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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
21 July 2022 (21.07.2022)

WIPO | PCT

(10) International Publication Number
WO 2022/154841 A1

(51) International Patent Classification:
G06F 17/30 (2006.01)

(21) International Application Number:
PCT/US2021/053950

(22) International Filing Date:
07 October 2021 (07.10.2021)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
63/137,274 14 January 2021 (14.01.2021) US
17/477,134 16 September 2021 (16.09.2021) US

(71) Applicant: **TENCENT AMERICA LLC** [US/US]; 2747 Park Boulevard, Palo Alto, CA 94306 (US).

(72) Inventors: **ZHAO, Shuai**; c/o Tencent America LLC, 2747 Park Boulevard, Palo Alto, CA 94306 (US). **WENGER, Stephan**; c/o Tencent America LLC, 2747 Park Boulevard, Palo Alto, CA 94306 (US). **LIU, Shan**; c/o Tencent America LLC, 2747 Park Boulevard, Palo Alto, CA 94306 (US).

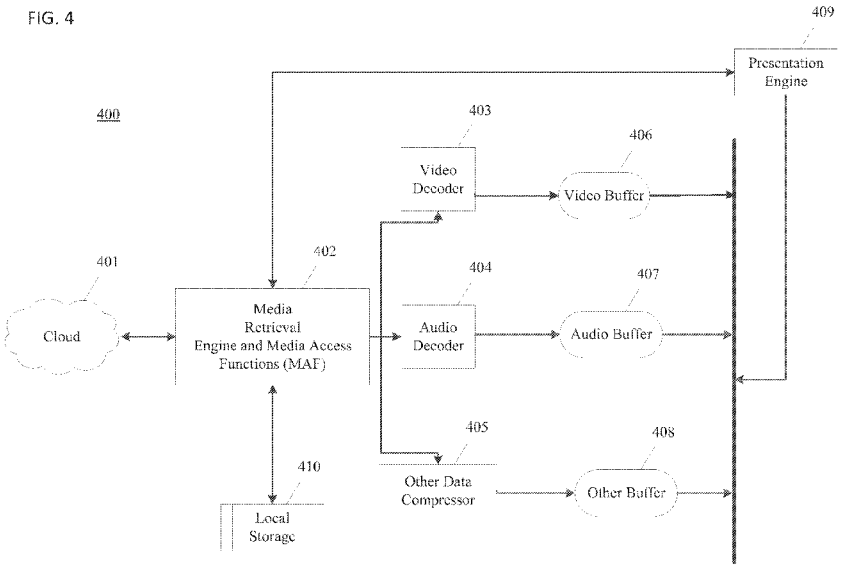
(74) Agent: **RABENA, John, F.** et al.; Sughrue Mion, PLLC, 2000 Pennsylvania Ave., N.W., Suite 9000, Washington, DC 20006 (US).

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

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM,

(54) Title: METHOD AND APPARATUS FOR MEDIA SCENE DESCRIPTION

FIG. 4



(57) Abstract: Systems, methods, and devices for managing media storage and delivery, including obtaining, by a media access function (MAF), a glTF file corresponding to a scene; determining that the glTF file has a CBOR format; converting the glTF file into a converted glTF file having a JSON format using a first CBOR parser function implemented by the MAF; and obtaining media content corresponding to the scene based on the converted glTF file.

[Continued on next page]

WO 2022/154841 A1

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

Published:

— *with international search report (Art. 21(3))*

METHOD AND APPARATUS FOR MEDIA SCENE DESCRIPTION

CROSS-REFERENCE TO RELATED APPLICATION

[0001] This application claims priority from U.S. Provisional Application No. 63/137,274, filed on January 14, 2021, and U.S. Application No. 17/477,134, filed on September 16, 2021, the disclosures of which are incorporated herein by reference in their entirety.

FIELD

[0002] Embodiments of the present disclosure relate to system design to support media objects using a 3D modeling syntax, implement media syntax to support various media codecs, containers, and formats, manage media storage and delivery method through predefined programming interfaces, and provide media buffer control and rendering functions.

BACKGROUND

[0003] The Graphics Language Transmission Format (glTF) is an API-neutral runtime asset 3D modeling delivery format. Compared with traditional 3D modeling tools, glTF provides a more efficient, extensible, interoperable format for the transmission and loading of 3D content. glTF2.0 is the most recent version of the glTF specification written by the Khronos 3D Group. This format supports a simple scene graph format that is generally capable of supporting static (untimed) objects in scenes, including “png” and “jpeg” image formats. glTF2.0 supports simple animations, including support for translate, rotate, and scale, of basic shapes described using the glTF primitives, i.e. for geometric objects. glTF2.0 does not support timed media, and hence does not support video nor audio.

[0004] "Information technology — Coding of audiovisual objects — Part 12: ISO base media file format", ISO/IEC 14496-12 (December 2015), "Draft of FDIS of ISO/IEC 23000-19 Common Media Application Format for Segmented Media", ISO/IEC JTC1/SC29/WG11 MPEG117/16819 (April 2017); and "Text of ISO/IEC FDIS 23009-1 4th edition", ISO/IEC JTC 1/SC 29/WG 11 N18609 (August 2019), and the glTF2.0 specification are incorporated herein by reference in their entirety.

SUMMARY

[0005] According to an embodiment, a method of managing media storage and delivery is implemented by at least one processor and includes: obtaining, by a media access function (MAF), a glTF file corresponding to a scene; determining that the glTF file has a CBOR format; converting the glTF file into a converted glTF file having a JSON format using a first CBOR parser function implemented by the MAF; and obtaining media content corresponding to the scene based on the converted glTF file.

[0006] According to an embodiment, a device for managing media storage and delivery includes at least one memory configured to store program code; and at least one processor configured to read the program code and operate as instructed by the program code, the program code including: first obtaining code configured to cause the at least one processor to obtain, by a media access function (MAF), a glTF file corresponding to a scene; first determining code configured to cause the at least one processor to determine that the glTF file has a CBOR format; first converting code configured to cause the at least one processor to convert the glTF file into a converted glTF file having a JSON format using a first CBOR parser function implemented by the MAF; and second obtaining code configured to cause the at least one processor to obtain media content corresponding to the scene based on the converted glTF file.

[0007] According to an embodiment, a non-transitory computer-readable medium storing instructions, the instructions including: one or more instructions that, when executed by at least one processor of a device for managing media storage and delivery, are configured to cause the at least one processor to: obtain, by a media access function (MAF), a glTF file corresponding to a scene; determine that the glTF file has a CBOR format; convert the glTF file into a converted glTF file having a JSON format using a first CBOR parser function implemented by the MAF; and obtain media content corresponding to the scene based on the converted glTF file.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] Further features, the nature, and various advantages of the disclosed subject matter will be more apparent from the following detailed description and the accompanying drawings in which:

[0009] FIG. 1 is a diagram of an environment in which methods, apparatuses and systems described herein may be implemented, according to embodiments.

[0010] FIG. 2 is a block diagram of example components of one or more devices of FIG. 1, according to embodiments.

[0011] FIG. 3 is a schematic illustration of glTF scene description objects, according to embodiments.

[0012] FIG. 4 is a schematic illustration of the media scene description system reference architecture, according to embodiments.

[0013] FIG. 5 is an example of glTF JavaScript Object Notation (JSON) format representation, according to embodiments.

[0014] FIG. 6 is an example of MPEG glTF extension, according to embodiments.

[0015] FIG. 7A is an illustration of a file having a JSON format, according to embodiments.

[0016] FIG. 7B is an illustration of a file having a CBOR format, according to embodiments.

[0017] FIG. 8 is an illustration of an example of glTF syntax, according to embodiments.

[0018] FIGS. 9A-9C are diagrams of example processes for managing media storage and delivery according to embodiments.

DETAILED DESCRIPTION

[0019] FIG. 1 is a diagram of an environment 100 in which methods, apparatuses, and systems described herein may be implemented, according to embodiments. As shown in FIG. 1, the environment 100 may include a user device 110, a platform 120, and a network 130. Devices of the environment 100 may interconnect via wired connections, wireless connections, or a combination of wired and wireless connections.

[0020] The user device 110 includes one or more devices capable of receiving, generating, storing, processing, and/or providing information associated with platform 120. For example, the user device 110 may include a computing device (*e.g.*, a desktop computer, a laptop computer, a tablet computer, a handheld computer, a smart speaker, a server, etc.), a mobile phone (*e.g.*, a smart phone, a radiotelephone, etc.), a wearable device (*e.g.*, a pair of smart glasses or a smart watch), or a similar device. In some implementations, the user device 110 may receive information from and/or transmit information to the platform 120.

[0021] The platform 120 includes one or more devices as described elsewhere herein. In some implementations, the platform 120 may include a cloud server or a group of cloud servers. In some implementations, the platform 120 may be designed to be modular such that

software components may be swapped in or out depending on a particular need. As such, the platform 120 may be easily and/or quickly reconfigured for different uses.

[0022] In some implementations, as shown, the platform 120 may be hosted in a cloud computing environment 122. Notably, while implementations described herein describe the platform 120 as being hosted in the cloud computing environment 122, in some implementations, the platform 120 may not be cloud-based (*i.e.*, may be implemented outside of a cloud computing environment) or may be partially cloud-based.

[0023] The cloud computing environment 122 includes an environment that hosts the platform 120. The cloud computing environment 122 may provide computation, software, data access, storage, etc. services that do not require end-user (*e.g.*, the user device 110) knowledge of a physical location and configuration of system(s) and/or device(s) that hosts the platform 120. As shown, the cloud computing environment 122 may include a group of computing resources 124 (referred to collectively as "computing resources 124" and individually as "computing resource 124").

[0024] The computing resource 124 includes one or more personal computers, workstation computers, server devices, or other types of computation and/or communication devices. In some implementations, the computing resource 124 may host the platform 120. The cloud resources may include compute instances executing in the computing resource 124, storage devices provided in the computing resource 124, data transfer devices provided by the computing resource 124, etc. In some implementations, the computing resource 124 may communicate with other computing resources 124 via wired connections, wireless connections, or a combination of wired and wireless connections.

[0025] As further shown in FIG. 1, the computing resource 124 includes a group of cloud resources, such as one or more applications ("APPs") 124-1, one or more virtual

machines ("VMs") 124-2, virtualized storage ("VSs") 124-3, one or more hypervisors ("HYPs") 124-4, or the like.

[0026] The application 124-1 includes one or more software applications that may be provided to or accessed by the user device 110 and/or the platform 120. The application 124-1 may eliminate a need to install and execute the software applications on the user device 110. For example, the application 124-1 may include software associated with the platform 120 and/or any other software capable of being provided via the cloud computing environment 122. In some implementations, one application 124-1 may send/receive information to/from one or more other applications 124-1, via the virtual machine 124-2.

[0027] The virtual machine 124-2 includes a software implementation of a machine (*e.g.*, a computer) that executes programs like a physical machine. The virtual machine 124-2 may be either a system virtual machine or a process virtual machine, depending upon use and degree of correspondence to any real machine by the virtual machine 124-2. A system virtual machine may provide a complete system platform that supports execution of a complete operating system ("OS"). A process virtual machine may execute a single program, and may support a single process. In some implementations, the virtual machine 124-2 may execute on behalf of a user (*e.g.*, the user device 110), and may manage infrastructure of the cloud computing environment 122, such as data management, synchronization, or long-duration data transfers.

[0028] The virtualized storage 124-3 includes one or more storage systems and/or one or more devices that use virtualization techniques within the storage systems or devices of the computing resource 124. In some implementations, within the context of a storage system, types of virtualizations may include block virtualization and file virtualization. Block virtualization may refer to abstraction (or separation) of logical storage from physical storage so that the storage system may be accessed without regard to physical storage or

heterogeneous structure. The separation may permit administrators of the storage system flexibility in how the administrators manage storage for end users. File virtualization may eliminate dependencies between data accessed at a file level and a location where files are physically stored. This may enable optimization of storage use, server consolidation, and/or performance of non-disruptive file migrations.

[0029] The hypervisor 124-4 may provide hardware virtualization techniques that allow multiple operating systems (*e.g.*, "guest operating systems") to execute concurrently on a host computer, such as the computing resource 124. The hypervisor 124-4 may present a virtual operating platform to the guest operating systems, and may manage the execution of the guest operating systems. Multiple instances of a variety of operating systems may share virtualized hardware resources.

[0030] The network 130 includes one or more wired and/or wireless networks. For example, the network 130 may include a cellular network (*e.g.*, a fifth generation (5G) network, a long-term evolution (LTE) network, a third generation (3G) network, a code division multiple access (CDMA) network, etc.), a public land mobile network (PLMN), a local area network (LAN), a wide area network (WAN), a metropolitan area network (MAN), a telephone network (*e.g.*, the Public Switched Telephone Network (PSTN)), a private network, an ad hoc network, an intranet, the Internet, a fiber optic-based network, or the like, and/or a combination of these or other types of networks.

[0031] The number and arrangement of devices and networks shown in FIG. 1 are provided as an example. In practice, there may be additional devices and/or networks, fewer devices and/or networks, different devices and/or networks, or differently arranged devices and/or networks than those shown in FIG. 1. Furthermore, two or more devices shown in FIG. 1 may be implemented within a single device, or a single device shown in FIG. 1 may be implemented as multiple, distributed devices. Additionally, or alternatively, a set of

devices (*e.g.*, one or more devices) of the environment 100 may perform one or more functions described as being performed by another set of devices of the environment 100.

[0032] FIG. 2 is a block diagram of example components of one or more devices of FIG. 1. The device 200 may correspond to the user device 110 and/or the platform 120. As shown in FIG. 2, device 200 may include a bus 210, a processor 220, a memory 230, a storage component 240, an input component 250, an output component 260, and a communication interface 270.

[0033] The bus 210 includes a component that permits communication among the components of the device 200. The processor 220 is implemented in hardware, firmware, or a combination of hardware and software. The processor 220 is a central processing unit (CPU), a graphics processing unit (GPU), an accelerated processing unit (APU), a microprocessor, a microcontroller, a digital signal processor (DSP), a field-programmable gate array (FPGA), an application-specific integrated circuit (ASIC), or another type of processing component. In some implementations, the processor 220 includes one or more processors capable of being programmed to perform a function. The memory 230 includes a random access memory (RAM), a read only memory (ROM), and/or another type of dynamic or static storage device (*e.g.*, a flash memory, a magnetic memory, and/or an optical memory) that stores information and/or instructions for use by the processor 220.

[0034] The storage component 240 stores information and/or software related to the operation and use of the device 200. For example, the storage component 240 may include a hard disk (*e.g.*, a magnetic disk, an optical disk, a magneto-optic disk, and/or a solid state disk), a compact disc (CD), a digital versatile disc (DVD), a floppy disk, a cartridge, a magnetic tape, and/or another type of non-transitory computer-readable medium, along with a corresponding drive.

[0035] The input component 250 includes a component that permits the device 200 to receive information, such as via user input (*e.g.*, a touch screen display, a keyboard, a keypad, a mouse, a button, a switch, and/or a microphone). Additionally, or alternatively, the input component 250 may include a sensor for sensing information (*e.g.*, a global positioning system (GPS) component, an accelerometer, a gyroscope, and/or an actuator). The output component 260 includes a component that provides output information from the device 200 (*e.g.*, a display, a speaker, and/or one or more light-emitting diodes (LEDs)).

[0036] The communication interface 270 includes a transceiver-like component (*e.g.*, a transceiver and/or a separate receiver and transmitter) that enables the device 200 to communicate with other devices, such as via a wired connection, a wireless connection, or a combination of wired and wireless connections. The communication interface 270 may permit the device 200 to receive information from another device and/or provide information to another device. For example, the communication interface 270 may include an Ethernet interface, an optical interface, a coaxial interface, an infrared interface, a radio frequency (RF) interface, a universal serial bus (USB) interface, a Wi-Fi interface, a cellular network interface, or the like.

[0037] The device 200 may perform one or more processes described herein. The device 200 may perform these processes in response to the processor 220 executing software instructions stored by a non-transitory computer-readable medium, such as the memory 230 and/or the storage component 240. A computer-readable medium is defined herein as a non-transitory memory device. A memory device includes memory space within a single physical storage device or memory space spread across multiple physical storage devices.

[0038] Software instructions may be read into the memory 230 and/or the storage component 240 from another computer-readable medium or from another device via the communication interface 270. When executed, software instructions stored in the memory

230 and/or the storage component 240 may cause the processor 220 to perform one or more processes described herein. Additionally, or alternatively, hardwired circuitry may be used in place of or in combination with software instructions to perform one or more processes described herein. Thus, implementations described herein are not limited to any specific combination of hardware circuitry and software.

[0039] The number and arrangement of components shown in FIG. 2 are provided as an example. In practice, the device 200 may include additional components, fewer components, different components, or differently arranged components than those shown in FIG. 2. Additionally, or alternatively, a set of components (e.g. one or more components) of the device 200 may perform one or more functions described as being performed by another set of components of the device 200.

[0040] Referring to FIG. 3, the Graphics Language Transmission Format (glTF) is an application programming interface (API)-neutral runtime asset 3D modeling delivery format. Compared with traditional 3D modeling tools, glTF provides a more efficient, extensible, interoperable format for the transmission and loading of 3D content.

[0041] A glTF scene may be a combination of multiple glTF assets. The glTF assets may be JavaScript Object Notation (JSON)-formatted files containing a full scene description which may include, for example, a scene object 301, node 302, camera 303, mesh 304, light 305, animation 306, accessor 307, material 308, skin 309, bufferview 310, technique 311, texture 312, buffer 313, program 314, image 315, sampler 316, shader 317, plus supporting external data.

[0042] glTF also supports external data sources which may be referenced in any above-mentioned scene objects. In embodiments, a binary file may be used for animation 306 or other buffer-based data 313. An image file may be used for object textures 312.

[0043] Referring to FIG. 5, as mentioned above, a glTF scene may be organized in JSON format. A glTF asset may include zero or more scenes 503, which may be the set of visual objects to render. Scenes may be defined in a scene array. In example illustrated in FIG. 5, there is a single scene 506 with a single node 501, although embodiments are not limited thereto. Various parameters that may be associated with each node object. For example, name 502 may specify the name of the node object, and scene name 504 may specify the name of the single scene.

[0044] The glTF scene assets may be consumed by a presentation engine for rendering a 3D or immersive scene to users. The existing glTF syntax only supports 3D objects including static or computer-generated animations. There is no support for media types such as video or audio, let alone rendering those video/audio media types.

[0045] Meanwhile, existing glTF can not describe a scene using geographical coordinate systems, which in some media presentation scenarios, such a feature is desired.

[0046] Therefore, there is a need to extend the glTF to support media types includes traditional 2D flat video, immersive media content such as virtual reality (VR), augmented reality (AR), extended reality (XR), and spatial audios. This may require an extension to support video/audio syntax and a system for media deliveries and render

[0047] Moving Picture Experts Group (MPEG) defines some extensions on top of the glTF specification to support immersive media content. Referring to FIG.3, new extensions are MPEG_media 330, MPEG_scene_dynamic 331, MPEG_texture_video 333, MEPG_animation_timing 332, MPEG_audio_spatial 334, MPEG_accessor_timed 335, MPEG_buffer_circular 336. In FIG. 3 generally, elements with rounded outlines, for example elements 301-317, may be glTF elements, and elements with square outlines, for example elements 330-336, may correspond to MPEG-based extensions of the glTF specification, although embodiments are not limited thereto.

[0048] If MPEG_media 330 as a root identifier, if specified, then MPEG_media may be supported. Referring to FIG. 6, the syntax to support MPEG media may be declared as the top-level JSON syntax. Syntax from 601 to 604 in FIG. 6 may be presented exactly as shown if supported.

[0049] Scene Updates may be expressed using the JSON Patch protocol and MPEG_scene_dynamic 331 may be used to support JSON patch protocol.

[0050] MPEG texture video extension, identified by MPEG_texture_video 333, may provide the possibility to link a glTF texture object to MPEG media and its respective track, listed by an MPEG_media object. MPEG texture video extension may also provide a reference to the MPEG_accessor_timed 335, where the decoded timed texture will be made available.

[0051] The MPEG_audio_spatial 334 extension may support multiple audio types.

[0052] In order to support timed data access, the buffer element may be extended to provide circular buffer functionality. The extension is named MPEG_buffer_circular 336 and may be included as part of the glTF "buffers" objects, for example buffer 313.

[0053] Above MEPG extensions may allow for the creation of immersive experiences using glTF. Eventually the glTF assets with MPEG extension may be used to be load into a rendering engine for visualization.

[0054] Referring to FIG. 4, a reference media scene description architecture 400 illustrates an example of how MPEG extensions may be used to support media type such as audio/video. The media contents may be retrieved using a Media Retrieval Engine and Media Access Functions (MAF) 402 from external sources such as a media cloud 401, processed using video decoder 403, audio decoder 404, and other data compressor 405, buffered in video buffer 406, audio buffer 407, and other buffer 408, and rendered by a presentation engine 409. In some cases, media content may be stored in local storage 410.

[0055] Referring to FIG. 4, the MPEG scene description extensions may decouple the Presentation Engine 409 from the Media Retrieval Engine 402. Presentation Engine 409 and Media Retrieval Engine 402 may communicate through the predefined programming interfaces, which allows the Presentation Engine 409 to request media data required for the rendering of the scene. The Media Retrieval Engine 402 may retrieve the requested media and make it available in a timely manner and in a format that can be immediately processed by the Presentation Engine 409. For instance, a requested media asset may be compressed and residing in the network, so the Media Retrieval Engine 402 will retrieve and decode the asset and pass the resulting media data to the Presentation Engine 409 for rendering. The media data may be passed in form of buffers from the Media Retrieval Engine 402 to the Presentation Engine 409. The requests for media data may be passed through a Media Retrieval API from the Presentation Engine 409 to the Media Retrieval Engine 402. For flexible use of video decoding resource, the Video Decoder 403 may be used. When the Video Decoder 403 is used, the Presentation Engine 409 may provide information for input formatting and output formatting to the Video Decoder 403 through Application Configuration APIs.

[0056] As discussed above, glTF syntax may be expressed in a JSON file. The Internet Engineering Task Force (IETF) Concise Binary Object Representation (CBOR) may represent a concise data format compared with the traditional JSON format. CBOR relates to similar data objects like JSON in a name/value pair format, but expressed in a binary and compact way, also with much more support with key-value types. A size of a file in CBOR format may be smaller than a corresponding file in JSON format. In some cases, the CBOR file may be more than 50% smaller than the corresponding JSON file. CBOR is registered in Internet Assigned Numbers Authority (IANA) as “application/cbor”.

[0057] CBOR may be used as one of the glTF interchangeable compressed file formats which also has been widely supported due to its compact data size and interchangeability with JSON.

[0058] Information in CBOR is stored in binary form. Because many use cases for information includes machines to understand the data, a binary data format may have speed advantages over human-readable data formats like JSON or XML which may need to be parsed each time the computer or machine is used to understand the data stored.

[0059] FIG. 7A illustrates an example of a file in JSON format, and FIG. 7B illustrates an example of a corresponding file in CBOR format. For example, the character "a" (711) in the JSON formatted file of FIG. 7A may correspond to 0x61 (721) in the CBOR formatted file of FIG. 7B. Similarly, the character "b" (712) in the JSON formatted file of FIG. 7A may correspond to 0x62 (722) in the CBOR formatted file of FIG. 7B, and the character "c" (713) in the JSON formatted file of FIG. 7A may correspond to 0x63 (723) in the CBOR formatted file of FIG. 7B.

[0060] The usage of CBOR compared with JSON for scene description may bring advantages in terms of small data size, support of multiple key-value types instead of just String object in JSON. Function programming interfaces may be used in the presented media scene description reference architecture, more precisely in the media access function module.

[0061] Because the support of CBOR by glTF is gaining popularity, such support may be added into MPEG scene description in order to, for example, increase glTF file format interoperability, reduce file size for local storage or cache, and reduce glTF file transfer latency with minimum processing power at MAF 402.

[0062] A CBOR parser function according to embodiments may be implemented by MAF 402 to translate CBOR input into glTF native supported JSON format and also could be used as a file compressor to save the large glTF file into local storage or cache 410.

[0063] The CBOR parser API offers one of the methods such as cbor2Json(), json2Cbor and save(), as shown in the Table 1 below:

TABLE 1

Method	Brief Description
`cbor2Json(FILE)`	Convert a CBOR format into a JSON format
`json2Cbor(FILE)`	Convert a JSON format into a CBOR format
`cbor2Json(Object)`	Convert a CBOR data blob into a JSON format

[0064] A detailed interface description may be as follows:

```
interface InputFileParser {
    readonly attribute FILE    inputFileName;
   readonly attribute FILE   outputFileName;
   readonly attribute CBOR cborDataBlob;
    FILE  cbor2Json()(FILE cborInput);
   FILE   json2Cbor(FILE jsonInput);
   FILE     cbor2Json(CBOR cborDataBlob);
    bool  save();
};};
```

[0065] The above proposed functions may be used for example in various scenarios as follows.

[0066] Referring to Fig. 8, a glTF "url" or "uri" syntax may point to a CBOR binary data blob (802). In embodiments, there may be two ways to specify if binary is indeed a CBOR data format. According to Example 1, a Multipurpose Internet Mail Extension (MIME) type may be signaled which specifies a "mimeTypes" with "application/cbor" (801). According to Example 2, a prefix "application/cbor;" may be included in front of actual binary data. Examples 1 and 2 may be used together. In any case, a function called "cbor2Json(Object)" which takes a CBOR binary data may be called to parse the CBOR file format into a JSON.

[0067] If an input glTF is in CBOR format, the output may be a glTF by using cbor2Json() API

[0068] If an input is in a native glTF format, then no conversion may be necessary.

[0069] For local storage or cache purpose, a glTF file may be saved as a CBOR by using json2Cbor() and save() interface.

[0070] Accordingly, embodiments may relate to methods of providing glTF file format interoperability with CBOR, reducing file size for local storage or cache, increasing, data transfer speed, reducing file transfer latency.

[0071] Referring to FIGS. 9A-9C, processes 900A, 900B, and 900C for managing media storage and delivery are described below.

[0072] FIG. 9A is a flowchart of an example process 900A for managing media storage and delivery.

[0073] As shown in FIG. 9A, process 900A may include obtaining, by a media access function (MAF), a glTF file corresponding to a scene (block 911). In embodiments, the MAF may correspond to MAF 402.

[0074] As further shown in FIG. 9A, process 900A may include determining that the glTF file has a CBOR format (block 912).

[0075] As shown in FIG. 9A, process 900A may include converting the glTF file into a converted glTF file having a JSON format using a first CBOR parser function implemented by the MAF (block 913).

[0076] As shown in FIG. 9A, process 900A may include obtaining media content corresponding to the scene based on the converted glTF file (block 914).

[0077] In embodiments, the converted glTF file having the JSON format may be larger than the glTF file having the CBOR format.

[0078] In embodiments, the MAF may be included in a Moving Picture Experts Group (MPEG) scene description architecture. In embodiments, the MPEG scene description architecture may correspond to media scene description architecture 400.

[0079] In embodiments, the first CBOR parser function may be implemented using an API associated with the MAF. In embodiments, the API may correspond to any API discussed above.

[0080] FIG. 9B is a flowchart of an example process 900B for managing media storage and delivery. In embodiments, one or more blocks of process 900B may be performed in combination with one or more blocks of process 900A. For example, one or more blocks of process 900B may be performed after one or more blocks of process 900A.

[0081] As shown in FIG. 9B, process 900B may include obtaining from the converted glTF file a uniform resource locator (URL) parameter indicating a binary data blob (block 921).

[0082] As further shown in FIG. 9B, process 900B may include determining that the binary data blob has the CBOR format (block 922).

[0083] As further shown in FIG. 9B, process 900B may include converting the binary data blob into an object having the JSON format using a second CBOR parser function implemented by the MAF (block 923).

[0084] In embodiments, the binary data blob may be determined to have the CBOR format based on a Multipurpose Internet Mail Extension (MIME) type that is signaled in the glTF file.

[0085] In embodiments, the binary data blob may be determined to have the CBOR format based on a prefix included at a beginning of the binary data blob.

[0086] In embodiments, the binary data blob may be determined to have the CBOR format based on a Multipurpose Internet Mail Extension (MIME) type that is signaled in the glTF file and a prefix included at a beginning of the binary data blob.

[0087] FIG. 9C is a flowchart of an example process 900C for managing media storage and delivery. In embodiments, one or more blocks of process 900C may be performed in combination with one or more blocks of processes 900A and/or 900B. For example, one or more blocks of process 900C may be performed after one or more blocks of process 900A, or after one or more blocks of process 900B.

[0088] As shown in FIG. 9C, process 900C may include re-converting the converted glTF file into a re-converted glTF having the CBOR format using a JSON parser function implemented by the MAF (block 931).

[0089] As further shown in FIG. 9C, process 900C may include storing the re-converted glTF file in at least one of a local storage or a cache (block 932).

[0090] In embodiments, the re-converted glTF file having the CBOR format may be smaller than the converted glTF file having the JSON format.

[0091] In embodiments, the JSON parser function may be implemented using an application programming interface associated with the MAF. In embodiments, the API may correspond to any API discussed above.

[0092] Although FIGS. 9A-9C show example blocks of processes 900A, 900B, and 900C, in some implementations, of processes 900A, 900B, and 900C may include additional blocks, fewer blocks, different blocks, or differently arranged blocks than those depicted in FIGS. 9A-9C. Additionally, or alternatively, two or more of the blocks of processes of processes 900A, 900B, and 900C may be performed in parallel. In embodiments, any one or more blocks of processes 900A, 900B, and 900C may be combined with any other one or

more blocks of processes 900A, 900B, and 900C in any order, and any one or more of any blocks of processes 900A, 900B, and 900C may be split or combined as desired.

[0093] Further, the proposed methods may be implemented by processing circuitry (e.g., one or more processors or one or more integrated circuits). In one example, the one or more processors execute a program that is stored in a non-transitory computer-readable medium to perform one or more of the proposed methods.

[0094] The foregoing disclosure provides illustration and description, but is not intended to be exhaustive or to limit the implementations to the precise form disclosed. Modifications and variations are possible in light of the above disclosure or may be acquired from practice of the implementations.

[0095] It will be apparent that systems and/or methods, described herein, may be implemented in different forms of hardware, firmware, or a combination of hardware and software. The actual specialized control hardware or software code used to implement these systems and/or methods is not limiting of the implementations. Thus, it should be understood that software and hardware may be designed to implement the systems and/or methods based on the description herein.

[0096] Even though particular combinations of features are recited in the claims and/or disclosed in the specification, these combinations are not intended to limit the disclosure of possible implementations. In fact, many of these features may be combined in ways not specifically recited in the claims and/or disclosed in the specification. Although each dependent claim listed below may directly depend on only one claim, the disclosure of possible implementations includes each dependent claim in combination with every other claim in the claim set.

[0097] No element, act, or instruction used herein should be construed as critical or essential unless explicitly described as such. Also, as used herein, the articles “a” and “an”

are intended to include one or more items, and may be used interchangeably with “one or more.” Furthermore, as used herein, the term “set” is intended to include one or more items (e.g., related items, unrelated items, a combination of related and unrelated items, etc.), and may be used interchangeably with “one or more.” Where only one item is intended, the term “one” or similar language is used. Also, as used herein, the terms “has,” “have,” “having,” or the like are intended to be open-ended terms. Further, the phrase “based on” is intended to mean “based, at least in part, on” unless explicitly stated otherwise.

WHAT IS CLAIMED IS:

1. A method of managing media storage and delivery, the method being implemented by at least one processor and comprising:

obtaining, by a media access function (MAF), a Graphics Language Transmission Format (glTF) file corresponding to a scene;

determining that the glTF file has a Concise Binary Object Representation (CBOR) format;

converting the glTF file into a converted glTF file having a JavaScript Object Notation (JSON) format using a first CBOR parser function implemented by the MAF; and

obtaining media content corresponding to the scene based on the converted glTF file.

2. The method of claim 1, wherein the converted glTF file having the JSON format is larger than the glTF file having the CBOR format.

3. The method of claim 1, wherein the MAF is included in a Moving Picture Experts Group (MPEG) scene description architecture.

4. The method of claim 1, wherein the first CBOR parser function is implemented using an application programming interface associated with the MAF.

5. The method of claim 1, further comprising:

re-converting the converted glTF file into a re-converted glTF file having the CBOR format using a JSON parser function implemented by the MAF; and

storing the re-converted glTF file in at least one of a local storage or a cache.

6. The method of claim 5, wherein the re-converted glTF file having the CBOR format is smaller than the converted glTF file having the JSON format.

7. The method of claim 5, wherein the JSON parser function is implemented using an application programming interface associated with the MAF.

8. A device for managing media storage and delivery, the device comprising:

at least one memory configured to store program code; and

at least one processor configured to read the program code and operate as instructed by the program code, the program code including:

first obtaining code configured to cause the at least one processor to obtain, by a media access function (MAF), a Graphics Language Transmission Format (glTF) file corresponding to a scene;

first determining code configured to cause the at least one processor to determine that the glTF file has a Concise Binary Object Representation (CBOR) format;

first converting code configured to cause the at least one processor to convert the glTF file into a converted glTF file having a JavaScript Object Notation (JSON) format using a first CBOR parser function implemented by the MAF; and

second obtaining code configured to cause the at least one processor to obtain media content corresponding to the scene based on the converted glTF file.

9. The device of claim 8, wherein the converted glTF file having the JSON format is larger than the glTF file having the CBOR format.

10. The device of claim 8, wherein the MAF is included in a Moving Picture Experts Group (MPEG) scene description architecture.

11. The device of claim 8, wherein the first CBOR parser function is implemented using an application programming interface associated with the MAF.

12. The device of claim 8, wherein the program code further includes:

re-converting code configured to cause the at least one processor to re-convert the converted glTF file into a re-converted glTF file having the CBOR format using a JSON parser function implemented by the MAF; and

storing code configured to cause the at least one processor to store the re-converted glTF file in at least one of a local storage or a cache.

13. The device of claim 12, wherein the re-converted glTF file having the CBOR format is smaller than the converted glTF file having the JSON format.

14. The device of claim 12, wherein the JSON parser function is implemented using an application programming interface associated with the MAF.

15. A non-transitory computer-readable medium storing instructions, the instructions comprising: one or more instructions that, when executed by at least one processor of a device for managing media storage and delivery, are configured to cause the at least one processor to:

obtain, by a media access function (MAF), a Graphics Language Transmission Format (glTF) file corresponding to a scene;

determine that the glTF file has a Concise Binary Object Representation (CBOR) format;

convert the glTF file into a converted glTF file having a JavaScript Object Notation (JSON) format using a first CBOR parser function implemented by the MAF; and

obtain media content corresponding to the scene based on the converted glTF file.

16. The non-transitory computer-readable medium of claim 15, wherein the converted glTF file having the JSON format is larger than the glTF file having the CBOR format.

17. The non-transitory computer-readable medium of claim 15, wherein the first CBOR parser function is implemented using an application programming interface associated with the MAF.

18. The non-transitory computer-readable medium of claim 15, wherein the one or more instructions are further configured to cause the at least one processor to:

re-convert the converted glTF file into a re-converted glTF file having the CBOR format using a JSON parser function implemented by the MAF; and

store the re-converted glTF file in at least one of a local storage or a cache.

19. The non-transitory computer-readable medium of claim 18, wherein the re-converted glTF file having the CBOR format is smaller than the converted glTF file having the JSON format.

20. The non-transitory computer-readable medium of claim 18, wherein the JSON parser function is implemented using an application programming interface associated with the MAF.

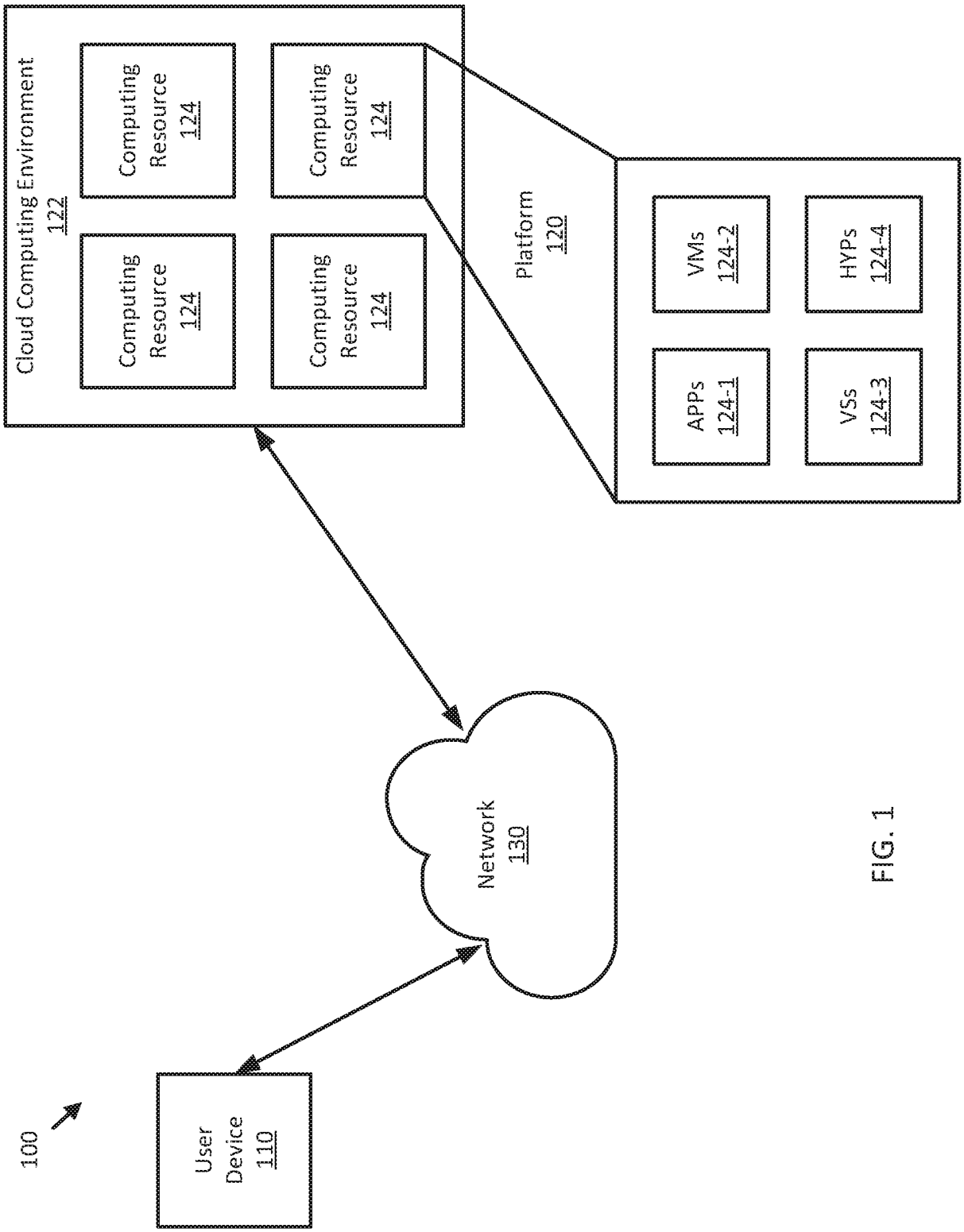

100
User Device 110
Network 130
Cloud Computing Environment 122
Computing Resource 124
Computing Resource 124
Computing Resource 124
Computing Resource 124
Platform 120
APPs 124-1
VMs 124-2
VSs 124-3
HYPs 124-4

FIG. 1

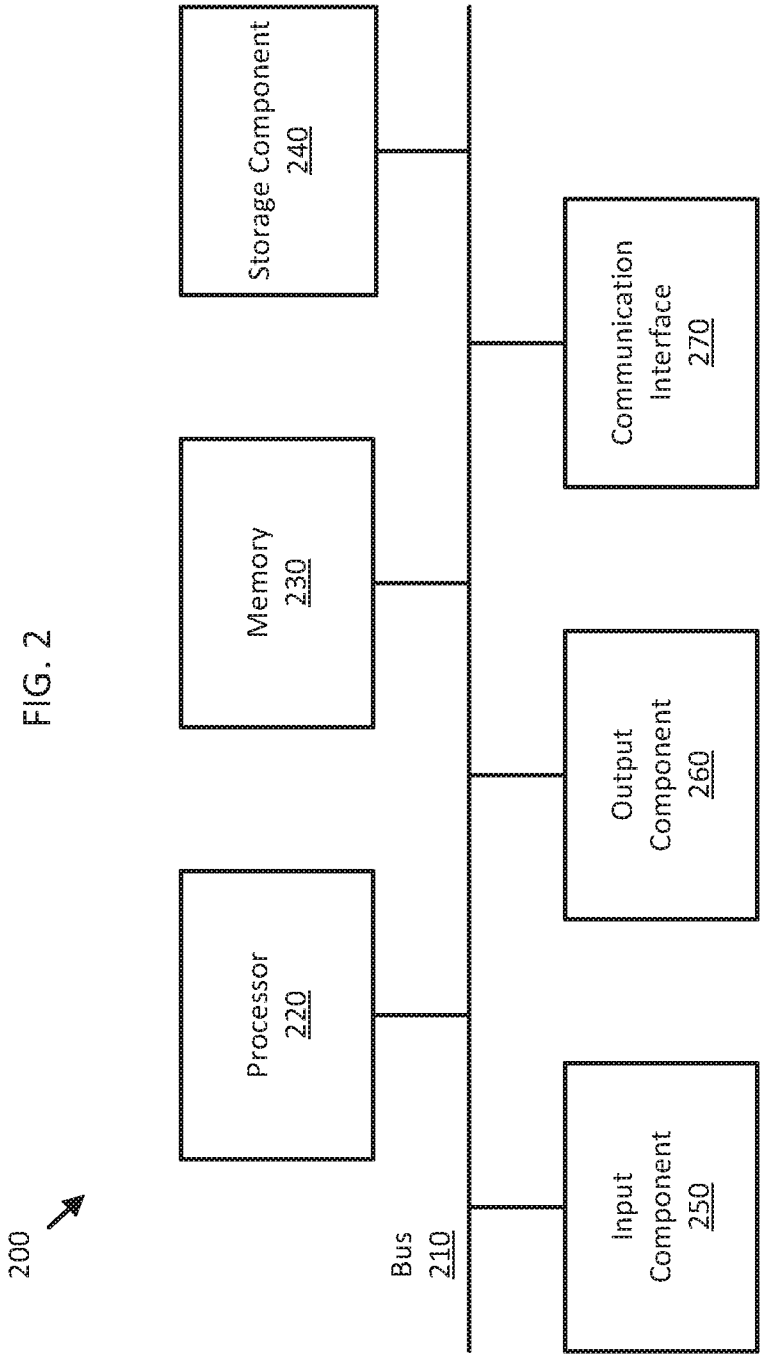

200
Bus
210
Processor
220
Memory
230
Storage Component
240
Input
Component
250
Output
Component
260
Communication
Interface
270

FIG. 2

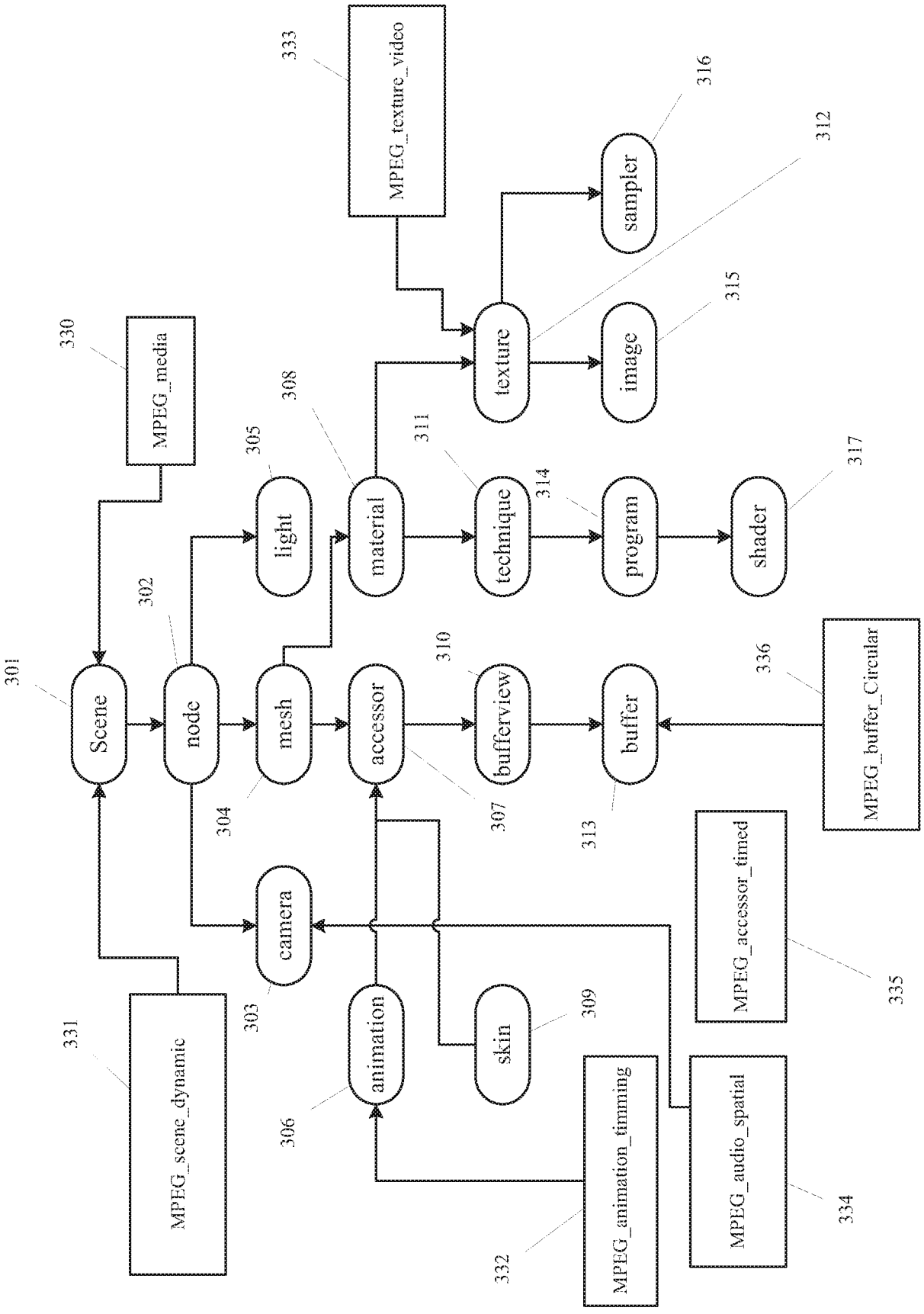

Scene
node
mesh
accessor
bufferview
buffer
MPEG_buffer_Circular
camera
animation
skin
MPEG_scene_dynamic
MPEG_animation_timming
MPEG_audio_spatial
MPEG_accessor_timed
light
material
technique
program
shader
texture
image
sampler
MPEG_texture_video
MPEG_media
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
330
331
332
333
334
335
336

FIG. 3

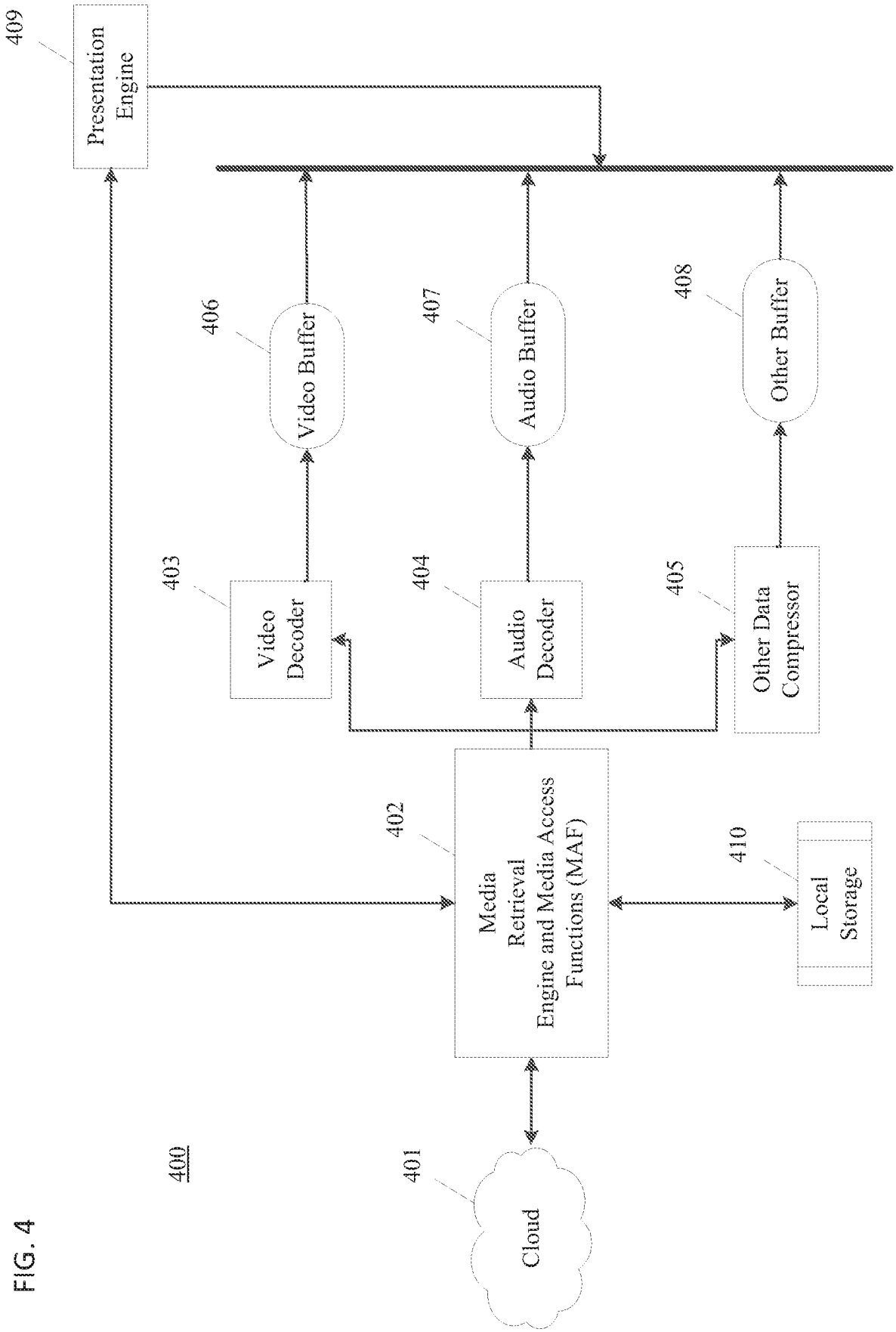

400
401
Cloud
402
Media Retrieval Engine and Media Access Functions (MAF)
403
Video Decoder
404
Audio Decoder
405
Other Data Compressor
406
Video Buffer
407
Audio Buffer
408
Other Buffer
409
Presentation Engine
410
Local Storage

FIG. 4

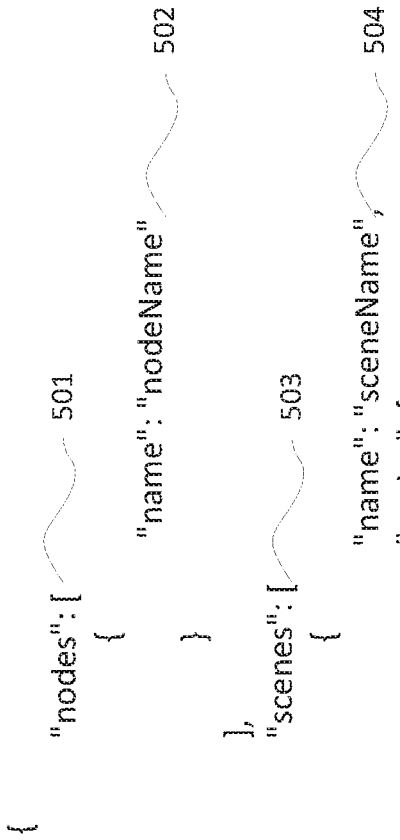

{
"nodes": [501
{
"name": "nodeName" 502
}
],
"scenes": [503
{
"name": "sceneName", 504
"nodes": [505
0
]
}
],
"scene": 0 506
}

FIG. 5

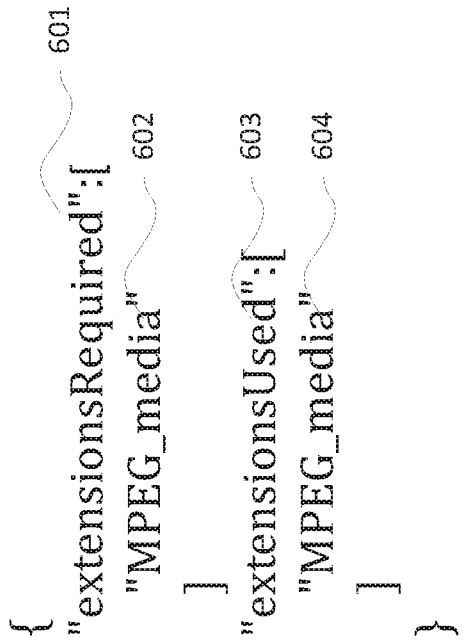

{
"extensionsRequired":[
"MPEG_media"
]
"extensionsUsed":[
"MPEG_media"
]
}
601
602
603
604

FIG. 6

JSON

```
[
    "a",
    {
        "b": "c"
    }
]
```

711
712
713

FIG. 7A

CBOR

```
82          # array(2)
   61       # text(1)          405
      61    # "a"              721
   BF       # map(*)
      61    # text(1)
         62 # "b"              722
      61    # text(1)
         63 # "c"              723
   FF       # primitive(*)
```

FIG. 7B

FIG. 8

```
{
"MPEG_media":
    { "media":[
        "alternatives":
            [   {   "mimeTypes": "application/cbor",        801
                    "uri": application/cbor;xxxxxx",        802
                }
    ] ,
}
```

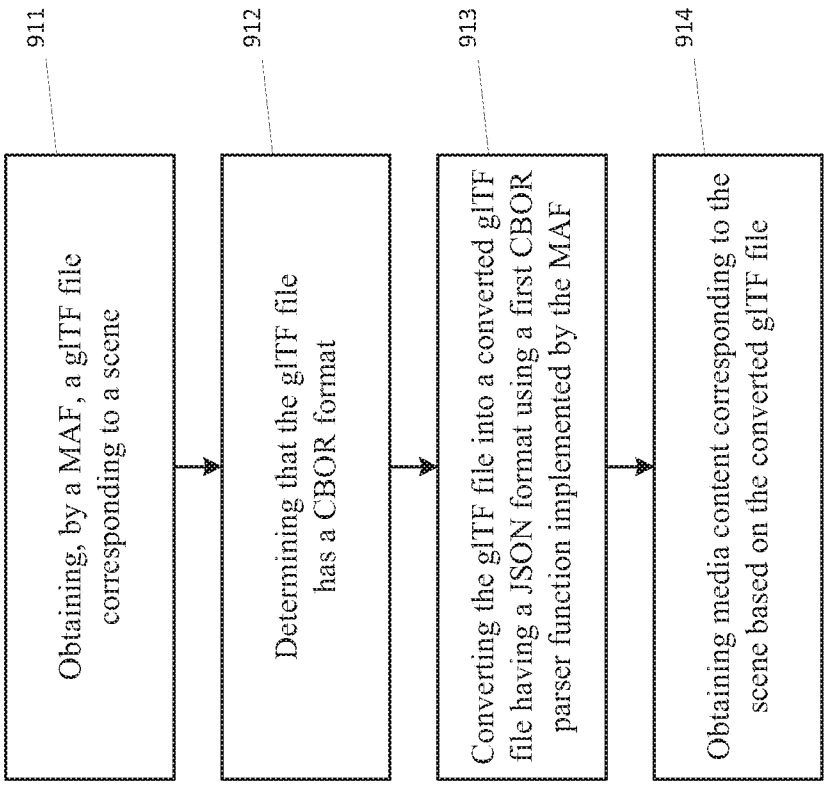

911
Obtaining, by a MAF, a glTF file corresponding to a scene
912
Determining that the glTF file has a CBOR format
913
Converting the glTF file into a converted glTF file having a JSON format using a first CBOR parser function implemented by the MAF
914
Obtaining media content corresponding to the scene based on the converted glTF file

FIG. 9A 900A

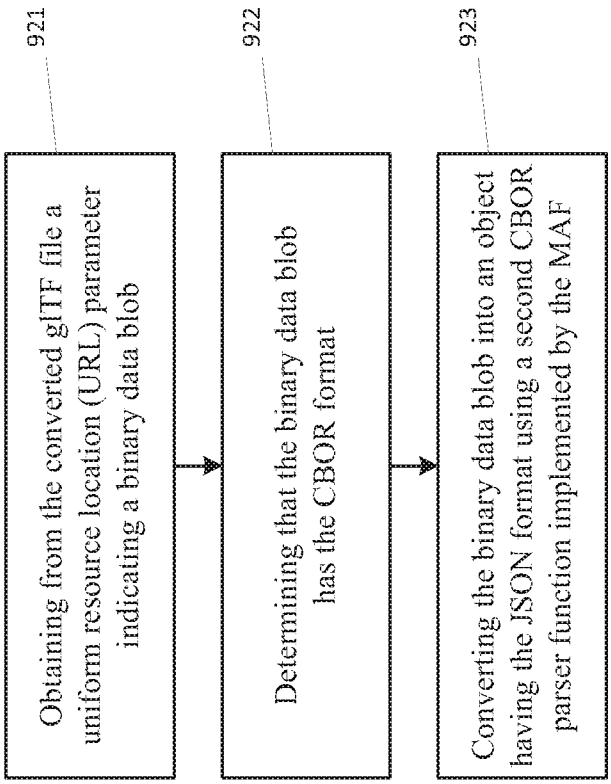

921
Obtaining from the converted glTF file a uniform resource location (URL) parameter indicating a binary data blob
922
Determining that the binary data blob has the CBOR format
923
Converting the binary data blob into an object having the JSON format using a second CBOR parser function implemented by the MAF

FIG. 9B

900B

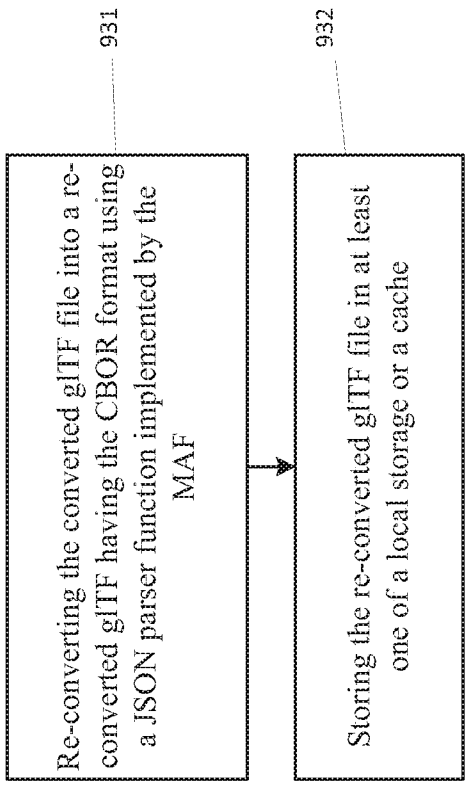

931
Re-converting the converted glTF file into a re-converted glTF having the CBOR format using a JSON parser function implemented by the MAF
932
Storing the re-converted glTF file in at least one of a local storage or a cache

FIG. 9C 900C

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 21/53950

A. CLASSIFICATION OF SUBJECT MATTER

IPC - G06F 17/30 (2021.01)

CPC - G06F 16/1787, H04L 67/06, G06F 16/40, H04L 67/18, G06F 16/122, G06F 16/178

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
See Search History document

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

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
See Search History document

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2020/0066022 A1 (Microsoft Technology Licensing, LLC) 27 February 2020 (27.02.2020), entire document especially para [0011], [0036]-[0040], [0100], [0120]	1-20
Y	US 2019/0265931 A1 (CANON KABUSHIKI KAISHA) 29 August 2019 (29.08.2019), entire document especially para [0040], [0065], [0070]	1-20
Y	US 2019/0028691 A1 (CABLE TELEVISION LABORATORIES, INC) 24 January 2019 (24.01.2019), entire document especially para [0094], [0096], [0115]	3, 10

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

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

Date of the actual completion of the international search	Date of mailing of the international search report
03 December 2021 (03.12.2021)	JAN 25 2022
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-8300	Authorized officer Kari Rodriquez Telephone No. PCT Helpdesk: 571-272-4300

Form PCT/ISA/210 (second sheet) (July 2019)