

METHOD AND APPARATUS FOR PROCESSING OF AUDIO DATA

CROSS-REFERENCE TO RELATED APPLICATION

5 [0001] This application claims priority of the following priority application: US provisional application 63/328,035, filed 06 April 2022 and EP application 22172243.2, filed 09 May 2022.

TECHNOLOGY

10 [0002] The present disclosure relates generally to a method of metadata-based dynamic processing of audio data for playback and, in particular, for determining and applying one or more processing parameters to the audio data for loudness leveling and/or dynamic range compression in combination with personalization settings (dialog enhancement, home- or away-commentary, etc.). The present disclosure further relates to a method of encoding audio data and metadata for loudness leveling and/or dynamic range compression into a bitstream.

15 The present disclosure yet further relates to a respective decoder and encoder as well as to a respective system and computer program products. The present disclosure further relates to a method of processing audio data for playback, a decoder for processing audio data for playback, and respective computer program products.

20 [0003] While some embodiments will be described herein with particular reference to that disclosure, it will be appreciated that the present disclosure is not limited to such a field of use and is applicable in broader contexts.

BACKGROUND

[0004] Any discussion of the background art throughout the disclosure should in no way be considered as an admission that such art is widely known or forms part of common general knowledge in the field.

[0005] In playing back audio content, loudness is the individual experience of sound pressure. In cinematic or television content, the loudness of dialogue in a program has been found to be the most crucial parameter determining the perception of program loudness by a listener.

30 [0006] To determine the average loudness of a program, either of the full program or

dialogue only, analysis of the entire program must be performed. The average loudness is typically required for loudness compliance (for example, the CALM act in the US), and is also used for aligning dynamic range control (DRC) parameters. The dynamic range of a program is the difference between its quietest and loudest sounds. The dynamic range of a
5 program depends on its content, for example, an action movie may have a different and wider dynamic range than a documentary, and reflects a creator's intent. However, capabilities of devices to play back audio content in the original dynamic range vary strongly. Besides loudness management, dynamic range control is thus a further key factor in providing optimal listening experience.

10 **[0007]** To perform loudness management and dynamic range control, the entire audio program or an audio program segment must be analyzed and the resulting loudness and DRC parameters can be delivered along with audio data or encoded audio data to be applied in a decoder or playback device.

[0008] When analysis of an entire audio program or an audio program segment prior to
15 encoding is not available, for example in real-time (dynamic) encoding, loudness processing or levelling is used to ensure loudness compliance and, if applicable, potential dynamic range constraints depending on playback requirements. This approach delivers processed audio that is “optimized” for a single playback environment.

[0009] There is thus an existing need for metadata-based processes that deliver “original”
20 unprocessed audio with accompanying metadata allowing the playback device to use the metadata to modify the audio dynamically depending on device constraints, user requirements and user settings (e.g., audio personalization settings).

[0010] Moreover, industry audio standards include descriptions and syntax for enabling loudness control and/or loudness management. For example, the Moving Picture Experts
25 Group (MPEG), which is an alliance of working groups established jointly by the International Organization for Standardisation (ISO) and International Electrotechnical Commission (IEC), that sets standards for media coding, including audio coding. MPEG is organized under ISO/IEC SC 29, and the audio group is presently identified as working group (WG) 6. This WG-6 helped establish an MPEG-H 3D audio standard that includes
30 compatibility for loudness control and/or loudness management (DRC) technology, however there is a need for further revising existing standards to process the above described metadata.

SUMMARY

[0011] In accordance with a first aspect of the present disclosure there is provided a method of metadata-based dynamic processing of audio data for playback. The method may include receiving, by a decoder, a bitstream including audio data and metadata for loudness leveling.

5 The method may further include decoding, by the decoder, the audio data and the metadata to obtain decoded audio data and the metadata. The method may further include determining, by the decoder, from the metadata, one or more processing parameters for loudness leveling based on a playback condition. The method may further include applying the determined one or more processing parameters to the decoded audio data to obtain processed audio data. And
10 the method may include outputting the processed audio data for playback.

[0012] In some embodiments, the metadata may be indicative of processing parameters for loudness leveling for a plurality of playback conditions.

[0013] In some embodiments, said determining the one or more processing parameters may further include determining one or more processing parameters for dynamic range
15 compression, DRC, based on the playback condition.

[0014] In some embodiments, the playback condition may include one or more of a device type of the decoder, characteristics of a playback device, characteristics of a loudspeaker, a loudspeaker setup, characteristics of background noise, characteristics of ambient noise, personalization experience selected on the device and characteristics of the acoustic
20 environment.

[0015] The personalization experience may be based on a version of the audio, such as the language, or user experience, such as enhancing the dialog. It could also include the ability to choose different experiences or perspectives, for example choosing the home team commentary versus away team commentary or choosing the home or away crowd as the
25 background.

[0016] The personalization experiences may be dependent on previous listening experiences and/or capabilities of listening devices. Or the personalization experience could be selected by the device (including via including external data via cloud) based on previous listening preferences.

30 **[0017]** The personalization experiences may be encoded in real-time, for example sports with home and away commentary, where loudness leveling would be used to ensure the audio is compliant with respect to loudness compliance (for example, the CALM act in the US). For a metadata-based solution, loudness leveling metadata, which may also include DRC metadata,

would be generated for each of the various personalized experiences, device capabilities.

[0018] In some embodiments, said determining the one or more processing parameters may further include selecting, by the decoder, at least one of a set of DRC sequences, DRCSet, a set of equalizer parameters, EQSet, and a downmix, corresponding to the playback condition.

5 **[0019]** In some embodiments, said determining the one or more processing parameters may further include identifying a metadata identifier indicative of the at least one selected DRCSet, EQSet and downmix to determine the one or more processing parameters from the metadata.

[0020] In some embodiments, the metadata may include one or more processing parameters
10 relating to average loudness values and optionally one or more processing parameters relating to dynamic range compression characteristics.

[0021] In some embodiments, the bitstream may further include additional metadata for static loudness adjustment to be applied to the decoded audio data.

[0022] In some embodiments, the bitstream may be an MPEG-D DRC bitstream and the
15 presence of metadata may be signaled based on MPEG-D DRC bitstream syntax.

[0023] In some embodiments, a uniDrcConfigExtension()-element may be used to carry the metadata as a payload.

[0024] In some embodiments, the metadata may comprise one or more metadata payloads, wherein each metadata payload may include a plurality of sets of parameters and identifiers,
20 with each set including at least one of a DRCSet identifier, drcSetId, an EQSet identifier, eqSetId, and a downmix identifier, downmixId, in combination with one or more processing parameters relating to the identifiers in the set.

[0025] In some embodiments, said determining the one or more processing parameters may involve selecting a set among the plurality of sets in the payload based on the at least one
25 DRCSet, EQSet, and downmix selected by the decoder, wherein the one or more processing parameters determined by the decoder may be the one or more processing parameters relating to the identifiers in the selected set.

[0026] In accordance with a second aspect of the present disclosure there is provided a decoder for metadata-based dynamic processing of audio data for playback. The decoder may
30 comprise one or more processors and non-transitory memory configured to perform a method including receiving, by the decoder, a bitstream including audio data and metadata for loudness leveling; decoding, by the decoder, the audio data and the metadata to obtain decoded audio data and the metadata; determining, by the decoder, from the metadata, one or more processing parameters for loudness leveling based on a playback condition; applying

the determined one or more processing parameters to the decoded audio data to obtain processed audio data; and outputting the processed audio data for playback.

[0027] In accordance with a third aspect of the present disclosure there is provided a method of encoding audio data and metadata for loudness leveling, into a bitstream. The method may include inputting original audio data into a loudness leveler for loudness processing to obtain, as an output from the loudness leveler, loudness processed audio data. The method may further include generating the metadata for loudness leveling based on the loudness processed audio data and the original audio data. And the method may include encoding the original audio data and the metadata into the bitstream.

[0028] In some embodiments, the method may further include generating additional metadata for static loudness adjustment to be used by a decoder.

[0029] In some embodiments, said generating metadata may include comparison of the loudness processed audio data to the original audio data, wherein the metadata may be generated based on a result of said comparison.

[0030] In some embodiments, said generating metadata may further include measuring the loudness over one or more pre-defined time periods, wherein the metadata may be generated further based on the measured loudness.

[0031] In some embodiments, the measuring may comprise measuring overall loudness of the audio data.

[0032] In some embodiments, the measuring may comprise measuring loudness of dialogue in the audio data.

[0033] In some embodiments, the bitstream may be an MPEG-D DRC bitstream and the presence of the metadata may be signaled based on MPEG-D DRC bitstream syntax.

[0034] In some embodiments, a uniDrcConfigExtension()-element may be used to carry the metadata as a payload.

[0035] In some embodiments, the metadata may comprise one or more metadata payloads, wherein each metadata payload may include a plurality of sets of parameters and identifiers, with each set including at least one of a DRCSets identifier, drcSetId, an EQSet identifier, eqSetId, and a downmix identifier, downmixId, in combination with one or more processing parameters relating to the identifiers in the set, and wherein the one or more processing parameters may be parameters for loudness leveling by a decoder.

[0036] In some embodiments, the at least one of the drcSetId, the eqSetId, and the downmixId may be related to at least one of a set of DRC sequences, DRCSets, a set of equalizer parameters, EQSet, and downmix, to be selected by the decoder.

[0037] In accordance with a fourth aspect of the present disclosure there is provided an encoder for encoding in a bitstream original audio data and metadata for loudness leveling. The encoder may comprise one or more processors and non-transitory memory configured to perform a method including inputting original audio data into a loudness leveler for loudness processing to obtain, as an output from the loudness leveler, loudness processed audio data; generating the metadata for loudness leveling based on the loudness processed audio data and the original audio data; and encoding the original audio data and the metadata into the bitstream.

[0038] In accordance with a fifth aspect of the present disclosure there is provided a system of an encoder for encoding in a bitstream original audio data and metadata for loudness leveling, and a decoder for metadata-based dynamic processing of audio data for playback.

[0039] In accordance with a sixth aspect of the present disclosure there is provided a computer program product comprising a computer-readable storage medium with instructions adapted to cause the device to carry out a method of metadata-based dynamic processing of audio data for playback or a method of encoding audio data and metadata for loudness leveling, into a bitstream when executed by a device having processing capability.

[0040] In accordance with a seventh aspect of the present disclosure there is provided a computer-readable storage medium storing the computer program product described herein.

[0041] In accordance with an eighth aspect of the present disclosure there is provided a method of processing audio data for playback. The method may include receiving, by a decoder, a bitstream including encoded audio data and metadata, wherein the metadata includes one or more dynamic range control (DRC) sets, and for each DRC set, an indication of whether the DRC set is configured for providing a loudness leveling effect. The method may further include parsing the metadata, by the decoder, to identify DRC sets that are configured for providing the loudness leveling effect. The method may further include decoding, by the decoder, the encoded audio data to obtain decoded audio data. The method may further include selecting, by the decoder, one of the identified DRC sets configured for providing the loudness leveling effect. The method may further include applying to the decoded audio data, by the decoder, the one or more DRC gains corresponding to the selected DRC set to obtain dynamic loudness compensated audio data. And the method may include outputting the dynamic loudness compensated audio data for playback.

[0042] In some embodiments, the metadata may include a plurality of DRC sets configured for providing the loudness leveling, wherein each of the plurality of DRC sets may also be associated with one or more playback conditions, and wherein the selecting may be

performed in response to an indication of a playback condition provided to the decoder.

[0043] In some embodiments, in addition to providing a loudness leveling effect, the one or more DRC sets may also be configured to provide dynamic range control.

[0044] In some embodiments, the playback condition may include one or more of a device
5 type of the decoder, characteristics of a playback device, characteristics of a loudspeaker, a loudspeaker setup, characteristics of background noise, characteristics of ambient noise and characteristics of the acoustic environment.

[0045] In some embodiments, the indication of whether the DRC set is configured for
10 providing the loudness leveling effect may be provided in a parameter indicating one or more effects provided by the DRC set.

[0046] In some embodiments, the parameter indicating one or more effects provided by the DRC set may be a drcSetEffect bitfield of an MPEG-D DRC bitstream, wherein individual bits of the drcSetEffect bitfield correspond to different effects, and one of the bits of the drcSetEffect bitfield corresponds to the loudness leveling effect.

[0047] In some embodiments, the indication of whether the DRC set is configured for
15 providing the loudness leveling effect may be whether the DRC set is specified in a loudness leveling bitstream payload.

[0048] In some embodiments, the loudness leveling bitstream payload may be included in an extension field of a previously defined bitstream syntax.

[0049] In some embodiments, the extension field may be a uniDrcConfigExtension field of
20 an MPEG-D DRC bitstream, and the loudness leveling bitstream payload may be included only for specific values of a uniDrcConfigExtType parameter.

[0050] In some embodiments, a plurality of loudness leveling payloads specifying a plurality of DRC sets configured for providing the loudness leveling effect may be included in the
25 extension field of the previously defined bitstream syntax.

[0051] In some embodiments, the indication of whether the DRC set is configured for providing the loudness leveling effect may be a field of a previously existing configuration element of a previously defined bitstream syntax.

[0052] In some embodiments, the field may be a levelingPresent parameter, and the
30 previously existing configuration element may be a downmixInstructions element, a drcInstructionsBasic element, or a drcInstructionsUniDRC element of an MPEG-D DRC bitstream.

[0053] In some embodiments, the field may be a previously existing field reserved for future use.

[0054] In some embodiments, the indication of whether the DRC set is configured for providing the loudness leveling effect may be a field of an updated version of a previously existing configuration element of a previously defined bitstream syntax.

5 [0055] In some embodiments, the field may be a levelingPresent parameter, and the updated version of the previously existing configuration element may be a downmixInstructionsV2 element or a drcInstructionsUniDrcV2 element.

[0056] In some embodiments, an indication that a loudness leveling effect is desired may be provided to the decoder through an interface, and the DRC set may be selected in response to the indication provided to the decoder through the interface.

10 [0057] In some embodiments, indications of additional desired effects may be provided to the decoder through the interface, the metadata may include a plurality of DRC sets configured to provide the loudness leveling effect, and the selection may depend on the additional desired effects.

15 [0058] In some embodiments, the indication that a loudness leveling effect is desired may be provided through a drcEffectTypeRequest parameter of a dynamicRangeControllerInterface payload.

[0059] In some embodiments, the metadata may include one or more static loudness values configured for providing static loudness adjustment to the decoded audio data.

20 [0060] In some embodiments, static loudness adjustment may be applied, in response to one or more of the static loudness values, to the decoded audio data or the dynamic loudness compensated audio data.

[0061] In some embodiments, a first one of the DRC sets may be configured for providing dynamic range control, and the first DRC set may comprise an indication that the selected DRC set configured for providing the loudness leveling effect may be configured for
25 application in combination with the first DRC set.

[0062] In some embodiments, the selected DRC set may comprise an indication of whether the one or more DRC gains corresponding to the selected DRC set may only be applied in combination with DRC gains corresponding to the first DRC set.

30 [0063] In some embodiments, DRC gains corresponding to the first DRC set may be extracted from the bitstream and may be applied to the decoded audio data.

[0064] In accordance with a ninth aspect of the present disclosure there is provided a decoder for processing audio data for playback. The decoder may comprise one or more processors

and non-transitory memory configured to perform a method of processing audio data for playback as described above.

[0065] In accordance with a tenth aspect of the present disclosure there is provided a computer program product comprising a computer-readable storage medium with instructions
5 adapted to cause the device to carry out a method of processing audio data for playback as described above.

[0066] In accordance with an eleventh aspect of the present disclosure there is provided a computer-readable storage medium storing the computer program product described herein.

10 BRIEF DESCRIPTION OF THE DRAWINGS

[0067] Example embodiments of the disclosure will now be described, by way of example only, with reference to the accompanying drawings in which:

[0068] FIG. 1 illustrates an example of a decoder for metadata-based dynamic processing of audio data for playback.

15 [0069] FIG. 2 illustrates an example of a method of metadata-based dynamic processing of audio data for playback.

[0070] FIG. 3A illustrates an example of an encoder for encoding in a bitstream original audio data and metadata for loudness leveling.

20 [0071] FIG. 3B illustrates an example of an encoder for encoding in a bitstream original audio data and metadata for loudness leveling and personalization.

[0072] FIG. 4A illustrates an example of a method of encoding audio data and metadata for loudness leveling, into a bitstream.

[0073] FIG. 4B illustrates an example of a method of encoding audio data and metadata for personalized, loudness leveling, into a bitstream.

25 [0074] FIG. 5 illustrates an example of a device comprising one or more processors and non-transitory memory configured to perform the methods described herein.

[0075] FIG. 6 illustrates an example of a method of processing audio data for playback.

DESCRIPTION OF EXAMPLE EMBODIMENTS

30 *Overview*

[0076] The average loudness of a program or dialogue is the main parameter or value used

for loudness compliance of broadcast or streaming programs. The average loudness is typically set to -24 or -23 LKFS. With audio codecs that support loudness metadata, this single loudness value representing the loudness of the entire program, is carried in the bitstream. Using this value in the decoding process allows gain adjustments that result in predictable playback levels, so that programs play back at known consistent levels.

Therefore, it is important that this loudness value is set properly and accurately. Since the average loudness is dependent on measuring the entire program prior to encoding, for real-time situations such as dynamic encoding with unknown loudness and dynamic range variation, this is, however, not possible.

[0077] When it is not possible to measure the loudness of the entire file prior to encoding, a dynamic loudness leveler is often used to modify or contour the audio data prior to encoding so that it meets the required loudness. This type of loudness management is often seen as an inferior method to meet compliance, as it often changes the dynamic range intercorrelation in the audio content and may thus change the creative intent. This is especially the case when it is desired to distribute one audio asset for all playback devices, which is one of the benefits of metadata driven codec and delivering systems.

[0078] In some approaches, audio content is mixed with the required target loudness, and the corresponding loudness metadata is set to that value. A loudness leveler might still be used in those situations as it will be used to help steer the audio content to the target loudness but it will not be that „active“ and is only used to when the audio content starts to deviate from the required target loudness.

[0079] In view of the above, methods and apparatus described herein aim at making real-time processing situations, also denoted as dynamic processing situations, also metadata driven. The metadata allow for loudness leveling and dynamic range compression in real-time situations.

The methods and apparatus as described advantageously enable:

- Use of real-time loudness adjustment and DRC in MPEG-D DRC and MPEG-H 3D audio syntax;
- Use of real-time loudness adjustment and DRC in combination with downmixId;
- Use of real-time loudness adjustment and DRC in combination with drcSetId;
- Use of real-time loudness adjustment and DRC in combination with eqSetId.
- Use of real-time loudness adjustment and DRC in combination with personalization settings identified using mae_groupID and mae_groupPresetID.

[0080] That is, depending on decoder settings (e.g., DRCSet, EQSet, downmix and personalization settings), a decoder can search based on the syntax a given payload for an appropriate set of parameters and identifiers, by matching the aforementioned settings to the identifiers. The parameters included in the set whose identifiers best match the settings can then be selected as the processing parameters for loudness leveling to be applied to received original audio data for correction.

[0081] Further, multiple sets of parameters for dynamic processing (multiple instances of dynLoudCompValue) can be transmitted.

[0082] The metadata-driven loudness leveling, in addition to correcting the overall loudness, can also be used to “center” the DRC gain calculation and application. This centering may be a result of correcting the loudness of the content, via the loudness leveling, and how DRC is typically calculated and applied. In this sense, metadata for loudness leveling can be said to be used for aligning DRC parameters.

Metadata-based dynamic processing of audio data

[0083] Referring to the example of Figure 1, a decoder 100 for metadata-based dynamic processing of audio data for playback is described. The decoder 100 may comprise one or more processors and non-transitory memory configured to perform a method including the processes as illustrated in the example of Figure 2 by means of steps S101 to S105.

[0084] The decoder 100, may receive a bitstream including audio data and metadata and may be able to output the unprocessed (original) audio data, the processed audio data after application of dynamic processing parameters determined from the metadata and/or the metadata itself depending on requirements. The metadata includes personalization metadata (including interactivity metadata) as described herein. The personalization metadata may comprise information for identifying a personalization experience.

[0085] The personalization experience may be based on a version of the audio, such as the language, or user experience, such as enhancing the dialog. It could also include the ability to choose different experiences or perspectives, for example choosing the home team commentary versus away team commentary or choosing the home or away crowd as the background.

[0086] The personalization experiences may be dependent on pervious listening experiences and/or capabilities of listening devices. Or the personalization experience could be selected by the device (including via including external data via cloud) based on previous listening preferences.

[0087] The personalization experiences may be encoded in real-time, for example sports with home and away commentary, where loudness leveling would be used to ensure the audio is compliant with respect to loudness compliance (for example, the CALM act in the US). For a metadata-based solution, loudness leveling metadata, which may also include DRC metadata, would be generated for each of the various personalized experiences, device capabilities.

[0088] Referring to the example of Figure 2, in step S101 the decoder 100 may receive a bitstream including audio data and metadata for loudness leveling and optionally, dynamic range compression (DRC). The audio data may be encoded audio data, the audio data may further be unprocessed. That is, the audio data may be said to be original audio data.

[0089] While the format of the bitstream is not limited, in an embodiment, the bitstream may be an MPEG-D DRC bitstream. The presence of metadata for dynamic processing of audio data may then be signaled based on MPEG-D DRC bitstream syntax. In an embodiment, a uniDrcConfigExtension()-element may be used to carry the metadata as a payload as detailed further below. Alternatively, the bit stream may be an MPEG-H 3D audio compatible bit stream and the metadata may be signaled using MPEG-H 3D audio bit stream syntax, for example using MPEG-H Audio Stream (MHAS) packets.

[0090] In step S102, the audio data and the metadata may then be decoded, by the decoder, to obtain decoded audio data and the metadata. In an embodiment, the metadata may include one or more processing parameters relating to average loudness values and optionally one or more processing parameters relating to dynamic range compression characteristics. The metadata allows to apply dynamic or real-time correction. For example, when encoding and decoding for live real-time playout, the application of the “real-time” or dynamic loudness metadata is desired to ensure that the live playout audio is properly loudness managed. The metadata further comprises personalization metadata as referenced herein, which may be generalized metadata for allowing of processing by various devices and device capabilities.

[0091] In step S103, the decoder then determines, from the metadata, one or more processing parameters for loudness leveling based on a playback condition. In an embodiment, a playback condition may include one or more of a device type of the decoder, characteristics of a playback device, characteristics of a loudspeaker, a loudspeaker setup, characteristics of background noise, characteristics of ambient noise and characteristics of the acoustic environment. Moreover, the playback condition may include information about personalization experience as discussed herein. The consideration of a playback condition allows the decoder for a targeted selection of processing parameters for loudness leveling with regard to device and environmental constraints.

[0092] In an embodiment, the process of determining the one or more processing parameters in step S103 may further include selecting, by the decoder, at least one of a set of DRC sequences, DRCSet, set of equalizer parameters, EQSet, and a downmix, corresponding to the playback condition. Thus, the at least one of a DRCSet, EQSet and downmix correlates
5 with or is indicative of the individual device and environmental constraints due to the playback condition.

[0093] In an embodiment, the process of determining in step S103 may further include identifying a metadata identifier indicative of the at least one selected DRCSet, EQSet and DownmixSet to determine the one or more processing parameters from the metadata. The
10 metadata identifier thus enables to connect the metadata with a corresponding selected DRCSet, EQSet, and/or downmix, and thus with a respective playback condition.

[0094] In an embodiment, the metadata may comprise one or more metadata payloads (e.g., dynLoudComp() payloads, such as shown in Table 5 below), wherein each metadata payload may include a plurality of sets of parameters (e.g., parameters dynLoudCompValue) and
15 identifiers, with each set including at least one of a DRCSet identifier, drcSetId, an EQSet identifier, eqSetId, a downmix identifier, downmixId and personalization identifiers, mae_groupID/mae_groupPresetID, in combination with one or more processing parameters relating to the identifiers in the set. That is, each payload may comprise an array of entries, each entry including processing parameters and identifiers (e.g., drcSetId, eqSetId,
20 downmixId, mae_groupID/mae_groupPresetID). In a further embodiment, the determining in step S103 may thus involve selecting a set among the plurality of sets in the payload based on the at least one DRCSet, EQSet, and downmix selected by the decoder, wherein the one or more processing parameters determined in step S103 may be the one or more processing parameters relating to the identifiers in the selected set. That is, depending on settings (e.g.,
25 DRCSet, EQSet, downmix, personalization settings) present in the decoder, the decoder can search a given payload for an appropriate set of parameters and identifiers, by matching the aforementioned settings to the identifiers. The parameters included in the set whose identifiers best match the settings can then be selected as the processing parameters for loudness leveling.

[0095] In step S104, the determined one or more processing parameters may then be applied, by the decoder, to the decoded audio data to obtain processed audio data. The processed audio data, for example live real-time audio data, are thus properly loudness managed.

[0096] In step S105, the processed audio data may then be output for playback.

[0097] In an embodiment, the bitstream may further include additional metadata for static

loudness adjustment to be applied to the decoded audio data. Static loudness adjustment refers in contrast to dynamic processing for real-time situations to processing performed for general loudness normalization.

5 [0098] Carrying the metadata for dynamic processing separately from the additional metadata for general loudness normalization allows to not have the “real-time” correction applied.

[0099] For example, when encoding and decoding for live real-time playout, the application of dynamic processing is desired to ensure that the live playout audio is properly loudness managed. But for a non-real-time playout, or a transcoding where the dynamic correction is not desired or required, the dynamic processing parameters determined from the metadata do
10 not have to be applied.

[0100] By further keeping the (dynamic/real-time) metadata for dynamic processing separate from the additional metadata, the originally unprocessed content can be retained, if desired. The original audio is encoded along with the metadata. This allows the playback device to selectively apply the dynamic processing and to further enable playback of original audio
15 content on high-end devices capable of playing back original audio.

[0101] Keeping the dynamic loudness metadata distinct from the long-term loudness measurement/information, such as contentLoudness (in ISO/IEC 23003-4) as described above has some advantages. If combined, the loudness of the content (or what it should be after the dynamic loudness metadata is applied) would not indicate the actual loudness of the content,
20 as the metadata available would be a composite value. Besides removing this ambiguity of what the content loudness (or program or anchor loudness) is, there are some cases where this would be particularly beneficial:

[0102] Keeping the metadata for dynamic processing separate allows the decoder or playback device to turn off the application of dynamic processing and to apply an implemented real-
25 time loudness leveler instead to avoid cascading leveling. This situation may occur, for example, if the device’s own real-time leveling solution is superior to the one used with the audio codec or, for example, if the device’s own real-time leveling solution cannot be disabled and therefore will always be active, resolution in further processing leading to a compromised playback experience.

30 [0103] Keeping the metadata for dynamic processing separate further allows transcoding to a codec that does not support loudness leveling and one wishes to apply their own loudness processing prior to re-encoding.

[0104] Live to broadcast, with a single encode for a live feed would be a further example. The dynamic processing metadata may be used or stored for archive or on-demand services.

Therefore, for the archive or on-demand services, a more accurate, or compliant loudness measurement, based on the entire program can be carried out, and the appropriate metadata reset.

5 [0105] For use-cases where a fixed target loudness is used throughout the workflow, for example in a R128 compliant situation where -23 LKFS (or LUFS) is recommended, this is also beneficial. In this scenario, the addition of the dynamic processing metadata is a “safety” measure, where the content is assumed and close to the required target and the addition of the dynamic processing metadata is a secondary check. Thus, having the ability to turn it off is desirable.

10 *Encoding audio data and metadata for loudness leveling*

[0106] Referring to the examples of Figures 3A, 3B and 4, an encoder for encoding in a bitstream original audio data and metadata for loudness leveling and optionally, dynamic range compression, DRC, is described which may comprise one or more processors and non-transitory memory configured to perform a method including the processes as illustrated in the steps in the examples of Figures 4A and 4B.

[0107] Referring to the examples of Figures 3A and 4A, in step S201, original audio data may be input into a loudness leveler, 201, for loudness processing to obtain, as an output from the loudness leveler, 201, loudness processed audio data.

20 [0108] In step S202, metadata for loudness leveling may then be generated based on the loudness processed audio data and the original audio data. Appropriate smoothing and time frames may be used to reduce artifacts.

[0109] In an embodiment, step S202 may include comparison of the loudness processed audio data to the original audio data, by an analyzer, 202, wherein the metadata may be generated based on a result of said comparison. The metadata thus generated can emulate the effect of the leveler at the decoder site. The metadata may include:

- Gain (wideband and/or multiband) processing parameters such that when applied to the original audio will produced loudness compliant audio for playback;
- Processing parameters describing the dynamics of the audio such as
 - 30 ○ Peak – sample and true peak
 - Short-term loudness values
 - Change of short-term loudness values.

[0110] In an embodiment, step S202 may further include measuring, by the analyzer, 202, the loudness over one or more pre-defined time periods, wherein the metadata may be generated further based on the measured loudness. In an embodiment, the measuring may comprise measuring overall loudness of the audio data. Alternatively, or additionally, in an

5 embodiment, the measuring may comprise measuring loudness of dialogue in the audio data.

[0111] In step S203, the original audio data and the metadata may then be encoded into the bitstream. While the format of the bitstream is not limited, in an embodiment, the bitstream may be an MPEG-D DRC bitstream and the presence of the metadata may be signaled based on MPEG-D DRC bitstream syntax. In this case, in an embodiment, a

10 uniDrcConfigExtension()-element may be used to carry the metadata as a payload as detailed further below.

[0112] In an embodiment, the metadata may comprise one or more metadata payloads, wherein each metadata payload may include a plurality of sets of parameters and identifiers, with each set including at least one of a DRCSet identifier, drcSetId, an EQSet identifier,

15 eqSetId, a downmix identifier, downmixId and personalization identifiers,

mae_groupID/mae_groupPresetID, in combination with one or more processing parameters relating to the identifiers in the set, and wherein the one or more processing parameters may be parameters for loudness leveling by a decoder. In this case, in an embodiment, the at least one of the drcSetId, the eqSetId, and the downmixId may be related to at least one of a set of

20 DRC sequences, DRCSet, a set of equalizer parameters, EQSet, and a downmix, to be selected by the decoder.

[0113] In an embodiment, the method may further include generating additional metadata for static loudness adjustment to be used by a decoder. Keeping the metadata for loudness leveling and the additional metadata separate in the bitstream and encoding further the

25 original audio data into the bitstream has several advantages as detailed above.

[0114] Referring to the examples of Figures 3B and 4B, in step S200B original audio data may be input into a personalization block 200B. The personalization block 200B may process the original audio (and/or additional information or metadata) in order to determine and/or generate personalized audio and personalization metadata.

30 [0115] In step S201B, personalized audio data from 200B is input into a loudness leveler, 201B, for loudness processing to obtain, as an output from the loudness leveler, 201B, personalized, loudness processed audio data. The loudness leveler 201B may be similar or same to the loudness leveler, 201, described in context with Figures 3A and 3B.

[0116] In step S202B, metadata for loudness leveling may then be generated based on the

personalized, loudness processed audio data output from S201B/analyzer 201B and the personalized audio data output from S200B/personalization block 200B. In an embodiment, step S202 may include comparison of the loudness processed audio data to the original audio data, by an analyzer, 202B, wherein the metadata may be generated based on a result of said comparison. Step 202B of Figure 3B may be performed in accordance with the description of step S202 of Figure 3A.

[0117] In step S203B, the personalized audio data from S200B/200B and the metadata from S200B and/or S202B may be encoded into the bitstream. Step 203B outputs encoded audio data and metadata, including personalization metadata as described herein. While the format of the bitstream is not limited, in an embodiment, the bitstream may be an MPEG-D DRC bitstream and the presence of the metadata may be signaled based on MPEG-D DRC bitstream syntax. Alternatively, the bitstream may be an MPEG-H 3D audio compatible bitstream and the metadata may be signaled based on MPEG-H 3D audio bitstream syntax. In an embodiment, the metadata may comprise one or more metadata payloads as described in context with step S203 of Figure 3A. The methods described herein (for example in context with Figures 3A and 3B) may be implemented on a decoder or an encoder, respectively, wherein the decoder and the encoder may comprise one or more processors and non-transitory memory configured to perform said methods. An example of a device having such processing capability is illustrated in the example of Figure 5 showing said device, 300, including two processors, 301, and non-transitory memory, 302.

[0118] It is noted that the methods described herein with regard to Figures 3A and 3B can further be implemented on a system of an encoder for encoding in a bitstream original audio data and metadata for loudness leveling and optionally, dynamic range compression, DRC, and a decoder for metadata-based dynamic processing of audio data for playback as described herein.

[0119] The methods may further be implemented as a computer program product comprising a computer-readable storage medium with instructions adapted to cause the device to carry out said methods when executed by a device having processing capability. The computer program product may be stored on a computer-readable storage medium.

MPEG-D DRC modified bitstream syntax

[0120] In the following, it will be described how the MPEG-D DRC bitstream syntax, as described in ISO/IEC 23003-4, may be modified in accordance with embodiments described herein.

[0121] The MPEG-D DRC syntax may be extended, e.g. the *loudnessInfoSetExtension()*-element shown in Table 1 below, in order to also carry the dynamic processing metadata as a frame-based *dynLoudComp* update.

- [0122] For example, another switch-case UNIDRCLOUDTEXT_DYNLOUDCOMP may be added in the *loudnessInfoSetExtension()*-element as shown in Table 1. The switch-case UNIDRCLOUDTEXT_DYNLOUDCOMP may be used to identify a new element *dynLoudComp()* as shown in Table 5. The *loudnessInfoSetExtension()*-element may be an extension of the *loudnessInfoSet()*-element as shown in Table 2. Further, the *loudnessInfoSet()*-element may be part of the *uniDRC()*-element as shown in Table 3.

Syntax	No. of bits	Mnemonic
<pre> loudnessInfoSetExtension() { while (loudnessInfoSetExtType != UNIDRCLOUDTEXT_TERM) { extSizeBits = bitSizeLen + 4; extBitSize = bitSize + 1; switch (loudnessInfoSetExtType) { UNIDRCLOUDTEXT_EQ: loudnessInfoV1AlbumCount; loudnessInfoV1Count; for (i=0; i<loudnessInfoV1AlbumCount; i++) { loudnessInfoV1(); } for (i=0; i<loudnessInfoV1Count; i++) { loudnessInfoV1(); } break; UNIDRCLOUDTEXT_DYNLOUDCOMP: dynLoudCompCount; for (i=0; i<dynLoudCompCount; i++) { dynLoudComp(); } break; /* add future extensions here */ default: for (i=0; i<extBitSize; i++) { </pre>	4 4 extSizeBits 6 6 6	uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf

```

}
    }
    }
    otherBit;
    1
    bsbf
}

```

Table 1: Syntax of `loudnessInfoSetExtension()`-element

Syntax	No. of bits	Mnemonic
loudnessInfoSet() { loudnessInfoAlbumCount; loudnessInfoCount; for (i=0; i<loudnessInfoAlbumCount; i++) { loudnessInfo(); } for (i=0; i<loudnessInfoCount; i++) { loudnessInfo(); } loudnessInfoSetExtPresent; if (loudnessInfoSetExtPresent==1) { loudnessInfoSetExtension(); } }	6 6 1	uimsbf uimsbf bslbf

Table 2: Syntax of `loudnessInfoSet()`-element

Syntax	No. of bits	Mnemonic
<pre> uniDrc() { uniDrcLoudnessInfoSetPresent; if (uniDrcLoudnessInfoSetPresent==1) { uniDrcConfigPresent; if (uniDrcConfigPresent==1) { uniDrcConfig(); } loudnessInfoSet(); } uniDrcGain(); } </pre>	1 1	bslbf bslbf

Table 3: Syntax of uniDRC()-element

5

Symbol	Value of loudnessInfoSetExtType	Purpose
UNIDRCLOUDEXT_TERM	0×0	Termination tag
UNIDRCLOUDEXT_EQ	0×1	Extension for equalization
UNIDRCLOUDEXT_DYNLOUDCOMP	0×2	Extension for dynamic processing
(reserved)	(All remaining values)	For future use

*Table 4: loudnessInfoSet extension types*10 *New dynLoudComp():*

Syntax	No. of bits	Mnemonic
dynLoudComp() { levelingPresent = 1 drcSetId; eqSetId; downmixId; mae_groupID; mae_groupPresetID; dynLoudCompValue; }	 6 6 7 7 5 10	 uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf

Table 5: Syntax of dynLoudComp()-element

- The drcSetId enables dynLoudComp (relating to the metadata) to be applied per DRC-set.

- The eqSetId enables dynLoudComp to be applied in combination with different settings for the equalization tool.
- The downmixId enables dynLoudComp to be applied per DownmixID.
- The mae_groupID enables dynLoudComp to be applied depending on personalization settings set at the device. The mae_groupPresetID enables dynLoudComp to be applied depending on personalization preset settings set at the device.

[0123] In some cases, in addition to the above parameters, it may be beneficial for the dynLoudComp() element to also include a **methodDefinition** parameter (specified by, e.g., 4 bits) specifying a loudness measurement method used for deriving the dynamic program loudness metadata (e.g., anchor loudness, program loudness, short-term loudness, momentary loudness, etc.) and/or a **measurementSystem** parameter (specified by, e.g., 4 bits) specifying a loudness measurement system used for measuring the dynamic program loudness metadata (e.g., EBU R.128, ITU-R BS-1770 with or without preprocessing, ITU-R BS-1771, etc.). Such parameters may, e.g., be included in the dynLoudComp() element between the **downmixId** and **dynLoudCompValue** parameters.

Alternative Syntax 1

Syntax	No. of bits	Mnemonic
loudnessInfoSetExtension() { while (loudnessInfoSetExtType != UNIDRCLOUDEXTERM) { extSizeBits = bitSizeLen + 4; extBitSize = bitSize + 1; switch (loudnessInfoSetExtType) { UNIDRCLOUDEXTEQ: loudnessInfoV1AlbumCount ; loudnessInfoV1Count ; for (i=0; i<loudnessInfoV1AlbumCount; i++) { loudnessInfoV1(); } for (i=0; i<loudnessInfoV1Count; i++) { loudnessInfoV1(); } } } }	4 4 extSizeBits 6 6	uimsbf uimsbf uimsbf uimsbf uimsbf

break;		
UNIDRCLOUDEX_T_DYNLOUDCOMP:		
loudnessInfoV2Count;	6	uimsbf
for (i=0; i< loudnessInfoV2Count; i++) {		
loudnessInfoV2();		
}		
break;		
/* add future extensions here */		
default:		
for (i=0; i<extBitSize; i++) {		
otherBit;	1	bslbf
}		
}		
}		
}		

Table 6: Syntax of loudnessInfoSetExtension()-element

Symbol	Value of loudnessInfoSetExtType	Purpose
UNIDRCLOUDEX_T_TERM	0×0	Termination tag
UNIDRCLOUDEX_T_EQ	0×1	Extension for equalization
UNIDRCLOUDEX_T_DYNLOUDCOMP	0×2	Extension for dynamic processing
(reserved)	(All remaining values)	For future use

Table 7: loudnessInfoSet extension types

Syntax	No. of bits	Mnemonic
loudnessInfoV2()		
{		
drcSetId;	6	uimsbf
eqSetId;	6	uimsbf
downmixId;	7	uimsbf
samplePeakLevelPresent;	1	bslbf
if (samplePeakLevelPresent==1) {		

bsSamplePeakLevel;	12	uimsbf
}		
truePeakLevelPresent;	1	bslbf
if (truePeakLevelPresent==1) {		
bsTruePeakLevel;	12	uimsbf
measurementSystem;	4	uimsbf
reliability;	2	uimsbf
}		
measurementCount;	4	uimsbf
for (i=0; i<measurementCount; i++) {		
methodDefinition;	4	uimsbf
methodValue;	2..8	vlclbf
measurementSystem;	4	uimsbf
reliability;	2	uimsbf
}		
levelingPresent;	1	bslbf
if (levelingPresent==1) {		
dynLoudCompValue;	10	uimsbf
}		
}		

Table 8: Syntax of loudnessInfoV2() payload

- [0124] In some cases, it may be beneficial to modify the syntax shown above in Table 8 such that the **levelingPresent** and (if levelingPresent==1) **dynLoudCompValue** parameters follow the **reliability** parameter within the measurementCount loop of the loudnessInfoV2() payload, rather than being outside the measurementCount loop. Furthermore, it may also be beneficial to set dynLoudCompValue equal to 0 in cases where levelingPresent is 0.

Alternative Syntax 2

- [0125] Alternatively, the dynLoudComp()-element could be placed into the uniDrcGainExtension()-element.

Syntax	No. of bits	Mnemonic
uniDrcGain()		

<pre> { nDrcGainSequences = gainSequenceCount; /* from drcCoefficientsUniDrc or drcCoefficientsUniDrc V1 */ for (s=0; s<nDrcGainSequences; s++) { if (gainCodingProfile[s]<3) { drcGainSequence(); } } uniDrcGainExtPresent; if (uniDrcGainExtPresent==1) { uniDrcGainExtension(); } } </pre>		
	1	bslbf

Table 9: Syntax of uniDrcGain()-element

Syntax	No. of bits	Mnemonic
<pre> uniDrcGainExtension() { while (uniDrcGainExtType != UNIDRCGAINEXT_TERM) { extSizeBits = bitSizeLen + 4; extBitSize = bitSize + 1; switch (uniDrcGainExtType) { UNIDRCLOUDEXT_DYNLOUDCOMP: dynLoudCompCount; for (i=0; i<dynLoudCompCount; i++) { dynLoudComp(); } break; /* add future extensions here */ default: for (i=0; i<extBitSize; i++) { otherBit; } } } } </pre>		
	4	uimsbf
	3	uimsbf
	extSizeBits	uimsbf
	6	uimsbf
	1	bslbf

}

Table 10: Syntax of *uniDrcGainExtension()*-element

Symbol	Value of <i>uniDrcGainExtType</i>	Purpose
UNIDRCGAINEXT_TERM	0x0	Termination tag
UNIDRCGAINEXT_DYNLOUDCOMP	0x1	Extension for dynamic processing
(reserved)	(All remaining values)	For future use

Table 11: *UniDrc* gain extension types

5

Semantics

dynLoudCompValue This field contains the value for *dynLoudCompDb*. The values are encoded according to Table 12. The default value is 0 dB.

Encoding	Size	Mnemonic	Value in [dB]	Approximate range
μ	10 bits	uimsbf	$dynLoudCompDb = -16 + \mu 2^{-5}$	-16 ... 16 dB, 0.0312 dB step size

Table 12: Coding of *dynLoudCompValue* field

10 [0126] In one example, the *dynLoudCompValue* may be transported in the bitstream using syntax for transporting DRC gains related to a DRC Set.

Updated MPEG-D DRC Loudness Normalization Processing

```

if (targetLoudnessPresent) {
    if (levelingPresent) {
        loudnessNormalizationGainDb = targetLoudness – contentLoudness +
dynLoudCompDb;
    } else {
        loudnessNormalizationGainDb = targetLoudness – contentLoudness;
    }
} else {
    loudnessNormalizationGainDb = 0.0;
}
if (loudnessNormalizationGainDbMaxPresent) {
    loudnessNormalizationGainDb = min(loudnessNormalizationGainDb,
loudnessNormalizationGainDbMax);
}
if (loudnessNormalizationGainModificationDbPresent) {
    gainNorm = pow(2.0, (loudnessNormalizationGainDb +
loudnessNormalizationGainModificationDb) / 6);
} else {
    gainNorm = pow(2.0, loudnessNormalizationGainDb / 6);
}
for (t=0; t<drcFrameSize; t++) {
    for (c=0; c<nChannels; c++) {
        audioSample[c][t] = gainNorm * audioSample[c][t];
    }
}

```

Table 13: Loudness normalization processing

Pseudo-Code for selection and processing of dynLoudComp

- [0127]** Additional parameters, such as *mae_groupID* or *mae_PresetID*, identifying different combinations of audio elements for user personalization settings, may be used for the selection process.

/* Selection Process */

/* The following settings would be derived from user/decoder settings

drcSetId = 1;

10 eqSetID = 2;

downmixId = 3;

mae_groupID = 4;

mae_groupPresetID = 5;

```

*/

findMatchingDynLoudComp (drcSetId, eqSetID, downmixId) {

    dynLoudComp = UNDEFINED;

    /* Check if matching loudnessInfo set is present */

5     if (targetLoudnessPresent(drcSetId, eqSetID, downmixId, mae_groupID,
    mae_groupPresetID)) continue;

    else {

        levelingPresent = false;

        exit(0);

10    }

    /* If all values are defined */

    if (drcSetId != UNDEFINED && eqSetID != UNDEFINED && downmixId !=
    UNDEFINED) {

        for (num=0; num<num_of_dynLoudCompValues; num++) {

15            if (dynLoudCompArray[num].drcSetId == drcSetId &&

                dynLoudCompArray[num].eqSetID == eqSetID &&

                dynLoudCompArray[num].downmixId == downmixId) {

                /* Correct entry found, assign to dynLoudComp */

                dynLoudComp = dynLoudCompArray[num].dynLoudCompValue;

20            }

        }

    }

    else if (drcSetId == UNDEFINED) {

        for (num=0; num<num_of_dynLoudCompValues; num++) {

25            if (dynLoudCompArray[num].eqSetID == eqSetID &&

```

```
        dynLoudCompArray[num].downmixId == downmixId) {  
            /* Correct entry found, assign to dynLoudComp */  
            dynLoudComp = dynLoudCompArray[num].dynLoudCompValue;  
        }  
5    }  
    }  
    else if (eqSetID == UNDEFINED) {  
        for (num=0; num<num_of_dynLoudCompValues; num++) {  
            if (dynLoudCompArray[num].drcSetId == drcSetId &&  
10        dynLoudCompArray[num].downmixId == downmixId) {  
                /* Correct entry found, assign to dynLoudComp */  
                dynLoudComp = dynLoudCompArray[num].dynLoudCompValue;  
            }  
        }  
15    }  
    else if (downmixId == UNDEFINED) {  
        for (num=0; num<num_of_dynLoudCompValues; num++) {  
            if (dynLoudCompArray[num].drcSetId == drcSetId &&  
20        dynLoudCompArray[num].eqSetID == eqSetID) {  
                /* Correct entry found, assign to dynLoudComp */  
                dynLoudComp = dynLoudCompArray[num].dynLoudCompValue;  
            }  
        }  
    }  
}
```

```
else if (drcSetId == UNDEFINED && downmixId == UNDEFINED) {  
    for (num=0; num<num_of_dynLoudCompValues; num++) {  
        if (dynLoudCompArray[num].eqSetID == eqSetID) {  
            /* Correct entry found, assign to dynLoudComp */  
5            dynLoudComp = dynLoudCompArray[num].dynLoudCompValue;  
        }  
    }  
}  
  
else if (drcSetId == UNDEFINED && eqSetID == UNDEFINED) {  
10    for (num=0; num<num_of_dynLoudCompValues; num++) {  
        if (dynLoudCompArray[num].downmixId == downmixId) {  
            /* Correct entry found, assign to dynLoudComp */  
            dynLoudComp = dynLoudCompArray[num].dynLoudCompValue;  
        }  
15    }  
}  
  
else if (eqSetID == UNDEFINED && downmixId == UNDEFINED) {  
    for (num=0; num<num_of_dynLoudCompValues; num++) {  
        if (dynLoudCompArray[num].eqSetID == eqSetID) {  
20            /* Correct entry found, assign to dynLoudComp */  
            dynLoudComp = dynLoudCompArray[num].dynLoudCompValue;  
        }  
    }  
}
```

- ```
 if (dynLoudComp == UNDEFINED){
 levelingPresent = false;
 exit(1);
 }
5 else {
 levelingPresent = true;
 exit(0);
 }
}
```
- 10 **[0128]** In some cases, in addition to the selection process shown in pseudo-code above (e.g., taking drcSetId, eqSetID, downmixId, mae\_groupID and mae\_PresetID into consideration for selecting a dynLoudCompValue parameter), it may be beneficial for the selection process to also take a **methodDefinition** parameter and/or a **measurementSystem** parameter into consideration for selecting a dynLoudCompValue parameter.
- 15 *Alternative Updated MPEG-D DRC Loudness Normalization Processing*

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>if (targetLoudnessPresent) {     loudnessNormalizationGainDb = targetLoudness – contentLoudness + dynLoudCompDb; } else {     loudnessNormalizationGainDb = 0.0; } if (loudnessNormalizationGainDbMaxPresent) {     loudnessNormalizationGainDb = min(loudnessNormalizationGainDb, loudnessNormalizationGainDbMax); } if (loudnessNormalizationGainModificationDbPresent) {     gainNorm = pow(2.0, (loudnessNormalizationGainDb + loudnessNormalizationGainModificationDb) / 6); } else {     gainNorm = pow(2.0, loudnessNormalizationGainDb / 6); } for (t=0; t&lt;drcFrameSize; t++) {     for (c=0; c&lt;nChannels; c++) {         audioSample[c][t] = gainNorm * audioSample[c][t];     } }</pre> |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Table 14: Alternative Loudness normalization processing

[0129] In cases where the alternative loudness normalization processing of Table 14 above is used, the loudness normalization processing pseudo-code described above may be replaced by the following alternative loudness normalization processing pseudo-code. Note that a default value of dynLoudCompDb, e.g., 0 dB, may be assumed to ensure that the value of dynLoudCompDb is defined, even for cases where loudness leveling metadata is not present in the bitstream.

Alternative Syntax 3

[0130] In some cases, it may be beneficial to combine the syntax described above in Table 1 – Table 5 with Alternative Syntax 1 described above in Table 6 – Table 8, as shown in the following Tables, to allow increased flexibility for transmission of the loudness leveling values.

| Syntax                          | No. of bits | Mnemonic |
|---------------------------------|-------------|----------|
| loudnessInfoSetExtension()<br>{ |             |          |

|                                                              |             |        |
|--------------------------------------------------------------|-------------|--------|
| while ( <b>loudnessInfoSetExtType</b> != UNIDRCLOUDEXTERM) { | 4           | uimbsf |
| extSizeBits = <b>bitSizeLen</b> + 4;                         | 4           | uimbsf |
| extBitSize = <b>bitSize</b> + 1;                             | extSizeBits | uimbsf |
| switch (loudnessInfoSetExtType) {                            |             |        |
| UNIDRCLOUDEXTEQ:                                             |             |        |
| <b>loudnessInfoV1AlbumCount</b> ;                            | 6           | uimbsf |
| <b>loudnessInfoV1Count</b> ;                                 | 6           | uimbsf |
| for (i=0; i<loudnessInfoV1AlbumCount; i++) {                 |             |        |
| loudnessInfoV1();                                            |             |        |
| }                                                            |             |        |
| for (i=0; i<loudnessInfoV1Count; i++) {                      |             |        |
| loudnessInfoV1();                                            |             |        |
| }                                                            |             |        |
| break;                                                       |             |        |
| UNIDRCLOUDEXTDYNLOUDCOMP:                                    |             |        |
| <b>loudnessInfoV2Count</b> ;                                 | 6           | uimbsf |
| for (i=0; i<loudnessInfoV2Count; i++) {                      |             |        |
| loudnessInfoV2();                                            |             |        |
| }                                                            |             |        |
| break;                                                       |             |        |
| UNIDRCLOUDEXTDYNLOUDCOMP2:                                   |             |        |
| <b>dynLoudCompCount</b> ;                                    | 6           | uimbsf |
| for (i=0; i<dynLoudCompCount; i++) {                         |             |        |
| dynLoudComp();                                               |             |        |
| }                                                            |             |        |
| break;                                                       |             |        |
| /* add future extensions here */                             |             |        |
| default:                                                     |             |        |
| for (i=0; i<extBitSize; i++) {                               |             |        |
| <b>otherBit</b> ;                                            | 1           | bslbf  |
| }                                                            |             |        |
| }                                                            |             |        |
| }                                                            |             |        |

Table 15: Alternate Syntax 3 of loudnessInfoSetExtension()-element

| Symbol                     | Value of loudnessInfoSetExtType | Purpose                          |
|----------------------------|---------------------------------|----------------------------------|
| UNIDRCLOUDEST_TERM         | 0×0                             | Termination tag                  |
| UNIDRCLOUDEST_EQ           | 0×1                             | Extension for equalization       |
| UNIDRCLOUDEST_DYNLOUDCOMP  | 0×2                             | Extension for dynamic processing |
| UNIDRCLOUDEST_DYNLOUDCOMP2 | 0×3                             | Extension for dynamic processing |
| (reserved)                 | (All remaining values)          | For future use                   |

Table 16: Alternate Syntax 3 of loudnessInfoSet extension types

| Syntax                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | No. of bits                                        | Mnemonic                                                                                                     |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------|--------------------------------------------------------------------------------------------------------------|
| <pre> loudnessInfoV2() {     drcSetId;     eqSetId;     downmixId;     samplePeakLevelPresent;     if (samplePeakLevelPresent==1) {         bsSamplePeakLevel;     }     truePeakLevelPresent;     if (truePeakLevelPresent==1) {         bsTruePeakLevel;         measurementSystem;         reliability;     }     measurementCount;     for (i=0; i&lt;measurementCount; i++) {         methodDefinition;         methodValue;         measurementSystem;         reliability;         levelingPresent;         if (levelingPresent==1) {             dynLoudCompValue; </pre> | <pre> 6 6 7 1 12 1 12 4 2 4 4 2..8 4 2 1 10 </pre> | <pre> uimsbf uimsbf uimsbf bslbf uimsbf bslbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf </pre> |

```

 }
 else {
 dynLoudCompValue=0;
 }
}
}

```

**Table 17: Alternate Syntax 3 of loudnessInfoV2() payload**

*Alternate Syntax 3 of dynLoudComp():*

| Syntax                                                                                                                                                                                                                                                     | No. of bits                                       | Mnemonic                                                                                                                                         |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| dynLoudComp()<br>{<br><i>levelingPresent = 1</i><br><b>drcSetId;</b><br><b>eqSetId;</b><br><b>downmixId;</b><br><b>mae_groupID;</b><br><b>mae_groupPresetID;</b><br><b>methodDefinition;</b><br><b>measurementSystem;</b><br><b>dynLoudCompValue;</b><br>} | <br><br><br>6<br>6<br>7<br>7<br>5<br>4<br>4<br>10 | <br><br><br><b>uimsbf</b><br><b>uimsbf</b><br><b>uimsbf</b><br><b>uimsbf</b><br><b>uimsbf</b><br><b>uimsbf</b><br><b>uimsbf</b><br><b>uimsbf</b> |

**Table 18: Alternate Syntax 3 of dynLoudComp()-elementInterface Extension Syntax**

- [0131] In some cases, it may be beneficial to allow control, e.g., by an end user, of whether or not loudness leveling is performed, even when loudness leveling information is present in a received bitstream. Such control may be provided by updating the MPEG-D DRC interface syntax to include an additional interface extension (e.g., UNIDRCINTERFACEEXT\_LEVELING) which contains a loudness leveling control interface payload (e.g., levelingControlInterface()) as shown in the following tables.

| Syntax                                                                                                                                             | No. of bits    | Mnemonic                               |
|----------------------------------------------------------------------------------------------------------------------------------------------------|----------------|----------------------------------------|
| uniDrcInterfaceExtension()<br>{<br>while ( <b>uniDrcInterfaceExtType</b> !=UNIDRCINTERFACEEXT_TERM) {<br>extSizeBits = <b>bitSizeLen</b> + 4;<br>} | <br><br>4<br>4 | <br><br><b>uimsbf</b><br><b>uimsbf</b> |

| Syntax                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | No. of bits                                                               | Mnemonic                                                                         |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------|----------------------------------------------------------------------------------|
| <pre> extBitSize = <b>bitSize</b> + 1; switch (uniDrcInterfaceExtType) {     UNIDRCINTERFACEEXT_EQ:         <b>loudnessEqParameterInterfacePresent;</b>         if (loudnessEqParameterInterfacePresent == 1) {             loudnessEqParameterInterface();         }         <b>equalizationControlInterfacePresent;</b>         if (equalizationControlInterfacePresent == 1) {             equalizationControlInterface();         }         break;     UNIDRCINTERFACEEXT_LEVELING:         levelingControlInterface();         break;     /* add future extensions here */     default:         for (i=0; i&lt;extBitSize; i++) {             <b>otherBit;</b>         }     } } </pre> | <p><b>extSizeBits</b></p> <p><b>1</b></p> <p><b>1</b></p> <p><b>1</b></p> | <p><b>uimsbf</b></p> <p><b>bslbf</b></p> <p><b>bslbf</b></p> <p><b>bslbf</b></p> |

*Table 19: Syntax of uniDRCInterfaceExtension() payload*

| Syntax                                                                     | No. of bits     | Mnemonic            |
|----------------------------------------------------------------------------|-----------------|---------------------|
| <pre> levelingControlInterface() {     <b>loudnessLevelingOn;</b> } </pre> | <p><b>1</b></p> | <p><b>bslbf</b></p> |

*Table 20: Syntax of levelingControlInterface() payload*

| Symbol                  | Value of uniDRCInterfaceExtType | Purpose              |
|-------------------------|---------------------------------|----------------------|
| UNIDRCINTERFACEEXT_TERM | 0×0                             | Termination tag      |
| UNIDRCINTERFACEEXT_EQ   | 0×1                             | Equalization Control |

|                             |                        |                          |
|-----------------------------|------------------------|--------------------------|
| UNIDRCINTERFACEEXT_LEVELING | 0×2                    | Dynamic Loudness Control |
| (reserved)                  | (All remaining values) | For future use           |

**Table 21: UniDRC Interface extension types**

### *Interface Extension Semantics*

#### **loudnessLevelingOn**

This flag signals if Loudness Leveling shall be switched on or off. If Loudness Leveling is switched on, this flag shall be equal to 1. If Loudness Leveling is switched off, this flag shall be equal to 0. The default value is 1.

### *Additional Methods of Enabling Loudness leveling*

- 5    **[0132]** In addition to the above, additional methods for enabling loudness leveling are possible. For example, instead of including a parameter which indicates a specific loudness leveling value to be applied to the signal, it is possible to define a specific set of DRC gains which already include the loudness leveling, such that, when those specific DRC gains are applied by a decoder, loudness leveling is applied. DRC gain sets suitable for applying such
- 10    loudness leveling may be identified through specific bitstream elements, as explained further below. A benefit of such an approach is that it is not necessary to transmit an explicit loudness leveling gain in addition to other loudness information.

**[0133]** Additionally, a user can specify through an interface that the DRC gains including loudness leveling should be selected and applied, as explained further below.

- 15    **[0134]** Finally, additional bitstream elements may be provided to allow tighter creative control over whether and how loudness leveling is performed. For instance, a flag may indicate whether or not loudness leveling may be switched OFF by a user. In such cases, if the content creator allows for loudness leveling to be disabled, then loudness leveling will be applied or not as specified by the user.

### 20    *Specifying DRC Sets with Loudness leveling through the DRC Set Effect Field*

- [0135]** In the MPEG-D DRC standard, it is possible to indicate one or more DRC effects which are provided by a specific DRC set. For instance, a DRC set might be appropriate for “Late Night” viewing, for viewing in “Noisy Environments”, for viewing at “Low Playback Levels”, etc. A DRC set effect parameter may indicate which specific effects are provided by
- 25    the DRC set.

[0136] Furthermore, the MPEG-D DRC standard allows a user to specify one or more desired DRC effects, as well as one or more optional fallback DRC effects. Such information from a user may be used to select a most appropriate DRC set from the available DRC sets. For example, if a DRC set exists which matches the desired DRC effects, then that set is selected.

- 5 If such a set does not exist, but there is a set that matches a fallback DRC effect, that set may be selected.

[0137] The list of defined DRC effects may be extended or updated to include a DRC effect which provides loudness leveling. Furthermore, the list of DRC effects which can be specified by a user through an interface may also be extended or updated to include a DRC

- 10 set effect which provides loudness leveling. As such, loudness leveling may be provided by including a DRC set indicated as providing a loudness leveling effect, and indicating, through an interface, a desire for a DRC set which provides loudness leveling, in which case, the decoder would select and apply the DRC set that corresponds to the loudness leveling DRC effect.

- 15 [0138] As described above, a previous version of a table specifying DRC Set Effects that may be signaled in a bitstream (e.g., Table A.45 of ISO/IEC 23003-4:2020) may be updated to include an entry for a “Loudness leveling” effect as shown below. Specifically, a row with a particular bit position (e.g. 13) corresponding to a Loudness leveling effect may be added as shown in the table below. Alternatively, an existing row with particular bit position (e.g. 12)
- 20 corresponding to a “Ducking/Leveling self” effect may be updated as shown in the table below.

| bit position | drcSetEffect           | Short name | Description                                                                                           |
|--------------|------------------------|------------|-------------------------------------------------------------------------------------------------------|
| 1 (LSB)      | Late night             | “Night”    | For quiet environment, listening at low level, avoiding to disturb others.                            |
| 2            | Noisy environment      | “Noisy”    | Optimized to get the best experience in noisy environments, for instance by amplifying soft sections. |
| 3            | Limited playback range | “Limited”  | Reduced dynamic range to improve quality on playback devices with limited dynamic range.              |
| 4            | Low playback level     | “LowLevel” | Listening at a low playback level.                                                                    |
| 5            | Dialog enhancement     | “Dialog”   | The main effect is a more prominent dialogue within the content.                                      |
| 6            | General compression    | “General”  | A DRC effect that reduces the dynamic range and is applicable to multiple playback scenarios.         |

| bit position                                                                                                                                            | drcSetEffect          | Short name        | Description                                                                                                                                                                                                                                                                                                                                    |
|---------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7                                                                                                                                                       | Dynamic expansion     | “Expand”          | Dynamics enhancement.                                                                                                                                                                                                                                                                                                                          |
| 8                                                                                                                                                       | Artistic effect       | “Artistic”        | To create an artistic sound effect.                                                                                                                                                                                                                                                                                                            |
| 9                                                                                                                                                       | Clipping prevention   | “Clipping”        | The main purpose is clipping prevention.                                                                                                                                                                                                                                                                                                       |
| 10                                                                                                                                                      | Fade-in/fade-out      | “Fade”            | Fade-in and fade-out envelope for gapless content (applicable when not playing in Album mode). It has no dynamic compression.                                                                                                                                                                                                                  |
| 11                                                                                                                                                      | Ducking other         | “Duck other”      | An effect that attenuates all audio content except for the channelGroup it is associated with. It has no dynamic compression.                                                                                                                                                                                                                  |
| 12                                                                                                                                                      | Ducking/Leveling self | “Duck/Level self” | An effect that amplifies and/or attenuates all channelGroups that it is associated with. It has the same channelGroup assignment as regular DRC gains.<br><br>In the case that this effect is used for Ducking only ( <i>levelingPresent</i> is undefined or equal to 0), this effect attenuates all channelGroups that it is associated with. |
| 13                                                                                                                                                      | Loudness leveling     | “Leveling”        | An effect that compensates for dynamic loudness changes in the decoder.                                                                                                                                                                                                                                                                        |
| <i>Remaining values are reserved</i>                                                                                                                    |                       |                   |                                                                                                                                                                                                                                                                                                                                                |
| NOTE Any information on DRC sets with “Ducking” effect in this document shall refer to both “Ducking other” and “Ducking self” if not stated otherwise. |                       |                   |                                                                                                                                                                                                                                                                                                                                                |

**Table 22: Examples of DRC Effects**

[0139] In the above example, a bit in a bitfield (e.g. a drcSetEffect bitfield of a drcInstructionsBasic() or a drcInstructionsUniDrc() or a drcInstructionsUniDrcV1() payload) associated with a particular DRC Set may be set (e.g., bit position 13 or bit position 12) to indicate that the DRC Set provides the Loudness leveling effect.

[0140] Similarly, a previous version of a table specifying DRC Set Effects that may be specified to a decoder through an interface (e.g., Table 12 of ISO/IEC 23003-4:2020) may be updated to include an entry for a Loudness leveling effect as shown below. Specifically, a row with a particular Index Value (e.g., 9) corresponding to a Loudness leveling effect may be added as shown in the table below.

| Index | drcSetEffect | Short name | Description                                                                                    |
|-------|--------------|------------|------------------------------------------------------------------------------------------------|
| 0     | None         | “None”     | A DRC set that has no dynamic compression, for instance “Clipping”, or applying no DRC set for |

| Index | drcSetEffect           | Short name  | Description                                                                                           |
|-------|------------------------|-------------|-------------------------------------------------------------------------------------------------------|
|       |                        |             | compression.                                                                                          |
| 1     | Late night             | "Night"     | For quiet environment, listening at low level, avoiding to disturb others.                            |
| 2     | Noisy environment      | "Noisy"     | Optimized to get the best experience in noisy environments, for instance by amplifying soft sections. |
| 3     | Limited playback range | "Limited"   | Reduced dynamic range to improve quality on playback devices with limited dynamic range.              |
| 4     | Low playback level     | "LowLevel"  | Listening at a low playback level.                                                                    |
| 5     | Dialog enhancement     | "Dialog"    | The main effect is a more prominent dialogue within the content.                                      |
| 6     | General compression    | "General"   | A DRC effect that reduces the dynamic range and is applicable to multiple playback scenarios.         |
| 7     | Dynamic expansion      | "Expand"    | Dynamics enhancement.                                                                                 |
| 8     | Artistic effect        | "Artistic"  | To create an artistic sound effect.                                                                   |
| 9     | Loudness leveling      | "LoudLevel" | Enables loudness leveling in the decoder.                                                             |

**Table 23: Examples of DRC Effects that may be specified through a interface**

[0141] In the above example, an interface parameter (e.g., a drcEffectTypeRequest parameter of a dynamicRangeControllerInterface() payload) may be set to the specific value (e.g., 9) that corresponds to the Loudness leveling Effect to instruct the decoder that, if available, a DRC Set which provides the Loudness leveling Effect set should be selected and applied. However, because loudness leveling may be intended for use in combination with loudness normalization, in some cases a decoder may require that in order to select and apply a DRC set that provides a loudness leveling effect, a decoder must also perform loudness normalization. Such a requirement could be accomplished by only allowing the decoder to select a DRC set that provides a loudness leveling effect when loudness normalization is also enabled in the decoder. Alternatively, or in addition, such a requirement could be satisfied by modifying the interface to require that when requesting a DRC set which provides a loudness leveling effect, loudness normalization must also be enabled (e.g., by setting both a loudness normalization on flag and a target loudness value).

**Specifying DRC Sets with Loudness leveling using new extension payloads**

[0142] Alternatively, a new payload (e.g., dynLoudInstructions()) including instructions for loudness leveling could be defined. One or more of such payloads could, e.g., be included in an extension field (e.g., UNIDRCCONFEXT\_V2) of an existing bitstream. By including such payloads in an extension field, compatibility with existing decoders that were not designed to

parse the new payload could be maintained, as such decoders would simply ignore the additional payload(s). Each of such payloads would be assigned a unique identifier (e.g., drcSetId) which corresponds to a DRC set that provides loudness leveling, and, to help easily identify such sets, a flag (e.g., a levelingPresent flag) could be set to 1 for each such DRC set signaled through the new payload.

**[0143]** Additionally, the user interface described above for selecting DRC sets that provide the loudness leveling effect could be used for DRC sets signaled through this new type of payload (e.g., dynLoudInstructions() payload) as well. Specifically, a user could indicate to the decoder through the interface (e.g., using the drcEffectTypeRequest field) that a DRC set that provides loudness leveling is desired. The decoder would then select, if present, a DRC set signaled through the new payloads (e.g., a DRC set having a DRC Set ID that is identified as corresponding to loudness leveling, and, for instance, having a levelingPresent flag set to 1). However, as noted above, because loudness leveling may be intended for use in combination with loudness normalization, in some cases a decoder may require that in order to select and apply a DRC set that provides a loudness leveling effect, a decoder must also perform loudness normalization. Such a requirement could be accomplished by only allowing the decoder to select a DRC set that provides a loudness leveling effect when loudness normalization is also enabled in the decoder. Alternatively, or in addition, such a requirement could be satisfied by modifying the interface to require that when requesting a DRC set which provides a loudness leveling effect, loudness normalization must also be enabled (e.g., by setting both a loudness normalization on flag and a target loudness value).

**[0144]** An advantage to using this type of signaling (e.g., instead of signaling DRC sets which provide loudness leveling through an existing parameter, such as a drcSetEffect parameter), is that, as described above, additional parameters indicating whether and how a user can enable and/or disable loudness leveling can be included in such new payloads. For instance, the new payload may also include a parameter (e.g., a dynamicLoudCompSwOffAllowed parameter) which indicates whether or not switching loudness leveling off through the user interface is allowed for each DRC set that applies loudness leveling.

**[0145]** An example syntax for a new payload (e.g., a dynLoudInstructions() payload) for loudness leveling is shown in Table 24.

| Syntax                | No. of bits | Mnemonic |
|-----------------------|-------------|----------|
| dynLoudInstructions() |             |          |

| Syntax                                       | No. of bits | Mnemonic      |
|----------------------------------------------|-------------|---------------|
| {                                            |             |               |
| <i>levelingPresent</i> = 1;                  |             |               |
| <b>drcSetId</b> ;                            | 6           | <b>uimsbf</b> |
| <b>drcLocation</b> ;                         | 4           | <b>uimsbf</b> |
| <b>downmixId</b> ;                           | 7           | <b>uimsbf</b> |
| <b>mae_groupID</b> ;                         | 7           | <b>uimsbf</b> |
| <b>mae_groupPresetID</b> ;                   | 5           | <b>uimsbf</b> |
| <b>additionalDownmixIdPresent</b> ;          | 1           | <b>bslbf</b>  |
| if (additionalDownmixIdPresent==1) {         |             |               |
| <b>additionalDownmixIdCount</b> ;            | 3           | <b>uimsbf</b> |
| for (j=0; j<additionalDownmixIdCount; j++) { |             |               |
| <b>additionalDownmixId</b> ;                 | 7           | <b>uimsbf</b> |
| }                                            |             |               |
| } else {                                     |             |               |
| additionalDownmixIdCount = 0;                |             |               |
| }                                            |             |               |
| <b>dynamicLoudCompSwOffAllowed</b> ;         | 1           | <b>bslbf</b>  |
| }                                            |             |               |

**Table 24: Example Syntax of a Dynamic Loudness Instructions Payload**

[0146] As described above, a unique DRC Set ID, as well as a flag (e.g., *levelingPresent*) which specifically indicates the DRC Set as a DRC set that provides loudness leveling, may be associated with the payload. Furthermore, the payload may be associated with one or more downmix identification parameters (e.g., *downmixId* parameters) which indicate that the DRC set is intended for use with one or more specific downmixes / downmix configurations of the audio program. Such DRC Set identifier (e.g., *drcSetId* parameter), as well as the flag (e.g., *levelingPresent* flag) allow for selection of the DRC Set associated with the payload in response to a user indication, provided through an interface, that a loudness leveling DRC effect is desired by the user. Additionally, the inclusion of one or more downmix identifiers (e.g., *downmixID* parameters) enables the selection of a DRC set that provides loudness leveling which is intended for use with a particular downmix or downmixes of the audio program. Furthermore, the payload may be associated with one or more personalization identification parameters (e.g., *mae\_groupID*, *mae\_groupPresetID*) which indicate that the DRC set is intended for use with one or more specific personalization configurations of the audio program. Finally, as also described above, the inclusion of a parameter (e.g.,

dynamicLoudComSwOffAllowed) indicating whether or not turning off loudness leveling is allowed, enables a content creator additional control over whether or not loudness leveling may be applied by decoders.

**[0147]** Finally, as indicated above, such additional payloads (e.g., dynLoudInstructions()

- 5 payloads) may be contained in an extension payload of a configuration payload (e.g., a uniDrcConfigExtension() payload). A specific extension type (e.g., a uniDrcConfigExtType of UNIDRCCONFEXT\_V2 having a value of 0x2) may be used to indicate the presence of such additional payloads within the extension field of the configuration payload.

- [0148]** An example of a syntax for an extension field of a configuration payload (e.g., a uniDrcConfigExtension() payload) is shown in Table 25.

| Syntax                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | No. of bits                                                | Mnemonic                                                                               |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------|----------------------------------------------------------------------------------------|
| uniDrcConfigExtension()<br>{<br>while ( <b>uniDrcConfigExtType</b> != UNIDRCCONFEXT_TERM) {<br>extSizeBits = <b>bitSizeLen</b> + 4;<br>extBitSize = <b>bitSize</b> + 1;<br>switch (uniDrcConfigExtType) {<br>case UNIDRCCONFEXT_PARAM_DRC:<br>drcCoefficientsParametricDrc();<br><b>parametricDrcInstructionsCount</b> ;<br>for (i=0; i<parametricDrcInstructionsCount; i++) {<br>parametricDrcInstructions ();<br>}<br>break;<br>case UNIDRCCONFEXT_V1:<br><b>downmixInstructionsV1Present</b> ;<br>if (downmixInstructionsV1Present==1) {<br><b>downmixInstructionsV1Count</b> ;<br>for (i=0; i<downmixInstructionsV1Count; i++) {<br>downmixInstructionsV1();<br>}<br>}<br><b>drcCoeffsAndInstructionsUniDrcV1Present</b> ;<br>if (drcCoeffsAndInstructionsUniDrcV1Present==1) {<br><b>drcCoefficientsUniDrcV1Count</b> ;<br>for (i=0; i<drcCoefficientsUniDrcV1Count; i++) {<br>drcCoefficientsUniDrcV1();<br>}<br>}<br>}<br>}<br>}<br>} | 4<br>4<br>extSizeBits<br><br>4<br><br>1<br>7<br><br>1<br>3 | uimsbf<br>uimsbf<br>uimsbf<br><br>uimsbf<br><br>bslbf<br>uimsbf<br><br>bslbf<br>uimsbf |

| Syntax                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | No. of bits | Mnemonic |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|----------|
| <pre>     }     drcInstructionsUniDrcV1Count;     for (i=0; i&lt;drcInstructionsUniDrcV1Count; i++) {         drcInstructionsUniDrcV1();     } } </pre>                                                                                                                                                                                                                                                                                                                       | 6           | uimsbf   |
| <pre>     loudEqInstructionsPresent;     if (loudEqInstructionsPresent==1) {         loudEqInstructionsCount;         for (i=0; i&lt;loudEqInstructionsCount; i++) {             loudEqInstructions();         }     } } </pre>                                                                                                                                                                                                                                               | 1           | bslbf    |
| <pre>     eqPresent;     if (eqPresent==1) {         eqCoefficients();         eqInstructionsCount;         for (i=0; i&lt;eqInstructionsCount; i++) {             eqInstructions();         }     } } break; case UNIDRCCONFEXT_V2:     downmixInstructionsV1Present;     if (downmixInstructionsV1Present==1) {         downmixInstructionsV1Count;         for (i=0; i&lt;downmixInstructionsV1Count; i++) {             downmixInstructionsV1();         }     } } </pre> | 1           | bslbf    |
| <pre>     drcCoeffsAndInstructionsUniDrcV1Present;     if (drcCoeffsAndInstructionsUniDrcV1Present==1) {         drcCoefficientsUniDrcV1Count;         for (i=0; i&lt;drcCoefficientsUniDrcV1Count; i++) {             drcCoefficientsUniDrcV1();         }     }     drcInstructionsUniDrcV1Count;     for (i=0; i&lt;drcInstructionsUniDrcV1Count; i++) { </pre>                                                                                                            | 4           | uimsbf   |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 4           | uimsbf   |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 1           | bslbf    |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 7           | uimsbf   |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 1           | bslbf    |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 3           | uimsbf   |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 6           | uimsbf   |

| Syntax                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | No. of bits                                                    | Mnemonic                                                                                                                                       |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> drcInstructionsUniDrcV1();     } } <b>loudEqInstructionsPresent;</b> if (loudEqInstructionsPresent==1) {     <b>loudEqInstructionsCount;</b>     for (i=0; i&lt;loudEqInstructionsCount; i++) {         loudEqInstructions();     } } <b>eqPresent;</b> if (eqPresent==1) {     eqCoefficients();     <b>eqInstructionsCount;</b>     for (i=0; i&lt;eqInstructionsCount; i++) {         eqInstructions();     } } <b>dynLoudPresent;</b> if (dynLoudPresent ==1) {     <b>dynLoudInstructionsCount;</b>     for (i=0; i&lt;eqInstructionsCount; i++) {         dynLoudInstructions();     } } break; /* add future extensions here */ default:     for (i=0; i&lt;extBitSize; i++) {         <b>otherBit;</b>     } } } </pre> | <p>1</p> <p>4</p> <p>1</p> <p>4</p> <p>1</p> <p>4</p> <p>1</p> | <p><b>bslbf</b></p> <p><b>uimsbf</b></p> <p><b>bslbf</b></p> <p><b>uimsbf</b></p> <p><b>bslbf</b></p> <p><b>uimsbf</b></p> <p><b>bslbf</b></p> |

**Table 25: Example for Updated Syntax of a Configuration Payload including Extension Fields**

[0149] The data associated with the specific extension type of the extension field of the configuration payload contains a parameter which indicates whether loudness leveling instructions are present (e.g., a *dynLoudPresent* parameter), and if so, a parameter which indicates the number of sets of loudness leveling instructions in the extension payload (e.g., a *dynLoudInstructionsCount* parameter), and each set of loudness leveling instructions (e.g., each *dynLoudInstructions()* payload), where the syntax for the loudness leveling instructions may be as shown in Table 25.

*Signaling DRC Sets for Loudness leveling through existing configuration elements*

[0150] Still alternatively, it may be possible to indicate that a particular DRC set provides loudness leveling by adding a new field to an existing configuration element. For example, a new parameter (e.g., a *levelingPresent* parameter) that indicates whether or not a DRC set provides loudness leveling could be included in an existing configuration element (e.g., in a *downmixInstructions()* element, a *drcInstructionsBasic()* element, or a *drcInstructionsUniDrc()* element). Ideally, to avoid compatibility issues with legacy decoders, the new element could be included in a field which is already ignored by legacy decoders (e.g., in a reserved field). Alternatively, the new parameter could be included in an updated version of an existing configuration element (e.g., in a *downmixInstructionsV2()* element or a *drcInstructionsUniDrcV2()* element), which would be ignored by legacy decoders.

[0151] In one example, signaling DRC Sets for loudness leveling through existing configuration elements may be using the extension of the *uniDrcConfigExtension()*-element shown in Table 26, in order to carry instructions to control and apply loudness leveling metadata received as dynamic processing metadata as a frame-based update, based on DRC Sets.

[0152] A switch-case *UNIDRCCONFEXT\_LEVELING* may be added in the *uniDrcConfigExtension()*-element as shown in Table 26. The switch-case *UNIDRCCONFEXT\_LEVELING* may be used to identify new elements including *levelingPresent* and *duckingOnlyDrcSetPresent*. The *uniDrcConfigExtension()*-element may be an extension of the *uniDrcConfig()*-element as described in ISO/IEC 23003-4. The new parameters including *levelingPresent* and *duckingOnlyDrcSetPresent* may be used for extending *drcInstructionsBasic()*, *drcInstructionsUniDrc()*- and *drcInstructionsUniDrcVI()*-elements as defined in ISO/IEC 23003-4.

[0153] The new elements including *levelingPresent* and *duckingOnlyDrcSetPresent* may be used for DRC set selection as described in ISO/IEC 23003-4, 6.3.7, as standalone parameters

or as parameters included in *drcInstructionsBasic()*, *drcInstructionsUniDrc()*- and *drcInstructionsUniDrcV1()*-elements.

- [0154] The leveling gains which need to be applied as part of the loudness leveling process (such as the parameter *dynLoudCompValue* in other syntax examples) may be included in the
- 5 bitstream using the respective *uniDrcGain()*-element.

| Syntax                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | No. of bits                                                                                       | Mnemonic                                                                                                                                                                                   |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> uniDrcConfigExtension() {     while (<b>uniDrcConfigExtType</b> != UNIDRCCONTEXT_TERM) {         extSizeBits = <b>bitSizeLen</b> + 4;         extBitSize = <b>bitSize</b> + 1;         switch (uniDrcConfigExtType) {             case UNIDRCCONTEXT_PARAM_DRC:                 drcCoefficientsParametricDrc();                 <b>parametricDrcInstructionsCount</b>;                 for (i=0; i&lt;parametricDrcInstructionsCount; i++) {                     parametricDrcInstructions ();                 }                 break;             case UNIDRCCONTEXT_V1:                 <b>downmixInstructionsV1Present</b>;                 if (downmixInstructionsV1Present==1) {                     <b>downmixInstructionsV1Count</b>;                     for (i=0; i&lt;downmixInstructionsV1Count; i++) {                         downmixInstructionsV1();                     }                 }                 <b>drcCoeffsAndInstructionsUniDrcV1Present</b>;                 if (drcCoeffsAndInstructionsUniDrcV1Present==1) {                     <b>drcCoefficientsUniDrcV1Count</b>;                     for (i=0; i&lt;drcCoefficientsUniDrcV1Count; i++) {                         drcCoefficientsUniDrcV1();                     }                     <b>drcInstructionsUniDrcV1Count</b>;                     for (i=0; i&lt;drcInstructionsUniDrcV1Count; i++) {                         drcInstructionsUniDrcV1();                     }                 }             }         }     } </pre> | <p>4</p> <p>4</p> <p><b>extSizeBits</b></p> <p>4</p> <p>1</p> <p>7</p> <p>1</p> <p>3</p> <p>6</p> | <p><b>uimsbf</b></p> <p><b>uimsbf</b></p> <p><b>uimsbf</b></p> <p><b>uimsbf</b></p> <p><b>bslbf</b></p> <p><b>uimsbf</b></p> <p><b>bslbf</b></p> <p><b>uimsbf</b></p> <p><b>uimsbf</b></p> |

| Syntax                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | No. of bits | Mnemonic |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|----------|
| <pre> } <b>loudEqInstructionsPresent;</b> if (loudEqInstructionsPresent==1) {     <b>loudEqInstructionsCount;</b>     for (i=0; i&lt;loudEqInstructionsCount; i++) {         loudEqInstructions();     } } </pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | 1           | bslbf    |
| <pre>     <b>loudEqInstructionsCount;</b>     for (i=0; i&lt;loudEqInstructionsCount; i++) {         loudEqInstructions();     } } </pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | 4           | uimsbf   |
| <pre> <b>eqPresent;</b> if (eqPresent==1) {     eqCoefficients();     <b>eqInstructionsCount;</b>     for (i=0; i&lt;eqInstructionsCount; i++) {         eqInstructions();     } } break; </pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | 1           | bslbf    |
| <pre>     <b>eqInstructionsCount;</b>     for (i=0; i&lt;eqInstructionsCount; i++) {         eqInstructions();     } } break; </pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | 4           | uimsbf   |
| <pre> case UNIDRCCONFEXT_LEVELING:     for (i=0; i&lt;drcInstructionsUniDrcV1Count; i++) {         dse = drcInstructionsUniDrcV1[i].drcSetEffect;         if ((dse &amp; (1&lt;&lt;11)) != 0) { /* if Ducking/Leveling self */             <b>levelingPresent;</b>             /* drcInstructionsUniDrcV1[i].levelingPresent */             if (levelingPresent==1) {                 <b>duckingOnlyDrcSetPresent;</b>                 /* drcInstructionsUniDrcV1[i].                 duckingOnlyDrcSetPresent */                 if (duckingOnlyDrcSetPresent ==1) {                     /* "Ducking only" DRC set */                     drcInstructionsUniDrcV1();                 }             }         }     }     break; /* add future extensions here */ default: </pre> | 1           | bslbf    |
| <pre>             <b>levelingPresent;</b>             /* drcInstructionsUniDrcV1[i].levelingPresent */             if (levelingPresent==1) {                 <b>duckingOnlyDrcSetPresent;</b>                 /* drcInstructionsUniDrcV1[i].                 duckingOnlyDrcSetPresent */                 if (duckingOnlyDrcSetPresent ==1) {                     /* "Ducking only" DRC set */                     drcInstructionsUniDrcV1();                 }             }         }     }     break; /* add future extensions here */ default: </pre>                                                                                                                                                                                                                          | 1           | bslbf    |

| Syntax                                                             | No. of bits | Mnemonic |
|--------------------------------------------------------------------|-------------|----------|
| <pre> for (i=0; i&lt;extBitSize; i++) {     otherBit; } } } </pre> | 1           | bslbf    |

**Table 26: Alternate Syntax of *uniDrcConfigExtension()*-element**

[0155] The semantics of the new elements including *levelingPresent* and *duckingOnlyDrcSetPresent* may be defined as shown in Table 27.

|                          |                                                                                                                                                                                                                       |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| levelingPresent          | A flag which signals if the gains in this DRC set contain data for loudness leveling. The default value shall be equal to 0.                                                                                          |
| duckingOnlyDrcSetPresent | A flag which signals if there is a "Ducking only" DRC set. If loudness leveling is switched off, "Ducking only" DRC sets shall be enabled for the DRC set selection procedure. The default value shall be equal to 0. |

**Table 27: Semantics of *levelingPresent* and *duckingOnlyDrcSetPresent* element**

- 5 [0156] Finally, similarly to the selection process described above, DRC Sets that are signaled as providing loudness leveling in this manner could be selected through an interface to the decoder by indicating that a DRC Set is desired that provides a loudness leveling effect. When such indication is provided through the interface to the decoder, the decoder will identify and select a DRC Set which is indicated as providing loudness leveling (e.g., which
- 10 has a levelingPresent parameter that equals 1). If multiple DRC sets are indicated as providing loudness leveling, a decoder may use other rules (e.g., predefined rules, such as those defined in ISO/IEC 23003-4:2020) for selecting the most appropriate DRC set of the multiple DRC sets that provide loudness leveling.

- [0157] However, because loudness leveling may be intended for use in combination with
- 15 loudness normalization, in some cases a decoder may require that in order to select and apply a DRC set that provides a loudness leveling effect, a decoder must also perform loudness normalization. Such a requirement could be accomplished by only allowing the decoder to select a DRC set that provides a loudness leveling effect when loudness normalization is also enabled in the decoder. Alternatively, or in addition, such a requirement could be satisfied by
- 20 modifying the interface to require that when requesting a DRC set which provides a loudness leveling effect, loudness normalization must also be enabled (e.g., by setting both a loudness normalization on flag and a target loudness value).

[0158] Figure 6 illustrates an example method of audio data for playback as described above.

Referring to the example of Figure 6, in step S301 the decoder 100 may receive a bitstream including encoded audio data and metadata, wherein the metadata includes one or more dynamic range control (DRC) sets, and for each DRC set, an indication of whether the DRC set is configured for providing a loudness leveling effect matching selected personalization experience, e.g., as described above. The metadata may include personalization information and/or indications as described herein.

[0159] While the format of the bitstream is not limited, in an embodiment, the bitstream may be an MPEG-D DRC bitstream. Alternatively, the bit stream may be an MPEG-H 3D audio compliant bit stream. The presence of metadata for providing a loudness leveling effect may then be signaled based on MPEG-D DRC bitstream syntax, e.g., as described above.

[0160] In step S302, the decoder may then parse the metadata to identify DRC sets that are configured for providing the loudness leveling effect matching the selected personalization experience, e.g., as described above. Moreover, in step S302 may parse the metadata to identify personalization metadata to allow for providing the loudness leveling effect matching the selected personalization experience.

[0161] In step S303, the decoder may then decode the audio data to obtain decoded audio data, e.g., as described above.

[0162] In step S304, the decoder may then select one of the identified DRC sets configured for providing the loudness leveling effect matching the selected personalization experience, e.g., as described above. This selection may be based on the personalization metadata identified at step S302.

[0163] In step S305, the decoder may then extract one or more DRC gains corresponding to the selected DRC set from the bitstream, e.g., as described above.

[0164] In step S306, the decoder may then apply the one or more DRC gains corresponding to the selected DRC set to the decoded audio data to obtain loudness leveled audio data, e.g., as described above.

[0165] Finally, in step S307, the loudness leveled audio data may then be output for playback, e.g., as described above.

#### *Efficient Coding / Transmission of Loudness leveling Data*

[0166] As described above, loudness leveling, in combination with dynamic range control, may be achieved by transmitting a DRC set that contains gains which are a combination of dynamic range control gains and loudness leveling gains. However, doing so may require redundant transmission of dynamic range control gain data. For instance, a first DRC set may

contain gains for applying dynamic range control to a signal, and it may be important to allow for that DRC gain set to be applied independently of any other gains (e.g., loudness leveling gains). Therefore, if a DRC set is desired which contains gains for applying a combination of dynamic range control and loudness leveling, then a second DRC set needs to be specified which represents a combination of the dynamic range control gains of the first set and the desired loudness leveling gains. Unfortunately, doing so requires that both the first and the second DRC sets contain information about the dynamic range control gains of the first set, which is inefficient.

**[0167]** A more efficient way of transmitting data to accomplish the same goal is to provide only the dynamic range control gains in a first DRC set, and only the loudness leveling gains in a second DRC set, along with some additional metadata indicating the relationship between the two DRC sets. For instance, the first DRC set may include a parameter which indicates that there is another DRC set which depends on the first DRCset. Then, a decoder may understand that it is possible to apply the gains of the first set independently, e.g., in case only dynamic range control is desired. Additionally, the decoder will understand that it is also possible to combine the gains of the first and second DRC sets in order to obtain a combination of dynamic range control and loudness leveling. Finally, in cases where the gains of the second set are never intended to be applied alone, metadata can be included with the second DRC set to indicate whether the gains of that DRC set may only be used in conjunction with the first set, or whether they may be used independently of the gains of the first set.

**[0168]** For example, consider 2 DRC sets, with a first DRC set identified as DRC Set 1 (e.g., has a drcSetId equal to 1) providing gains for dynamic range control, and a second DRC Set identified as DRC Set 2 (e.g., has a drcSetId equal to 2) providing gains for loudness leveling.

Since DRC Set 2 is intended for use in combination with DRC Set 1, DRC Set 1 could include a parameter indicating that DRC Set 2 depends on DRC Set 1 (e.g., a dependsOnDrcSet parameter of DRC Set 1 may be equal to 2). Furthermore, if loudness leveling is only intended for use in combination with dynamic range control, then DRC Set 2 could include a parameter indicating that the gains of DRC Set 2 may not be used independently (e.g., a noIndependentUse flag of DRC Set 2 may be set to a value 1). Conversely, if DRC Set 2 was not intended only for use in combination with dynamic range control, then DRC Set 2 could include a parameter indicating that the gains of DRC Set 2 may be used independently (e.g., a noIndependentUse flag of DRC Set 2 may be set to a value of 0).

[0169] The above described Efficient Coding/Transmission of Loudness leveling data may be used with any of the above described Additional Methods of Enabling Loudness leveling.

[0170] For instance, in the first case, where the presence of a DRC Set which provides a loudness leveling effect is signaled through an existing effect flag (e.g., a bit of a drcSetEffect bitfield), DRC Set 1 could have a drcSetEffect parameter in which the bit corresponding to the loudness leveling effect is set to 0, while DRC Set 2 could have the bit corresponding to the loudness leveling effect set to 1, and all other bits of the bitfield may be the same for both DRC Sets. Through such signaling, when a user requests a loudness leveling effect through the interface, a decoder will extract and apply the gains of DRC Set 1 and DRC Set 2 in order to apply a combination of dynamic range control and loudness leveling.

[0171] Table 28 shows exemplary values of parameters accomplishing the above signaling of efficient transmission of loudness leveling data.

| <i>Parameter</i>          | <i>DRC Set 1</i>                                         | <i>DRC Set 2</i>                                         |
|---------------------------|----------------------------------------------------------|----------------------------------------------------------|
| <i>drcSetID (Example)</i> | 0x01                                                     | 0x02                                                     |
| <i>dependsOnDrcSet</i>    | 0x02                                                     | Not set                                                  |
| <i>noIndependentUse</i>   | 0                                                        | 1                                                        |
| <i>drcSetEffect</i>       | “Dynamic compensation flag”<br>set to 0, otherwise same. | “Dynamic compensation flag”<br>set to 1, otherwise same. |

**Table 28: Exemplary Values of parameters for signaling efficient transmission of loudness leveling data**

[0172] In the second case, where the presence of a DRC Set which provides a loudness leveling effect is signaled using a new payload, DRC Set 2 would be signaled through the new payload (e.g., the drcSetId parameter of the dynLoudInstructions payload would be set to 2). DRC Set 1 would be signaled as described above (e.g., a drcSetId parameter would be set to 1, and a dependsonDrcSet parameter would be set to 2, indicating that DRC Set 2 depends on DRC Set 1). Finally, it would not be possible to indicate through the new payload described above (e.g., dynLoudInstructions) that DRC Set 2 should not be used independently (e.g., that it should always be used with DRC Set 1). As such, an additional parameter could be added to the new payload (e.g., a noIndependentUse parameter could be added to the dynLoudInstructions payload), such that, when the new parameter is set to 1, DRC Set 2 is only used with DRC Set 1, while when the new parameter is set to 0, DRC Set 2 may be used independently of DRC Set 1. Similarly to the first case, when a user requests a loudness leveling effect through the interface, a decoder will extract and apply the gains of both DRC Set 1 and DRC Set 2 in order to apply a combination of dynamic range control and

loudness leveling.

[0173] Finally, in the third case, existing configuration elements are used which already include all of the necessary parameters for efficiently enabling loudness leveling as described above. Specifically, DRC Set 1 would include an indication that DRC Set 2 depends on DRC Set 1 as described above, and DRC Set 2 would include an indication of whether or not it may be used independently, or only in conjunction with DRC Set 1. Again, similarly to the first two cases, when a user requests a loudness leveling effect through the interface, a decoder will extract and apply the gains of DRC Set 1 and DRC Set 2 in order to apply a combination of dynamic range control and loudness leveling.

[0174] As indicated above, a benefit of using dependent DRC Sets to enable loudness leveling is that doing so allows for more efficient transmission of the data required for loudness leveling, because it is eliminates the need to transmit redundant dynamic range control gains in the DRC Set which contains the loudness leveling gains.

*Differences for DRC Implementation of MPEG-D DRC in MPEG-H 3D Audio*

[0175] As described above, references herein to MPEG-D and MPEG-D DRC references the standard in ISO/IEC 23003-4, which is also identified as Information technology — MPEG audio technologies — Part 4: Dynamic Range Control. This standard is further incorporated into the MPEG-H 3D audio standard specified as ISO/IEC 23008-3 (MPEG-H Part 3). The MPEG-H 3D audio standard is an audio coding standard developed by MPEG to support coding audio as audio channels, audio objects, or higher order ambisonics (HOA).

[0176] Although the MPEG-H 3D audio standard references the MPEG-D standard, in order to provide appropriate technological compatibility, it includes several modifications to the syntax and technology of the MPEG-D standard. In particular, the main differences for MPEG-D DRC in MPEG-H Audio that are relevant to the present invention include:

- In MPEG-D DRC (and therefore in MPEG-D USAC, specified as ISO/IEC 23003-3), DRC instructions are defined in the Dynamic Loudness Control Profile, using the *drcInstructionsUniDrcVI()*-syntax element. In the Low Complexity and Baseline Profile of MPEG-H 3D audio, DRC instructions are defined using the *drcInstructionsUniDrc()*-syntax element.
- MPEG-H 3D audio offers a possibility for personalizing the user experience. Therefore, there are additional steps for the DRC set selection process, as defined in 6.4.4.2 and 6.4.4.3 of ISO/IEC 23008-3 (MPEG-H 3D audio).
- The MPEG-D DRC decoder interface in MPEG-H 3D audio is realized via a

bitstream interface based on MPEG-H Audio Stream (MHAS) packets, using the Packet Type of PACTYP\_LOUDNESS\_DRC.

*Inventive Syntax Updates to MPEG-H 3D Audio*

- 5 **[0177]** As MPEG-H 3D audio uses the *drcInstructionsUniDrc()*-syntax element to define instructions used for the DRC Set selection process, the *uniDrcConfigExtension()* syntax shown in Table 26 needs to be updated as shown in Table 29, and particularly the inventive element illustrated in the for loop of “for (i=0; i<drcInstructionsUniDrcCount; i++). Such syntax may be part of the playback condition considered in the figures, such as Figs. 2 and 6.
- 10 **[0178]** This additional syntax enables mapping the DRC instruction parameters needed for loudness leveling (e.g., *levelingPresent* and *duckingOnlyDrcSetPresent*) to the DRC instruction parameters needed for selecting the appropriate DRC Set matching the playback condition (including the selected personalization experience, e.g., *mae\_groupID* and *mae\_groupPresetID*). Further, this mapping enables decoders to select the appropriate DRC
- 15 Set based on the playback condition (including the selected personalization experience) and the loudness leveling process being switched on or off (e.g., using the *levelingControlInterface()*).

| Syntax                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | No. of bits                                  | Mnemonic                                                        |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------|-----------------------------------------------------------------|
| uniDrcConfigExtension()<br>{<br>while ( <b>uniDrcConfigExtType</b> != UNIDRCCONFEXT_TERM) {<br>extSizeBits = <b>bitSizeLen</b> + 4;<br>extBitSize = <b>bitSize</b> + 1;<br>switch (uniDrcConfigExtType) {<br>case UNIDRCCONFEXT_PARAM_DRC:<br>drcCoefficientsParametricDrc();<br><b>parametricDrcInstructionsCount</b> ;<br>for (i=0; i<parametricDrcInstructionsCount; i++) {<br>parametricDrcInstructions ();<br>}<br>break;<br>case UNIDRCCONFEXT_V1:<br><b>downmixInstructionsV1Present</b> ;<br>if (downmixInstructionsV1Present==1) {<br><b>downmixInstructionsV1Count</b> ;<br>}<br>}<br>}<br>} | 4<br>4<br>extSizeBits<br><br>4<br><br>1<br>7 | uimsbf<br>uimsbf<br>uimsbf<br><br>uimsbf<br><br>bslbf<br>uimsbf |

| Syntax                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | No. of bits                                                             | Mnemonic                                                                                                    |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------|
| <pre> for (i=0; i&lt;downmixInstructionsV1Count; i++) {     downmixInstructionsV1(); } } <b>drcCoeffsAndInstructionsUniDrcV1Present;</b> if (drcCoeffsAndInstructionsUniDrcV1Present==1) {     <b>drcCoefficientsUniDrcV1Count;</b>     for (i=0; i&lt;drcCoefficientsUniDrcV1Count; i++) {         drcCoefficientsUniDrcV1();     }     <b>drcInstructionsUniDrcV1Count;</b>     for (i=0; i&lt;drcInstructionsUniDrcV1Count; i++) {         drcInstructionsUniDrcV1();     } } <b>loudEqInstructionsPresent;</b> if (loudEqInstructionsPresent==1) {     <b>loudEqInstructionsCount;</b>     for (i=0; i&lt;loudEqInstructionsCount; i++) {         loudEqInstructions();     } } <b>eqPresent;</b> if (eqPresent==1) {     eqCoefficients();     <b>eqInstructionsCount;</b>     for (i=0; i&lt;eqInstructionsCount; i++) {         eqInstructions();     } } break; case UNIDRCCONFEXT_DYNLOUD:     for (i=0; i&lt;drcInstructionsUniDrcV1Count; i++) {         dse = drcInstructionsUniDrcV1[i].drcSetEffect;         if ((dse &amp; (1&lt;&lt;11)) != 0) { /* if Ducking/Leveling self */             <b>levelingPresent;</b>             /* drcInstructionsUniDrcV1[i].levelingPresent = levelingPresent */ </pre> | <p>1</p> <p>3</p> <p>6</p> <p>1</p> <p>4</p> <p>1</p> <p>4</p> <p>1</p> | <p>bslbf</p> <p>uimsbf</p> <p>uimsbf</p> <p>bslbf</p> <p>uimsbf</p> <p>bslbf</p> <p>uimsbf</p> <p>bslbf</p> |

| Syntax                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | No. of bits                                  | Mnemonic                                                         |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------|------------------------------------------------------------------|
| <pre> if (levelingPresent==1) {     <b>duckingOnlyDrcSetPresent;</b>     /* drcInstructionsUniDrcV1[i].        duckingOnlyDrcSetPresent */     if (duckingOnlyDrcSetPresent ==1) {         /* "Ducking only" DRC set */         drcInstructionsUniDrcV1();     } } } } for (i=0; i&lt;drcInstructionsUniDrcCount; i++) {     dse = drcInstructionsUniDrc[i].drcSetEffect;     if ((dse &amp; (1&lt;&lt;11)) != 0) { /* if Ducking/Leveling self */         <b>levelingPresent;</b>         /* drcInstructionsUniDrc[i].levelingPresent */         if (levelingPresent==1) {             <b>duckingOnlyDrcSetPresent;</b>             /* drcInstructionsUniDrc[i].                duckingOnlyDrcSetPresent */             if (duckingOnlyDrcSetPresent ==1) {                 /* "Ducking only" DRC set */                 drcInstructionsUniDrc();             }         }     } } break; /* add future extensions here */ default:     for (i=0; i&lt;extBitSize; i++) {         <b>otherBit;</b>     } } } } </pre> | <p>1</p> <p>1</p> <p>1</p> <p>1</p> <p>1</p> | <p>bslbf</p> <p>bslbf</p> <p>bslbf</p> <p>bslbf</p> <p>bslbf</p> |

Table 29: Syntax updates to uniDrcConfigExtension()

[0179] Alternatively, the syntax between “case UNIDRCCONFEXT\_DYNLOUD:” and “break;” may be referred to as a separate syntax element, e.g. *levelingInstructions()* as shown in Table 30.

| Syntax                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | No. of bits                         | Mnemonic                                                                        |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------|---------------------------------------------------------------------------------|
| <pre> levelingInstructions() {     for (i=0; i&lt;drcInstructionsUniDrcV1Count; i++) {         dse = drcInstructionsUniDrcV1[i].drcSetEffect;         if ((dse &amp; (1&lt;&lt;11)) != 0) { /* if Ducking/Leveling self */             <b>levelingPresent;</b>             /* drcInstructionsUniDrcV1[i].levelingPresent */             if (levelingPresent==1) {                 <b>duckingOnlyDrcSetPresent;</b>                 /* drcInstructionsUniDrcV1[i]. duckingOnlyDrcSetPresent */                 if (duckingOnlyDrcSetPresent ==1) {                     /* “Ducking only” DRC set */                     drcInstructionsUniDrcV1();                 }             }         }     }     for (i=0; i&lt;drcInstructionsUniDrcCount; i++) {         dse = drcInstructionsUniDrc[i].drcSetEffect;         if ((dse &amp; (1&lt;&lt;11)) != 0) { /* if Ducking/Leveling self */             <b>levelingPresent;</b>             /* drcInstructionsUniDrc[i].levelingPresent */             if (levelingPresent==1) {                 <b>duckingOnlyDrcSetPresent;</b>                 /* drcInstructionsUniDrc[i]. duckingOnlyDrcSetPresent */                 if (duckingOnlyDrcSetPresent ==1) {                     /* “Ducking only” DRC set */                     drcInstructionsUniDrc();                 }             }         }     } } </pre> | <p>1</p> <p>1</p> <p>1</p> <p>1</p> | <p><b>bslbf</b></p> <p><b>bslbf</b></p> <p><b>bslbf</b></p> <p><b>bslbf</b></p> |

*Table 30: Syntax of levelingInstructions() payload*

*MPEG-H 3D Audio Differences in DRC set selection process*

[0180] In the functionality and syntax set out by MPEG-H 3D audio, there may be more than one DRC set with a ducking/leveling effect, with different *maeGroupID* or *maePresetID* values.

- 5 [0181] An inventive aspect includes taking the newly added parameters related to loudness leveling in any syntax-element (e.g., *levelingPresent*, *duckingOnlyDrcSetPresent* in the *uniDrcConfigExtension()*-element) into consideration for the DRC set selection process as described in Section 6.4.4 of MPEG-H 3D audio, in order to allow for the selection of the correct DRC Set depending on the current playback condition (including the selected
- 10 personalization experience; e.g., *mae\_groupID* and *mae\_groupPresetID*). The selection of the correct DRC Set may further be dependent on the status of loudness leveling (e.g., switched on or off), which may be controlled using a MPEG-D DRC decoder interface. The MPEG-D DRC decoder interface in MPEG-H 3D audio may be realized via a bitstream interface based on MHAS, using a Packet Type of *PACTYP\_LOUDNESS\_DRC*, enabling
- 15 transport of the *mpegh3daLoudnessDrcInterface()*-element including *uniDrcInterface()*-element and the *uniDrcInterfaceExtension()*-element. The *uniDrcInterfaceExtension()*-element may be updated as shown in Table 19 to Table 21.

*Interpretation*

- [0182] Unless specifically stated otherwise, as apparent from the following discussions, it is
- 20 appreciated that throughout the disclosure discussions utilizing terms such as “processing”, “computing”, “determining”, “analyzing” or the like, refer to the action and/or processes of a computer or computing system, or similar electronic computing devices, that manipulate and/or transform data represented as physical, such as electronic, quantities into other data similarly represented as physical quantities.

- 25 [0183] In a similar manner, the term “processor” may refer to any device or portion of a device that processes electronic data, e.g., from registers and/or memory to transform that electronic data into other electronic data that, e.g., may be stored in registers and/or memory. A “computer” or a “computing machine” or a “computing platform” may include one or more processors.

- 30 [0184] The methodologies described herein are, in one example embodiment, performable by one or more processors that accept computer-readable (also called machine-readable) code containing a set of instructions that when executed by one or more of the processors carry out

at least one of the methods described herein. Any processor capable of executing a set of instructions (sequential or otherwise) that specify actions to be taken are included. Thus, one example is a typical processing system that includes one or more processors. Each processor may include one or more of a CPU, a graphics processing unit, and a programmable DSP  
5 unit. The processing system further may include a memory subsystem including main RAM and/or a static RAM, and/or ROM. A bus subsystem may be included for communicating between the components. The processing system further may be a distributed processing system with processors coupled by a network. If the processing system requires a display, such a display may be included, e.g., a liquid crystal display (LCD) or a cathode ray tube  
10 (CRT) display. If manual data entry is required, the processing system also includes an input device such as one or more of an alphanumeric input unit such as a keyboard, a pointing control device such as a mouse, and so forth. The processing system may also encompass a storage system such as a disk drive unit. The processing system in some configurations may include a sound output device, and a network interface device. The memory subsystem thus  
15 includes a computer-readable carrier medium that carries computer-readable code (e.g., software) including a set of instructions to cause performing, when executed by one or more processors, one or more of the methods described herein. Note that when the method includes several elements, e.g., several steps, no ordering of such elements is implied, unless specifically stated. The software may reside in the hard disk, or may also reside, completely  
20 or at least partially, within the RAM and/or within the processor during execution thereof by the computer system. Thus, the memory and the processor also constitute computer-readable carrier medium carrying computer-readable code. Furthermore, a computer-readable carrier medium may form, or be included in a computer program product.

**[0185]** In alternative example embodiments, the one or more processors operate as a  
25 standalone device or may be connected, e.g., networked to other processor(s), in a networked deployment, the one or more processors may operate in the capacity of a server or a user machine in server-user network environment, or as a peer machine in a peer-to-peer or distributed network environment. The one or more processors may form a personal computer (PC), a tablet PC, a Personal Digital Assistant (PDA), a cellular telephone, a web appliance, a  
30 network router, switch or bridge, or any machine capable of executing a set of instructions (sequential or otherwise) that specify actions to be taken by that machine.

**[0186]** Note that the term “machine” shall also be taken to include any collection of machines that individually or jointly execute a set (or multiple sets) of instructions to perform any one

or more of the methodologies discussed herein.

[0187] Thus, one example embodiment of each of the methods described herein is in the form of a computer-readable carrier medium carrying a set of instructions, e.g., a computer program that is for execution on one or more processors, e.g., one or more processors that are  
5 part of web server arrangement. Thus, as will be appreciated by those skilled in the art, example embodiments of the present disclosure may be embodied as a method, an apparatus such as a special purpose apparatus, an apparatus such as a data processing system, or a computer-readable carrier medium, e.g., a computer program product. The computer-readable carrier medium carries computer readable code including a set of instructions that when  
10 executed on one or more processors cause the processor or processors to implement a method. Accordingly, aspects of the present disclosure may take the form of a method, an entirely hardware example embodiment, an entirely software example embodiment or an example embodiment combining software and hardware aspects. Furthermore, the present disclosure may take the form of carrier medium (e.g., a computer program product on a  
15 computer-readable storage medium) carrying computer-readable program code embodied in the medium.

[0188] The software may further be transmitted or received over a network via a network interface device. While the carrier medium is in an example embodiment a single medium, the term “carrier medium” should be taken to include a single medium or multiple media  
20 (e.g., a centralized or distributed database, and/or associated caches and servers) that store the one or more sets of instructions. The term “carrier medium” shall also be taken to include any medium that is capable of storing, encoding or carrying a set of instructions for execution by one or more of the processors and that cause the one or more processors to perform any one or more of the methodologies of the present disclosure. A carrier medium may take many  
25 forms, including but not limited to, non-volatile media, volatile media, and transmission media. Non-volatile media includes, for example, optical, magnetic disks, and magneto-optical disks. Volatile media includes dynamic memory, such as main memory. Transmission media includes coaxial cables, copper wire and fiber optics, including the wires that comprise a bus subsystem. Transmission media may also take the form of acoustic or light waves, such  
30 as those generated during radio wave and infrared data communications. For example, the term “carrier medium” shall accordingly be taken to include, but not be limited to, solid-state memories, a computer product embodied in optical and magnetic media; a medium bearing a propagated signal detectable by at least one processor or one or more processors and

representing a set of instructions that, when executed, implement a method; and a transmission medium in a network bearing a propagated signal detectable by at least one processor of the one or more processors and representing the set of instructions.

**[0189]** It will be understood that the steps of methods discussed are performed in one example embodiment by an appropriate processor (or processors) of a processing (e.g., computer) system executing instructions (computer-readable code) stored in storage. It will also be understood that the disclosure is not limited to any particular implementation or programming technique and that the disclosure may be implemented using any appropriate techniques for implementing the functionality described herein. The disclosure is not limited to any particular programming language or operating system.

**[0190]** Reference throughout this disclosure to “one embodiment”, “some embodiments” or “an example embodiment” means that a particular feature, structure or characteristic described in connection with the embodiment is included in at least one embodiment of the present disclosure. Thus, appearances of the phrases “in one embodiment”, “in some embodiments” or “in an example embodiment” in various places throughout this disclosure are not necessarily all referring to the same example embodiment. Furthermore, the particular features, structures or characteristics may be combined in any suitable manner, as would be apparent to one of ordinary skill in the art from this disclosure, in one or more example embodiments.

**[0191]** As used herein, unless otherwise specified the use of the ordinal adjectives “first”, “second”, “third”, etc., to describe a common object, merely indicate that different instances of like objects are being referred to and are not intended to imply that the objects so described must be in a given sequence, either temporally, spatially, in ranking, or in any other manner.

**[0192]** In the claims below and the description herein, any one of the terms comprising, comprised of or which comprises is an open term that means including at least the elements/features that follow, but not excluding others. Thus, the term comprising, when used in the claims, should not be interpreted as being limitative to the means or elements or steps listed thereafter. For example, the scope of the expression a device comprising A and B should not be limited to devices consisting only of elements A and B. Any one of the terms including or which includes or that includes as used herein is also an open term that also means including at least the elements/features that follow the term, but not excluding others. Thus, including is synonymous with and means comprising.

[0193] It should be appreciated that in the above description of example embodiments of the disclosure, various features of the disclosure are sometimes grouped together in a single example embodiment, Fig., or description thereof for the purpose of streamlining the disclosure and aiding in the understanding of one or more of the various inventive aspects.

5 This method of disclosure, however, is not to be interpreted as reflecting an intention that the claims require more features than are expressly recited in each claim. Rather, as the following claims reflect, inventive aspects lie in less than all features of a single foregoing disclosed example embodiment. Thus, the claims following the Description are hereby expressly incorporated into this Description, with each claim standing on its own as a separate example  
10 embodiment of this disclosure.

[0194] Furthermore, while some example embodiments described herein include some but not other features included in other example embodiments, combinations of features of different example embodiments are meant to be within the scope of the disclosure, and form different example embodiments, as would be understood by those skilled in the art. For  
15 example, in the following claims, any of the claimed example embodiments can be used in any combination.

[0195] In the description provided herein, numerous specific details are set forth. However, it is understood that example embodiments of the disclosure may be practiced without these specific details. In other instances, well-known methods, structures and techniques have not  
20 been shown in detail in order not to obscure an understanding of this description.

[0196] Thus, while there has been described what are believed to be the best modes of the disclosure, those skilled in the art will recognize that other and further modifications may be made thereto without departing from the spirit of the disclosure, and it is intended to claim all such changes and modifications as fall within the scope of the disclosure. For example, any  
25 formulas given above are merely representative of procedures that may be used. Functionality may be added or deleted from the block diagrams and operations may be interchanged among functional blocks. Steps may be added or deleted to methods described within the scope of the present disclosure.

[0197] In the following, two sets of enumerated example embodiments (EEEA's and EEEB's)  
30 describe some structures, features, and functionalities of some aspects of the example embodiments disclosed herein.

EEEA1. A method of metadata-based dynamic processing of audio data for playback, the

method including processes of:

(a) receiving, by a decoder, a bitstream including audio data and metadata for loudness leveling;

5 (b) decoding, by the decoder, the audio data and the metadata to obtain decoded audio data and the metadata;

(c) determining, by the decoder, from the metadata, one or more processing parameters for loudness leveling based on a playback condition;

(d) applying the determined one or more processing parameters to the decoded audio data to obtain processed audio data; and

10 (e) outputting the processed audio data for playback.

EEEEA2. The method according to EEEA1, wherein the metadata is indicative of processing parameters for loudness leveling for a plurality of playback conditions.

EEEEA3. The method according to EEEA1 or EEEA2, wherein said determining the one or more processing parameters further includes determining one or more processing parameters  
15 for dynamic range compression, DRC, based on the playback condition

EEEEA4. The method according to any one of EEEA1 to EEEA3, wherein the playback condition includes one or more of a device type of the decoder, characteristics of a playback device, characteristics of a loudspeaker, a loudspeaker setup, characteristics of background noise, characteristics of ambient noise and characteristics of the acoustic environment.

20 EEEA5. The method according to any one of EEEA1 to EEEA4, wherein process (c) further includes selecting, by the decoder, at least one of a set of DRC sequences, DRCSet, a set of equalizer parameters, EQSet, and a downmix, corresponding to the playback condition.

EEEEA6. The method according to EEEA5, wherein process (c) further includes identifying a metadata identifier indicative of the at least one selected DRCSet, EQSet, and downmix to  
25 determine the one or more processing parameters from the metadata.

EEEEA7. The method according to any one of EEEA1 to EEEA6, wherein the metadata includes one or more processing parameters relating to average loudness values and optionally one or more processing parameters relating to dynamic range compression characteristics.

30 EEEA8. The method according to any one of EEEA1 to EEEA7, wherein the bitstream further includes additional metadata for static loudness adjustment to be applied to the

decoded audio data.

EEEEA9. The method according to any one of EEEA1 to EEEA8, wherein the bitstream is an MPEG-D DRC bitstream and the presence of metadata is signaled based on MPEG-D DRC bitstream syntax.

- 5 EEEA10. The method according to EEEA9, wherein a uniDrcConfigExtension()-element is used to carry the metadata as a payload.

EEEEA11. The method according to any one of EEEA1 to EEEA10, wherein the metadata comprises one or more metadata payloads, wherein each metadata payload includes a plurality of sets of parameters and identifiers, with each set including at least one of a

- 10 DRCSet identifier, drcSetId, an EQSet identifier, eqSetId, and a downmix identifier, downmixId, in combination with one or more processing parameters relating to the identifiers in the set.

EEEEA12. The method according to EEEA11 when depending on EEEA5, wherein process (c) involves selecting a set among the plurality of sets in the payload based on the at least one

- 15 DRCSet, EQSet, and downmix selected by the decoder, and wherein the one or more processing parameters determined at process (c) are the one or more processing parameters relating to the identifiers in the selected set.

EEEEA13. A decoder for metadata-based dynamic processing of audio data for playback, wherein the decoder comprises one or more processors and non-transitory memory

- 20 configured to perform a method including processes of:

(a) receiving, by a decoder, a bitstream including audio data and metadata for loudness leveling;

(b) decoding, by the decoder, the audio data and the metadata to obtain decoded audio data and the metadata;

- 25 (c) determining, by the decoder, from the metadata, one or more processing parameters for loudness leveling based on a playback condition;

(d) applying the determined one or more processing parameters to the decoded audio data to obtain processed audio data; and

(e) outputting the processed audio data for playback.

- 30 EEEA14. A method of encoding audio data and metadata for loudness leveling, into a bitstream, the method including processes of:

- (a) inputting original audio data into a loudness leveler for loudness processing to obtain, as an output from the loudness leveler, loudness processed audio data;
- (b) generating metadata for loudness leveling based on the loudness processed audio data and the original audio data; and
- 5 (c) encoding the original audio data and the metadata into the bitstream.

EEEE15. The method according to EEEA14, wherein the method further includes generating additional metadata for static loudness adjustment to be used by a decoder.

- EEEE16. The method according to EEEA14 or EEEA15, wherein process (b) includes comparison of the loudness processed audio data to the original audio data, and wherein the
- 10 metadata is generated based on a result of said comparison.

EEEE17. The method according to EEEA16, wherein process (b) further includes measuring the loudness over one or more pre-defined time periods, and wherein the metadata is generated further based on the measured loudness.

- EEEE18. The method according to EEEA17, wherein the measuring comprises measuring
- 15 overall loudness of the audio data.

EEEE19. The method according to EEEA17, wherein the measuring comprises measuring loudness of dialogue in the audio data.

- EEEE20. The method according to any one of EEEA14 to EEEA19, wherein the bitstream is an MPEG-D DRC bitstream and the presence of the metadata is signaled based on MPEG-D
- 20 DRC bitstream syntax.

EEEE21. The method according to EEEA20, wherein a `uniDrcConfigExtension()`-element is used to carry the metadata as a payload.

- EEEE22. The method according to any one of EEEA14 to EEEA21, wherein the metadata comprises one or more metadata payloads, wherein each metadata payload includes a
- 25 plurality of sets of parameters and identifiers, with each set including at least one of a DRCSet identifier, `drcSetId`, an EQSet identifier, `eqSetId`, and a downmix identifier, `downmixId`, in combination with one or more processing parameters relating to the identifiers in the set, and wherein the one or more processing parameters are parameters for loudness leveling by a decoder.

- 30 EEEA23. The method according to EEEA22, wherein the at least one of the `drcSetId`, the

eqSetId, and the downmixId is related to at least one of a set of DRC sequences, DRCSet, a set of equalizer parameters, EQSet, and a downmix, to be selected by the decoder.

EEEE24. An encoder for encoding in a bitstream original audio data and metadata for loudness leveling, wherein the encoder comprises one or more processors and non-transitory  
5 memory configured to perform a method including the processes of:

(a) inputting original audio data into a loudness leveler for loudness processing to obtain, as an output from the loudness leveler, loudness processed audio data;  
(b) generating the metadata for loudness leveling based on the loudness processed audio data and the original audio data; and

10 (c) encoding the original audio data and the metadata into the bitstream.

EEEE25. A system of an encoder for encoding in a bitstream original audio data and metadata for loudness leveling and/or dynamic range compression, DRC, according to EEEA24 and a decoder for metadata-based dynamic processing of audio data for playback according to EEEA13.

15 EEEA26. A computer program product comprising a computer-readable storage medium with instructions adapted to cause the device to carry out the method according to any one of EEEA1 to EEEA12 or EEEA14 to EEEA23 when executed by a device having processing capability.

EEEE27. A computer-readable storage medium storing the computer program product of  
20 EEEA26.

EEEE28. The method according to any one of EEEA1 to EEEA12 further comprising receiving, by the decoder, through an interface, an indication of whether or not to perform the metadata-based dynamic processing of audio data for playback, and when the decoder receives an indication not to perform the metadata-based dynamic processing of audio data  
25 for playback, bypassing at least the step of applying the determined one or more processing parameters to the decoded audio data.

EEEE29. The method according to EEEA28, wherein until the decoder receives, through the interface, the indication of whether or not to perform the metadata-based dynamic processing of audio data for playback, the decoder bypasses at least the step of applying the determined  
30 one or more processing parameters to the decoded audio data.

EEEE30. The method of any one of EEEA1 to EEEA12, EEEA28, or EEEA29, wherein the

metadata is indicative of processing parameters for loudness leveling for a plurality of playback conditions, and the metadata further includes a parameter specifying a loudness measurement method used for deriving a processing parameter of the plurality of processing parameters.

- 5      EEEA31. The method of any one of EEEA1 to EEEA12, or EEEA28 to EEEA30, wherein the metadata is indicative of processing parameters for loudness leveling for a plurality of playback conditions, and the metadata further includes a parameter specifying a loudness measurement system used for measuring a processing parameter of the plurality of processing parameters.
- 10     EEEB1. A method of metadata-based dynamic processing of audio data for playback, the method including:  
receiving, by a decoder, a bitstream including audio data and metadata for dynamic loudness adjustment;  
decoding, by the decoder, the audio data and the metadata to obtain decoded audio data and  
15     the metadata;  
selecting, by the decoder, at least one set of decoder settings corresponding to a playback condition and determining, from the metadata, one or more processing parameters for dynamic loudness adjustment based on the at least one selected set;  
applying the determined one or more processing parameters to the decoded audio data to  
20     obtain processed audio data; and  
outputting the processed audio data for playback.
- EEEB2. The method according to EEEB1, wherein the metadata is indicative of processing parameters for dynamic loudness adjustment for a plurality of playback conditions.
- EEEB3. The method according to EEEB1 or EEEB2, wherein said determining the one or  
25     more processing parameters further includes determining one or more processing parameters for dynamic range compression, DRC, based on the playback condition.
- EEEB4. The method according to any one of EEEBs1 to 3, wherein the playback condition includes one or more of a device type of the decoder, characteristics of a playback device, characteristics of a loudspeaker, a loudspeaker setup, characteristics of background noise,  
30     characteristics of ambient noise and characteristics of the acoustic environment.
- EEEB5. The method according to any one of EEEBs1 to 4, wherein the at least one set of decoder settings includes a set of DRC sequences, DRCSet, a set of equalizer parameters, EQSet, and a downmix.

EEEEB6. The method according to EEEB5, wherein said determining the one or more processing parameters further includes identifying a metadata identifier indicative of the at least one selected DRCSet, EQSet, and downmix, to determine the one or more processing parameters from the metadata.

- 5 EEEB7. The method according to any one of EEEBs1 to 6, wherein the metadata includes one or more processing parameters relating to average loudness values and optionally one or more processing parameters relating to dynamic range compression characteristics.

EEEEB8. The method according to any one of EEEBs1 to 7, wherein the bitstream further includes additional metadata for static loudness adjustment to be applied to the decoded audio  
10 data.

EEEEB9. The method according to any one of EEEBs1 to 8, wherein the bitstream is an MPEG-D DRC bitstream and the presence of metadata is signaled based on MPEG-D DRC bitstream syntax.

- EEEEB10. The method according to EEEB9, wherein a loudnessInfoSetExtension()-element  
15 is used to carry the metadata as a payload.

- EEEEB11. The method according to any one of EEEBs 1 to 10, wherein the metadata comprises one or more metadata payloads, wherein each metadata payload includes a plurality of sets of parameters and identifiers, with each set including at least one of a DRCSet identifier, drcSetId, an EQSet identifier, eqSetId, and a downmix identifier,  
20 downmixId, in combination with one or more processing parameters relating to the identifiers in the set.

- EEEEB12. The method according to EEEB11 when depending on EEEB5, wherein said determining the one or more processing parameters involves selecting a set among the plurality of sets in the payload based on the at least one DRCSet, EQSet, and downmix  
25 selected by the decoder, and wherein the one or more processing parameters determined by the decoder are the one or more processing parameters relating to the identifiers in the selected set.

- EEEEB13. A decoder for metadata-based dynamic processing of audio data for playback, wherein the decoder comprises one or more processors and non-transitory memory  
30 configured to perform a method including:  
receiving, by the decoder, a bitstream including audio data and metadata for dynamic loudness adjustment;  
decoding, by the decoder, the audio data and the metadata to obtain decoded audio data and the metadata;

- selecting, by the decoder, at least one set of decoder settings corresponding to a playback condition and determining from the metadata, one or more processing parameters for dynamic loudness adjustment based on the at least one selected set;
- applying the determined one or more processing parameters to the decoded audio data to
- 5 obtain processed audio data; and
- outputting the processed audio data for playback.

## CLAIMS

1. A method of processing audio data for playback, the method including:  
receiving, by a decoder, a bitstream including encoded audio data and metadata,  
wherein the metadata includes one or more dynamic range control (DRC) sets, and for each  
5 DRC set, an indication of whether the DRC set is configured for providing a loudness  
leveling effect, wherein the metadata further includes personalization metadata;  
parsing the metadata, by the decoder, to identify DRC sets that are configured for  
providing the loudness leveling effect, wherein the DRC sets are identified at least in part  
based on the personalization metadata;  
10 decoding, by the decoder, the encoded audio data to obtain decoded audio data;  
selecting, by the decoder, one of the identified DRC sets configured for providing the  
loudness leveling effect;  
extracting from the bitstream, by the decoder, one or more DRC gains corresponding  
to the selected DRC set;  
15 applying to the decoded audio data, by the decoder, the one or more DRC gains  
corresponding to the selected DRC set to obtain dynamic loudness compensated audio data;  
and  
outputting the dynamic loudness compensated audio data for playback.
- 20 2. The method of claim 1, wherein the bitstream is a bitstream compatible with the  
MPEG-H 3D audio standard.
3. The method according to claim 2, wherein the personalization metadata includes  
mae\_groupID and maegroupPresetID syntax and semantics as described within the MPEG-H  
25 3D audio standard.
4. The method according to any one of claims 1-3, wherein the metadata includes a  
plurality of DRC sets configured for providing the loudness leveling effect, wherein each of  
the plurality of DRC sets is also associated with one or more playback conditions, and  
30 wherein the selecting is performed in response to an indication of a playback condition  
provided to the decoder.
5. The method according to any one of claims 1-4, wherein, in addition to providing a

loudness leveling effect, the one or more DRC sets are also configured to provide dynamic range control.

6. The method according to any one of claims 1-4, wherein the playback condition includes one or more of a device type of the decoder, characteristics of a playback device, characteristics of a loudspeaker, a loudspeaker setup, characteristics of background noise, characteristics of ambient noise, capabilities of the device type, personalization experience selected on the device and characteristics of the acoustic environment.

7. The method according to claim 6, wherein the personalization experience, selected by user or device, is based on user preferences, such as language, user experience, previous listening selections, and/or device capabilities.

8. The method according to any one of claims 1 to 7, wherein the indication of whether the DRC set is configured for providing the loudness leveling effect is provided in a parameter indicating one or more effects provided by the DRC set.

9. The method according to claim 8, wherein the parameter indicating one or more effects provided by the DRC set is a drcSetEffect bitfield of an MPEG-D DRC bitstream, wherein individual bits of the drcSetEffect bitfield correspond to different effects, and one of the bits of the drcSetEffect bitfield corresponds to the loudness leveling effect.

10. The method according to any one of claims 1 to 9, wherein the indication of whether the DRC set is configured for providing the loudness leveling effect is whether the DRC set is specified in a loudness leveling bitstream payload.

11. The method according to claim 10, wherein the loudness leveling bitstream payload is included in an extension field of a previously defined bitstream syntax.

12. The method according to claim 11, wherein the extension field is a uniDrcConfigExtension field of an MPEG-H 3D audio bitstream, and wherein the loudness leveling bitstream payload is included only for specific values of a uniDrcConfigExtType parameter.

13. The method according to claim 11 or claim 12 wherein a plurality of loudness leveling payloads specifying a plurality of DRC sets configured for providing the loudness leveling effect are included in the extension field of the previously defined bitstream syntax.

5           14. The method of any one of claims 1 to 13, wherein the indication of whether the DRC set is configured for providing the loudness leveling effect is a field of a previously existing configuration element of a previously defined bitstream syntax.

10           15. The method of any one of claims 1 to 14, wherein the indication of whether the DRC set is configured for providing the loudness leveling effect is a field of an updated version of a previously existing configuration element of a previously defined bitstream syntax.

15           16. The method of any one of claims 1 to 15, wherein an indication that a loudness leveling effect is desired is provided to the decoder through an interface, and wherein the DRC set is selected in response to the indication provided to the decoder through the interface.

20           17. The method of claim 16, wherein the interface receives the indication from a MHAS compatible syntax.

25           18. The method of claim 17, wherein indications of additional desired effects are provided to the decoder through the interface, wherein the metadata includes a plurality of DRC sets configured to provide the loudness leveling effect, and wherein the selection depends on the additional desired effects.

30           19. The method of any one of claims 16-18, wherein the indication that a loudness leveling effect is desired is provided through a loudnessLevelingOn parameter of a levelingControlInterface payload.

20. The method of any one of claims 1 to 19, wherein the metadata includes one or more static loudness values configured for providing static loudness adjustment to the decoded audio data.

21. The method of claim 20, comprising applying static loudness adjustment, in response to one or more of the static loudness values, to the decoded audio data or the dynamic loudness compensated audio data.

5           22. The method of claim 1, wherein a first one of the DRC sets is configured for providing dynamic range control, wherein the first DRC set comprises an indication that the selected DRC set configured for providing the loudness leveling effect is configured for application in combination with the first DRC set.

10           23. The method of claim 22, wherein the selected DRC set comprises an indication of whether the one or more DRC gains corresponding to the selected DRC set may only be applied in combination with DRC gains corresponding to the first DRC set.

15           24. The method of claim 22 or claim 23, further comprising extracting from the bitstream the DRC gains corresponding to the first DRC set, and applying the DRC gains corresponding to the first DRC set to the decoded audio data.

20           25. A decoder for processing audio data for playback, wherein the decoder comprises one or more processors and non-transitory memory configured to perform the method of any one of claims 1 to 24.

25           26. A computer program product comprising a computer-readable storage medium with instructions adapted to cause a device to carry out the method according to any one of claims 1 to 24.

27. A computer-readable storage medium storing the computer program product of claim 26.