Specification
Media Reference Container, Virtual Container, Video Reference Object, Media Container Reference, Referenced Media Container, Container Proxy, Media Descriptor, Video Descriptor, Virtual Media Container, Reference-Based Container
Container Metadata, Metadata-Only Representation, File Container Metadata, Media Container Structure, Container Information, Multimedia Container Metadata, Container Descriptors, Container Headers, References To Media Data, Timestamps, Locations Of Samples, The Media Structure, Moov Box, Offset Box, STCO, CO64, STSC, STSZ, STTS, CTTS
Virtualized Video, Virtual Media, Virtual Video Content, Abstract Video, Referenced Video, Proxy Video, Logical Video, Simulated Video, Virtual Media Asset, Lightweight Video Representation
Mdat, Media Data Box, Media Data Atom, Media Payload, Media Payload Section, Encoded Media Data, Sample Data Section, Media Data Area
Temporal Position, Timestamp, Presentation Timestamp (Pts), Decode Timestamp (Dts), Playback Position, Frame Number, Presentation Order, Temporal Index, Sample Time
Video Playback, Media Playback, Content Playback, Media Rendering, Video Rendering, Stream Playback, Media Presentation, Content Presentation, Video Output, Real-Time Media Playback
Graph-Controlled Video Assembly, Traversal-Based Video Composition, Dynamic Video Assembly, Path-Controlled Video Assembly, Graph-Directed Video Streaming, Traversal-Driven Media Assembly, Dynamic Media Composition, Route-Based Video Assembly, Adaptive Video Composition, Conditional Video Assembly
Raw Media Payload Data, Media Content Data, Media Payload Information, Underlying Media Data, Raw Media Content, Media Payload Content
Traversal Reference, Media Reference, Content Reference, Video Link, Navigation Reference
Recursively Nested Graph, Recursive Content Graph, Hierarchical Media Graph, Nested Video Graph, Recursive Traversal Graph

7.2 Patent Search Strings

S No.	Patseer Search Strings
1	TACD:((VIDEO OR AUDIOVISUAL OR “DIGITAL VIDEO” OR MULTIMEDIA OR STREAM*) AND ((VIRTUAL OR MEDIA OR VIDEO OR METADATA) W2 CONTAINER) AND (“MEDIA INFORMATION” OR METADATA OR MDAT OR TIMING OR DURATION OR STBL OR “STRUCTURAL DATA”) AND ((CONTINUOUS OR SEAMLESS OR UNINTERRUPTED OR PERSISTENT OR "REAL-TIME") W5 (PLAYBACK OR STREAM* OR PLAY* OR CONTENT)) AND (((TRAVERSAL OR NESTED OR RECURSIVE* OR HIERARCHICAL) W3 GRAPH) OR BRANCH*) AND ((TRAVERSAL OR MEDIA OR CONTENT OR VIDEO OR NAVIGATION) W3 (REFERENCE OR LINK)) AND ((CRYPTOGRAPHIC OR ACCESS OR AUTHORIZ* OR DRM) W3 (KEY OR TOKEN)) AND (ENCRYPT* OR DECRYPT* OR PROTECT*) AND (“PRE-RENDERED” OR “PRE-GENERATED” OR PREPROCESSED OR (STORED W1 RENDERED W1 CONTENT) OR PREASSEMBLED OR PRECOMPUTED))
2	TACD:((VIDEO OR AUDIOVISUAL OR “DIGITAL VIDEO” OR MULTIMEDIA OR STREAM*) AND ((VIRTUAL OR MEDIA OR VIDEO OR METADATA) W2 CONTAINER) AND (((TRAVERSAL OR NESTED OR RECURSIVE* OR HIERARCHICAL) W3 GRAPH) OR BRANCHING OR TREE) AND ((RUNTIME OR DYNAMIC OR “REAL-TIME”) W2 SELECTION))
3	TACD:(((REMOV* OR STRIP* OR OMIT* OR EXCLUDE OR DETACH OR ELIMINAT*) W5 ((RAW OR MEDIA OR PAYLOAD) WD2 (MEDIA OR CONTENT OR SEGMENT*))) AND (VIDEO OR AUDIOVISUAL OR MULTIMEDIA OR STREAM*) AND ((VIRTUAL OR MEDIA OR VIDEO OR METADATA) W2 CONTAINER) AND (((TRAVERSAL OR NESTED OR RECURSIVE* OR HIERARCHICAL) W3 GRAPH) OR BRANCHING OR TREE) AND ((CONTINUOUS OR SEAMLESS OR UNINTERRUPTED OR PERSISTENT OR "REAL-TIME") W5 (PLAYBACK OR STREAM* OR PLAY* OR CONTENT)))
4	AC:(H04N7/1675 OR G11B27/10 OR H04N21/8541 OR H04N21/8456) AND TACD:((VIDEO OR AUDIOVISUAL OR “DIGITAL VIDEO” OR MULTIMEDIA OR STREAM*) AND ((VIRTUAL OR MEDIA OR VIDEO OR METADATA) W2 CONTAINER) AND (“MEDIA INFORMATION” OR METADATA OR MDAT OR TIMING OR DURATION OR STBL OR “STRUCTURAL DATA”) AND ((CONTINUOUS OR SEAMLESS OR UNINTERRUPTED OR PERSISTENT OR REAL-TIME) W5 (PLAYBACK OR STREAM* OR PLAY* OR CONTENT)) AND (((TRAVERSAL OR NESTED OR RECURSIVE* OR HIERARCHICAL) W3 GRAPH) OR BRANCHING))
5	ALLCTOF(PNC:US9236091B2 OR US2015063781A1)
6	AASN:(NETFLIX OR ONMED OR (CONVERGENT W2 SOLUTIONS) OR (FRAUNHOFER W2 FOERDERUNG W2 ANGEWANDTEN W2 FORSCHUNG) OR DIVX OR (JBF W2 INTERLUDE) OR DISNEY OR (GUANGZHOU W2 TECHNOLOGY)) AND TACD:((VIDEO OR AUDIOVISUAL OR "DIGITAL VIDEO" OR MULTIMEDIA OR STREAM*) AND ((VIRTUAL OR MEDIA OR VIDEO OR METADATA) W2 CONTAINER) AND (((TRAVERSAL OR NESTED OR RECURSIVE* OR HIERARCHICAL) W3 GRAPH) OR BRANCHING OR TREE))
7	TACD:((VIRTUALIZATION OR EMULATION OR SIMULATION) AND ((PLURALITY OR VARIOUS) WP (VIDEO OR CLIP OR FOOTAGE))) AND TACD:(((VIDEO OR CLIP OR FOOTAGE) WP (REFERENCE OR ADDRESS OR LOCATOR)) AND (CONTAINER OR WRAPPER OR PACKAGE)) AND TACD:(((REMOVE OR CLEAR OR DELETE) WP (RAW OR UNSTRUCTURED)) AND ("MEDIA SAMPLE PAYLOAD DATA" OR MDAT))
8	TACD: ((PLURALITY OR VARIOUS OR MULTIPLE OR MULTI) AND (VIDE* OR CLIP* OR FOOTAG* OR RECORD*) AND (REFERENCE OR ADDRESS OR LOCATOR) AND (CONTAINER OR WRAPPER OR PACKAGE) AND (REMOVE OR CLEAR OR DELETE) AND (RAW OR UNSTRUCTURED) AND ("MEDIA SAMPLE PAYLOAD DATA" OR MDAT) AND (STTS OR SBTL OR "SAMPLE TABLE BOX" OR "SAMPLE TIMING") AND (TRAVERSAL OR PROGRESSION) AND (GRAPH OR TREE OR TOPOLOGY OR NETWORK))
9	TACD: ((((VIRTUALIZATION OR ABSTRACTION OR EMULATION OR SIMULATION) AND ((VIRTUAL* OR DIGITAL*) WD ((VIDEO* OR FOOTAGE* OR CLIP* OR SCENE*) W5 (REFERENCE OR ADDRESS OR LOCATOR))))) AND (REMOVE OR CLEAR OR CLEAN) AND (MDAT OR “PAYLOAD DATA” OR “MEDIA SAMPLE PAYLOAD DATA” OR RAW_DATA OR UNSTRUCTURED_DATA) AND (STOR* OR SAV* OR REPOST*

	OR ARCHIVE) AND (STBL OR “SAMPLE TIMING METADATA” OR STTS OR “SAMPLE TABLE BOX” OR TIMESTAMP OR DURATION))
10	TACD:((VIDEO OR MEDIA) AND (REMOVE OR REMOVING OR OMIT OR OMISSION OR STRIP OR STRIPPED) AND (MDAT OR "MEDIA DATA") AND (RETAIN OR RETAINING OR PRESERVE OR PRESERVED) AND (STBL OR "SAMPLE TABLE" OR "SAMPLE TIMING") AND (CONTAINER OR FILE))
11	TACD:((VIDEO OR MEDIA) AND (TRAVERSAL OR NAVIGATION OR RESOLUTION) AND (TOKEN OR "CRYPTOGRAPHIC TOKEN" OR AUTHORIZATION OR AUTHORISATION) AND (VALIDATE OR VALIDATION OR VERIFY OR VERIFICATION) AND (CONTAINER OR SEGMENT OR OBJECT)) AND ("REAL-TIME SIGNAL*" OR RUNTIME) AND (("VIDEO CONTAINER") OR ("VIRTUAL CONTAINER"))
12	TAC:((VIDEO OR MEDIA) AND (PLAYBACK OR ASSEMBLY OR STREAMING) AND (CONTAINER OR OBJECT) AND (MDAT OR "MEDIA DATA")AND (STBL OR "SAMPLE TABLE")) AND PBD:[2004-01-01 TO 2026-10-07]
13	TAC:(VIDEO OR MEDIA) AND (CONTAINER OR SEGMENT OR FRAGMENT) AND ("TRAVERSAL REFERENCE" OR "REFERENCE LINK" OR "CONTAINER REFERENCE" OR "MEDIA REFERENCE") AND (RECURSIVE OR NESTED OR HIERARCHICAL) AND PBD:[2004-01-01 TO 2026-10-07]
14	TAC:(VIDEO OR MEDIA) AND(CONTAINER OR SEGMENT OR FRAGMENT) AND (TRAVERSAL OR BRANCHING OR "GRAPH TRAVERSAL") AND (AUTHORIZATION OR LICENSING OR DRM OR "RIGHTS MANAGEMENT") AND (PREFETCH OR CACHING OR "DECRYPTION KEY") AND ("DYNAMIC" OR RUNTIME OR "REAL-TIME") AND PBD:[2004-01-01 TO 2026-10-07]
15	TACD:((VIDEO OR AUDIO OR FILM OR SCENE OR SCENIC OR PICTURE OR MEDIA OR TV OR TELEVISION OR STREAMING) W5 (PART* OR SEGMENT* OR CUTS OR PIECES OR BIFERCAT*)) AND TACD:(((SELECTIVE OR CUSTOMIZED OR SELECTED) W5 (PLAYBACK OR RENDER* OR BUFFER*)) AND AUTHENTICAT* AND (VIDEO OR AUDIO OR FILM OR SCENE OR SCENIC OR PICTURE OR MEDIA OR TV OR TELEVISION OR STREAMING) AND ((REFERENCE OR VIRTUAL OR ARTIFICIAL) W5 (CONTAINER OR STORAGE OR STORING)) AND TRAVERSAL)
16	INV:((AKINORI TAKEO) OR (ANDREW LEE JENKS) OR (BARAK FELDMAN) OR (DEEPALI ARYA) OR (ELENA KVOCHKO) OR (ERIC HORVITZ) OR (ERIC HSUMING CHEN) OR (GAVIN STUART PETER MILLER) OR (HARALD FUCHS) OR (HUNG-JU LEE) OR (JASON N. WANG) OR (JONATHAN BLOCH) OR (JOSEPH PEYTON LEA) OR (JUNRUI LU) OR (KEVIN PEI-WEN LEE) OR (KFIR Y. ROTBARD) OR (KOLTUN GERSHI) OR (LAURA BUFFARD) OR (LAWRENCE E. JONES) OR (LEON SILVERMAN) OR (LIU YONGLIANG) OR (MALVAR HENRIQUE) OR (MARK ARANA) OR (MARK BAKER) OR (MAX GRAY EDELL) OR (MICHAEL ROBERT BLOOD) OR (NIKLAS RYSTEDT) OR (NIKOLAUS FAERBER) OR (RATHISH KRISHNAN) OR (REBECCA NICOLE BURKE-AGUERO) OR (RICHARD KELLEY) OR (SAMUEL JOHN WENKER) OR (STEFAN DOEHLA) OR (STEFAN KRAEGELOH) OR (TAL ZUBALSKY) OR (TONISSON ALAN VALEV) OR (XINYI HAN) OR (XIONG XIAOLI) OR (XUEJUN XU) OR (YAFFE YOEL) OR (YAN HUANG) OR (YUVAL HOFSHY)) AND TACD:((SELECTIVE OR CUSTOMIZED OR SELECTED) W5 (PLAYBACK OR RENDER* OR BUFFER*))
17	AASN:((ACRONIS INTERNATIONAL) OR (ADOBE) OR (ALIBABA) OR (AMAZON) OR (APPLE) OR (AT&T) OR (BANK OF AMERICA) OR (BISON PATENT LICENSING) OR (CANON) OR (CISCO) OR (CONVERGENT MEDIA SOLUTIONS) OR (CYBERCUBE) OR (DIGITAL TRANSACTION) OR (DISNEY ENTERPRISES) OR (DIVX) OR (EZDRM) OR (FRANCE TELECOM) OR (FUTUREWEI TECHNOLOGIES) OR (GUANGZHOU HUYA TECHNOLOGY) OR (HUAWEI) OR (IBM) OR (ILUMINTEL) OR (INSTART LOGIC) OR (INTEL) OR (IRDETO) OR (JBF INTERLUDE) OR (KOREA ELECTRONICS TECHNOLOGY INSTITUTE) OR (LG) OR (MEDIATEK) OR (MICROSOFT) OR (NAGRAVISION SARL) OR (NETFLIX) OR (NOKIA) OR (NVIDIA) OR (NXP) OR (ONMED) OR (ORANGE) OR (PLS IV) OR (SAMSUNG) OR (SONY) OR (SWEAR) OR (TRUEPIC) OR (TRUFO) OR (TYCO FIRE AND SECURITY) OR (UNVEILED LABS)) AND TACD:(((SELECTIVE OR CUSTOMIZED OR SELECTED) W5 (PLAYBACK OR RENDER* OR BUFFER*)) AND AUTHENTICAT* AND (VIDEO OR AUDIO OR FILM OR SCENE OR SCENIC OR PICTURE OR MEDIA OR TV OR TELEVISION OR STREAMING) AND ((REFERENCE OR VIRTUAL OR ARTIFICIAL) W5 (CONTAINER OR STORAGE OR STORING)) AND TRAVERSAL)

S No.	USPTO Search Strings
1	(H04N21/8541) AND ((VIDEO OR MEDIA) NEAR3 (GRAPH OR TREE OR BRANCH)) AND ((PREFETCH OR PREBUFFER OR DOWNLOAD) NEAR3 (SEGMENT OR VIDEO OR MEDIA)) AND ((SELECTION OR CHOICE OR DECISION) NEAR3 (USER)) AND ((PLAYBACK OR STREAM OR ASSEMBLE) NEAR3 (VIDEO OR MEDIA))
2	(H04N21/8541 OR H04N21/47202 OR H04N21/4627) AND ((VIDEO OR MEDIA OR MULTIMEDIA) NEAR3 (CONTAINER OR SEGMENT OR SAMPLE)) AND ((VIRTUAL OR REFERENCE OR EXTERNAL OR REMOTE) NEAR3 (VIDEO OR MEDIA OR CONTAINER OR SAMPLE)) AND ((GRAPH OR TREE OR BRANCH OR TRAVERSAL) NEAR3 (VIDEO OR MEDIA OR CONTAINER OR SEGMENT)) AND ((SELECT OR SELECTION OR CHOICE OR DECISION OR INPUT) NEAR3 (USER)) AND ((TOKEN OR LICENSE OR RIGHTS OR AUTHORIZATION OR CRYPTOGRAPHIC) NEAR3 (VIDEO OR MEDIA OR SEGMENT)) AND ((PREFETCH OR PREBUFFER OR BUFFER OR PLAYBACK OR ASSEMBLE OR STREAM) NEAR3 (VIDEO OR MEDIA OR SEGMENT))
3	(VIDEO OR MEDIA) AND (VIRTUALIZED OR VIRTUALIZATION OR REFERENCE OR LIGHTWEIGHT) AND (GRAPH OR TRAVERSAL OR NAVIGATION) AND (PLAYBACK OR ASSEMBLY OR STREAMING) AND (CONTAINER OR OBJECT) AND (MDAT OR "MEDIA DATA") AND (STBL OR "SAMPLE TABLE")
4	((VIDEO OR MEDIA) AND (VIRTUAL CONTAINER OR VIRTUALIZED CONTAINER OR REFERENCE CONTAINER) AND (GRAPH OR "TRAVERSABLE GRAPH" OR TRAVERSAL) AND (CONTAINER OR OBJECT) AND (PLAYBACK OR STREAMING) AND (REFERENCE OR VIRTUALIZED OR LIGHTWEIGHT)) AND (STBL OR "SAMPLE TABLE" OR "SAMPLE TIMING") AND (CONTAINER OR FILE)
5	("VIRTUAL VIDEO" OR "VIRTUAL MEDIA") AND (MDAT OR "MEDIA DATA" OR "MEDIA PAYLOAD") AND ("EXTERNAL MEDIA" OR "EXTERNALLY STORED MEDIA" OR "EXTERNAL MEDIA SAMPLES") AND @AY>="2004"
6	(VIDEO OR MEDIA) AND (CONTAINER OR SEGMENT OR FRAGMENT) AND (TRAVERSAL OR BRANCHING OR "GRAPH TRAVERSAL") AND (AUTHORIZATION OR LICENSING OR DRM OR "RIGHTS MANAGEMENT") AND (PREFETCH OR CACHING OR "DECRYPTION KEY") AND ("DYNAMIC" OR RUNTIME OR "REAL-TIME") AND ((H04W12/06).CPC.) AND @AY>="2004"
7	(VIDEO OR MULTIMEDIA) AND (CONTAINER OR FILE)AND (MDAT OR "MEDIA DATA") AND (STBL OR "SAMPLE TABLE") AND ("METADATA ONLY" OR "WITHOUT MEDIA DATA" OR "MEDIA PAYLOAD REMOVED") AND @AY>="2004"
S No.	**WIPO Search Strings**
1	EN_ALLTXT:((VIDEO OR MEDIA OR MULTIMEDIA) NEAR3 (CONTAINER OR SEGMENT OR SAMPLE)) AND (USER NEAR3 (INPUT OR CHOICE OR SELECTION)) AND ((PLAY OR FETCH OR RENDER OR BUFFER) NEAR3 (VIDEO OR SEGMENT OR SAMPLE)) AND ((AUTHORIZATION OR AUTHORISATION OR ENCRYPTION) NEAR3 (KEY OR TOKEN)) AND CLASSIF: ((H04N21/4627 OR H04L2209/603) AND (H04N21/47202))
2	EN_ALLTXT:((VIDEO OR MEDIA) NEAR2 (VIRTUAL OR REFERENCE OR CONTAINER)) AND (("STBL" OR "SAMPLE TABLE" OR METADATA) NEAR3 (TIMING OR STRUCTURE OR REFERENCE OR TIMESTAMP)) AND (("MDAT" OR "MEDIA DATA" OR PAYLOAD OR SAMPLE) NEAR3 (EXTERNAL OR REMOTE OR SEPARATE)) AND ((GRAPH OR TREE OR BRANCH OR TRAVERSAL) NEAR3 (CONTAINER OR SEGMENT OR VIDEO)) AND ((VALIDATION OR ENCRYPTION OR AUTHORIZATION OR CYPTOGRAPHIC) NEAR3 (KEY OR TOKEN))
3	EN_ALLTXT:((VIDEO OR MEDIA OR MULTIMEDIA) NEAR3 (CONTAINER OR SEGMENT OR SAMPLE)) AND ((VIRTUAL OR REFERENCE OR EXTERNAL OR REMOTE) NEAR3 (VIDEO OR MEDIA OR CONTAINER OR "MEDIA DATA" OR "MDAT")) AND ((GRAPH OR TREE OR BRANCH OR TRAVERSAL) NEAR3 (VIDEO OR MEDIA OR CONTAINER OR SEGMENT)) AND ((SELECTION OR CHOICE OR DECISION OR INPUT) NEAR3 (USER)) AND ((TOKEN OR LICENSE OR RIGHTS OR AUTHORIZATION OR CRYPTOGRAPHIC) NEAR3 (VIDEO

	OR MEDIA OR CONTAINER OR SEGMENT OR PLAYBACK)) AND ((PREFETCH OR PREBUFFER OR BUFFER OR PLAYBACK OR ASSEMBLY OR STREAM) NEAR3 (VIDEO OR MEDIA OR SEGMENT OR SAMPLE)) AND CLASSIF:H04N21/8541
S No.	**J-PlatPat Search Strings**
1	[5C164GA08+5C164YA12/FT]*[TRAVERSAL/TX]
2	[5C164MC01/FT]
3	[5C164MB41+5C164MB42+5C164MB44/FT]
S No.	**KIPRIS Search Strings**
1	(VIRTUAL * (VIDEO + MEDIA)) * (CONTAINER + MANIFEST) * (TRAVERSAL + ROUTING) * (GRAPH + NETWORK)
2	(VIDEO + MEDIA) * (CONTAINER + MANIFEST) * (TRAVERSAL + ROUTINE)
S No.	**FPO Search Strings**
1	SPEC/ ((VIRTUALIZATION OR EMULATION OR SIMULATION) AND (PLURALITY OR VARIOUS OR MULTIPLE) (VIDEO OR CLIP OR FOOTAGE) AND (REFERENCE OR LINK OR ADDRESS) AND (REMOV* OR DELETE*) AND (RAW OR UNSTRUCTURED) AND (MDAT OR "RAW MEDIA SAMPLE PAYLOAD DATA") AND (RENDERED OR FINALIZED OR COMPLIED) AND ("SAMPLE TIMING METADATA" OR STBL))
2	ACLM/ (((VIRTUAL* OR ARTIFICIAL) AND (MEDIA OR VIDEO) AND (CONTAINER* OR FILE* OR REPRESENTATION*)) AND ((SAMPLE* OR FRAME* OR "MEDIA DATA" OR PAYLOAD) AND (METADATA OR TIMING OR DURATION OR OFFSET*)) AND ((RAW OR ACTUAL OR UNDERLYING OR ORIGINAL) AND (MEDIA OR SAMPLE* OR PAYLOAD OR DATA)) AND ((REMOVE* OR OMIT* OR DETACH* OR REFERENCE* OR VIRTUALIZE*) AND (MEDIA OR SAMPLE* OR PAYLOAD OR DATA)))
3	ACLM/ (((VIDEO OR MEDIA OR MULTIMEDIA OR CONTENT) AND (GRAPH OR TOPOLOGY)) AND ((NODE* OR CONTAINER* OR SEGMENT* OR CLIP*) AND (REFERENCE* OR POINTER* OR LINK* OR EDGE* OR TRANSITION*)) AND ((TRAVERS* OR NAVIGAT* OR BRANCH*)) AND ((RUNTIME OR "REAL TIME" OR DYNAMIC) AND (ROUTING OR TRAVERSAL OR PLAYBACK)) AND (TOKEN AND CRYPTO*))
4	ACLM/ (((VIDEO OR MEDIA OR CONTENT) AND (SEGMENT* OR CLIP* OR CONTAINER* OR REPRESENTATION*)) AND ((DYNAMIC OR RUNTIME OR REAL-TIME OR CONDITIONAL OR CONTEXTUAL OR ADAPTIVE) AND (CONDITION* OR SIGNAL* OR PARAMETER* OR INPUT* OR EVENT*)) AND ((SELECT* OR DETERMIN* OR IDENTIFY* OR CHOOSE* OR ROUTE* OR DIRECT*) AND (TARGET OR CANDIDATE OR SEGMENT* OR CONTAINER* OR MEDIA*)) AND ((TRAVERSAL OR BRANCH* OR TRANSITION*) AND (VIDEO OR MEDIA OR PLAYBACK)) AND (TOKEN AND CRYPTO*))
5	(VIDEO PLAYBACK OR VIDEO STREAMING OR MULTIMEDIA) AND ("DYNAMIC BRANCH" OR BRANCHING OR "BRANCH CONDITION") AND ("REAL-TIME SIGNAL" OR RUNTIME) AND ("VIDEO CONTAINER") OR ("VIRTUAL CONTAINER") AND APD/1/1/2004->10/07/2026 AND IPC/H04W12/06
6	(VIDEO OR MEDIA) AND (VIRTUALIZED OR REFERENCE OR LIGHTWEIGHT) AND (GRAPH OR TRAVERSAL OR NAVIGATION) AND (PLAYBACK OR ASSEMBLY OR STREAMING) AND (CONTAINER OR OBJECT) AND (MDAT OR "MEDIA DATA")AND (STBL OR "SAMPLE TABLE") AND APD/1/1/2004->10/07/2026
7	((VIDEO OR MEDIA) AND (REMOVE OR REMOVING OR OMIT OR OMISSION OR STRIP OR STRIPPED) AND (MDAT OR "MEDIA DATA") AND (RETAIN OR RETAINING OR PRESERVE OR PRESERVED) AND (STBL OR "SAMPLE TABLE" OR "SAMPLE TIMING") AND (CONTAINER OR FILE)) AND APD/1/1/2004->10/07/2026

8	(VIDEO OR MEDIA) AND (CONTAINER OR SEGMENT OR FRAGMENT) AND ("TRAVERSAL REFERENCE" OR "REFERENCE LINK" OR "CONTAINER REFERENCE" OR "MEDIA REFERENCE") AND (RECURSIVE OR NESTED OR HIERARCHICAL) AND APD/1/1/2004->10/07/2026
S No.	**Espacenet Search Strings**
1	(FTXT ANY "VIRTUALIZATION" AND (FTXT ANY "PLURALITY" OR FTXT ANY "VARIOUS") AND (FTXT ANY "VIDEO" OR FTXT ANY "CLIP" OR FTXT ANY "FOOTAGE")) AND ((FTXT ANY "REFERENCE" OR FTXT ANY "ADDRESS") AND (FTXT ANY "REMOVE" OR FTXT ANY "UNSTRUCTURED") AND ((FTXT ANY "VIDEO" OR FTXT ANY "CLIP" OR FTXT ANY "FOOTAGE") AND (FTXT=(" TIMING " PROX/DISTANCE<3 "METADATA") OR FTXT ANY "SBTL"))) AND ((FTXT ANY "TRAVERSAL" OR FTXT ANY "PROGRESSION") AND (FTXT ANY "GRAPH" OR FTXT ANY "TREE" OR FTXT ANY "TOPOLOGY")) AND ((FTXT ANY "USER" OR FTXT ANY "CLIENT" OR FTXT ANY "HOST") AND (FTXT ANY "INPUT" OR FTXT ANY "INSTRUCTION" OR FTXT ANY "SELECTION"))
2	((FTXT ANY "VIRTUAL" OR FTXT ANY "DIGITAL") AND (FTXT ANY "VIDEO" OR FTXT ANY "CLIP" OR FTXT ANY "FOOTAGE" OR FTXT ANY "SCENE") AND (FTXT ANY "REFERENCE" OR FTXT ANY "ADDRESS" OR FTXT ANY "LOCATOR") AND (FTXT ANY "CONTAINER" OR FTXT ANY "PACKAGE")) AND ((FTXT ANY "REMOVE" OR FTXT ANY "CLEAN") AND (FTXT ANY "MDAT" OR FTXT=("PAYLOAD " PROX/DISTANCE<3 "DATA") OR FTXT=("SAMPLE" PROX/DISTANCE<3 "DATA"))) AND ((FTXT ANY "DURATION" OR FTXT ANY "TIMESTAMP" OR FTXT=("SAMPLE" PROX/DISTANCE<3 "BOX") OR FTXT ANY "STTS" OR FTXT ANY "STBL") AND (FTXT ANY "STORE" OR FTXT ANY "REPOSITORY" OR FTXT ANY "ARCHIVE")) AND ((FTXT ANY "GRAPH" OR FTXT ANY "TOPOLOGY" OR FTXT ANY "TREE") AND FTXT ANY "TRAVERSAL") AND ((FTXT ANY "VIDEO" OR FTXT ANY "SCENE" OR FTXT ANY "CLIP" OR FTXT ANY "FOOTAGE" OR FTXT ANY "PORTION") AND (FTXT ANY "REFERENCE" OR FTXT ANY "LOCATOR" OR FTXT ANY "ADDRESS") AND (FTXT ANY "INTERCONNECTED" OR FTXT ANY "NESTED" OR FTXT ANY "LINKED")) AND ((FTXT ANY "AUTHORIZATION" OR FTXT ANY "VALIDATE") AND (FTXT ANY "CRYPTOGRAPHY" OR FTXT ANY "CRYPTO"))
3	(FTXT=("VIRTUAL " PROX/DISTANCE<3 "REFERENCE_CONTAINER*") OR FTXT=("VIRTUAL " PROX/DISTANCE<3 "VIDEO_CONTAINER*") OR FTXT=("VIRTUALIZED " PROX/DISTANCE<3 "VIDEO_CONTAINER*") OR FTXT=("VIDEO " PROX/DISTANCE<3 "REFERENCE_CONTAINER*") OR FTXT=("MEDIA " PROX/DISTANCE<3 "REFERENCE_CONTAINER*") OR FTXT=("MEDIA " PROX/DISTANCE<3 "CONTAINER*")) AND (FTXT=("MDAT" PROX/DISTANCE<3 "REMOV*") OR FTXT=("MEDIA " PROX/DISTANCE<3 "PAYLOAD") OR FTXT=("MEDIA " PROX/DISTANCE<3 "PAYLOAD_REMOVAL") OR FTXT=("MEDIA " PROX/DISTANCE<3 "PAYLOAD*")) AND (FTXT=("STBL" PROX/DISTANCE<3 "RETAIN*") OR FTXT=("SAMPLE " PROX/DISTANCE<3 "TABLE*") OR FTXT=("SAMPLE " PROX/DISTANCE<3 "METADATA") OR FTXT=("STRUCTURAL " PROX/DISTANCE<3 "METADATA") OR FTXT=("TIMING " PROX/DISTANCE<3 "METADATA"))
4	(FTXT=("VIRTUAL " PROX/DISTANCE<3 "REFERENCE_CONTAINER*") OR FTXT=("VIRTUAL " PROX/DISTANCE<3 "VIDEO_CONTAINER*") OR FTXT=("DIGITAL " PROX/DISTANCE<3 "CONTAINER*") OR FTXT=("MEDIA " PROX/DISTANCE<3 "REFERENCE_CONTAINER") OR FTXT=("VIRTUALIZED" PROX/DISTANCE<3 "CONTAINER")) AND (FTXT=("MEDIA " PROX/DISTANCE<3 "GRAPH ") OR FTXT=("TRAVERSABLE" PROX/DISTANCE<3 "GRAPH ") OR FTXT=("LINKED " PROX/DISTANCE<3 "MEDIA_CONATINER") OR FTXT=("INTERCONNECTED " PROX/DISTANCE<3 "CONTAINER "))
5	(FTXT=("VIRTUAL " PROX/DISTANCE<3 "REFERENCE_CONTAINER*") OR FTXT=("VIRTUAL " PROX/DISTANCE<3 "VIDEO_CONTAINER*") OR FTXT=("DIGITAL " PROX/DISTANCE<3 "CONTAINER*") OR FTXT=("MEDIA " PROX/DISTANCE<3 "REFERENCE_CONTAINER") OR FTXT=("VIRTUALIZED" PROX/DISTANCE<3 "CONTAINER") OR FTXT=("MEDIA " PROX/DISTANCE<3 "PAYLOAD_REMOVAL ") OR FTXT=("MDAT" PROX/DISTANCE<3 "REMOVAL ")) AND (FTXT=("SAMPLE " PROX/DISTANCE<3 "TIM*") OR FTXT=("STRUCTURAL " PROX/DISTANCE<3 "METADATA ") OR FTXT=("STRUCTURAL " PROX/DISTANCE<3 "CHARACTERISTICS") OR FTXT=("STRUCTURAL " PROX/DISTANCE<3 "PROPERTIES") OR FTXT=("FORMAT " PROX/DISTANCE<3 "METADATA ")) AND (FTXT=("GRAPH" PROX/DISTANCE<3 "CONNECTIVITY") OR

	FTXT=("GRAPH" PROX/DISTANCE<3 "COUPLING") OR FTXT=("GRAPH" PROX/DISTANCE<3 "LINKAGE") OR FTXT=("GRAPH" PROX/DISTANCE<3 "ARRANGEMENT") OR FTXT=("INTERCONNECTED " PROX/DISTANCE<3 "GRAPH"))
6	(CLAIMS=("VIRTUAL " PROX/DISTANCE<3 "CONTAINER*") OR CLAIMS=("MEDIA " PROX/DISTANCE<3 "CONTAINER*") OR CLAIMS=("DIGITAL " PROX/DISTANCE<3 "CONTAINER*") OR CLAIMS=("REFERENCE " PROX/DISTANCE<3 "CONTAINER*")) AND (FTXT=("TIM*" PROX/DISTANCE<3 "METADATA") OR FTXT ALL "STBL" OR FTXT=("SAMPLE " PROX/DISTANCE<3 "METADATA ") OR FTXT=("SAMPLE " PROX/DISTANCE<3 "TIM*") OR FTXT=("STRUCTURAL " PROX/DISTANCE<3 "METADATA")) AND (FTXT=("TRAVERSABLE " PROX/DISTANCE<3 "GRAPH ") OR FTXT=("LINKED" PROX/DISTANCE<3 "GRAPH ") OR FTXT=("MEDIA " PROX/DISTANCE<3 "GRAPH ") OR FTXT=("GRAPH " PROX/DISTANCE<3 "COMPOSITION") OR FTXT ALL "INTERCONNECTED" OR FTXT ALL "LINK" OR FTXT ALL "CONNECT") AND (FTXT=("RECURSIVE " PROX/DISTANCE<3 "NESTING ") OR FTXT=("NESTED " PROX/DISTANCE<3 "REFERENCE") OR FTXT=("HIERARCHICAL " PROX/DISTANCE<3 "CONTAINER ") OR FTXT=("CONTAINER " PROX/DISTANCE<3 "NESTING"))
7	(CLAIMS=("VIRTUAL " PROX/DISTANCE<3 "CONTAINER*") OR CLAIMS=("MEDIA " PROX/DISTANCE<3 "CONTAINER*") OR CLAIMS=("DIGITAL " PROX/DISTANCE<3 "CONTAINER*") OR CLAIMS=("REFERENCE " PROX/DISTANCE<3 "CONTAINER*")) AND (FTXT=("TIM*" PROX/DISTANCE<3 "METADATA") OR FTXT ALL "STBL" OR FTXT=("SAMPLE " PROX/DISTANCE<3 "METADATA ") OR FTXT=("SAMPLE " PROX/DISTANCE<3 "TIM*") OR FTXT=("STRUCTURAL " PROX/DISTANCE<3 "METADATA")) AND (FTXT=("RECURSIVE " PROX/DISTANCE<3 "NESTING ") OR FTXT=("NESTED " PROX/DISTANCE<3 "REFERENCE") OR FTXT=("HIERARCHICAL " PROX/DISTANCE<3 "CONTAINER ") OR FTXT=("CONTAINER " PROX/DISTANCE<3 "NESTING"))
8	(CTXT=("STRUCTURAL" PROX/DISTANCE<3 "METADATA") OR CTXT=("SAMPLE " PROX/DISTANCE<3 "TABLE ") OR CTXT=("SAMPLE " PROX/DISTANCE<3 "MEATDATA") OR CTXT=("TIM*" PROX/DISTANCE<3 "METADATA ")) AND (FTXT=("MEDIA " PROX/DISTANCE<3 "GRAPH ") OR FTXT=("CONTENT " PROX/DISTANCE<3 "GRAPH ") OR FTXT=("GRAPH " PROX/DISTANCE<3 "STRUCTURE "))
S No	**Google Patent Search Strings**
1	TAC=(VIDEO OR AUDIOVISUAL OR “DIGITAL VIDEO” OR MULTIMEDIA OR STREAM*) ((VIRTUAL OR MEDIA OR VIDEO OR METADATA) NEAR2 CONTAINER) ((CONTINUOUS OR SEAMLESS OR UNINTERRUPTED OR PERSISTENT OR "REAL-TIME") NEAR5 (PLAYBACK OR STREAM* OR PLAY* OR CONTENT)) TAC=(((TRAVERSAL OR NESTED OR RECURSIVE* OR HIERARCHICAL) NEAR3 GRAPH) OR BRANCHING) ((TRAVERSAL OR MEDIA OR CONTENT OR VIDEO OR NAVIGATION) NEAR3 (REFERENCE OR LINK)) ((CRYPTOGRAPHIC OR ACCESS OR AUTHORIZ* OR DRM) NEAR3 (KEY OR TOKEN)) (“MEDIA INFORMATION” OR METADATA OR MDAT OR TIMING OR DURATION OR STBL OR “STRUCTURAL DATA”) (“PRE-RENDERED” OR “PRE-GENERATED” OR PREPROCESSED OR "STORED RENDERED" OR PREASSEMBLED OR PRECOMPUTED) ((VIDEO OR MEDIA OR CONTENT OR STREAM) NEAR2 (RESOLUTION OR SELECTION)) ((RUNTIME OR DYNAMIC OR REAL-TIME) NEAR2 SELECTION)
2	AB=(VIDEO OR AUDIOVISUAL OR "DIGITAL VIDEO" OR MULTIMEDIA OR STREAM*) AB=((VIRTUAL OR MEDIA OR VIDEO OR METADATA) NEAR2 CONTAINER) ((CONTINUOUS OR SEAMLESS OR UNINTERRUPTED OR PERSISTENT OR "REAL-TIME") NEAR5 (PLAYBACK OR STREAM* OR PLAY* OR CONTENT)) (((TRAVERSAL OR NESTED OR RECURSIVE* OR HIERARCHICAL) NEAR3 GRAPH) OR BRANCHING OR TREE) ((CRYPTOGRAPHIC OR ACCESS OR AUTHORIZ* OR DRM) NEAR3 (KEY OR TOKEN)) (“MEDIA INFORMATION” OR METADATA OR MDAT OR TIMING OR DURATION OR STBL OR “STRUCTURAL DATA”) (“PRE-RENDERED” OR “PRE-GENERATED” OR PREPROCESSED OR "STORED RENDERED" OR PREASSEMBLED OR PRECOMPUTED) ((VIDEO OR MEDIA OR CONTENT OR STREAM) NEAR2 (RESOLUTION OR SELECTION)) ((RUNTIME OR DYNAMIC OR REAL-TIME) NEAR2 SELECTION)
3	TAC=(VIDEO OR AUDIOVISUAL OR "DIGITAL VIDEO" OR MULTIMEDIA OR STREAM*) AB=((VIRTUAL OR MEDIA OR VIDEO OR METADATA) NEAR2 CONTAINER) ((CONTINUOUS OR SEAMLESS OR

	UNINTERRUPTED OR PERSISTENT OR "REAL-TIME") NEAR5 (PLAYBACK OR STREAM* OR PLAY* OR CONTENT)) (((TRAVERSAL OR NESTED OR RECURSIVE* OR HIERARCHICAL) NEAR3 GRAPH) OR BRANCHING OR TREE) ((CRYPTOGRAPHIC OR ACCESS OR AUTHORIZ* OR DRM) NEAR3 (KEY OR TOKEN)) (“MEDIA INFORMATION” OR METADATA OR MDAT OR TIMING OR DURATION OR STBL OR “STRUCTURAL DATA”) (“PRE-RENDERED” OR “PRE-GENERATED” OR PREPROCESSED OR "STORED RENDERED" OR PREASSEMBLED OR PRECOMPUTED) ((VIDEO OR MEDIA OR CONTENT OR STREAM) NEAR2 (RESOLUTION OR SELECTION)) ((RUNTIME OR DYNAMIC OR REAL-TIME) NEAR2 SELECTION) ((GAZE OR EYE) NEAR3 (BASED OR AWARE OR DIRECTED OR RESPONSIVE)) (RESOLUTION OR QUALITY)
4	CPC=(H04N7/1675 OR G11B27/10 OR H04N21/8541 OR H04N21/8456) ((VIRTUAL OR MEDIA OR VIDEO OR METADATA) NEAR2 CONTAINER) ((CONTINUOUS OR SEAMLESS OR UNINTERRUPTED OR PERSISTENT OR "REAL-TIME") NEAR5 (PLAYBACK OR STREAM* OR PLAY* OR CONTENT)) (((TRAVERSAL OR NESTED OR RECURSIVE* OR HIERARCHICAL) NEAR3 GRAPH) OR BRANCHING OR TREE) ((CRYPTOGRAPHIC OR ACCESS OR AUTHORIZ* OR DRM) NEAR3 (KEY OR TOKEN)) (“MEDIA INFORMATION” OR METADATA OR MDAT OR TIMING OR DURATION OR STBL OR “STRUCTURAL DATA”) (“PRE-RENDERED” OR “PRE-GENERATED” OR PREPROCESSED OR "STORED RENDERED" OR PREASSEMBLED OR PRECOMPUTED) ((VIDEO OR MEDIA OR CONTENT OR STREAM) NEAR2 (RESOLUTION OR SELECTION)) ((RUNTIME OR DYNAMIC OR REAL-TIME) NEAR2 SELECTION) ((GAZE OR EYE) NEAR3 (BASED OR AWARE OR DIRECTED OR RESPONSIVE)) (RESOLUTION OR QUALITY) (VIDEO OR AUDIOVISUAL OR "DIGITAL VIDEO" OR MULTIMEDIA OR STREAM*)
5	((REMOV* OR STRIP* OR OMIT* OR EXCLUDE OR DETACH OR ELIMINAT*) NEAR5 ((RAW OR MEDIA OR PAYLOAD) NEAR2 (MEDIA OR CONTENT OR SEGMENT*))) (VIDEO OR AUDIOVISUAL OR “DIGITAL VIDEO” OR MULTIMEDIA OR STREAM*) ((VIRTUAL OR MEDIA OR VIDEO OR METADATA) NEAR2 CONTAINER) TAC=(((TRAVERSAL OR NESTED OR RECURSIVE* OR HIERARCHICAL) NEAR3 GRAPH) OR BRANCHING OR TREE) ((CRYPTOGRAPHIC OR ACCESS OR AUTHORIZ* OR DRM) NEAR3 (KEY OR TOKEN)) (“MEDIA INFORMATION” OR METADATA OR MDAT OR TIMING OR DURATION OR STBL OR “STRUCTURAL DATA”) ((GAZE OR EYE) NEAR3 (BASED OR AWARE OR DIRECTED OR RESPONSIVE OR TRACK*)) ((RUNTIME OR DYNAMIC OR REAL-TIME) NEAR2 SELECTION) ((CONTINUOUS OR SEAMLESS OR UNINTERRUPTED OR PERSISTENT OR "REAL-TIME") NEAR5 (PLAYBACK OR STREAM* OR PLAY* OR CONTENT)) CPC=(H04N7/1675 OR G11B27/10 OR H04N21/8541 OR H04N21/8456 OR H04N19/96 OR H04N19/162)
6	((REMOV* OR STRIP* OR OMIT* OR EXCLUDE OR DETACH OR ELIMINAT*) NEAR5 ((RAW OR MEDIA OR PAYLOAD) NEAR2 (MEDIA OR CONTENT OR SEGMENT*))) (VIDEO OR AUDIOVISUAL OR “DIGITAL VIDEO” OR MULTIMEDIA OR STREAM*) (“PRE-RENDERED” OR “PRE-GENERATED” OR PREPROCESSED OR "STORED RENDERED" OR PREASSEMBLED OR PRECOMPUTED) (((TRAVERSAL OR NESTED OR RECURSIVE* OR HIERARCHICAL) NEAR3 GRAPH) OR BRANCHING OR TREE) ((CRYPTOGRAPHIC OR ACCESS OR AUTHORIZ* OR DRM) NEAR3 (KEY OR TOKEN)) (“MEDIA INFORMATION” OR METADATA OR MDAT OR TIMING OR DURATION OR STBL OR “STRUCTURAL DATA”) ((GAZE OR EYE) NEAR3 (BASED OR AWARE OR DIRECTED OR RESPONSIVE OR TRACK*)) ((RUNTIME OR DYNAMIC OR REAL-TIME) NEAR2 SELECTION) ((CONTINUOUS OR SEAMLESS OR UNINTERRUPTED OR PERSISTENT OR "REAL-TIME") NEAR5 (PLAYBACK OR STREAM* OR PLAY* OR CONTENT)) ((VIRTUAL OR MEDIA OR VIDEO OR METADATA) NEAR2 CONTAINER) CPC=(H04N7/1675 OR G11B27/10 OR H04N21/8541 OR H04N21/8456 OR H04N19/96 OR H04N19/162)
7	TAC=(VIDEO OR AUDIOVISUAL OR “DIGITAL VIDEO” OR MULTIMEDIA OR STREAM*) ((VIRTUAL OR MEDIA OR VIDEO OR METADATA) NEAR2 CONTAINER) ((CONTINUOUS OR SEAMLESS OR UNINTERRUPTED OR PERSISTENT OR "REAL-TIME") NEAR5 (PLAYBACK OR STREAM* OR PLAY* OR CONTENT)) TAC=(((TRAVERSAL OR NESTED OR RECURSIVE* OR HIERARCHICAL) NEAR3 GRAPH) OR BRANCHING) ((TRAVERSAL OR MEDIA OR CONTENT OR VIDEO OR NAVIGATION) NEAR3 (REFERENCE OR LINK)) ((CRYPTOGRAPHIC OR ACCESS OR AUTHORIZ* OR DRM) NEAR3 (KEY OR TOKEN)) (“MEDIA INFORMATION” OR METADATA OR MDAT OR TIMING OR DURATION OR STBL OR “STRUCTURAL DATA”) (“PRE-RENDERED” OR “PRE-GENERATED” OR PREPROCESSED OR "STORED RENDERED" OR PREASSEMBLED OR PRECOMPUTED) ((VIDEO OR MEDIA OR CONTENT OR STREAM) NEAR2 (RESOLUTION OR SELECTION)) ((RUNTIME OR DYNAMIC OR REAL-TIME) NEAR2 SELECTION)

8	((VIRTUAL* OR EMULAT* OR SIMULATION) WITH ((PLURALITY OR MULTIPLE OR VARIOUS) NEAR4 (VIDEO OR CLIP* OR FOOTAGE*))) (((REFERENCE OR LOCATOR OR ADDRESS) NEAR3 (CONTAINER OR PACKAGE)) WITH ((RAW OR UNSTRUCTURED) NEAR7 (MDAT OR "MEDIA SAMPLE PAYLOAD DATA"))) (((REND* OR FINALIZED OR COMPLIED) NEAR4 (VIDEO OR CLIP* OR FOOTAGE*)) WITH ("SAMPLE TIMING METADATA" OR STBL OR STTS OR " SAMPLE TABLE BOX")) (GRAPH OR TREE OR TOPOLOGY)
9	(((VARIOUS OR MULTIPLE OR MULTI) NEAR4 (VIDEO OR CLIP OR FOOTAGE)) WITH ((REFERENC* OR LOCATOR OR ADDRESS) NEAR4 (CONTAINER OR PACKAGE))) ((REMOVE OR CLEAR OR CLEAN OR DELETE) NEAR4 (MDAT OR "MEDIA SAMPLE PAYLOAD DATA")) (((RENDER* OR FINAL* OR COMPLIED) NEAR4 (VIDEO OR CLIP OR FOOTAGE)) WITH (SBTL OR "SAMPLE TIMING METADATA" OR STTS OR "SAMPLE TABLE BOX")) ((DECISION OR TRAVERSAL) SAME (GRAPH OR TREE OR TOPOLOGY)) (((VARIOUS OR MULTIPLE OR MULTI) NEAR4 (SCENE OR SEGMENT OR SLICE OR PORTION)) SAME (VIDEO OR CLIP OR FOOTAGE)) ((USER OR CLIENT OR HOST) NEAR3 (SELECT* OR CHOOSE*))
10	((PLURALITY OR VARIOUS OR MULTIPLE OR MULTI) NEAR4 ((VIDE* OR CLIP* OR FOOTAG* OR RECORD*) NEAR4 (PACKET* OR PORTION*))) ((REFERENCE OR ADDRESS OR LOCATOR) NEAR4 (CONTAINER OR WRAPPER OR PACKAGE)) (((REMOVE OR CLEAR OR DELETE) NEAR3 (RAW OR UNSTRUCTURED)) WITH ("MEDIA SAMPLE PAYLOAD DATA" OR MDAT)) (STTS OR SBTL OR "SAMPLE TABLE BOX" OR "SAMPLE TIMING") ((TRAVERSAL OR PROGRESSION) NEAR3 (GRAPH OR TREE OR TOPOLOGY OR NETWORK))
11	((VIDEO OR MEDIA OR CONTENT OR SAMPLE* OR SEGMENT*) NEAR5 (REFERENCE* OR CONTAINER* OR OBJECT* OR NODE*)) (((CRYPTOGRAPH* AND (TOKEN* OR KEY*))) NEAR5 (REFERENCE* OR CONTAINER* OR SEGMENT*)) ((VALIDAT* OR VERIF* OR AUTHENTICAT* OR AUTHORIZ* OR PERMIT* OR DENY* OR RESTRICT*) NEAR5 (ACCESS OR TRAVERSAL OR RESOLUTION OR PLAYBACK OR SELECTION)) CPC=(H04N21/2389 OR H04N21/23895 OR H04N21/23897)
12	((VIDEO* OR MEDIA OR MULTIMEDIA OR CONTENT) NEAR5 (SEGMENT* OR PORTION* OR CLIP* OR SEQUENCE*)) ((VIRTUAL*) NEAR5 (CONTAINER* OR PACKAGE* OR WRAPPER*)) WITH ((REFERENCE* OR LOCATOR*) NEAR5 (SAMPLE* OR SEGMENT* OR CLIP* OR SCENE*)) (TRAVERSAL) SAME ((GRAPH OR TOPOLOGY OR TREE OR NODE*) NEAR5 (REFERENCE*)) CPC=(H04N21/234) OR CPC=(H04N21/23412) OR CPC=(H04N21/44012)
13	(VIDEO OR MEDIA) (VIRTUALIZED OR REFERENCE OR LIGHTWEIGHT) (GRAPH OR TRAVERSAL OR NAVIGATION) (PLAYBACK OR ASSEMBLY OR STREAMING) (CONTAINER OR OBJECT) (H04L9/32) (H04N21/8456) COUNTRY:WO,US AFTER:PUBLICATION:20040101
14	(VIDEO OR MULTIMEDIA) AND (CONTAINER OR FILE) AND (MDAT OR "MEDIA DATA")AND (STBL OR "SAMPLE TABLE")AND ("METADATA ONLY" OR "WITHOUT MEDIA DATA" OR "MEDIA PAYLOAD REMOVED")
15	(VIDEO PLAYBACK OR VIDEO STREAMING OR MULTIMEDIA) ("DYNAMIC BRANCH" OR BRANCHING OR "BRANCH CONDITION") ("REAL-TIME SIGNAL*" OR RUNTIME) ("VIDEO CONTAINER") OR ("VIRTUAL CONTAINER")

7.3 Non-patent Search Strings

S No.	Strings (Google /Google Scholar/Science Direct)
1	(TRAVERSAL OR NAVIGATION) AND ((VIRTUALIZATION OR CONTAINERIZATION) AND (METADATA OR STBL)) AND (GRAPH OR LINKING)
2	(TRAVERSAL OR SEQUENCING) AND (VIRTUALIZATION OR PLACEHOLDER) AND (METADATA AND TIMESTAMP) AND (GRAPH OR INTERCONNECTION)
3	(FOVEAL OR PERIPHERAL) AND (SPATIAL OR RESOLUTION) AND (LICENSING OR CONSENT) AND (AUTHENTICATION) AND (CRYPTOGRAPHY)
4	((VIDEO OR MEDIA) NEAR3 (CONTAINER OR MANIFEST))AND (TRAVERSAL OR ROUTING) AND (GRAPH OR NETWORK)
5	((VIDEO OR MEDIA) NEAR7 (TRAVERSAL OR ROUTING)) AND (GRAPH) AND (NESTED OR INTERCONNECTION)
6	(VIDEO OR MEDIA OR CONTENT) AND (TOKEN* OR CREDENTIAL) AND (AUTHORIZATION OR VALIDATION) AND (ENCRYPTION OR DECRYPTION) AND ((TRAVERSAL) NEAR5 (GRAPH)) AND (NESTED OR INTERCONNECTION)
7	(MEDIA OR VIDEO OR MOVIE OR FILM OR "TV SHOW") ("VIRTUAL CONTAINERS" OR "REFERENCE CONTAINERS") AUTHORIZATION
8	"DYNAMIC VIDEO" COMPOSITION ("VIRTUAL CONTAINER" OR "REFERENCE CONTAINER")
9	"VIDEO SEGMENTATION" ENCODING TRAVERSAL ("REFERENCE CONTAINER" OR "VIRTUAL CONTAINER")
10	TRAVERSABLE "VIDEO GRAPH" ((REFERENCE CONTAINER) OR (VIRTUAL CONTAINER)) ENCODING (VIDEO SEGMENTATION)

7.4 Relevant Classifications

a. IPC/CPC Classification

IPC/CPC	Definition
G11B27/10	Indexing; Addressing; Timing Or Synchronising; Measuring Tape Travel
H04L2209/603	Digital Right Managament [DRM]
H04N19/162	User Input
H04N19/96	Tree Coding, E.G. Quad-Tree Coding
H04N21/234	Processing Of Video Elementary Streams, E.G. Splicing Of Video Streams Or Manipulating Encoded Video Stream Scene Graphs
H04N21/23412	For Generating Or Manipulating The Scene Composition Of Objects, E.G. Mpeg-4 Objects
H04N21/2389	Multiplex Stream Processing, E.G. Multiplex Stream Encrypting
H04N21/23895	Involving Multiplex Stream Encryption
H04N21/23897	By Partially Encrypting, E.G. Encrypting Only The Ending Portion Of A Movie
H04N21/44012	Involving Rendering Scenes According To Scene Graphs, E.G. Mpeg-4 Scene Graphs
H04N21/4627	Rights Management Associated To The Content (Security In Data Switching Network Management H04L41/28; Security Management Or Policies For Network Security H04L63/20; Access Security In Wireless Networks H04W12/08)
H04N21/47202	For Requesting Content On Demand, E.G. Video On Demand
H04N21/8456	By Decomposing The Content In The Time Domain, E.G. In Time Segments
H04N21/8541	Involving Branching, E.G. To Different Story Endings
H04N7/1675	Providing Digital Key Or Authorisation Information For Generation Or Regeneration Of The Scrambling Sequence (Pseudo-Random Number Generators In General G06F7/58)

b. US Classification

US Classes	Definition
386/241	Video or audio bookmarking (e.g., bit rate, scene change, thumbnails, timed, entry points, user manual initiated, etc.)
713/176	Authentication by digital signature representation or digital watermark

c. F-Term

F-Term	Definition
5C164MB41	Structuring Contents
5C164MB42	Hierarchisations
5C164MB44	Having Divided Time Units E.G. Scenes
5C164MC01	Content Production Or Authoring
5C164GA08	Matching Lines Of Sights
5C164YA12	Users' Presence Information Lines Of Sights Biological Information Or The Like

7.5 Key Inventors:

Akinori Takeo	Kevin Pei-Wen LEE	Nikolaus Faerber
Andrew Lee Jenks	Kfir Y. Rotbard	Rathish Krishnan
Barak Feldman	Koltun Gershi	Rebecca Nicole Burke-Aguero
Deepali Arya	Laura Buffard	Richard Kelley
Elena Kvochko	Lawrence E. Jones	Samuel John Wenker
Eric Horvitz	Leon Silverman	Stefan DOEHLA
Eric Hsuming Chen	Liu Yongliang	Stefan Kraegeloh
Gavin Stuart Peter Miller	Malvar Henrique	Tal Zubalsky
Harald Fuchs	Mark Arana	Tonisson Alan Valev
Hung-Ju Lee	Mark Baker	Xinyi Han
Jason N. Wang	Max Gray Edell	Xiong Xiaoli
Jonathan BLOCH	Michael Robert Blood	Xuejun Xu
Joseph Peyton Lea	Niklas Rystedt	Yaffe Yoel

7.6 Key Assignee(s):

Acronis International	Ezdrm	Nagravision Sarl
Adobe	France Telecom	Netflix
Alibaba	Futurewei Technologies	Nokia
Amazon	Guangzhou Huya Technology	Nvidia
Apple	Huawei	Nxp
AT&T	Ibm	Onmed
Bank Of America	Ilumintel	Orange
Bison Patent Licensing	Instart Logic	Pls Iv
Canon	Intel	Samsung
Cisco	Irdeto	Sony
Convergent Media Solutions	JBF Interlude	Swear
Cybercube	Korea Electronics Technology Institute	Truepic
Digital Transaction	Lg	Trufo
Disney Enterprises	MediaTek	Tyco Fire And Security
Divx	Microsoft	Unveiled Labs

8. Disclaimer

This report is work of analysis and interpretation of publically available information on various free and paid online sources and should not be construed as a legal opinion. This report is shared with you with mutual understanding and trust that no part of this report shall be publically distributed or used by you without explicit permission.