INTERNATIONAL STANDARD

ISO/IEC 23003-4

First edition
2015-11-15

AMENDMENT 1
2017-12

Information technology — MPEG audio technologies —

Part 4: **Dynamic Range Control**

AMENDMENT 1: Parametric DRC, gain mapping and equalization tools

Technologies de l'information — Technologies audio MPEG —

Partie 4: Contrôle de gamme dynamique

AMENDEMENT 1: Outils de DRC paramétrique, de mappage des gains et d'égalisation

Reference number
ISO/IEC 23003-4:2015/Amd.1:2017(E)

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/IEC JTC 1, *Information technology*, SC 29, *Coding of audio, picture, multimedia and hypermedia information.*

A list of all parts in the ISO/IEC 23003 series can be found on the ISO website.

Information technology — MPEG audio technologies —

Part 4:
Dynamic Range Control

AMENDMENT 1: Parametric DRC, gain mapping and equalization tools

Page vi, Introduction

Add the following at the end of the Introduction:

Loudness normalization is fully integrated with DRC and peak control to avoid clipping. A metadata-controlled equalization tool is provided to compensate for playback scenarios that impact the spectral balance, such as downmix or DRC. Furthermore, the DRC tool supports metadata-based loudness equalization to compensate the effect of playback level changes on the spectral balance.

Page 2, Clause 4

Insert the following new definitions and maintain the alphabetical order:

mod	modulo operator: (x mod y) = x-y floor (x/y)
sizeof(x)	size operator that returns the bit size of a field

Page 3, Clause 5

Replace the first paragraph:

The technology described in this part of ISO/IEC 23003 is called DRC tool. It provides efficient control of dynamic range, loudness, and clipping based on metadata generated at the encoder. The decoder can choose to selectively apply the metadata to the audio signal to achieve a desired result. Metadata for dynamic range compression consists of encoded time-varying gain values that can be applied to the audio signal. Hence, the main blocks of the DRC tool include a DRC gain encoder, a DRC gain decoder, a DRC gain modification block, and a DRC gain application block. These blocks are exercised on a frame-by-frame basis during audio processing. Various DRC configurations can be conveyed in a separate bitstream element, such as configurations for a downmix or combined DRCs. The DRC set selection block decides based on the playback scenario and the applicable DRC configurations which DRC gains to apply to the audio signal. Moreover, the DRC tool supports loudness normalization based on loudness metadata.

With:

The technology described in this document is called the “DRC tool”. It provides efficient control of dynamic range, loudness, and clipping based on metadata generated at the encoder. The decoder can choose to selectively apply the metadata to the audio signal to achieve a desired result. Metadata for dynamic range compression consists of encoded time-varying gain values that can be applied to the audio signal. Hence, the main blocks of the DRC tool include a DRC gain encoder, a DRC gain decoder, a DRC gain modification block, and a DRC gain application block. These blocks are exercised on a frame-by-frame basis during audio processing. In addition to encoded time-varying gain values, the DRC gain decoder can also receive parametric DRC metadata for generation of time-varying gain values at the decoder. Various DRC configurations can be conveyed in a separate bitstream element, such as configurations for a downmix or combined DRCs. The DRC set selection block decides based on the playback scenario and the applicable DRC configurations which DRC gains to apply to the audio signal. Moreover, the DRC tool supports loudness normalization based on loudness metadata.

Page 3, Clause 5

Add the following at the end of the clause:

The DRC tool provides support for loudness equalization, or sometimes called "loudness compensation", that can be applied to compensate for the effect of the playback level on the spectral balance. For this purpose, time-varying loudness information can be recovered from DRC gain sequences to dynamically control the compensation module. While the compensation module is out of scope, the interface describes in which frequency ranges the loudness information should be applied.

A flexible tool for generic metadata-controlled equalization is provided. The tool can be used to reach the desired spectral balance of the reproduced audio signal depending on a wide variety of playback scenarios, such as downmix, DRC, or playback room size. It can operate in the sub-band domain of an audio decoder and in the time domain.

Page 4, 6.1.1

Replace the following list after Table 1:

The static payload is divided into five logical blocks:

— channelLayout()

— downmixInstructions ()

— drcCoefficientsBasic(), drcCoefficientsUniDrc()

— drcInstructionsBasic(), drcInstructionUniDrc()

— loudnessInfo()

With:

The static payload is divided into six logical blocks:

— channelLayout();

— downmixInstructions(), downmixInstructionsV1();

— drcCoefficientsBasic(), drcCoefficientsUniDrc(), drcCoefficientsUniDrcV1();

— drcInstructionsBasic(), drcInstructionUniDrc(), drcInstructionUniDrcV1();

— loudnessInfo(), loudnessInfoV1();

— loudEqInstructions().

Page 4, 6.1.1

Replace the last two paragraphs:

uniDrcConfig() contains all blocks except for the loudnessInfo() blocks which are bundled in loudnessInfoSet(). The last part of the uniDrcConfig() payload can include future extension payloads. In the event that a *uniDrcConfigExtType* value is received that is not equal to UNIDRCCONFEXT_TERM, the DRC tool parser shall read and discard the bits (otherBit) of the extension payload. Similarly, the last part of the loudnessInfoSet() payload can include future extension payloads. In the event that a *loudnessInfoSetExtType* value is received that is not equal to UNIDRCLOUDEXT_TERM, the DRC tool parser shall read and discard the bits (otherBit) of the extension payload.

The top level fields of uniDrcConfig() include the audio sample rate, which is a fundamental parameter for the decoding process (if not present, the audio sample rate is inherited from the employed audio codec). Moreover, the top level fields of uniDrcConfig() include the number of instances of each of the logical blocks, except for the channelLayout() block which appears only once. The top level fields of loudnessInfoSet() only include the number of loudnessInfo() blocks. The five logical blocks are described in the following.

With:

uniDrcConfig() contains all blocks except for the loudnessInfo() blocks which are bundled in loudnessInfoSet(). The last part of the uniDrcConfig() payload can include future extension payloads. In the event that a *uniDrcConfigExtType* value is received that is not equal to UNIDRCCONFEXT_TERM, the DRC tool parser shall read and discard the bits (otherBit) of the extension payload. Similarly, the last part of the loudnessInfoSet() payload can include future extension payloads. In the event that a *loudnessInfoSetExtType* value is received that is not equal to UNIDRCLOUDEXT_TERM, the DRC tool parser shall read and discard the bits (otherBit) of the extension payload. Each extension payload type in uniDrcConfig() or loudnessInfoSet() shall not appear more than once in the bitstream if not stated otherwise. An extension payload of type UNIDRCCONFEXT_V1 shall preceed an extension payload of type UNIDRCCONFEXT_PARAM_DRC in the bitstream if both payloads are present. Note that for ISO/IEC 14496-12, configuration extension payloads are provided according to Table AMD1.26.

The top level fields of uniDrcConfig() include the audio sample rate, which is a fundamental parameter for the decoding process (if not present, the audio sample rate is inherited from the employed audio codec). Moreover, the top level fields of uniDrcConfig() include the number of instances of each of the logical blocks, except for the channelLayout() block which appears only once. The top level fields of loudnessInfoSet() only include the number of loudnessInfo() blocks. The six logical blocks are described in the following.

Page 5, 6.1.2.2

Replace:

6.1.2.2 downmixInstructions()

This block includes a unique non-zero downmix identifier (downmixId) that can be used externally to refer to this downmix. The targetChannelCount specifies the number of channels after downmixing to the target layout. It may also contain downmix coefficients, unless they are specified elsewhere. For use cases where the base audio signal represents objects or other audio content, the downmixId can be used to refer to a specific target channel configuration of a present rendering engine.

With:

6.1.2.2 downmixInstructions() and downmixInstructionsV1()

This block includes a unique non-zero downmix identifier (*downmixId*) that can be used externally to refer to this downmix. The *targetChannelCount* specifies the number of channels after downmixing to the target layout. It may also contain downmix coefficients, unless they are specified elsewhere. For use cases where the base audio signal represents objects or other audio content, the *downmixId* can be used to refer to a specific target channel configuration of a present rendering engine. In contrast to downmixInstructions(), the downmixInstructionsV1() payload includes an offset for all downmix coefficients and the coefficient decoding does not depend on the LFE channel assignment. The downmixInstructions() box for ISO/IEC 14496-12 contains the corresponding metadata of either one of the in-stream payloads as indicated by the *version* parameter of the box.

Page 5, 6.1.2.3

Replace the heading of 6.1.2.3 with:

6.1.2.3 drcCoefficientsBasic(), drcCoefficientsUniDrc(), and drcCoefficientsUniDrcV1()

Page 5, 6.1.2.3

Replace the second paragraph with:

The DRC location field encoding depends on the audio codec. A codec specification may include this specification, and use values 1 to 4 to refer to codec-specific locations as indicated in Table 2. For example, for AAC (ISO/IEC 14496-3), the codec-specific values of the DRC location field are encoded as shown in Table 3.

Page 6, 6.1.2.3

Add new paragraph before 6.1.2.4:

The drcCoefficientsUniDrcV1() payload is defined in Table AMD1.24. It contains the same information as drcCoefficientsUniDrc() except for the assignment of DRC gain sequences to gain sets and the optional specification of a number of parametric DRC characteristics. The drcCoefficientsUniDrc() payload assigns gain sequences in order of transmission. In contrast, the drcCoefficientsUniDrcV1() payload maps a gain sequence by index to *gainSets*. The latter permits to refer to the same gain sequence for multiple DRC bands which is not possible when using drcCoefficientsUniDrc(). If a drcCoefficientsUniDrcV1() payload is present, any drcCoefficientsUniDrc() payload for the same location is ignored.

The drcCoefficientsUniDrcV1() payload can also include information about dynamic equalization filters if the field *shapingFiltersPresent*==1. There can be a number of filters that are indexed in order of appearance. The DRC sets defined in drcInstructionsUniDrcV1() can refer to specific filters using their indices (see 6.4.11).

Page 6, 6.1.2.4

Replace the heading of 6.1.2.4 with:

6.1.2.4 drcInstructionsBasic(), drcInstructionsUniDrc(), and drcInstructionsUniDrcV1()

Page 6, 6.1.2.4

Insert the following after the first paragraph of 6.1.2.4:

The drcInstructionsUniDrcV1() payload is defined in Table AMD1.25. Compared to the drcInstructionsUniDrc() payload, it contains the same information plus several enhancements. However, the *downmixIDs* that appear in the two payloads are interpreted differently. For drcInstructionsUniDrcV1(), the *downmixIDs* indicate which downmix configuration is permitted in combination with this DRC, but it does not specify whether the DRC is applied to the downmix or the base layout. This is controlled by the *drcApplyToDownmix* flag instead.

The enhanced metadata of drcInstructionsUniDrcV1() compared to drcInstructionsUniDrc() includes references to target DRC characteristics and dynamic equalization filters. If a target characteristic is referenced, the corresponding DRC gain values shall be mapped to the target characteristic unless the host system overrides the target characteristic as specified in Table 16. If dynamic equalization filters are referenced, they shall be applied when the corresponding DRC set is selected and when this feature is supported by the decoder implementation (see 6.4.11).

Page 7, 6.1.2.4

Replace the paragraphs:

A second DRC set may be specified for certain configurations. These configurations include cases where, e.g. one DRC set is used for dynamic range compression and the other for clipping prevention ("Clipping" bit is set); or, e.g. one DRC set is applied before and the other after the downmix. In those cases, the second DRC set contains a non-zero field *dependsOnDrcSet* that has the value of the *drcSetId* of the first DRC set it depends on. The declared DRC set effects of the second DRC set do not take into account the effects of the first DRC set. If the first DRC set is not designed to be used without combining it with another DRC set, the *noIndependentUse* flag shall be set to 1. In that case, the DRC set can only be used in combination with another DRC set as indicated by the *dependsOnDrcSet* field of the other set that is combined with it.

Usually, each audio channel is assigned to a DRC gain sequence. A collection of channels assigned to the same DRC gain sequence is called "channel group". The assignment of a DRC gain sequence to a channel group is done in the order of first appearance of the sequence index when iterating through all channels (see also Table 14). A DRC gain sequence index *bsSequenceIndex*==0 indicates that the assigned channel will be passed through by the DRC tool without processing unless otherwise noted. Note that therefore *bsSequenceIndex* is effectively 1-based, whereas the corresponding indices (*sequenceIndex*) for processing are zero-based.

If subsequent channels are assigned the same sequence index, the field *repeatSequenceCount* indicates how many channels will have the same sequence not including the first.

With:

A second DRC set may be specified for certain configurations. These configurations include cases where, for example, one DRC set is used for dynamic range compression and the other for clipping prevention ("Clipping" bit is set); or, for example, one DRC set is applied before and the other after the downmix. In those cases, the first DRC set contains a non-zero field *dependsOnDrcSet* that has the value of the *drcSetId* of the second DRC set it depends on. The declared DRC set effects of the first DRC set do not take into account the effects of the second DRC set. If the second DRC set is not designed to be used without combining it with another DRC set, the *noIndependentUse* flag shall be set to 1. In that case, the DRC set can only be used in combination with another DRC set as indicated by the *dependsOnDrcSet* field of the other set that is combined with it.

If a second DRC is specified by the *dependsOnDrcSet* field of a drcInstructionsUniDrcV1() payload, the combined DRCs cannot be applied if the DRC decoder downmix configuration does not match any *downmixID* in any of the two corresponding drcInstructionsUniDrcV1() payloads. Please note that these downmix configurations may include the output of the base layout without downmix, which is specified by a *downmixID*==0. Otherwise, the combined DRC is compatible with the downmix.

A DRC gain set is a set of DRC gain sequences, where the sequences are assigned to the bands of a multi-band DRC. For a single band DRC, the gain set contains just one gain sequence. Usually, each audio channel is assigned to a DRC gain set. A collection of channels assigned to the same DRC gain set is called "channel group". The assignment of a DRC gain set to a channel group is done in the order of first appearance of the set index when iterating through all channels (see also Table 14). A DRC gain set index *bsGainSetIndex*==0 indicates that the assigned channel will be passed through by the DRC tool without processing unless otherwise noted. Note that therefore, *bsGainSetIndex* is effectively 1-based, whereas the corresponding indices (*gainSetIndex*) are zero-based.

If subsequent channels are assigned the same gain set index, the field *repeatGainSetCount* indicates how many channels will have the same gain set not including the first.

Page 7, 6.1.2.5

Replace the heading of 6.1.2.5 with:

6.1.2.5 loudnessInfo() and loudnessInfoV1()

Page 8, 6.1.2.5

Replace the second paragraph:

If *downmixId* is zero, then loudnessInfo() applies to the base layout. If the *drcSetId* is zero, then loudnessInfo() applies to the audio signal without DRC processing.

With:

If *downmixId* is zero, then loudnessInfo() applies to the base layout. If the *drcSetId* is zero, then loudnessInfo() applies to the audio signal without DRC processing. If *drcSetId* is 0x3F, then loudnessInfo() applies to any DRC processing including no DRC.

Page 8, 6.1.2.5

Replace the third paragraph:

The fields *samplePeakLevel* and *truePeakLevel* represent the level of the maximum sample magnitude in dBFS and the true peak in dBTP, respectively, of the associated audio content before or after audio encoding as defined in Reference [4]. The *measurementSystem* field includes standardized systems and others (see Table A.37). System 3 is defined as ITU-R BS.1770-3 with pre-processing. The pre-processing is a high-pass filter that models the typical limited frequency response of portable device loudspeakers. System 4 is defined as “User”. It means that the corresponding *methodValue* reflects a (subjective) user preference. System 5 is defined as “Expert/Panel”. It means that the corresponding *methodValue* represents a (subjective) expert or panel preference.

With:

The fields *samplePeakLevel* and *truePeakLevel* represent the level of the maximum sample magnitude in dBFS and the true peak in dBTP, respectively, of the associated audio content before or after audio encoding as defined in Reference [4]. The *measurementSystem* field includes standardized systems and others (see Table A.37). System 3 is defined as ITU-R BS.1770-4 with pre-processing. The pre-processing is a high-pass filter that models the typical limited frequency response of portable device loudspeakers. System 4 is defined as “User”. It means that the corresponding *methodValue* reflects a (subjective) user preference. System 5 is defined as “Expert/Panel”. It means that the corresponding *methodValue* represents a (subjective) expert or panel preference.

Page 8, 6.1.2.5

Add a new subclause after 6.1.2.5:

6.1.2.6 loudEqInstructions()

The loudEqInstructions() payload includes information relevant for the loudness equalization support. Each instance of this payload defines a set of loudness equalizer metadata and which combinations of downmix, DRC, and EQ it can be applied to. The metadata includes references to the corresponding DRC gain sequences and associated parameters needed to derive the acoustic level data. A typical way of generation and use of the metadata is given in D.2.10.

Page 11, 6.1.3

Replace the paragraph before Table 7:

Table 6 includes functions to check the availability and to retrieve peak-related information from loudnessInfo() and a drcInstructions block which can have the basic or uniDrc format. Table 7 shows pseudo code for some of the functions for the truePeakLevel and limiterPeakTarget. The functions for samplePeakLevel can be implemented by replacing truePeakLevel with samplePeakLevel.

With:

Table 6 includes functions to check the availability and to retrieve peak-related information from loudnessInfo() and a drcInstructions block which can have the basic or uniDrc format. Table 7 shows pseudo code for some of the functions for the *truePeakLevel* and *limiterPeakTarget* based on downmixInstructions(), drcInstructionsUniDrc(), and loudnessInfo() payloads. The functions shall be adapted accordingly for the downmixInstructionsV1(), drcInstructionsUniDrcV1(), or loudnessInfoV1() payloads, if present. The functions for *samplePeakLevel* can be implemented by replacing *truePeakLevel* with *samplePeakLevel*.

Page 9, 6.1.3

Add the following paragraph and table after Table 5:

The applicable loudnessInfo() structure for determination of *programLoudness* and *anchorLoudness* is determined as specified in Table AMD1.1. The final loudness metadata value is extracted based on the selection steps specified in 6.6.2.

Table AMD1.1 — Determination of applicable loudnessInfo() structure for selection of programLoudness or anchorLoudness for a specific DRC set

```
getApplicableLoudnessInfoStructure(drcSetId, downmixIdRequested) {
  /* default value */
  loudnessInfo = getLoudnessInfoStructure(drcSetId, downmixIdRequested);
  /* fallback values */
  if (loudnessInfo == NULL) {
    loudnessInfo = getLoudnessInfoStructure(drcSetId, 0x7F);
  } else if (loudnessInfo == NULL) {
    loudnessInfo = getLoudnessInfoStructure(0x3F, downmixIdRequested);
  } else if (loudnessInfo == NULL) {
    loudnessInfo = getLoudnessInfoStructure(0, downmixIdRequested);
  } else if (loudnessInfo == NULL) {
    loudnessInfo = getLoudnessInfoStructure(0x3F, 0x7F);
  } else if (loudnessInfo == NULL) {
```

Table AMD1.1 *(continued)*

```
    loudnessInfo = getLoudnessInfoStructure(0, 0x7F);
  } else if (loudnessInfo == NULL) {
    loudnessInfo = getLoudnessInfoStructure(drcSetId, 0);
  } else if (loudnessInfo == NULL) {
    loudnessInfo = getLoudnessInfoStructure(0x3F, 0);
  } else if (loudnessInfo == NULL) {
    loudnessInfo = getLoudnessInfoStructure(0, 0);
  }
  return loudnessInfo;
}
getLoudnessInfoStructure(drcSetId, downmixId) {
  if (useAlbumMode == 1) count = loudnessInfoAlbumCount;
  else count = loudnessInfoCount;
  for (i=0; i<count; i++) {
    if (loudnessInfo[i]->drcSetId == drcSetId) &&
       (loudnessInfo[i]->downmixId == downmixId) {
      for (j=0; j<loudnessInfo[i]->measurmentCount; j++) {
        if ((loudnessInfo[i]->measure[j]->methodDefinition==1) ||
            (loudnessInfo[i]->measure[j]->methodDefinition==2)) {
          return loudnessInfo[i];
        }
      }
    }
  }
  return NULL;
}
```

Page 10, Table 6

Replace Table 6 with:

Table 6 — Determination of signalPeakLevel for a specific DRC set

```
getSignalPeakLevelForDrcSet (drcSetId, downmixIdRequested) {
  dmxId = downmixIdRequested;
  if truePeakLevelIsPresent(drcSetId, dmxId) {
    signalPeakLevel = getTruePeakLevel(drcSetId, dmxId);
  } else if samplePeakLevelIsPresent(drcSetId, dmxId) {
    signalPeakLevel = getSamplePeakLevel(drcSetId, dmxId);
  } else if truePeakLevelIsPresent(0x3F, dmxId) {
    signalPeakLevel = getTruePeakLevel(0x3F, dmxId);
  } else if samplePeakLevelIsPresent(0x3F, dmxId) {
    signalPeakLevel = getSamplePeakLevel(0x3F, dmxId);
  } else if limiterPeakTargetIsPresent(drcSetId, dmxId) {
    signalPeakLevel = getLimiterPeakTarget(drcSetId, dmxId);
  } else if (dmxId != 0) {
    signalPeakLevelTmp = 0.0;
    downmixPeakLevelLinear = 0.0;
    if downmixCoefficientsArePresent(dmxId) {
      for (i=0; i<targetChannelCount; i++) {
```

Table 6 *(continued)*

```
        downmixPeakLevelLinearTmp = 0.0;
        for (j=0; j<baseChannelCount; j++) {
          downmixPeakLevelLinearTmp +=
              pow(10.0, getDownmixCoefficient(dmxId, i, j)/20.0);
        }
        if (downmixPeakLevelLinear < downmixPeakLevelLinearTmp) {
          downmixPeakLevelLinear = downmixPeakLevelLinearTmp;
        }
      }
    }
    if truePeakLevelIsPresent(drcSetId, 0) {
      signalPeakLevelTmp = getTruePeakLevel(drcSetId, 0);
    } else if samplePeakLevelIsPresent(drcSetId, 0) {
      signalPeakLevelTmp = getSamplePeakLevel(drcSetId, 0);
    } else if truePeakLevelIsPresent(0x3F, 0) {
      signalPeakLevelTmp = getTruePeakLevel(0x3F, 0);
    } else if samplePeakLevelIsPresent(0x3F, 0) {
      signalPeakLevelTmp = getSamplePeakLevel(0x3F, 0);
    } else if limiterPeakTargetIsPresent(drcSetId, 0) {
      signalPeakLevelTmp = getLimiterPeakTarget(drcSetId, 0);
    }
    signalPeakLevel = signalPeakLevelTmp + 20.0*log10(downmixPeakLevelLinear);
  } else {
    signalPeakLevel = 0.0;     /* worst case estimate */
  }
  return signalPeakLevel
}
```

Page 12, 6.3.1

Replace the paragraph:

The most relevant metadata for the selection process is summarized in Table 8. The bit fields of the *drcSetEffect* field are described in detail in Table A.32. All parameters that can be supplied by the host to control loudness normalization and dynamic range compression are summarized in Table A.40 and Table A.41, respectively.

With:

The most relevant metadata for the selection process is summarized in Table 8. The bit fields of the *drcSetEffect* field are described in detail in Table A.32. All parameters that can be supplied by the host to control loudness normalization and dynamic range compression are summarized in Table A.40 and Table A.41, respectively.

If the DRC tool is configured to support equalization (EQ) according to 6.8, the selection process includes the EQ-related metadata and the final selection also considers requests for specific EQ, such as EQ dependent on the playback room size. Otherwise, the preselection discards all DRC sets that require EQ, i.e. which have a value of requiresEq==1.

Page 14, 6.3.2.1

Replace:

Table 9 — Requirements for DRC pre-selection

#	Requirement	Applicability	Comment
1	DownmixId of DRC set matches the requested downmixId.	If a downmixId is requested	See 6.3.2.2.
2	Output channel layout of DRC set matches the requested layout.	If a target channel layout is requested	See 6.3.2.2.
3	Channel count of DRC set matches the requested channel count.	If a target channel count is requested	See 6.3.2.2.
4	The DRC set is not a “Fade-” or “Ducking-” only DRC set.	Always	DRC sets with “Fade” or “Ducking” effect are selected automatically. They are not subject to this selection process.
5	The number of DRC bands is supported.	Always	DRC sets that exceed the number of supported DRC bands are discarded. For time-domain DRC, the maximum is four bands.
6	Independent use of DRC set is permitted.	If the DRC set is not used in combination with another DRC set.	DRC sets with a noIndependentUse flag value of 1 can only be used in combination with a second DRC set.
7	The range of the target loudness specified for a DRC set has to include the requested decoder target loudness.	If drcSetTargetLoudnessPresent==1 and no explicit peak information is available for that DRC set.	See 6.3.2.2.2.
8	Clipping is minimized.	Except for DRC sets which were already selected in pre-selection step #7.	See 6.3.2.2.3.

With:

Table 9 — Requirements for DRC pre-selection

#	Requirement	Applicability	Comment
1	DownmixId of DRC set matches the requested downmixId.	If a downmixId is requested	See 6.3.2.2.1.
2	Output channel layout of DRC set matches the requested layout.	If a target channel layout is requested	See 6.3.2.2.1.
3	Channel count of DRC set matches the requested channel count.	If a target channel count is requested	See 6.3.2.2.1.
4	The DRC set is not a “Fade-” or “Ducking-” only DRC set.	Always	DRC sets with “Fade” or “Ducking” effect are selected automatically. They are not subject to this selection process.
5	The number of DRC bands is supported.	Always	DRC sets that exceed the number of supported DRC bands are discarded. For time-domain DRC, the maximum is four bands.