BS ISO/IEC 23003-4:2020+A1:2022



BSI Standards Publication

Information technology — MPEG audio technologies

Part 4: Dynamic range control

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 23003-4:2020+A1:2022

National foreword

This British Standard is the UK implementation of ISO/IEC 23003-4:2020+A1:2022. It supersedes BS ISO/IEC 23003-4:2015, which is withdrawn.

The start and finish of text introduced or altered by amendment is indicated in the text by tags. Tags indicating changes to ISO/IEC text carry the number of the ISO/IEC amendment. For example, text altered by ISO/IEC amendment 1 is indicated by 〈A1〉 〈/A1〉.

The UK participation in its preparation was entrusted to Technical Committee IST/37, Coding of picture, audio, multimedia and hypermedia information.

A list of organizations represented on this committee can be obtained on request to its committee manager.

Contractual and legal considerations

This publication has been prepared in good faith, however no representation, warranty, assurance or undertaking (express or implied) is or will be made, and no responsibility or liability is or will be accepted by BSI in relation to the adequacy, accuracy, completeness or reasonableness of this publication. All and any such responsibility and liability is expressly disclaimed to the full extent permitted by the law.

This publication is provided as is, and is to be used at the recipient's own risk.

The recipient is advised to consider seeking professional guidance with respect to its use of this publication.

This publication is not intended to constitute a contract. Users are responsible for its correct application.

© The British Standards Institution 2022
Published by BSI Standards Limited 2022

ISBN 978 0 539 17368 0

ICS 35.040.40

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 October 2022.

Amendments/corrigenda issued since publication

Date	Text affected

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 23003-4:2020+A1:2022

INTERNATIONAL STANDARD

ISO/IEC 23003-4

Second edition
2020-06

Information technology — MPEG audio technologies —

Part 4: Dynamic range control

Technologies de l'information — Technologies audio MPEG —

Partie 4: Contrôle de gamme dynamique

Reference number
ISO/IEC 23003-4:2020(E)

Documents
Document Sharing Hub
© ISO/IEC 2020

Full document details: BS ISO/IEC 23003-4:2020+A1:2022



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2020

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Fax: +41 22 749 09 47
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

© ISO/IEC 2020 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 23003-4:2020+A1:2022

Contents

Page

Foreword vi

Introduction vii

1 Scope 1

2 Normative references 1

3 Terms, definitions and mnemonics 1
3.1 Terms and definitions 1
3.2 Mnemonics 3

4 Symbols (and abbreviated terms) 3

5 Technical overview 4

6 DRC decoder 6
6.1 DRC decoder configuration 6
6.1.1 Overview 6
6.1.2 Description of logical blocks 7
6.1.3 Derivation of peak and loudness values 12
6.2 Dynamic DRC gain payload 16
6.3 DRC set selection 16
6.3.1 Overview 16
6.3.2 Pre-selection based on Signal Properties and Decoder Configuration 17
6.3.3 Selection based on requests 20
6.3.4 Final selection 22
6.3.5 Applying multiple DRC sets 23
6.3.6 Album mode 23
6.3.7 Ducking 23
6.3.8 Precedence 24
6.4 Time domain DRC application 24
6.4.1 Overview 24
6.4.2 Framing 24
6.4.3 Time resolution 25
6.4.4 Time alignment 25
6.4.5 Decoding 26
6.4.6 Gain modifications and interpolation 29
6.4.7 Spline interpolation 35
6.4.8 Look-ahead in decoder 36
6.4.9 Node reservoir 37
6.4.10 Applying the compression 38
6.4.11 Dynamic equalization 41
6.4.12 Multi-band DRC filter bank 43
6.5 Sub-band domain DRC 47
6.6 Generation of DRC gain values at the decoder 51
6.6.1 Overview 51
6.6.2 Description of logical blocks 52
6.6.3 Algorithmic details 53
6.6.4 Combining parametric and non-parametric DRCs 60
6.7 Loudness equalization support 61
6.8 Equalization tool 62

Documents
Document Sharing Hub

© ISO/IEC 2020 – All rights reserved

Full document details: BS ISO/IEC 23003-4:2020+A1:2022

6.8.1 Overview ... 62
6.8.2 EQ payloads ... 62
6.8.3 EQ filter elements ... 63
6.8.4 EQ set selection ... 64
6.8.5 Application of EQ set ... 64
6.9 Complexity management ... 72
6.9.1 General ... 72
6.9.2 DRC and downmixing complexity estimation ... 72
6.9.3 EQ complexity estimation ... 74
6.10 Loudness normalization ... 75
6.10.1 Overview ... 75
6.10.2 Loudness normalization based on target loudness ... 76
6.11 DRC in streaming scenarios ... 79
6.11.1 DRC configuration ... 79
6.11.2 Error handling ... 79
6.12 DRC configuration changes during active processing ... 79

7 Syntax ... 81
7.1 Syntax of DRC payload ... 81
7.2 Syntax of DRC gain payload ... 81
7.3 Syntax of static DRC payload ... 82
7.4 Syntax of DRC gain sequence ... 109
7.5 Syntax of parametric DRC tool ... 110
7.6 Syntax of equalization tools ... 117

8 Reference software ... 131
8.1 Reference software structure ... 131
8.1.1 General ... 131
8.2 Bitstream decoding software ... 131
8.2.1 General ... 131
8.2.2 MPEG-D DRC decoding software ... 132

9 Conformance ... 132
9.1 General ... 132
9.2 Conformance testing ... 132
9.2.1 Conformance test data and test procedure ... 132
9.2.2 Naming conventions ... 134
9.2.3 File format definitions ... 136
9.3 Encoder Conformance for MPEG-D DRC bitstreams ... 138
9.3.1 Characteristics and test procedure ... 138
9.3.2 Configuration payload ... 139
9.3.3 Interface payload ... 153
9.3.4 Frame Payload ... 156
9.3.5 Requirements depending on profiles and levels ... 157
9.4 Decoder conformance test categories and conditions ... 158
9.4.1 General ... 158
9.4.2 Conformance test categories ... 158
9.4.3 Conformance test conditions ... 158

Annex A (normative) Tables ... 167

Annex B (normative) External Interface to DRC tool ... 207

Annex C (informative) Audio codec specific information ... 220

AVA Documents
Document Sharing Hub

© ISO/IEC 2020 – All rights reserved

Full document details: BS ISO/IEC 23003-4:2020+A1:2022

Annex D (informative) DRC gain generation and encoding **225**
Annex E (informative) DRC set selection and adjustment at decoder **236**
Annex F (informative) Loudness normalization **243**
Annex G (informative) Peak limiter **244**
Annex H (informative) Equalization **249**
Annex I (normative) Profiles and levels **251**
Annex J (informative) Reference software disclaimer **260**
Annex K (informative) Reference software **261**
Bibliography **262**

Documents
Document Sharing Hub

© ISO/IEC 2020 – All rights reserved

Full document details: BS ISO/IEC 23003-4:2020+A1:2022

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.

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 document 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 and IEC 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) or the IEC list of patent declarations received (see http://patents.iec.ch).

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

For an explanation of 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 www.iso.org/iso/foreword.html.

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

This second edition cancels and replaces the first edition (ISO 23003-4:2015), which has been technically revised. It also incorporates the Amendments ISO 23003-4:2015/Amd.1:2017 and ISO 23003-4:2015/Amd.2:2017. The main changes compared to the previous edition are as follows:

— Amendments to the previous edition that include enhancements, definitions of profiles and levels, reference software, and conformance are integrated.

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

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

AVA Documents
Document Sharing Hub

© ISO/IEC 2020 – All rights reserved

Full document details: BS ISO/IEC 23003-4:2020+A1:2022

Introduction

Consumer audio systems and devices are used in a large variety of configurations and acoustical environments. For many of these scenarios, the audio reproduction quality can be improved by appropriate control of content dynamics and loudness.

This document provides a universal dynamic range control tool that supports loudness normalization. The DRC tool offers a bitrate efficient representation of dynamically compressed versions of an audio signal. This is achieved by adding a low-bitrate DRC metadata stream to the audio signal. The DRC tool includes dedicated sections for clipping prevention, ducking, and for generating a fade-in and fade-out to supplement the main dynamic range compression functionality. The DRC effects available at the DRC decoder are generated at the DRC encoder side. At the DRC decoder side, the audio signal may be played back without applying the DRC tool, or an appropriate DRC tool effect is selected and applied based on the given playback scenario.

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.

The International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) draw attention to the fact that it is claimed that compliance with this document may involve the use of a patent.

ISO and IEC take no position concerning the evidence, validity and scope of these patent rights

The holders of these patent rights have assured ISO and IEC that they are willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statements of the holders of these patent rights are registered with ISO and IEC. Information may be obtained from the patent database available at www.iso.org/patents.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those in the patent database. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

© ISO/IEC 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-4:2020+A1:2022

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 23003-4:2020+A1:2022

Information technology — MPEG audio technologies —

Part 4:
Dynamic range control

1 Scope

This document specifies technology for loudness and dynamic range control. It is applicable to most MPEG audio technologies. It offers flexible solutions to efficiently support the widespread demand for technologies such as loudness normalization and dynamic range compression for various playback scenarios.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 14496-12, *Information technology — Coding of audio-visual objects — Part 12: ISO base media file format*

ISO/IEC 14496-26:2010, *Information technology — Coding of audio-visual objects — Part 26: Audio Conformance*

ISO/IEC 23008-3:2019, *Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 3: 3D audio*

ISO/IEC 23091-3, *Information technology — Coding-independent code points — Part 3: Audio*

3 Terms, definitions and mnemonics

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 14496-12 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

3.1.1
DRC sequence
series of DRC gain values that can be applied to one or more audio channels

Documents
Document Sharing Hub

© ISO/IEC 2020 – All rights reserved

Full document details: BS ISO/IEC 23003-4:2020+A1:2022

3.1.2
DRC set
defined set of DRC sequences that produce a desired effect if applied to the audio signal

3.1.3
album
collection of audio recordings that are mastered in a consistent way

Note 1 to entry: Traditionally, a collection of songs released on a Compact Disk belongs into this category, for example.

3.1.4
conformance test bitstream
bitstream used for testing the conformance of MPEG-D DRC compliant audio decoders

3.1.5
conformance test case
conformance test category and a combination of one or more conformance test conditions for which a conformance test sequence is provided

3.1.6
conformance test condition
condition which applies to properties of a conformance test sequence in order to test a certain functionality of the MPEG-D DRC decoder

3.1.7
conformance test criteria
one or more conformance test tools and corresponding parameters applied to verify the conformance for a certain conformance test sequence

3.1.8
conformance test sequence
set of a conformance test bitstream, a decoder setting, an input audio file and a corresponding reference file

3.1.9
decoder input parameters
input parameters that are supplied to an MPEG-D DRC decoder in addition to a conformance test bitstream, a decoder interface bitstream and an input audio file

3.1.10
decoder setting
combination of a decoder interface bitstream and decoder input parameters that are supplied to an MPEG-D DRC decoder

3.1.11
input DRC set selection parameters
input parameter set for testing of a DRC gain decoder instance

Note 1 to entry: This parameter set is solely used for conformance testing in the context of the DRC gain decoder conformance test category (DrcGainDec).

3.1.12
reference audio file
decoded counterpart of a conformance test bitstream, a decoder setting and an input audio file

AVA Documents
Document Sharing Hub

© ISO/IEC 2020 – All rights reserved
Full document details: BS ISO/IEC 23003-4:2020+A1:2022

3.1.13
reference DRC set selection parameters
decoded counterpart of a conformance test bitstream and a decoder setting fed to the DRC set selection process

Note 1 to entry: This parameter set is an intermediate result of an MPEG-D DRC compliant decoder implementation solely used for conformance testing in the context of the DRC selection process test category (DrcSelProc).

3.1.14
reference file
reference audio file or reference DRC set selection parameters

3.2 Mnemonics

bslbf	bit string, left bit first, where "left" is the order in which bit strings are written in the ISO/IEC 14496 series NOTE Bit strings are written as a string of 1s and 0s within single quote marks, for example '1000 0001'. Blanks within a bit string are for ease of reading and have no significance.
byte_align()	number of bits to fill for byte alignment at the offset of n bits: byte_align(n) = 8 ceil (n/8) – n
uimsbf	unsigned integer, most significant bit first
vlclbf	variable length code, left bit first, where "left" refers to the order in which the variable length codes are written
bit(n)	a bit string with n bits in the same format as bslbf
unsigned int(n)	an unsigned integer with n bits in the same format as uimsbf
signed int(n)	a signed integer with n bits, most significant bit first
mod	modulo operator: (x mod y) = x-y floor (x/y)
sizeof(x)	size operator that returns the bit size of a field x
TRUE/FALSE	values of Boolean data type, which correspond to numerical 1 and 0, respectively

4 Symbols

a_i	filter coefficient
b	band index of DRC filter bank (starting at 0)
b_i	filter coefficient
deltaTmin	smallest permitted DRC gain sample interval in units of the audio sample interval

Documents
Document Sharing Hub

© ISO/IEC 2020 – All rights reserved

Full document details: BS ISO/IEC 23003-4:2020+A1:2022

f_c	cross-over frequency in Hz
$f_{c,norm}$	cross-over frequency expressed as fraction of the audio sample rate
$f_{c,norm,SB}(s)$	cross-over frequency of audio decoder sub-band s expressed as fraction of the audio sample rate
	NOTE The cross-over frequency is the upper band edge frequency of the sub-band.
f_s	audio sample rate in Hz
	NOTE If an audio decoder is present, it is the sample rate of the decoded time-domain audio signal.
M_{DRC}	DRC frame size in units of the audio sample interval $1/f_s$
N_{DRC}	maximum permitted number of DRC samples per DRC frame
	NOTE Identical to the number of intervals with a duration of *deltaTmin* per DRC frame.
N_{Codec}	codec frame size in units of the audio sample interval $1/f_s$
π	ratio of a circle's circumference to its diameter
s	audio decoder sub-band index (starting at 0)
z	complex variable of the z-transform

5 Technical overview

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.

A typical system for loudness and dynamic range control in the time domain is shown in Figure 1. A more complex system including downmixer and peak limiter is shown in Figure 2. The decoder part of the DRC tool is driven by metadata that efficiently represents the DRC gain samples and parameters for interpolation. The gain samples can be updated as fast as necessary to accurately represent gain changes down to at least 1 ms update intervals. In the following, the decoder part of the DRC tool is referred to as "DRC decoder", which includes everything except the audio decoder and associated bitstream de-multiplexing.

AVA Documents Document Sharing Hub

© ISO/IEC 2020 – All rights reserved
Full document details: BS ISO/IEC 23003-4:2020+A1:2022

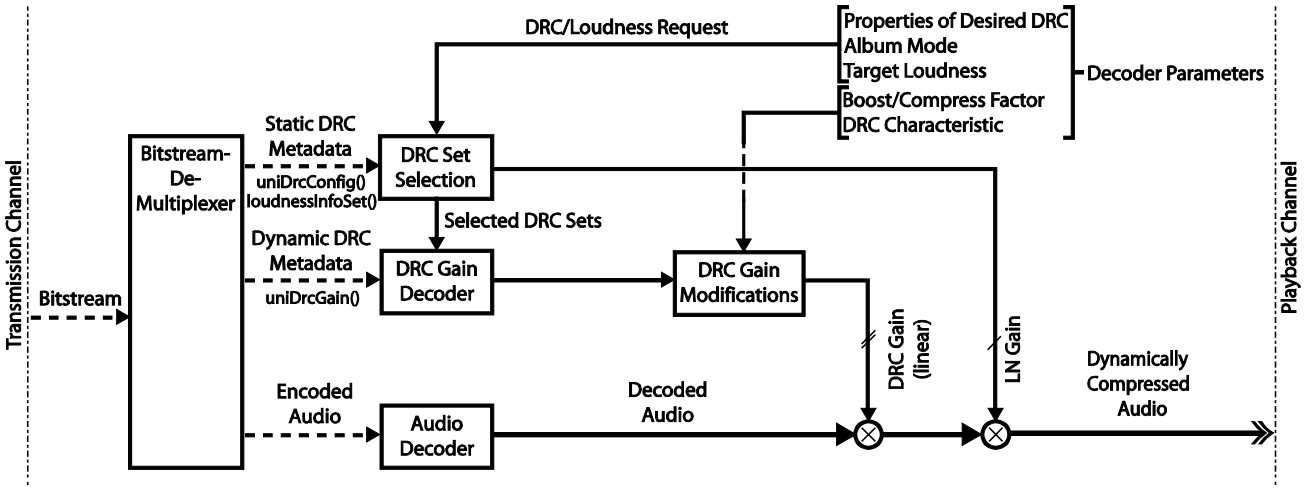


Figure 1 — Block diagram of a typical system with audio decoder and DRC tool modules to achieve loudness normalization (LN) and dynamic range control

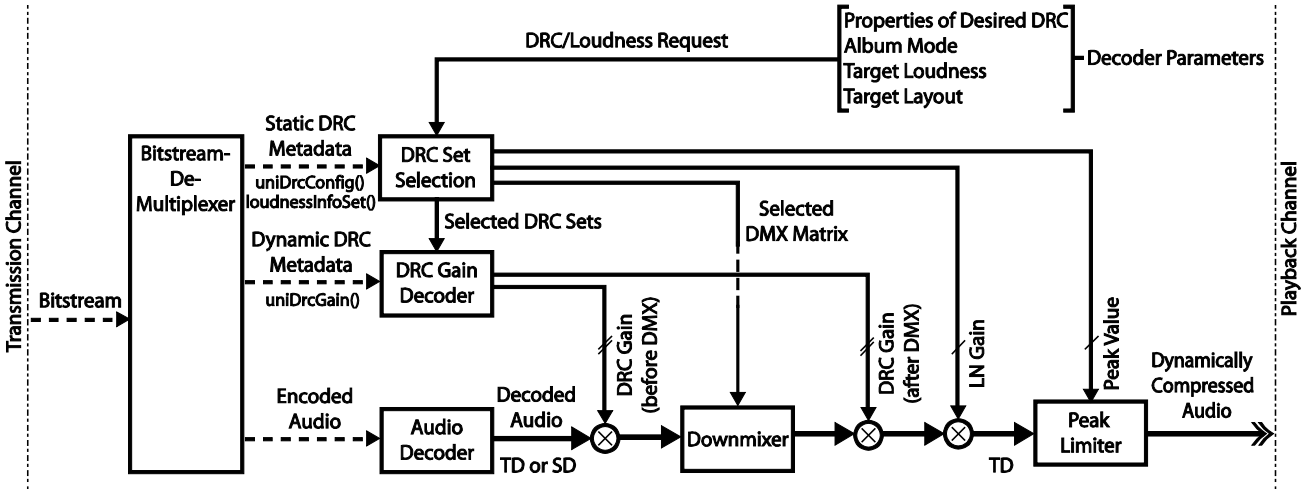


Figure 2 — Block diagram of a more complex system including downmixer and peak limiter (TD = time-domain, SD = subband-domain)

The DRC tool provides support for loudness equalization, 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.

The DRC tool is specified in Clause 6. The tool may be subject to profiles and levels that shall be in accordance with Annex I. The bitstream field decoding of the DRC tool shall be in accordance with Annex A. If an interface for external parameter control of the DRC tool is used, it shall conform to Annex B.

Documents
Document Sharing Hub

© ISO/IEC 2020 – All rights reserved

Full document details: BS ISO/IEC 23003-4:2020+A1:2022

6 DRC decoder

6.1 DRC decoder configuration

6.1.1 Overview

The DRC configuration information can be received in-stream using the static payloads uniDrcConfig() and loudnessInfoSet() described below, or it can be delivered by a higher layer, such as in ISO/IEC 14496-12 (see Table 1). The basic decoding process of the static information is virtually the same. The difference consists mainly in a few syntax changes and reduced field sizes to increase the bit rate efficiency of the in-stream configuration. The syntax of the in-stream static payload is given in 7.3. The associated metadata encoding is given in A.6. The static DRC payload is evaluated once at the beginning of the decoding process and it is monitored subsequently. For static DRC payload changes during playback, see 6.12.

Table 1 — Overview of configuration (setup) and separate metadata track in ISO/IEC 14496-12

	Sample entry code	Setup (in sample entry)	Track reference	Sample format
Audio track	As specified for the audio codec in use (unchanged)	DRCInstructions box using negative values for *drcLocation*	"adrc" referring to the metadata tracks carrying gain values	As specified for the audio codec in use (unchanged)
Metadata track	"unid"	(none)	(none)	Each sample is a uniDrcGain() payload

The static payload is divided into several logical blocks:

— channelLayout();

— downmixInstructions(), downmixInstructionsV1();

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

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

— loudnessInfo(), loudnessInfoV1();

— drcCoefficientsParametricDrc();

— parametricDrcInstructions();

— loudEqInstructions();

— eqCoefficients();

— eqInstructions().

Except for the channelLayout(), drcCoefficientsParametricDrc(), and eqCoefficients(), multiple instances of a logical block can appear. The DRC decoder combines the information of the matching instances of the logical blocks for a given playback scenario. Matching instances are found by matching several identifiers (labels) contained in the blocks.

Documents
Document Sharing Hub

© ISO/IEC 2020 – All rights reserved
Full document details: BS ISO/IEC 23003-4:2020+A1:2022

From the static payload, the decoder can also extract information about the effect of a particular DRC and various associated loudness information, if present. If multiple DRCs are available, this information can be used to select a particular DRC based on target criteria for dynamics and loudness (see 6.3)

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 〚A1〉 UNIDRCCONFEXT_V1 or UNIDRCCONFEXT_V2 〈A1〛 shall preceed an extension payload of type UNIDRCCONFEXT_PARAM_DRC in the bitstream if both payloads are present. For ISO/IEC 14496-12, configuration extension payloads are provided according to Table 76.

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 logical blocks are described in the following.

6.1.2 Description of logical blocks

6.1.2.1 channelLayout()

The channelLayout() block includes the channel count of the audio signal in the base layout. It may also include the base layout unless it is specified elsewhere. For use cases where the base audio signal represents objects or other audio content, the base channel count represents the total number of base content channels. The base channel count value shall serve as the value of baseChannelCount for parsing the downmixInstructions(), downmixInstructionsV1(), drcInstructionsUniDrc(), drcInstructionsUniDrcV1() and eqInstructions() payloads as specified in Clause 7.

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.

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

A drcCoefficients block describes all available DRC gain sequences in one location. The block can have the basic format or the uniDrc format. The basic format, drcCoefficientsBasic(), contains a subset of information included in drcCoefficientsUniDrc() that can be used to describe DRCs other than the ones specified in this document. drcCoefficientsUniDrc() contains for each sequence several indicators on how it is encoded, the time resolution, time alignment, the number of DRC sub-bands and corresponding crossover frequencies and DRC characteristics. The crossover frequencies shall increase

© ISO/IEC 2020 – All rights reserved

Documents Document Sharing Hub
Full document details: BS ISO/IEC 23003-4:2020+A1:2022

with increasing band index. Alternatively, explicit indices in a decoder sub-band domain can be specified for the assignment of DRC sub-bands. The sub-band indices shall also increase with increasing band index. If the DRC gains are applied in the time-domain by using the multi-band DRC filter bank specified in 6.4.12, explicit index signalling is not allowed. The index of the DRC characteristic indicates which compression characteristic was used to produce the gain sequence. The DRC location describes where these gain sequences can be found in the bitstream. The DRC gain sequences in that location are inherently enumerated according to their order of appearance starting with 1.

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.

Table 2 — Encoding of *drcLocation* for in-stream payload

drcLocation n	Payload
0	*Reserved*
1	Location 1 (Codec-specific use)
2	Location 2 (Codec-specific use)
3	Location 3 (Codec-specific use)
4	Location 4 (Codec-specific use)
$n > 4$	*reserved*

Table 3 — Codec-specific encoding of *drcLocation* for MPEG-4 Audio

drcLocation n	Payload
1	uniDrc() (defined in Clause 7)
2	dyn_rng_sgn[i] / dyn_rng_ctl[i] in dynamic_range_info() (defined in ISO/IEC 14496-3:2009 subpart 4)
3	compression_value in MPEG4_ancillary_data() (defined in ISO/IEC 14496-3:2009/AMD 4:2013)
4	*reserved*

The DRC frame size can optionally be specified. It shall be provided if the DRC frame size deviates from the default size specified in 6.4.2. If not specified, the default frame size is used.

The in-stream drcCoefficient syntax is given in Table 65, Table 67 and Table 68. The syntax for the corresponding block for ISO/IEC 14496-12 (ISO base media file format) is shown in Table 66 and Table 69. The corresponding blocks carry essentially the same information. Values that are identically included in both blocks are coded the same way except for *drcLocation*.

〚A1〛 The drcCoefficientsUniDrc() payload for ISO/IEC 14496-12 (see Table 69) for version=2 and *characteristicV1Override=1* carries essentially the same information as the extension UNIDRCCONFEXT_ V2. The corresponding bitstream fields are coded the same way as specified in Table A.10. 〚A1〛

In ISO base media file format (see ISO/IEC 14496-12), for each codec that can be carried in MP4 files and that also carries DRC information, there is a specific definition of how the location is coded, using the *DRC_location* field (see Table 4). A negative value of *DRC_location* indicates that a DRC payload is in an associated meta-data track. That track is the *n*-th linked via a track reference of type "adrc" (audio DRC) from the audio track, where *n* = abs(DRC_location), and the sample-entry type in the meta-data track indicates in which format the coefficients are stored. Table 3 defines the specific entries of the *drcLocation* field for AAC. Some example use cases are discussed in C.10.

AVA Documents Document Sharing Hub

© ISO/IEC 2020 – All rights reserved
Full document details: BS ISO/IEC 23003-4:2020+A1:2022

If the uniDrc() payload is stored in a separate track in the ISO base media file format (ISO/IEC 14496-12), then the track is a metadata track with the sample entry identifier "unid" (uniDrc), with no required boxes added to the sample entry. The time synchronization with the linked audio track is the same as if the payload was in-stream.

Table 4 — Encoding of drcLocation for ISO/IEC 14496-12

drcLocation *n*	Payload
n < 0	*DRC payload located in |n|-th linked meta-data track*
0	*reserved*
1	Location 1 (Codec-specific use)
2	Location 2 (Codec-specific use)
3	Location 3 (Codec-specific use)
4	Location 4 (Codec-specific use)
n > 4	*reserved*

The drcCoefficientsUniDrcV1() payload is defined in Table 68. 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).

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

A drcInstructions block includes information about one specific DRC set that can be applied to achieve a desired effect. This block can have the basic format or the uniDrc format. The basic format, drcInstructionsBasic(), contains a subset of information included in drcInstructionsUniDrc() that can be used to describe DRCs other than the ones specified in this document. The information included in drcInstructionsUniDrc() consists mainly of pre-defined description elements such as the DRC set effect and the DRC gain sequences that are applied. The *drcSetEffect* field contains several effect bits as listed in Table A.45. Multiple bits can be set unless otherwise noted. If no effect bit is set at all, the DRC set is ignored in the DRC set selection (see 6.3). Each drcInstructions block carries a unique non-zero identifier *drcSetId*. A *downmixId* is included to indicate if this DRC set applies to a certain downmix with this identifier. A *downmixId* of zero indicates that the DRC set is applied to the base layout. A *downmixId* of 0x7F indicates that the DRC set can be applied before or after the downmix. Since such a DRC can be applied to any downmix, it has only one channel group including all channels. If a “Ducking” bit is set in the *drcSetEffect* field, the DRC set is applied before any downmix specified by the downmix ID, i.e. the DRC set is always applied to the base layout and the downmix is generated thereafter. The *downmixId* 0x7F is not permitted for a ducking DRC set. In all other cases, the DRC set is applied to the channel configuration indicated by the *downmixId*.

Documents
Document Sharing Hub

© ISO/IEC 2020 – All rights reserved

Full document details: BS ISO/IEC 23003-4:2020+A1:2022

The drcInstructionsUniDrcV1() payload is defined in Table 73. 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 *downmixID* indicates 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 17. 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).

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. 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 15). 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. 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.

The *drcLocation* field is used in the same way as the *drcLocation* field in the drcCoefficients (see 6.1.2.3). Certain entries of the *drcLocation* field allow adding drcInstructions information to gain sequences defined elsewhere. Some use cases are discussed in C.10.

The field *limiterPeakTarget* declares the peak target level used by the encoder-side DRC, if applicable. For example, if a limiter is used to generate the DRC gain sequence, it is configured to control the audio sample magnitude to not exceed this peak target level. *limiterPeakTarget* is represented in dBFS and encoded according to Table A.40.

If *limiterPeakTarget* is present, and the only *drcSetEffect* is "clipping prevention", the gain sequence is to be shifted by the negative sum of *loudnessNormalizationGainDb* and *limiterPeakTarget* if the negative sum is greater than 0. Afterwards, the gain sequence is saturated at the threshold of 0 dB so that only

AVA Documents
Document Sharing Hub

© ISO/IEC 2020 – All rights reserved
Full document details: BS ISO/IEC 23003-4:2020+A1:2022

negative gains (dB) occur. With this mechanism, it is possible to send gains for clipping prevention in expectation of a high *loudnessNormalizationGainDb*. If *loudnessNormalizationGainDb* is lower than expected, the gains are applied only as far as needed, and the dynamic range can be kept as high as possible.

If *gainScalingPresent* == 1, the gain scaling coefficients shall be applied to the channel group. If *gainOffsetPresent* == 1, the gain offset value shall be applied to the channel group as shown in Table 17. Similarly, if *duckingScalingPresent* == 1, the scaling factor shall be applied to the associated ducking gain sequence for that channel group.

The in-stream drcInstructions syntax is given in Table 70, Table 72 and Table 73. The syntax for the corresponding block for ISO/IEC 14496-12 is shown in Table 71 and Table 74. The corresponding blocks carry essentially the same information. Values that are identically included in both blocks are coded the same way except for *drcLocation*. Further information on the coding of *drcLocation* is defined in 6.1.2.3.

6.1.2.5 loudnessInfo() and loudnessInfoV1()

A loudnessInfo() block includes loudness and peak information. A downmix identifier and DRC set identifier indicate which configuration the information applies to. Hence, this block can be associated with the audio signal without DRC and without downmix, or with any specific DRC and/or downmix applied. If a DRC with a dependent DRC set is applied, the loudness information describes the output of the combined DRCs. A loudnessInfo() block can either represent an individual content item or the entire album. Typically, all content items of an album include identical album loudnessInfo() blocks.

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.

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.50). 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.

The *methodDefinition* field according to Table A.49 specifies how the *methodValue* is derived. The mixing level is compatible with “mixlevel” in ATSC A/52[1]. It indicates the absolute acoustic sound pressure level of an individual channel during the final audio mixing session. The peak mixing level is the acoustic level of a sine wave in a single channel whose peaks reach 100 percent in the PCM representation. The absolute SPL value is typically measured by means of pink noise with an RMS value of −20 or −30 dB with respect to the peak RMS sine wave level. The value of mixing level is not typically used within the DRC tool, but may be used by other parts of the audio reproduction system.

The room type field is compatible with “roomtyp” in ATSC A/52[1]. It indicates the type and calibration of the mixing room used for the final audio mixing session. The value of *roomtyp* is not typically used by the DRC tool, but may be used by other parts of the audio reproduction system.

The loudnessInfoSet() payload contains all loudnessInfo() blocks. The in-stream syntax of loudnessInfoSet() is given in Table 58. For the ISO base media file format, the slightly different syntax of “LoudnessBox” is used as defined in ISO/IEC 14496-12.

Documents
Document Sharing Hub

© ISO/IEC 2020 – All rights reserved

Full document details: BS ISO/IEC 23003-4:2020+A1:2022

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.

6.1.3 Derivation of peak and loudness values

The loudnessInfo() blocks provide optional values that describe loudness and peak. Several DRC decoder processes depend on these values, hence, when the loudness information is partially or entirely absent, fallback values are used as shown in Table 5. For peak values, a default value shall be used. Some other values can be drawn from the loudnessInfo() block of the base layout.

Table 5 — Default and fallback values of loudnessInfo

Value	Default	1st fallback: use value from loudnessInfo() of base layout with same DRCsetId	2nd fallback: use value from loudnessInfo() of the base layout without DRC
truePeakLevel	0.0	No	No
samplePeakLevel	0.0	No	No
programLoudness	Undefined	Yes	Yes
anchorLoudness	Undefined	Yes	Yes
loudnessRange	Undefined	No	No
Maximum loudness range	Undefined	No	No
Maximum momentary loudness	Undefined	No	No
Maximum short-term loudness	Undefined	No	No
Short-term loudness	Undefined	No	No
Mixing level	Undefined	Yes	Yes
Room type	Undefined	Yes	Yes

The applicable loudnessInfo() structure for determination of *programLoudness* and *anchorLoudness* is determined as specified in Table 6. The final loudness metadata value is extracted based on the selection steps specified in 6.10.2.

Table 6 — 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) {
```

Documents
Document Sharing Hub

© ISO/IEC 2020 – All rights reserved
Full document details: BS ISO/IEC 23003-4:2020+A1:2022

```
        loudnessInfo = getLoudnessInfoStructure(0x3F, downmixIdRequested);
    } else if (loudnessInfo == NULL) {
        loudnessInfo = getLoudnessInfoStructure(0, downmixIdRequested);
    } else if (loudnessInfo == NULL) {
        loudnessInfo = getLoudnessInfoStructure(0x3F, 0x7F);
    } else if (loudnessInfo == NULL) {
        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;
}
```

The *signalPeakLevel* of a DRC set is determined as specified in Table 7, where peak related metadata entries are selected dependent on their availability and dependent on the *drcSetId*, and the requested *downmixId*. If no explicit peak information is available, *signalPeakLevel* is estimated from downmix coefficients and others. The estimates based on downmix coefficients hold for passive downmixers and might hold for specific active downmixers.

Table 7 — 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);
```

Documents
Document Sharing Hub

© ISO/IEC 2020 – All rights reserved

Full document details: BS ISO/IEC 23003-4:2020+A1:2022

```
    } 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++) {
                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
}
```

Table 7 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 8 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*.

Table 8 — Pseudo code for functions referenced in Table 7

```
truePeakLevelIsPresent(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) {
            if (loudnessInfo[i]->truePeakLevelPresent) return TRUE;
```

Documents
Document Sharing Hub

© ISO/IEC 2020 – All rights reserved
Full document details: BS ISO/IEC 23003-4:2020+A1:2022

```
      }
   }
   return FALSE;
}

getTruePeakLevel(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) {
         if (loudnessInfo[i]->truePeakLevelPresent) {
            return (loudnessInfo[i]->truePeakLevel);
         }
      }
   }
   return error;
}

limiterPeakTargetIsPresent(drcSetId, downmixId) {
   for (i=0; i<drcInstructionsCount; i++) {
      if    (drcInstructions[i]->drcSetId == drcSetId) &&
            ((drcInstructions[i]->downmixId == downmixId) ||
            (drcInstructions[i]->downmixId == 0x7F)) {
         if (drcInstructions[i]->limiterPeakTargetPresent) return TRUE;
      }
   }
   return FALSE;
}

getlimiterPeakTarget(drcSetId, downmixId) {
   for (i=0; i<drcInstructionsCount; i++) {
      if    (drcInstructions[i]->drcSetId == drcSetId) &&
            ((drcInstructions[i]->downmixId == downmixId) ||
            (drcInstructions[i]->downmixId == 0x7F)) {
         if (drcInstructions[i]->limiterPeakTargetPresent)) {
            return (drcInstructions[i]->limiterPeakTarget);
         }
      }
   }
   return error;
}
downmixCoefficientsArePresent(downmixId) {
   for (i=0; i<downmixInstructionsCount; i++) {
      if (downmixInstructions[i]->downmixId == downmixId) {
         if (downmixInstructions[i]->downmixCoefficientsPresent) return TRUE;
      }
   }
   return FALSE;
}
```

Documents
Document Sharing Hub

© ISO/IEC 2020 – All rights reserved

Full document details: BS ISO/IEC 23003-4:2020+A1:2022

```
getDownmixCoefficient(downmixId, outChan, inChan) {
   for (i=0; i<downmixInstructionsCount; i++) {
      if (downmixInstructions[i]->downmixId == downmixId) {
         if (downmixInstructions[i]->downmixCoefficientsPresent) {
            return (downmixInstructions[i]->downmixCoefficient[outChan][inChan]);
         }
      }
   }
   return error;
}
```

6.2 Dynamic DRC gain payload

The dynamic gain sequences for all DRCs are received in-stream or via a metadata track using the uniDrcGain() syntax given in Table 55. Each access unit contains gain sequences for the duration of *drcFrameSize* samples that are decoded according to 6.4.5.

The last part of the uniDrcGain() can include future extension payloads. If a *uniDrcGainExtType* is received that is different from UNIDRCGAINEXT_TERM, the extension payload (*otherBit*) shall be read and discarded.

When gain values are stored in a separate meta-data track under the sample entry code "unid" as described in 6.1.2.3, each sample is a uniDrcGain() padded with 0 to 7 trailing zero bits to the next byte boundary.

6.3 DRC set selection

6.3.1 Overview

A bitstream can carry multiple DRCs for various purposes. At the decoder, the DRC set that best matches the requirements for the given playback scenario is selected. The selection process is performed in three stages:

1. A pre-selection that discards all DRC sets that are not applicable because they do not match a target channel configuration, do not support the decoder target loudness, or have more clipping than requested (see 6.3.2).

2. A main selection process based on requested DRC set features (see 6.3.3).

3. A final selection that picks a single DRC set if multiple DRC sets are applicable (see 6.3.4).

The most relevant metadata for the selection process is summarized in Table 9. The bit fields of the *drcSetEffect* field are described in detail in Table A.45. All parameters that can be supplied by the host to control loudness normalization and dynamic range compression are summarized in Table A.102 and Table A.103, 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.

DRC sets with only a “Fade” or “Ducking” effect are automatically selected by the decoder without using the three-stage selection process. DRC sets with other features can be requested by using DRC decoder settings as described below.

Documents
Document Sharing Hub

© ISO/IEC 2020 – All rights reserved
Full document details: BS ISO/IEC 23003-4:2020+A1:2022

The pool of DRC sets that is subject to the three-stage selection process comprises not only the DRC sets defined in the bitstream (except for "Fade" and "Ducking") but also virtual DRC sets generated in the DRC tool. The virtual DRC sets are placeholders for the cases where no compression is applied to the audio signal, hence their drcSetEffect bits are zero and they correspond to the DRC effect request "None".

Since the DRC set selection is driven by requests, DRC processing shall not be applied if it is not requested. More specifically, dynamic range compression shall not be applied if there is no corresponding request, i.e. *dynamicRangeControlInterfacePresent* == 0 (see Annex B); loudness normalization shall not be applied if it is not requested, i.e. *loudnessNormalizationInterfacePresent* == 0 (see Annex B).

Conceptually it is possible to integrate other DRC sets specified by the drcInstructionsBasic() and drcCoefficientsBasic() syntax in the selection process. However, this is out of scope and not part of this document.

Table 9 — Most relevant metadata for DRC selection at the decoder

Field	Type
drcInstructionsUniDrc->downmixId	Identifier
drcInstructionsUniDrc->limiterPeakTarget	Value
drcInstructionsUniDrc->drcSetEffect	Bit field
drcInstructionsUniDrc->noIndependentUse	Flag
drcInstructionsUniDrc->drcSetTargetLoudnessValueUpper/-Lower	Value
loudnessInfo->truePeakLevel	Value
loudnessInfo->samplePeakLevel	Value
loudnessInfo->program loudness	Value
loudnessInfo->anchor loudness	Value
loudnessInfo->maximum short-term loudness	Value
loudnessInfo->maximum momentary loudness	Value
loudnessInfo->maximum loudness range	Value
drcCoefficientsUniDrc->drcCharacteristic[sequence][band]	Index
drcCoefficientsUniDrc->bandCount[sequence]	Value

6.3.2 Pre-selection based on Signal Properties and Decoder Configuration

6.3.2.1 Overview

The pre-selection selects all DRC sets that fulfill all requirements listed in Table 10. All available DRC sets are analysed in the given order of steps. If no DRC set is selected, no DRC can be applied except for fading or ducking.

Table 10 — 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.

Documents
Document Sharing Hub

© ISO/IEC 2020 – All rights reserved

Full document details: BS ISO/IEC 23003-4:2020+A1:2022

#	Requirement	Applicability	Comment
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.
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	DRC sets that require EQ are only permitted if EQ is supported.	For all drcInstructionsUniDrcV1() payloads.	If EQ is not supported, DRC sets with *requiresEq*==1 are discarded.
8	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.
9	Clipping is minimized.	Except for DRC sets which were already selected in pre-selection step #8.	See 6.3.2.2.3.
NOTE Pre-selection steps #8 and #9 are interpreted as pre-selection steps #7 and #8 in the first edition of this document (ISO/IEC 23003-4:2015). Pre-selection step #7 related to EQ support is first available with the second edition of this document (ISO/IEC 23003-4:2019).			

6.3.2.2 Detailed description of pre-selection steps

6.3.2.2.1 Pre-selection based on downmix ID, channel layout, or channel count (#1,2,3)

Requests for downmix IDs, target channel layout, or target channel count are supported. Only one of these requests will be used based on the following priority:

1. downmix ID(s)
2. target channel layout
3. target channel count

Documents
Document Sharing Hub

© ISO/IEC 2020 – All rights reserved
Full document details: BS ISO/IEC 23003-4:2020+A1:2022

If no downmix ID is requested, the request(s) will be mapped to one or more matching *downmixIds* that are used in the pre-selection process as defined in the following.

If a target channel layout is requested but a downmix ID is not requested, the channel layout is mapped to downmix IDs with a matching layout. If only a target channel count is requested, it is mapped to downmix IDs with a matching target channel count. If no matching downmix IDs can be found, no DRC set can be applied except for fading or ducking.

If no request is present, a *downmixId* of 0x0 (base layout) will be used for the pre-selection.

A DRC set is selected if the requested downmix ID (or the downmix ID, which was generated from a different kind of request) matches one of the defined *downmixId* values of the DRC set. DRC sets which define a *downmixId* of 0x0 or 0x7F automatically pass the pre-selection #1,2,3.

A downmix ID list contains all downmix IDs to be used for pre-selection as described in the previous steps. The pre-selection for these requests is done by selecting all DRC sets that have one of the *downmixIds* of the list or a *downmixId* of 0x0 or 0x7F. A requested *downmixId* of 0x7F is meaningless and therefore not permitted.

6.3.2.2.2 Pre-selection based on drcSetTargetLoudness (#8)

This pre-selection step addresses only DRC sets for which *drcSetTargetLoudnessPresent* equals one and for which no explicit peak information is available. From the DRC sets which match this criterion, only those are selected whose range defined by *drcTargetLoudnessValueUpper/-Lower* includes the requested decoder target loudness (*targetLoudness*). A range check shall include the upper boundary value and exclude the lower boundary value. Pre-selection step #9 is omitted for DRC sets selected in this step. Explicit peak information for a specific DRC set is available if at least one of the following results is TRUE:

— truePeakLevelIsPresent(*drcSetId*, *downmixIdRequested*)

— samplePeakLevelIsPresent(*drcSetId*, *downmixIdRequested*)

— limiterPeakTargetIsPresent(*drcSetId*, *downmixIdOfDrcSet*)

6.3.2.2.3 Pre-selection based on output peak level (#9)

This pre-selection step addresses the problem that can arise if due to loudness normalization or downmixing the maximum peak value exceeds full scale. Two mechanisms can be used to avoid clipping, a DRC set that reduces or eliminates clipping and/or a peak limiter. DRC sets resulting in too high peak values at the output possibly need to be discarded. The peak value of the output signal is computed as

$$outputPeakLevel = signalPeakLevel + loudnessNormalizationGainDb$$

where

signalPeakLevel is the peak value of the signal in dBFS according to Table 7;

loudnessNormalizationGainDb is the scaling factor for the loudness normalization in dB.

NOTE The *loudnessNormalizationGainDb* can be specific for each DRC set if individual loudnessInfo() blocks are present for each DRC set.

All DRC sets are discarded whose *outputPeakLevel* exceeds *outputPeakLevelMax.* The recommended value of *outputPeakLevelMax* is 0 dBFS when no peak limiter is applied after the DRC tool. If a peak

AVA Documents Document Sharing Hub

© ISO/IEC 2020 – All rights reserved

Full document details: BS ISO/IEC 23003-4:2020+A1:2022

limiter is subsequently applied, it is recommended to set *outputPeakLevelMax* to the maximum peak value the peak limiter can handle without introducing severe and audible distortions. By default, the DRC tool assumes that no peak limiter will be applied. The default value for *outputPeakLevelMax* is 0 dBFS. If a peak limiter is applied (*peakLimiterPresent*==1), the default value for *outputPeakLevelMax* is 6 dBFS.

If no DRC set was selected so far (including the pre-selection in step #8), select the DRC sets for which *drcSetTargetLoudnessPresent* equals one and whose range defined by *drcTargetLoudnessValueUpper/-Lower* includes the requested decoder target loudness. If after that no DRC set was selected, the requirement of clipping prevention needs to be relaxed until at least one DRC set is selected. Therefore, the DRC sets with the lowest *outputPeakLevel* are selected. In addition, the DRC sets are selected whose *outputPeakLevel* does not exceed the lowest *outputPeakLevel* by more than 1.0 dB.

At this point, the output signal can have audible distortions, either due to clipping or because a peak limiter is applied at too high levels. This is mitigated by lowering the output level resulting in an intentional deviation from the target loudness. The parameter *loudnessDeviationMax* (which is 63 dB by default) limits the gain reduction that is applied to reduce the distortions (e.g. *loudnessDeviationMax* of 3 dB permits a gain adjustment of −3 dB).

6.3.3 Selection based on requests

6.3.3.1 Overview

In the following, the selected DRC sets are matched with requested features that can describe certain aspects of a DRC set. The features are organized in an ordered list that is given to the DRC decoder. One or more features can be requested (see also Annex E). A feature can be requested multiple times. The decoder will search the available DRC sets to find the best match according to the following rules. Table 11 lists DRC features that are supported for the DRC search.

Table 11 — Requestable features of DRC sets

Index	DRC feature	Comment
0	“Effect type”	Specifies the type of DRC effect
1	“Dynamic range”	Specifies the dynamic range, see Table 13
2	“DRC characteristic”	Specifies the DRC characteristic at the encoder

The decoder works through the feature list item-by-item starting from the first. Conceptually, it takes for each item the selected DRC sets and selects only those that match the requested feature. The following specifies the selection rules for each requested feature in detail. If no features are requested, the selection process is performed as if the effect type feature with effect type “None” has been requested as described in the following section. If that request does not succeed, another feature request attempt is issued using the “General” effect type with the fallback effect types “Night”, “Noisy”, “Limited”, “LowLevel”.

6.3.3.2 Sub-selection by requesting an effect type feature

The effect type as specified by the field *drcSetEffect* in Table A.45 describes the result of applying the DRC set. For DRC sets that carry a non-zero entry in the *dependsOnDrcSet* field, the valid effect types for the combination of both sets are obtained from the effect types of both sets (the depending DRC set and the one with a corresponding *dependsOnDrcSet* value).

Documents
Document Sharing Hub

© ISO/IEC 2020 – All rights reserved
Full document details: BS ISO/IEC 23003-4:2020+A1:2022

Table 12 lists all effect types that can be requested. One effect type can be requested at a time. Requests for effect types can be repeated multiple times with different effect types. From the DRC sets selected so far, all DRC sets that match the requested effect type are selected.

An effect type request is ignored if none of the selected DRC sets match. Consequently, the sub-selection process proceeds with the results of the previous request, if applicable; otherwise the results of the pre-selection are used.

The list of requested effect types contains the desired effect types. Additionally, fallback effect types can follow the desired effect types. The fallback effect types are specified to allow selecting a DRC set even if DRC sets with the desired effect types are not available.

After a request sequence of desired effect types, the sub-selection ends if one or more DRC sets are selected and any fallback requests are ignored. Otherwise, the sub-selection continues with the fallback effects. In that case, the sub-selection ends if one or more matching DRC sets have been selected during the processing of fallback requests and the remaining fallback requests are ignored.

The requests can be specified as a list of requested effect types together with an index of the first fallback request in the list. If, for example, only one single desired effect type is specified, the fallback requests start at position 2. More detailed explanations are given in Annex E.

Table 12 — Requestable DRC effect types and short names for reference

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 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.

6.3.3.3 Sub-selection by requesting a “Dynamic Range” value

The decoder can receive a request for a single Dynamic Range measurement value or value range. The measurement is based on one of those provided in Table 13. If a value is specified, all DRC sets are selected with the closest matching value of the feature. If a range is specified, all DRC sets are selected whose feature value is within that range.

Each of the measurement values for “short-term loudness”, “momentary loudness”, and “top of loudness range” can be given for multiple measurement systems. Permitted measurement systems include EBU R128 and BS.1771-1. The first system shall be selected when searching the available values in order of

Documents
Document Sharing Hub

© ISO/IEC 2020 – All rights reserved

Full document details: BS ISO/IEC 23003-4:2020+A1:2022

increasing measurement system index (see Table A.50) including all reserved indices for future updates, i.e. if a reserved index is found, it shall be interpreted as a valid measurement system.

The program loudness values used in Table 13 shall be based on the following measurement systems (see Table A.50):

1. RMS_C
2. RMS_B
3. RMS_A
4. BS.1770-4

If multiple values are given, choose the first available using the order of the list.

Table 13 — Requestable dynamic range measurement values

Index	Label	Dynamic range measurement	Comment
0	StA	"Short-term loudness peak to average"	Max. short-term loudness minus program loudness (see 6.10.2)
1	MtA	"Momentary loudness peak to average"	Max. momentary loudness minus program loudness (see 6.10.2)
2	TtA	"Top of loudness range to average"	Top of loudness range minus program loudness (see 6.10.2)

6.3.3.4 Sub-selection by requesting a "DRC characteristic"

From the selected DRC sets, all sets are selected with the closest DRC characteristic index according to a matching order. If different DRC characteristics are used within the same DRC set, only the DRC characteristic indices are taken that are in the matching order list. The remaining ones are ignored. If no DRC set is found for the matching order of the requested DRC characteristic in Table 14, this selection step is ignored.

Table 14 — Matching order for DRC characteristic

		Requested DRC characteristic											
		1	2	3	4	5	6	7	8	9	10	11	c; c > 11
Matching order	first	1	2	3	4	5	6	7	8	9	10	11	c
	second	2	3	4	5	6	5	9	10	7	8	10	-
	third	-	1	2	3	4	-	-	-	-	-	9	-

6.3.4 Final selection

This clause uses the term "multiple DRC sets" for DRC sets that are independent of each other and do not include ducking or fading effects.

If the DRC tool is configured to support EQ and if there are still multiple DRC sets selected, select the ones that can be combined with an EQ of the requested "EQ purpose" (see B.3.7.2). If no DRC set can be selected, select the ones that can be combined with an EQ of the default "EQ purpose". If still no DRC set can be selected, ignore this step.

AVA Documents Document Sharing Hub

© ISO/IEC 2020 – All rights reserved
Full document details: BS ISO/IEC 23003-4:2020+A1:2022

If there are still multiple DRC sets selected, select the ones that result in an output peak value of 0.0 or less. If no such DRC set is found, from the ones selected, select the DRC sets that exceed the threshold by the minimum amount. DRC sets selected in pre-selection step #8 are assumed to have an output peak value of 0.0 or less.

If there are still multiple DRC sets selected, select the ones that exactly match the requested downmixId.

If there are still multiple DRC sets selected, select the ones with the minimum number of effect types where the effect bit for "General" is ignored. The reason for ignoring the effect bit "General" is that otherwise DRC sets without "General" would be unjustifiably preferred.

If there are still multiple DRC sets selected, discard any DRC set that was selected in the pre-selection step dealing with *drcSetTargetLoudness*. If all selected DRC sets are discarded in this step, select the DRC set with the smallest *drcSetTargetLoudnessValueUpper* instead.

If there are still multiple DRC sets selected, select the DRC sets for which *drcSetTargetLoudnessPresent* equals one and whose range defined by *drcSetTargetLoudnessValueUpper*/*-Lower* includes the requested decoder target loudness. If multiple DRC sets are selected, select the DRC set with the smallest *drcSetTargetLoudnessValueUpper*.

If there are still multiple DRC sets selected, select the DRC sets with the largest peak value. The output peak value also depends on the modifications of the gains as described in 6.4.6.

If there are still multiple DRC sets selected, choose the one with the largest *drcSetId*.

6.3.5 Applying multiple DRC sets

In the following cases, multiple DRC sets are applied simultaneously. First, if the DRC set selected in 6.3.4 carries a non-zero entry in the *dependsOnDrcSet* field, the depending DRC set is applied together with the selected one. Second, if a DRC set with "Fade" or "Ducking" effect was automatically selected, it is applied simultaneously with the DRC set selected in 6.3.4. Thus, if the DRC set selected in 6.3.4 has a non-zero *dependsOnDrcSet* value, a total of three DRC sets are applied, which is the maximum number permitted. If all three DRC sets are applied to the same layout (*downmixId*), the DRC set referenced in the *dependsOnDrcSet* field shall be applied first, the DRC set selected in 6.3.4 shall be applied thereafter, and the DRC set with "Fade" or "Ducking" effect shall be applied last. If none of the applied DRC sets is a parametric DRC (see 6.6), the DRC set with "Fade" or "Ducking" can alternatively be processed first. If only two DRC sets are applied to the same layout (*downmixId*), the same order applies. If a DRC set with "Fade" effect and another DRC set with "Ducking" effect were both automatically selected, the DRC set with "Fade" effect is ignored. A DRC set with "Fade" effect shall be applied after the downmix, if present, if any of the applied DRC sets is a parametric DRC.

6.3.6 Album mode

When the playback system is in album mode, i.e. successive songs of an album are played back, the DRC selection should be applied first using all DRCs in the loudnessInfo blocks for albums. These blocks may contain *drcSetIds* that point to matching DRCs. Only if there is no match with any DRC in a loudnessInfo block for albums, the selection should proceed to apply the logic described in 6.3.2 to all available DRCs.

In album mode, any "Fade" DRC is not applied. If not in album mode, if an applicable "Fade" DRC exists, it shall be applied. The "Fade" DRC can be applied simultaneously with any other DRC except "Ducking".

6.3.7 Ducking

The base layout and each specific downmix with a unique *downmixId* can have a maximum of one DRC set with a ducking effect. During configuration, the decoder scans all available DRC sets for the active downmix to identify the applicable DRC set for ducking if present. If ducking DRC sets are both defined

AVA Documents
Document Sharing Hub

© ISO/IEC 2020 – All rights reserved

Full document details: BS ISO/IEC 23003-4:2020+A1:2022

for the base layout and the active downmix, select the one that exactly matches the active downmix. If a ducking DRC set is identified and the associated overlaid audio signal is active, the ducking gain sequence is automatically applied to all channels except those that are members of the channel group associated with the ducking DRC set (*drcSetEffect*=="Duck other") or alternatively to all channels that are members of the channel group associated with the ducking DRC set (*drcSetEffect*=="Duck self"). The overlaid audio is defined to be active if at least one non-zero downmix coefficient is applied to it.

Ducking is always applied before any downmix, i.e to the base layout. Hence, the DRC channel groups for the ducking process refer to the base layout. The *downmixId* of the corresponding drcInstructionsUniDrc() indicates how to generate the downmix after the ducking was applied.

A ducking DRC set with *downmixId* 0x0 (baseLayout) is automatically applied independent of the requested *downmixId*. It is therefore recommended to define ducking DRC sets with *downmixId* 0x0 only for specific use cases, where the ducking DRC set should be always applied when DRC processing is enabled.

6.3.8 Precedence

If a bitstream contains the described DRC metadata and other DRC metadata such as MPEG light or heavy compression, the described metadata will take precedence unless the decoder is instructed to apply the other DRC metadata.

6.4 Time domain DRC application

6.4.1 Overview

The DRC gain can be applied to the time-domain audio signal or to the sub-band signals of the audio decoder. The following text first includes a full description of the time-domain DRC. Subsequently, the necessary modifications of the time-domain DRC are described to achieve DRC in sub-bands.

Since the encoder provides only sparsely sampled gain values, the decoder applies interpolation to achieve a smooth gain transition between the samples (see also Annex D for DRC gain generation and encoding). The sample rate of the interpolated gain is the audio sample rate. Each gain sequence can be configured to use either linear interpolation or spline interpolation. For linear interpolation, the interpolated values of one segment between two subsequent gain samples (nodes) are derived from the two gain samples at both ends of the segment. For spline interpolation, they are additionally derived from their slope (derivative). Hence, when transitioning from one segment to the next, the first derivative is continuous as both segments have the same slope at the transition point.

For applications of the DRC tool in tandem with an audio codec, the following parameters are provided to adjust the DRC frame size and time resolution so that codec and DRC processing can be done most efficiently in terms of complexity and delay. The parameters are:

— DRC frame size in units of the audio sample interval

— *deltaTmin* in units of the audio sample interval

— delay mode.

These parameters have default values, but an audio codec specification may override the defaults.

6.4.2 Framing

The DRC gain information is organized in DRC frames. Each DRC frame contains DRC data to generate the DRC gain for the duration of a DRC frame. The DRC frame duration is constant for a given audio item and it is a multiple of the audio sample interval. DRC frames do not overlap.

AVA Documents Document Sharing Hub

© ISO/IEC 2020 – All rights reserved
Full document details: BS ISO/IEC 23003-4:2020+A1:2022

In practice, whenever suitable, the DRC frame size M_{DRC} is recommended to correspond to the same duration as the codec's frame size to minimize delay and complexity. This is the default setting for frame sizes of 1 ms and more. For audio formats with a frame size below 1 ms, as is typically the case for PCM, the default DRC frame size is 32 *deltaTmin* as defined in Formula (1). The default frame size is used, unless the frame size is specified by *drcFrameSize* in the bitstream.

6.4.3 Time resolution

The DRC tool uses a uniform time grid to generate a sparse representation of the DRC gain. The spacing of this grid defines the highest available time resolution *deltaTmin*. The unit of *deltaTmin* is one sample interval at the audio sample rate. For efficiency, *deltaTmin* is chosen to be an integer multiple of the audio sample interval with a corresponding duration between 0.5 ms and 1.0 ms. Preferably, *deltaTmin* is an integer power of 2, so that sample rates can be efficiently converted between audio and DRC. The default values are computed based on Formulae (1) and (2):

$$f_s 0.0005 \text{ s} < deltaTmin \leq f_s 0.001 \text{ s} \tag{1}$$

and

$$deltaTmin = 2^M \tag{2}$$

where

f_s is the audio sample rate, in Hz;

M is the non-negative integer exponent.

Audio sample rates smaller than 1 kHz are not permitted.

For certain DRC frame sizes, it is not possible to find a value of M that fulfils Formula (2) and which results in a *deltaTmin* value that is an integer factor of the DRC frame size. In that case, the default value of *deltaTmin* is the integer value that results in a sample interval with the smallest deviation from 0.75 ms and which is an integer factor of the DRC frame size. The deviation ε is defined in Formula (3):

$$\varepsilon = \left| 1 - \frac{\text{deltaTmin}}{f_s 0.00075 s} \right| \tag{3}$$

If there are several values of deltaTmin that result in the same deviation, the largest value shall be chosen.

Alternatively, the value of *deltaTmin* can be transmitted in the DRC bitstream field *timeDeltaMin*. If it is present, it overrides the default value.

6.4.4 Time alignment

The time alignment specifies the temporal location of each gain sample within a block of *deltaTmin* audio sample intervals. The alignment is conveyed by the *timeAlignment* field in drcCoefficientsUniDrc(). A value of 0 indicates that the gain sample is located at the last audio sample interval of the block, i.e. the sample index at the audio rate is *deltaTmin*-1 assuming that index 0 is the first audio sample interval in the block. A value of 1 indicates that the gain value is located in the centre of the block, i.e. at the sample index of floor((*deltaTmin*-1)/2).

Documents
Document Sharing Hub

© ISO/IEC 2020 – All rights reserved

Full document details: BS ISO/IEC 23003-4:2020+A1:2022

In low-delay mode, a *timeAlignment* value of 1 is not supported. Hence, *timeAlignment* is set to 0 in low-delay mode. The *timeAlignment* value for the gain sequences of all *channelGroups* within a DRC set shall be identical.

6.4.5 Decoding

The DRC gains are derived from linear interpolation or spline interpolation, which is determined by the node coordinates and, in case of spline interpolation, by their associated slopes. The decoding process of the node coordinates consists of the following sequence of tasks:

1) Gather the DRC configuration information (once);

2) Select a DRC set (once);

3) Parse the DRC gain bitstream (per DRC frame);

4) Apply the code tables including Huffman decoding to decode the quantized values (per DRC frame);

5) Undo the differential encoding (per DRC frame).

The basic in-stream DRC configuration information is fully specified in this document including some parts of ISO base media file format that are not included in ISO/IEC 14496-12. The configuration of the core DRC gain decoding is essentially the same for both cases. The following configuration parameters are most relevant for decoding:

— The number of gain sequences: *nDrcGainSequences*;

— The profile for DRC gain coding: *drcGainCodingProfile*;

— The type of DRC gain interpolation: *gainInterpolationType*;

— The assignment of a gain set to each channel. Channels using the same gain set are referred to as channel groups. The total number of groups is *nDrcChannelGroups* (see Table 15);

— The number of DRC bands in a group: *nDrcBands*.

Given these parameters, the part of the DRC bitstream containing the DRC gains (uniDrcGain()) can be parsed and decoded using the algorithms of Table 16 up to Table 22 and the codes of Table A.1 and the following ones based on *gainCodingProfile*.

In the ISO base media file format, the audio content may be carried in multiple tracks where a base track contains the DRC metadata for all tracks. The additional tracks are referenced by the base track using a track reference of type "adda" (additional audio). The channels are all the channels in the base track, plus all the channels in track(s) referenced, in the order of the references. The channel groups apply to all those channels (even if they are channels in a track that is disabled or not currently being played).

Table 15 — Derivation of drcChannelGroups from gainSetIndices

```
uniqueIndex = {-10, -10, ...};
k=0;
if ((drcSetEffect & (3<<10)) != 0) {            /* Ducking */
    uniqueScaling = {-10, -10, ...};
    for (i=0; i<channelCount; i++) {
        match = FALSE;
```

Documents
Document Sharing Hub

© ISO/IEC 2020 – All rights reserved
Full document details: BS ISO/IEC 23003-4:2020+A1:2022

```
            idx = gainSetIndex[i];
            factor = duckingScaling[i];
            if (idx<0) channelGroupForChannel[i] = -1;
            else {
                for (n=0; n<k; n++) {
                    if ((uniqueIndex[n] == idx) && (uniqueScaling[n] == factor)) {
                        match = TRUE;
                        channelGroupForChannel[i] = n;
                        break;
                    }
                }
                if (match == FALSE) {
                    uniqueIndex[k] = idx;
                    uniqueScaling[k] = factor;
                    channelGroupForChannel[i] = k;
                    k++;
                }
            }
        }
    }
    else {
        for (i=0; i<channelCount; i++) {
            match = FALSE;
            idx = gainSetIndex[i];
            if (idx<0) channelGroupForChannel[i] = -1;
            else {
                for (n=0; n<k; n++) {
                    if (uniqueIndex[n] == idx) {
                        match = TRUE;
                        channelGroupForChannel[i] = n;
                        break;
                    }
                }
                if (match == FALSE) {
                    uniqueIndex[k] = idx;
                    channelGroupForChannel[i] = k;
                    k++;
                }
            }
        }
    }
    nDrcChannelGroups = k;
```

The application of codes is expressed in Table 16 by the pseudo-functions decodeInitialGain(), decodeDeltaGain(), decodeTimeDelta(), and decodeSlope(). Differentially encoded values are then converted into absolute values according to Table 16. The decoded result is represented by the gain values *gDRC*[*g*][b][*k*], the time values *tDRC*[*g*][b][*k*], and the slope values *sDRC*[*g*][b][*k*] where *g* is the channel group index, *b* is the band index, and *k* is the node index. The DRC gain sequences are assigned to channel groups and DRC bands using the information in the drcCoefficientsUniDrc() and drcInstructionsUniDrc() payloads or their V1 versions. For linear interpolation, the slope values are set to

AVA Documents Document Sharing Hub

© ISO/IEC 2020 – All rights reserved

Full document details: BS ISO/IEC 23003-4:2020+A1:2022

to 0. They will be neglected during interpolation. Time values are integer numbers relative to the beginning of the DRC frame in units of *deltaTmin*. The audio sample that coincides with the beginning of the DRC frame has a time value of *tDRC*=0.

For *gainCodingProfile* == 3, no dynamic gain sequence is transmitted. Hence, the decoded values are constant. This is useful for generating a constant DRC gain based on the *gainOffset* value.

Table 16 — Decoding of DRC gain sample coordinates and slopes in the dB domain

```
if (timeAlignment==0) {
    timeOffset = -1;
}
else {
    timeOffset = - deltaTmin + floor((deltaTmin-1)/2);
}
for(g=0; g<nDrcChannelGroups; g++) {
    if (gainCodingProfile[g]==3) {
        nDrcBands[g] = 1;
        nNodes[g][b] = 1;
        gDRC[g][b][0] = 0.0;
        tDRC[g][b][0] = drcFrameSize + timeOffset;
        sDRC[g][b][0] = 0.0;
    }
    else {
        for(b=0; b<nDrcBands[g]; b++) {
            gDRC[g][b][0] = decodeInitialGain(gainInitialCode[g][b]);
            if (drcGainCodingMode[g][b] == 0) {
                /* "simple" mode */
                tDRC[g][b][0] = drcFrameSize + timeOffset;
                sDRC[g][b][0] = 0.0;
            }
            else {
                for (k=1; k<nNodes[g][b]; k++) {
                    gDRC[g][b][k] = gDRC[g][b][k-1]+decodeDeltaGain(gainDeltaCode[g][b][k-1]);
                }
                tmp_timeOffset = timeOffset;
                if (frameEndFlag[g][b] == 1) {
                    nodeResFlag = 0;
                    for (k=0; k<nNodes[g][b]-1; k++) {
                        tDRC_Buf = tmp_timeOffset + deltaTmin * decodeTimeDelta(timeDeltaCode[g][b][k]);
                        if (tDRC_Buf > drcFrameSize + timeOffset) {          /* nodes from node reservoir */
                            if (nodeResFlag == 0) {
                                tDRC[g][b][k] = drcFrameSize + timeOffset;
                                nodeResFlag = 1;
                            }
                            tDRC[g][b][k+1] = tDRC_Buf;
                        }
                        else {
                           tDRC[g][b][k] = tDRC_Buf;
                        }
                        tmp_timeOffset = tDRC_Buf;
```

Documents
Document Sharing Hub

© ISO/IEC 2020 – All rights reserved

Full document details: BS ISO/IEC 23003-4:2020+A1:2022

```
                    }
                    if (nodeResFlag == 0) {
                        tDRC[g][b][k] = drcFrameSize + timeOffset;
                    }
                }
                else {
                    for (k=0; k<nNodes[g][b]; k++) {
                        tDRC[g][b][k] = tmp_timeOffset + deltaTmin * decodeTimeDelta(timeDeltaCode[g][b][k]);
                        tmp_timeOffset = tDrc[g][b][k];
                    }
                }
                for (k=0; k<nNodes[g][b]; k++) {
                    if (gainInterpolationType == 0) {
                        sDRC[g][b][k] = decodeSlope (slopeCode[g][b][k]);
                    }
                    else {
                        sDrc[g][b][k] = 0.0; /* slope will be neglected */
                    }
                }
            }
        }
    }
}
```

6.4.6 Gain modifications and interpolation

Before the gain can be applied to the audio signal, it shall be converted to the linear domain and gain values between gain samples shall be interpolated. To achieve lower complexity, the dB to linear conversion is done before the interpolation. Hence, the interpolation process is entirely done in the linear domain. Both the gain modification and conversion to the linear domain are done using the pseudo code of Table 17. The input variables are the gain sample and slope in the dB domain. The output consists of the gain sample and slope in the linear domain.

As described in E.2.4 and E.4, there are several ways to adapt the DRC characteristics in the DRC tool decoder. These adjustments are applied to the decoded gain samples in the dB domain. The function toLinear() includes all necessary steps to generate a linear gain sample from the logarithmic value in dB (see Table 17). It contains a mapping function mapGain() (see Table 18) that supports modifications of the DRC gain values with the purpose of achieving a different compression characteristic than the one used in the encoder. The mapping can be controlled by the host (*drcCharacteristicTarget* > 0) which selects one of the target characteristics based on an index. Otherwise, if specified in the drcInstructionsUniDrcV1() payload, a target characteristic is used and the DRC gain values are mapped to it using the mapGain() function. In that case, different characteristics or scalings can be applied independently to positive and negative gains (see Table 17). Otherwise, the encoder characteristic will not be replaced. A modified characteristic can be generated based on the encoder compression characteristic that is conveyed in the DRC configuration. Moreover, a compression and boost factor is supported to scale negative and positive gains, respectively. These factors have a value of 1.0, unless values in the range [0,1] are supplied by the user. Similarly, an encoder-controlled scaling is applied when *gainScalingPresent*==1 using the scale factors *attenuationScaling* and *amplificationScaling*. When ducking is active, the ducking gains in dB are scaled by the factor *duckingScaling,* if present. The *duckingScaling* factors are conveyed in the drcInstructionsUniDrc() payload for the channel they are applied to, which is in contrast to the *bsGainSetIndex* channel assignment for the "Duck other" effect. User-supplied compression and boost factors shall be applied to all DRC sets except DRC sets with ducking, fading, or clipping effect.

In a drcCoefficientsUniDrcV1() payload, custom DRC characteristics can be defined to support more flexible gain modifications. These parametric characteristics can be used to describe the encoder-side DRC and the target characteristic.

AVA Documents Document Sharing Hub

© ISO/IEC 2020 – All rights reserved

Full document details: BS ISO/IEC 23003-4:2020+A1:2022

If a target characteristic is defined, it shall be applied after inverting the encoder-side characteristic. Hence, for the most general use, the encoder-side characteristic shall be invertible, i.e. have either a negative or positive slope for the entire gain range. Moreover, if the target characteristic has a section with constant gain, the encoder-side characteristic may also have constant gain in those sections. The parametric characteristics are computed as indicated in Table 19 and Table 20. Further details can be found in E.4.

If the only *drcSetEffect* is "clipping prevention", the clipping prevention gains are shifted depending on the target loudness to provide just enough signal attenuation to avoid clipping. The scaling factor is derived based on the *limiterPeakTarget*. See 6.10.2 for the calculation of *loudnessNormalizationGainDb*. The shifting mechanism will not be enabled if there are additionally defined effect bits apart from "clipping prevention".

A1 If the encoder-side characteristic is provided in the bitstream, it is recommended to use linear gain interpolation. ISO/IEC 23091-3 (CICP) encoder characteristics in the range of 65 to 70 are only supported by drcCoefficientsUniDrcV1() as part of a UNIDRCCONFEXT_V2 extension. These characteristics shall not be used otherwise. When a CICP characteristic in this range is inverted in the decoder, loudness normalization shall be applied after the inversion based on the available loudness metadata and encoder normalization gain, if applicable. This is shown in the pseudo code of Table E.3, where the output of the inverse characteristic is computed with an offset depending on the value of *sourceLoudness* and *encDrcNormGainDb. sourceLoudness* is the integrated DRC input loudness at the encoder before any normalization. The value of *sourceLoudness* is obtained from the loudness metadata for the DRC. *encDrcNormGainDb* is the signal gain in dB applied at the encoder DRC input. This gain value is available in a UNIDRCCONFEXT_V2 extension payload to support legacy devices when *characteristicV1Override==1* (see also E.4). The value of *drcInputLoudnessTarget* is the target input loudness of the DRC characteristic applied to generate the DRC gains in the decoder. The drcCoefficientsUniDrc() payload of the Base Media File Format ISO/IEC 14496-12 supports CICP characteristics 65 to 70 only if *version>=2* (see also Table 69).

Table 17 — Conversion of a DRC gain sample and associated slope from dB to linear domain (*slopeIsNegative==1* if the source DRC characteristic has a negative slope)

```
toLinear (gainDb, slopeDb) {
    SLOPE_FACTOR_DB_TO_LINEAR = 0.1151f;        * ln(10) / 20 */
    EFFECT_BIT_CLIPPING = 0x0100;               * drcSetEffect 9 (Clip.Prev.) */
    EFFECT_BIT_FADE   = 0x0200;                 * drcSetEffect 10 (Fade) */
    EFFECT_BITS_DUCKING = 0x0400 | 0x0800;      /* drcSetEffect 11 or 12 (Ducking) */
    gainRatio = 1.0;
    gainDbMod = gainDb;
    if (((drcSetEffect & EFFECT_BITS_DUCKING) == 0) &&
                    (drcSetEffect != EFFECT_BIT_FADE) &&
                    (drcSetEffect != EFFECT_BIT_CLIPPING)) {
        if (drcCharacteristicTarget > 0) {
            gainDbMod = mapGain(gainDb); /* target characteristic from host */
        }
        else if (drcCoefficientsUniDrcV1Present == 1) {
            if (((gainDb >= 0.0) && (slopeIsNegative == 1)) || ((gainDb <= 0.0) && (slopeIsNegative == 0))) {
                if (targetCharacteristicLeftPresent == 1) {
                    gainDbMod = mapGain(gainDb); /* target characteristic in payload */
                }
            }
            else if (((gainDb <= 0.0) && (slopeIsNegative == 1)) || ((gainDb >= 0.0) && (slopeIsNegative == 0))) {
                if (targetCharacteristicRightPresent == 1) {
                    gainDbMod = mapGain(gainDb); /* target characteristic in payload */
                }
            }
        }
        if (gainDbMod < 0.0) {
            gainRatio *= compress;
        }
        else {
            gainRatio *= boost;
        }
```

AVA Documents Document Sharing Hub

© ISO/IEC 2020 – All rights reserved
Full document details: BS ISO/IEC 23003-4:2020+A1:2022

```
    }
    if (gainScalingPresent) {
        if (gainDbMod < 0.0) {
            gainRatio *= attenuationScaling;
        }
        else {
            gainRatio *= amplificationScaling;
        }
    }
    if (duckingScalingPresent && (drcSetEffect & EFFECT_BITS_DUCKING)) {
        gainRatio *= duckingScaling;
    }
    gainLin = pow(2.0, gainRatio * gainDbMod / 6.0);
    slopeLin = SLOPE_FACTOR_DB_TO_LINEAR * gainRatio * gainLin * slopeDb;
    if (gainOffsetPresent) {
        gainLin *= pow(2, gainOffset/6.0);
    }
    /* The only drcSetEffect is "clipping prevention" */
    if (limiterPeakTargetPresent && (drcSetEffect == EFFECT_BIT_CLIPPING)) {
        gainLin *= pow(2, max(0.0, -limiterPeakTarget-loudnessNormalizationGainDb
                -loudnessNormalizationGainModificationDb)/6.0);
        if (gainLin >= 1.0) {
            gainLin = 1.0;
            slopeLin = 0.0;
        }
    }
    return (gainLin, slopeLin);
}
```

Table 18 — DRC gain mapping according to a target DRC characteristic

```
mapGain(gainQuant) {
    inLevel = inverseCompressorIO(gainQuant);
    outgain = targetCompressorIO(inLevel);
    return outgain;
}
```

Table 19 — Pseudo code to compute the DRC gain based on the input level in dB (*characteristicFormat*==0)

```
drcInputLoudnessTarget = -31;
compressorIO_sigmoidLeft(inLevelDb) {
    if (inLevelDb > drcInputLoudnessTarget) {
        printf("ERROR!");
        return (0.0);
    }
    tmp = (drcInputLoudnessTarget – inLevelDb) * ioRatioLeft;
    if (expLeft < INF) {                                    /* INF means infinity */
        outGainDb = tmp / pow(1 + pow(tmp/gainDbLeft, expLeft), 1/expLeft);
    } else {
        outGainDb = tmp;
```

Documents
Document Sharing Hub

© ISO/IEC 2020 – All rights reserved

Full document details: BS ISO/IEC 23003-4:2020+A1:2022

```
    }
    if (flipSignLeft == 0) return outGainDb;
    else return -outGainDb;
}
compressorIO_sigmoidRight(inLevelDb) {
    if (inLevelDb < drcInputLoudnessTarget) {
        printf("ERROR!");
        return (0.0);
    }
    tmp = (drcInputLoudnessTarget – inLevelDb) * ioRatioRight;
    if (expRight < INF) {                                   /* INF means infinity */
        outGainDb = tmp / pow(1 + pow(tmp/gainDbRight, expRight), 1/expRight);
    } else {
        outGainDb = tmp;
    }
    if (flipSignRight == 0) return outGainDb;
    else return -outGainDb;
}
```

Table 20 — Pseudo code to compute the DRC gain based on the input level in dB (*characteristicFormat*==1)

```
drcInputLoudnessTarget = -31
compressorIO_nodesLeft(inLevelDb) {
    if (inLevelDb > drcInputLoudnessTarget) {
        printf("ERROR!");
        return (0.0);
    }
    nodeLevel[0] = drcInputLoudnessTarget;
    nodeGain[0] = 0.0;
    for (n=1; n<=characteristicNodeCount; n++) {
        nodeLevel[n] = nodeLevel[n-1] - nodeLevelDelta[n];
    }
    for (n=1; n<=characteristicNodeCount; n++) {
        if ((inLevelDb <= nodeLevel[n-1]) && (inLevelDb > nodeLevel[n])) {
            w = (nodeLevel[n]- inLevelDb)/(nodeLevel[n]-nodeLevel[n-1]);
            return (w * nodeGain[n-1] + (1.0-w) * nodeGain[n]);
        }
    }
    return (nodeGain[characteristicNodeCount]);
}
compressorIO_nodesRight(inLevelDb) {
    if (inLevelDb < drcInputLoudnessTarget) {
        printf("ERROR!");
        return (0.0);
    }
    nodeLevel[0] = drcInputLoudnessTarget;
    nodeGain[0] = 0.0;
    for (n=1; n<=characteristicNodeCount; n++) {
        nodeLevel[n] = nodeLevel[n-1] + nodeLevelDelta[n];
```

Documents
Document Sharing Hub

© ISO/IEC 2020 – All rights reserved
Full document details: BS ISO/IEC 23003-4:2020+A1:2022

```
    }
    for (n=1; n<=characteristicNodeCount; n++) {
        if ((inLevelDb >= nodeLevel[n-1]) && (inLevelDb < nodeLevel[n])) {
            w = (nodeLevel[n]- inLevelDb)/(nodeLevel[n]-nodeLevel[n-1]);
            return (w * nodeGain[n-1] + (1.0-w) * nodeGain[n]);
        }
    }
    return (nodeGain[characteristicNodeCount]);
}
```

The gain interpolation is implemented by the pseudo code in Table 21. Dependent on the configuration variable *gainInterpolationType*, either spline interpolation or linear interpolation is performed. The input variables are:

— the time difference between the two gain samples in units of the target sample rate interval *tGainStep*

— a pair of subsequent gain samples *gain0* and *gain1* in dB

— a pair of corresponding slope steepness values *slope0* and *slope1* in the dB domain. These are neglected for linear interpolation.

This function uses toLinear() to convert the variables to the linear domain. The result is a smooth sequence of gain values at the target sample rate located between the pair of gain samples. The target sample rate is the sample rate of the compressed audio signal.

Table 21 — Interpolation of the DRC gain for one spline or linear segment

```
interpolateDrcGain(tGainStep, gain0, gain1, slope0, slope1)
{
    int n;
    float k1, k2, a, b, c, d;

    float slopeLeft;
    float slopeRight;
    float gainLeft;
    float gainRight;

    (gainLeft, slopeLeft) = toLinear (gain0, slope0);
    (gainRight, slopeRight) = toLinear (gain1, slope1);

    if (gainInterpolationType == 0) {
        slopeLeft = slopeLeft / deltaTmin;
        slopeRight = slopeRight / deltaTmin;

        bool useCubicInterpolation = TRUE;
        int tConnect;
        float tConnectFloat;
        if (abs(slopeLeft) > abs(slopeRight)) {
            tConnectFloat = 2.0 * (gainRight - gainLeft - slopeRight * tGainStep) / (slopeLeft - slopeRight);
            tConnect = (floor) (0.5 + tConnectFloat);
            if ((tConnect >= 0) && (tConnect < tGainStep)) {
```

AVA Documents
Document Sharing Hub

© ISO/IEC 2020 – All rights reserved

Full document details: BS ISO/IEC 23003-4:2020+A1:2022

```
                useCubicInterpolation = FALSE;
                result[0] = gainLeft;
                result[tGainStep] = gainRight;
                a = (slopeRight - slopeLeft) / (tConnectFloat + tConnectFloat);
                b = slopeLeft;
                c = gainLeft;
                for (n=1; n<tConnect; n++) {
                    float t = (float) n;
                    result[n] = (a * t + b ) * t + c;
                    result[n] = max (0.0, result[n]);
                }
                a = slopeRight;
                b = gainRight;
                for (     ; n<tGainStep; n++) {
                    float t = (float) (n - tGainStep);
                    result[n] = a * t + b;
                }
            }
        }
        else if (abs(slopeLeft) < abs(slopeRight)) {
            tConnectFloat = 2.0 * (gainLeft - gainRight + slopeLeft * tGainStep) / (slopeLeft - slopeRight);
            tConnectFloat = tGainStep - tConnectFloat;
            tConnect = (floor) (0.5 + tConnectFloat);
            if ((tConnect >= 0) && (tConnect < tGainStep)) {
                useCubicInterpolation = FALSE;
                result[0] = gainLeft;
                result[tGainStep] = gainRight;
                a = slopeLeft;
                b = gainLeft;
                for (n=1; n<tConnect; n++) {
                    float t = (float) n;
                    result[n] = a * t + b;
                }
                a = (slopeRight - slopeLeft) / (2.0 * (tGainStep - tConnectFloat));
                b = - slopeRight;
                c = gainRight;
                for (     ; n<tGainStep; n++) {
                    float t = (float) (tGainStep-n);
                    result[n] = (a * t + b ) * t + c;
                    result[n] = max (0.0, result[n]);
                }
            }
        }
        if (useCubicInterpolation == TRUE) {
            float tGainStepInv = 1.0 / (float)tGainStep;
            float tGainStepInv2 = tGainStepInv * tGainStepInv;

            k1 = (gainRight - gainLeft) * tGainStepInv2;
            k2 = slopeRight + slopeLeft;
            a = tGainStepInv * (tGainStepInv * k2 - 2.0 * k1);
```

Documents
Document Sharing Hub

© ISO/IEC 2020 – All rights reserved

Full document details: BS ISO/IEC 23003-4:2020+A1:2022

```
                b = 3.0 * k1 - tGainStepInv * (k2 + slopeLeft);
                c = slopeLeft;
                d = gainLeft;
                result[0] = gainLeft;
                result[tGainStep] = gainRight;
                for (n=1; n<tGainStep; n++) {
                    float t = (float) n;
                    result[n] = (((a * t + b ) * t + c ) * t ) + d;
                    result[n] = max (0.0, result[n]);
                }
            }
        }
        else {
            a = (gainRight – gainLeft) / tGainStep;
            b = gainLeft;

            result[0] = gainLeft;
            result[tGainStep] = gainRight;

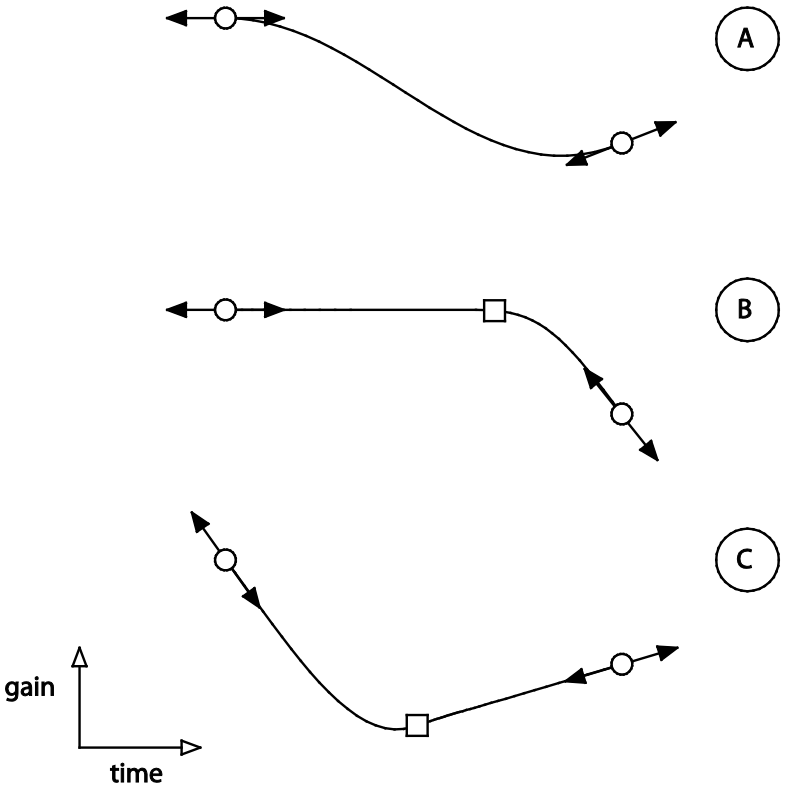
            for (n=1; n<tGainStep; n++) {
                float t = (float)n;
                result[n] = a * t + b;
            }
        }
        return result;
}
```

6.4.7 Spline interpolation

Interpolation of the DRC gain in the decoder is based on pairs of gain samples. For spline interpolation, in addition to the node coordinates (time and value in dB), also the slope information is given for each pair. The decoder will choose one of three available types of interpolation as illustrated in Figure 3. In most cases, cubic interpolation is chosen. However, under certain conditions, a hybrid interpolation combining linear and quadratic interpolation is applied instead. For the hybrid interpolation, a node is inserted between the two given nodes (shown as a square). At one side of that node, linear interpolation is applied and quadratic interpolation is applied at the other. The method is fully specified in 6.4.6.

© ISO/IEC 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-4:2020+A1:2022



NOTE A square indicates the connection point of the linear and quadratic sub-segment.

Figure 3 — Spline segment types: cubic interpolation (A), hybrid using linear and quadratic interpolation (B, C)

6.4.8 Look-ahead in decoder

The DRC tool decoder can be operated in one of two delay modes. The low-delay mode immediately applies the decoded DRC gain while the default mode applies the DRC gain with a delay of one DRC frame. The low-delay mode requires that the flag *fullFrame* is 1, which signals that a gain value sample is located at the end of each DRC frame. For the default mode, the flag *fullFrame* can also be 0. Then it supports gain sample interpolation from any position of the current DRC frame to any position of the next DRC frame.

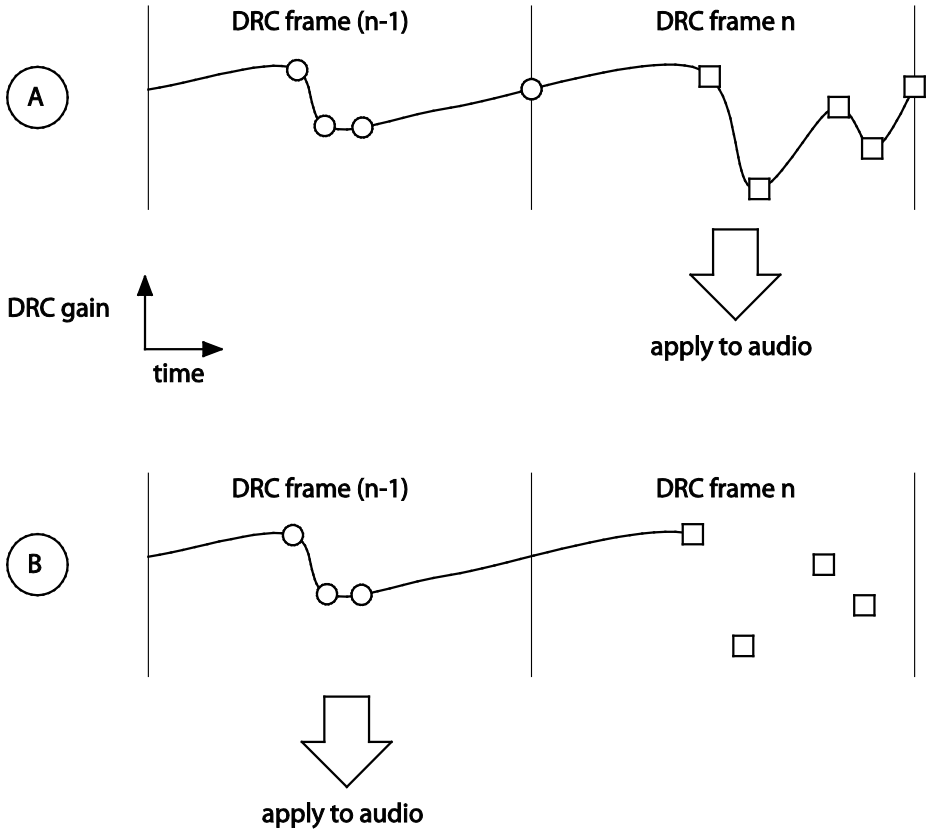
Figure 4 illustrates the two delay modes. The upper diagram shows that each DRC frame has a spline node at the end of the frame, so that the entire DRC gain curve for that frame can immediately be generated by interpolation. The lower diagram shows that the interpolated gain curve is applied with a delay of one DRC frame, since the interpolation for frame n-1 can only be completed after the first node of frame n is received.

For common perceptual codecs, the default delay mode does not require additional decoder delay. The delay is already required due to the overlap-add operation. The decoder appends audio samples with zeros at the end to reconstruct the last frame. If the DRC gains for the last frame do not cover the entire duration, the last gain shall be repeated until the end of the frame. If DRC gains are required for the first frame in default delay mode, they should be generated based on a node at the beginning of the first frame with 0 dB gain and, when in spline interpolation mode, with a slope of 0.

AVA Documents Document Sharing Hub

© ISO/IEC 2020 – All rights reserved

Full document details: BS ISO/IEC 23003-4:2020+A1:2022



Key

circles spline nodes received in frame *n*-1

squares spline nodes received in frame *n*

NOTE The solid line illustrates the interpolated DRC gain up to DRC frame *n.*

Figure 4 — Delay modes: low-delay and *fullFrame* = 1 (A), default delay and *fullFrame* = 0 (B).

The low-delay mode is suitable for decoders that do not have inherent delay such as a delay due to overlap-add. For instance, this is the case for some lossless codecs. For any PCM audio format, low-delay mode shall be used.

For ISO/IEC 14496 (ISO base media file format), the delay mode is explicitly conveyed in the drcCoefficientsUniDrc() payload in the *delayMode* field (see Table 69). The encoding of *delayMode* is given in Table A.20.

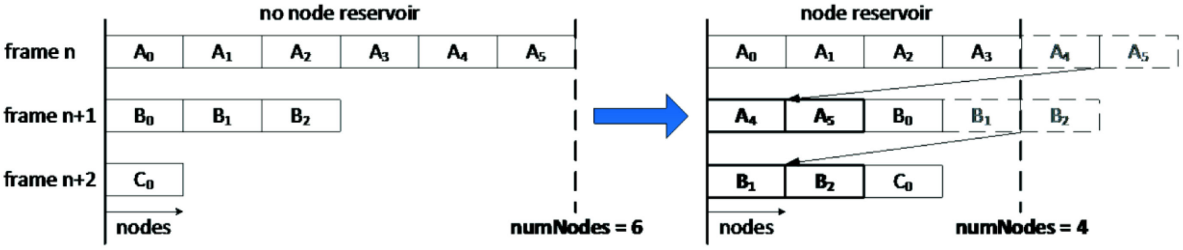
6.4.9 Node reservoir

To reduce bitrate peaks, a so-called “node reservoir” can be employed. The node reservoir can be only applied in the default delay mode, where the DRC gain is applied with a delay of one DRC frame (see 6.4.8). In addition, the *fullFrame* flag shall be set to 0. The node reservoir mechanism shifts nodes from the current frame into the subsequent frame (see Figure 5). In order to keep the decoding delay constant, it is not allowed to shift the first node of a frame since in default delay mode, the first node is needed to fully decode the previous frame. Thus, the maximum number of nodes that can be shifted to the subsequent frame is limited to *nNodes-1* nodes.

Documents
Document Sharing Hub

© ISO/IEC 2020 – All rights reserved

Full document details: BS ISO/IEC 23003-4:2020+A1:2022



NOTE A_x, B_x, and C_x are nodes regularly transmitted in DRC frame n, n+1, and n+2, respectively (left). When the node reservoir is used, some nodes (bold) are transmitted with a delay of one frame (right).

Figure 5 — Illustration of node reservoir mechanism

Each node is defined by time, gain, and slope data (*gainInterpolationType* = 0). The gain and slope data have chronological order in the bitstream, therefore the data from the node reservoir precedes the regular nodes of a particular frame in the bitstream. During decoding, the nodes of the node reservoir can be identified because their time values are encoded with an offset of *drcFrameSize*. Therefore, the use of the node reservoir is not explicitly signalled. The offset is subtracted in the decoding process to obtain the correct time value.

The number of nodes shifted to the node reservoir in each frame is not transmitted explicitly, but results implicitly from the number of time values larger than *drcFrameSize*. The decoding of time values precedes the decoding of gain and slope values. Therefore, the assignment of gain and slope values to time values is known after decoding the time values. The decoding process is fully specified in 6.4.5 and 6.4.10.

6.4.10 Applying the compression

The interpolated gain values of each interpolation segment are concatenated to generate a complete gain vector *gain*[*g*][*b*][*t*] for the entire DRC frame. Finally, the gain vector is applied as shown in Table 22. The function *channelInDrcGroup*() returns TRUE if the current channel *c* belongs to the current DRC channel group. The scheduling of the interpolation segments depends on the delay mode as indicated in Table 22.

Table 22 — Concatenation of interpolation segments to a gain vector and application of the DRC gain vector to the audio channels

```
for(g=0; g<nDrcChannelGroups; g++) {
  for(b=0; b<nDrcBands[g]; b++) {
    if (delayMode == DELAY_MODE_DEFAULT) {
      nNodesNodeReservoir = 0;
      for (k=0; k<nNodes[g][b]; k++) {
        if (tDRC[g][b][k] >= drcFrameSize) {
          nNodesNodeReservoir++;
        }
      }
      for (k=0; k<nNodesNodeReservoir; k++) {
        tDRCprev[g][b][nNodesPrev[g][b]+k]=tDRC[g][b][nNodes[g][b]-nNodesNodeReservoir+k]-drcFrameSize;
        gDRCprev[g][b][nNodesPrev[g][b]+k]=gDRC[g][b][k];
        sDRCprev[g][b][nNodesPrev[g][b]+k]=sDRC[g][b][k];
      }
      memmove(&gDRC[g][b][0],&gDRC[g][b][nNodesNodeReservoir], nNodes[g][b]-nNodesNodeReservoir);
```

AVA Documents Document Sharing Hub

© ISO/IEC 2020 – All rights reserved
Full document details: BS ISO/IEC 23003-4:2020+A1:2022

```
    memmove(&sDRC[g][b][0],&sDRC[g][b][nNodesNodeReservoir], nNodes[g][b]-nNodesNodeReservoir);
    nNodesPrev[g][b] = nNodesPrev[g][b] + nNodesNodeReservoir;
    nNodes[g][b] = nNodes[g][b] – nNodesNodeReservoir;
    for (k=0; k<nNodesPrev[g][b]-1; k++) {
      duration = tDRCprev[g][b][k+1] – tDRCprev[g][b][k];
      interpolationSegment= interpolateDrcGain(duration, gDRCprev[g][b][k],
                  gDRCprev[g][b][k+1], sDRCprev[g][b][k], sDRCprev[g][b][k+1]);
      for (t=0; t<duration; t++) {
        gain[g][b][t+tDRCprev[g][b][k]] = interpolationSegment[t];
      }
    }
    k = nNodesPrev[g][b]-1;
    duration = drcFrameSize + tDRC[g][b][0] – tDRCprev[g][b][k];
    interpolationSegment = interpolateDrcGain(duration, gDRCprev[g][b][k],
        gDRC[g][b][0], sDRCprev[g][b][k], sDRC[g][b][0]]);
    for (t=0; t<duration; t++) {
      gain[g][b][t+tDRCprev[g][b][k]] = interpolationSegment[t];
    }
  }
  else {
    k = nNodesPrev[g][b]-1;
    duration = tDRC[g][b][0]+1;
    interpolationSegment = interpolateDrcGain(duration, gDRCprev[g][b][k],
          gDRC[g][b][0], sDRCprev[g][b][k], sDRC[g][b][0]]);
    for (t=1; t<=duration; t++) {
      gain[g][b][t-1] = interpolationSegment [t];
    }
    for (k=0; k<nNodes[g][b]-1; k++) {
      duration = tDRC[g][b][k+1] – tDRC[g][b][k];
      interpolationSegment = interpolateDrcGain(duration, gDRC[g][b][k],
          gDRC[g][b][k+1], sDRC[g][b][k], sDRC[g][b][k+1]);
      for (t=1; t<=duration; t++) {
        gain[g][b][t+tDRC[g][b][k]] = interpolationSegment [t];
      }
    }
  }

  /* Apply gain to DRC bands of audio in each channel */

  for (c=0; c<nChannels; c++) {
    if (channelInDrcGroup(c, g)) {
      for (t=0; t<drcFrameSize; t++) {
        audioBandOut[c][b][t] = audioBandIn[c][b][t] * gain[g][b][t];
      }
    }
  }
  if (delayMode == DELAY_MODE_DEFAULT) {
    for (k=0; k<nNodes; k++) {
      gDRCprev[g][b][k] = gDRC[g][b][k];
      sDRCprev[g][b][k] = sDRC[g][b][k];
```

Documents
Document Sharing Hub

© ISO/IEC 2020 – All rights reserved

Full document details: BS ISO/IEC 23003-4:2020+A1:2022

```
            tDRCprev[g][b][k] = tDRC[g][b][k];
          }
          nNodesPrev[g][b] = nNodes[g][b];
          for (t=0; t<drcFrameSize; t++) {
            gain[g][b][t] = gain[g][b][t + drcFrameSize];
          }
        }
        else {
          k=nNodes[g][b]-1;
          gDRCprev[g][b][k] = gDRC[g][b][k];
          sDRCprev[g][b][k] = sDRC[g][b][k];
          nNodesPrev[g][b] = nNodes[g][b];
        }
      }
    }
    for(g=0; g<nDrcChannelGroups; g++) {
      for(c=0; c<nChannels; c++) {
        if (channelInDrcGroup(c, g)) {
          for (t=0; t<drcFrameSize; t++) {
            sum = 0.0;
            for(b=0; b<nDrcBands[g]; b++) {
              sum = sum + audioBandOut[c][b][t];
            }
            audioSampleOut[c][t] = sum;
          }
        }
      }
    }
```

Table 22 is based on the following assumption:

— *interpolationSegment* is a vector that contains the gain values of one interpolation segment.

— *duration* is an integer number describing the duration of the interpolation segment in units of audio sample intervals.

— *nNodes* is the number of gain values in the current DRC frame.

— *drcFrameSize* is the number of audio sample intervals in a DRC frame.

— Initialization of the following variables:

```
gDRCprev[g][b][0]=0.0,
sDRCprev[g][b][0]=0.0,
nNodesPrev[g][b]=1,
if (timeAlignment==0)
      tDRCprev[g][b][0]=drcFrameSize-1
else
      tDRCprev[g][b][0]=drcFrameSize-timeDeltaMin+floor((timeDeltaMin-1)/2).
```

Documents
Document Sharing Hub

© ISO/IEC 2020 – All rights reserved

Full document details: BS ISO/IEC 23003-4:2020+A1:2022