|  |
| --- |
| 3GPP TS 26.249 V0.2.0 (2024-04) |
| Technical Specification |
| 3rd Generation Partnership Project;Technical Specification Group SA;Immersive Audio for Split Rendering Scenarios;Detailed Algorithmic Description of Split Rendering Functions(Release 18) |
|   |
|  |  |
|  |
| The present document has been developed within the 3rd Generation Partnership Project (3GPP TM) and may be further elaborated for the purposes of 3GPP.The present document has not been subject to any approval process by the 3GPPOrganizational Partners and shall not be implemented.This Specification is provided for future development work within 3GPPonly. The Organizational Partners accept no liability for any use of this Specification.Specifications and Reports for implementation of the 3GPP TM system should be obtained via the 3GPP Organizational Partners' Publications Offices. |

|  |
| --- |
|  |
| ***3GPP***Postal address3GPP support office address650 Route des Lucioles - Sophia AntipolisValbonne - FRANCETel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16Internethttp://www.3gpp.org |
| ***Copyright Notification***No part may be reproduced except as authorized by written permission.The copyright and the foregoing restriction extend to reproduction in all media.© 2024, 3GPP Organizational Partners (ARIB, ATIS, CCSA, ETSI, TSDSI, TTA, TTC).All rights reserved.UMTS™ is a Trade Mark of ETSI registered for the benefit of its members3GPP™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational PartnersLTE™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational PartnersGSM® and the GSM logo are registered and owned by the GSM Association |

Contents

Foreword 4

Introduction 5

1 Scope 6

2 References 6

3 Definitions of terms, symbols and abbreviations 6

3.1 Terms 6

3.2 Symbols 6

3.3 Abbreviations 7

4 General description of split renderer 7

4.1 Introduction 7

4.2 ISAR system overview 7

5 ISAR baseline 8

5.1 Overview 8

5.2 Split pre-rendering 8

5.2.1 Clauses referenced to [4] 8

5.2.2 Interfaces for split pre-rendering 8

5.2.3 Interface requirements 8

5.3 Intermediate split renderer format 9

5.3.1 Clauses referenced to [4] 9

5.4 Split post-rendering 10

5.4.1 Clauses referenced to [4] 10

5.4.2 Interfaces for split post-rendering 10

5.4.3 Interface requirements 10

Annex A (informative): ISAR Application Programming Interfaces 11

ISAR pre-renderer API 11

ISAR post-renderer API 17

Annex B (normative): RTP Payload Format and SDP Parameters 21

Annex C (normative): ISAR Reference Source Code 21

Annex D (normative): Test Vectors 21

Annex E (informative): Change history 21

# Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

In the present document, modal verbs have the following meanings:

**shall** indicates a mandatory requirement to do something

**shall not** indicates an interdiction (prohibition) to do something

The constructions "shall" and "shall not" are confined to the context of normative provisions, and do not appear in Technical Reports.

The constructions "must" and "must not" are not used as substitutes for "shall" and "shall not". Their use is avoided insofar as possible, and they are not used in a normative context except in a direct citation from an external, referenced, non-3GPP document, or so as to maintain continuity of style when extending or modifying the provisions of such a referenced document.

**should** indicates a recommendation to do something

**should not** indicates a recommendation not to do something

**may** indicates permission to do something

**need not** indicates permission not to do something

The construction "may not" is ambiguous and is not used in normative elements. The unambiguous constructions "might not" or "shall not" are used instead, depending upon the meaning intended.

**can** indicates that something is possible

**cannot** indicates that something is impossible

The constructions "can" and "cannot" are not substitutes for "may" and "need not".

**will** indicates that something is certain or expected to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document

**will not** indicates that something is certain or expected not to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document

**might** indicates a likelihood that something will happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

**might not** indicates a likelihood that something will not happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

In addition:

**is** (or any other verb in the indicative mood) indicates a statement of fact

**is not** (or any other negative verb in the indicative mood) indicates a statement of fact

The constructions "is" and "is not" do not indicate requirements.

# Introduction

An essential architectural characteristic of XR clients is the reliance on a functional split between a set of composite pre-renderers that are implemented as parts of a presentation engine and a set of post-rendering operations implemented on an End Device prior to final output. Split rendering may be a necessity if the End Device is power constrained or limited in computational power. However, split rendering is not precluded from other End Devices that do not have such constraints. A discussion of relevant split rendering scenarios is provided in TR 26.865 [1], together with general design guidelines for immersive audio split rendering systems and specific design constraints and performance requirements for split rendering solutions for the 3GPP IVAS codec [1]. The latter are the basis for the split rendering feature of the IVAS codec. This TS presents ISAR split rendering solutions in a detailed algorithmic description, applicable even for other coding systems and renderers, whereby the split rendering solutions of the IVAS codec constitute a baseline set of the provided split rendering solutions.

# 1 Scope

The present document is a detailed algorithmic description of Split Rendering functions (ISAR) addressing Immersive Audio for Split Rendering Scenarios and that are applicable to a broad range of immersive audio coding systems and renderers. Functional solutions are described on an algorithmic level. Annexes of this document specify APIs, RTP payload format and SDP parameters as well as source code and test vectors.

# 2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non‑specific.

- For a specific reference, subsequent revisions do not apply.

- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TR 26.865: "Immersive Audio for Split Rendering Scenarios; Requirements ".

[3] 3GPP TS 26.250: "Codec for Immersive Voice and Audio Services (IVAS); General overview".

[4] 3GPP TS 26.253: "Codec for Immersive Voice and Audio Services (IVAS); Detailed Algorithmic Description incl. RTP payload format and SDP parameter definitions".

[5] 3GPP TS 26.258: "Codec for Immersive Voice and Audio Services (IVAS); C code (floating-point)".

[6] 3GPP TS 26.252: "Codec for Immersive Voice and Audio Services (IVAS); Test sequences".

# 3 Definitions of terms, symbols and abbreviations

## 3.1 Terms

Void

## 3.2 Symbols

Void

## 3.3 Abbreviations

For the purposes of the present document, the abbreviations given in 3GPP TR 21.905 [1] and the following apply. An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in 3GPP TR 21.905 [1].

IVAS Immersive Voice and Audio Services

ISAR Immersive Audio for Split Rendering

DoF/DOF Degree of Freedom

LCLD Low Complexity Low Delay

LC3plus Low Complexity Communication Codec Plus

CLDFB Complex Low-delay Filter Bank

# 4 General description of split renderer

## 4.1 Introduction

The main part of the present document is a detailed algorithmic description of functions for Split Rendering of immersive audio. It comprises

* Intermediate pre-rendered audio representation,
* Encoder, bitstream and decoder for the intermediate representation,
* Post-rendering of the decoded intermediate representation to provide binaural audio output with and without head-tracker input and post-rendering control metadata.

Along with the intermediate pre-rendered audio representation, functional requirements for pre-renderer operations are provided, which, if met, enable a Presentation Engine to connect to an ISAR compliant ISAR decoder and post-renderer. Interfaces are described allowing an immersive audio decoder/renderer in a Presentation Engine to connect to the ISAR pre-renderer.

The post-renderer procedures of this document are mandatory for implementation all User Equipment (UE)s claiming ISAR compliant post-rendering capabilities.

## 4.2 ISAR system overview

This clause provides a generic ISAR systems overview based on the example of the ISAR compliant IVAS split rendering feature, which define the baseline ISAR system illustrated in Figure 4.2-1.

The immersive audio rendering process is split between a capable device or network node (by the Presentation Engine relying on IVAS decoding and rendering) and a less capable device with limited computing and memory resources and motion-sensing for head-tracked binaural output. ISAR split rendering consists of the following core components:

* Pre-renderer & encoder with
	+ Pose correction metadata computation and metadata encoder
	+ Binaural audio encoder (transport codec encoder)
* ISAR decoder & post-renderer with
	+ Pose correction metadata decoder
	+ Binaural audio decoder (transport codec decoder)
	+ Pose corrective post renderer

The metadata-based pose correction scheme allows adjusting in a lightweight process a binaural audio signal originally rendered for a first pose according to a second pose. In split rendering context, the first pose is the potentially outdated lightweight-device pose available at the pre-renderer while the second pose is the current and accurate pose of the lightweight-device. The metadata is calculated at the capable device or network node based on additional binaural renditions at probing poses different from the first pose. For increasing degrees-of-freedom (DOF) an increasing number of additional binaural renditions at different probing poses is required. The metadata is transmitted to the lightweight device along with the coded binaural audio signal rendered for the first pose.

The pose correction metadata computation is done in CLDFB domain. Thus, unless the immersive audio decoder/renderer already operates in that domain, a conversion of the pre-rendered immersive audio signal and the additional binaural renditions to that domain is required. The binaural audio signal rendered to the first pose is encoded using one of the two codecs, LCLD or LC3plus, whereby the former of these codecs operates in CLDFB domain. The two codecs have complementary properties giving implementors the freedom to make individual trade-offs between complexity, memory, latency, and rate-distortion performance and to implement a design that is optimized for a given immersive audio service and hardware configuration. It is also possible to use another transport codec for the binaural audio signal.

ISAR split rendering can operate at various DOFs, ranging from 0-DOF (no pose correction) to 3-DOF (pose correction on the three rotational axes yaw, pitch, roll) at bit rates from 256 kbps (0-DOF) to 384 - 768 kbps (3-DOF).



Figure 4.2‑1: ISAR Split Rendering baseline system

# 5 ISAR baseline

## 5.1 Overview

ISAR baseline functionality is largely defined by the algorithmic description of the IVAS codec split rendering feature in TS 26.253 [4]. In the following, an overview of the functional components of the ISAR baseline is provided along the corresponding description or with a reference in what clause of TS 26.253 they are described.

## 5.2 Split pre-rendering

### 5.2.1 Functional components/topics specified TS 26.253

The following list displays where a given functional component or topic is described in TS 26.253 [4]. The righthand side specifies the respective clause in [4].

Overview 7.6.2.1

Complex Low-delay Filter Bank (CLDFB) analysis 6.2.5.1

### 5.2.2 Interfaces for split pre-rendering

Split pre-rendering (or ISAR pre-rendering) API is described in detailed in Annex A. The interface for split pre-renderer takes in the user head pose and generates N probing head poses, where N >= 0 and depends on Degree of Freedom (DoF) and rotation axis. Then, the renderer of the immersive audio decoder/renderer system (see Figure 4.2-1) is called N+1 times (one call per head pose) to generate N+1 binaural signals. The split pre-rendering interface takes in the N+1 binaural signals and generates split rendering bitstream. The interface provides the support to take the binaural signals in either time domain or CLDFB domain.

### 5.2.3 Interface requirements

Split rendering supports sampling frequency of 48 kHz only. The supported frame sizes and bitrates are described in IVAS specification TS 26.253 [4] clause 7.6.2.2 and clause 7.6.2.3.

## 5.3 Intermediate split renderer format

### 5.3.1 Functional components/topics specified TS 26.253

The following list displays where a given functional component or topic is described in TS 26.253 [4]. The righthand side specifies the respective clause in [4].

Supported Split Rendering bitrates 7.6.2.2/3

Intermediate split renderer metadata format 7.6.3

 Overview 7.6.3.1

 Metadata computation, quantization and coding 7.6.3.2

 Metadata computation for deviations about Yaw axis 7.6.3.2.1

 Quantization and coding of Yaw metadata 7.6.3.2.2

 Metadata computation for deviations about Pitch axis 7.6.3.2.3

 Quantization and coding of Pitch metadata 7.6.3.2.4

 Metadata computation for deviations about Roll axis 7.6.3.2.5

 Quantization and coding of Roll metadata 7.6.3.2.6

 Common split rendering metadata quantization and coding strategies 7.6.3.3

 Intermediate split renderer metadata decoder 7.6.3.4

 Intermediate split renderer metadata loss concealment 7.6.3.5

LCLD coded intermediate split renderer binaural audio format 7.6.4

 LCLD codec overview 7.6.4.1

 LCLD encoder 7.6.4.2

 Overview 7.6.4.2.1

 Perceptual Banding 7.6.4.2.2

 Joint Channel Coding 7.6.4.2.3

 Overview 7.6.4.2.3.1

 Bitstream Syntax 7.6.4.2.3.2

 Parameter Computation and Quantization 7.6.4.2.3.3

 Joint Coding Type Decision 7.6.4.2.3.4

 Entropy Coding of Parameters 7.6.4.2.3.5

 Temporal Grouping 7.6.4.2.4

 RMS Envelope 7.6.4.2.5

 RMS Envelope Calculation 7.6.4.2.5.1

 Normalizing the CLDFB Coefficients with the RMS Envelope 7.6.4.2.5.2

 RMS Envelope Transmission 7.6.4.2.5.3

 Perceptual Model 7.6.4.2.6

 Linear Prediction 7.6.4.2.7

 Overview 7.6.4.2.7.1

 Bitstream Syntax 7.6.4.2.7.2

 Prediction for 20ms frames 7.6.4.2.7.3

 Prediction for frames shorter than 20ms 7.6.4.2.7.4

 Prediction Signalling 7.6.4.2.7.5

 Quantization of Prediction Parameters 7.6.4.2.7.6

 Estimation of Prediction Parameters 7.6.4.2.7.7

 Bit Allocation 7.6.4.2.8

 Quantization of the Normalized CLDFB Coefficients 7.6.4.2.9

 Overview 7.6.4.2.9.1

 Differential Coding of the Normalized CLDFB Coefficients 7.6.4.2.9.2

 Quantization of Normalized CLDFB coefficients and Prediction Residuals 7.6.4.2.9.3

 Huffman Coding of Quantized Normalized CLDFB coefficients and

 Quantized Prediction Residuals 7.6.4.2.9.4

 LCLD decoder 7.6.4.3

 Overview 7.6.4.3.1

 Decoding Group Information 7.6.4.3.2

 Decoding RMS Envelope Information 7.6.4.3.3

 Perceptual Model 7.6.4.3.4

 Bit Allocation 7.6.4.3.5

 Normalized CLDFB Coefficient and Prediction Residual Huffman Decoding and

 Inverse Quantization 7.6.4.3.6

 Inverse Prediction 7.6.4.3.7

 Overview 7.6.4.3.7.1

 Status Tracking after Frame Loss 7.6.4.3.7.2

 Inverse RMS Envelope Normalization 7.6.4.3.8

 Inverse Joint Stereo Processing 7.6.4.3.9

 LCLD packet loss concealment 7.6.4.4

 General 7.6.4.4.1

 Synthesis model 7.6.4.4.2

 Analysis and parameter estimation 7.6.4.4.3

 Tonality determination 7.6.4.4.4

 Sinusoidal extension 7.6.4.4.5

 Predictive extension 7.6.4.4.6

 Cross-fade 7.6.4.4.7

 Burst-loss handling 7.6.4.4.8

LC3plus coded intermediate split renderer binaural audio format 7.6.5

 Introduction (Informative) 7.6.5.1

 Overview 7.6.5.2

 Encoder 7.6.5.3

 Decoder 7.6.5.4

 Frame Structure 7.6.5.5

 Packet Loss Concealment 7.6.5.6

 LC3 interoperable mode 7.6.5.7

Bit allocation for Split rendering 7.6.7

## 5.4 Split post-rendering

### 5.4.1 Functional components/topics specified TS 26.253

The following list displays where a given functional component or topic is described in TS 26.253 [4]. The righthand side specifies the respective clause in [4].

Overview 7.6.6.1

Post rendering with pose correction 7.6.6.2

 Metadata decoding 7.6.6.2.1

 Metadata interpolation or extrapolation 7.6.6.2.2

 Matrix mixing 7.6.6.2.3

Post rendering in 0 DOF mode 7.6.6.3

Complex Low-delay Filter Bank (CLDFB) synthesis 5.3.7

### 5.4.2 Interfaces for split post-rendering

Split post-rendering (or ISAR post-rendering) API is described in detailed in Annex A. The interface for split post-renderer takes in the ISAR bitstream and decodes it to generate reference binaural signal and pose correction metadata. The interface also takes in the latest user head pose and performs pose correction on reference binaural signal to generate output binaural signal in time domain.

### 5.4.3 Interface requirements

Split rendering supports sampling frequency of 48 KHz only. The post renderer supports output frame sizes of 5, 10 and 20ms. The output frame size of post renderer needs to be less than or equal to the frame size of pre-renderer.

Annex A (informative):
ISAR Application Programming Interfaces

## A.1 Overview

ISAR API is available in lib\_isar\lib\_isar\_pre\_rend.h and lib\_isar\lib\_isar\_post\_rend.h header files. These header files are present in source code as part of the IVAS codec floating-point code specification [5]. The tables below provide a detailed description of ISAR pre-renderer and post renderer API functions.

## A.2 ISAR pre-renderer API

Table 1 provides an overview of ISAR pre-renderer API.

Table 1: ISAR pre-renderer API functions

|  |  |  |  |
| --- | --- | --- | --- |
| Function names | Description of function | Function arguments | Description of arguments |
|  |  |  |  |
| ISAR\_PRE\_REND\_open() | Allocates and initializes the buffers/handles inside SPLIT\_REND\_WRAPPER | SPLIT\_REND\_WRAPPER \*hSplitBinRend | * Input/output parameter.
* The wrapper handle hSplitBinRend should be allocated prior to calling this function.
* hSplitBinRend->multiBinPoseData should be set as per ISAR\_PRE\_REND\_GetMultiBinPoseData() prior to calling this function
* ISAR pre-renderer handle described in detail in Table 3
 |
| ISAR\_SPLIT\_REND\_CONFIG\_DATA \*pSplitRendConfig | * Input/output parameter
* The handle pSplitRendConfig should be allocated prior to calling this function.
* ISAR pre-renderer config handle described in detail in Table 2
 |
| const int32\_t output\_Fs | Input parameter.Output sampling rate |
| const int16\_t cldfb\_in\_flag | Input parameter.1: if multi-binaural signal input is in CLDFB domain only0: Otherwise  |
| const int16\_t pcm\_out\_flag | Input parameter.1: if output binaural signal is in PCM format (BINAURAL\_SPLIT\_PCM config) 0: Otherwise |
| const int16\_t num\_subframes | Input parameter.Number of 5ms subframes to process.Supported values:1/2/4: For 0-DOF cases4: For non-zero DOF cases |
| const int16\_t mixed\_td\_cldfb\_flag | Input parameter.1: if multi-binaural signal input is in both CLDFB and time domain0: Otherwise |
|  |  |  |  |
|  |  |  |  |
| ISAR\_PRE\_REND\_GetMultiBinPoseData() | * Get the Pose correction metadata configuration
* Should be called during runtime if rot\_axis changes
 | const ISAR\_SPLIT\_REND\_CONFIG\_DATA \*pSplit\_rend\_config | Input parameter.ISAR pre-renderer config handle described in detail in Table 2 |
| MULTI\_BIN\_REND\_POSE\_DATA \*pMultiBinPoseData | Output parameter.ISAR pre-renderer pose config data handle described in detail in Table 4 |
| const ISAR\_SPLIT\_REND\_ROT\_AXIS rot\_axis | Input parameter.ISAR rotation axis described in detail in Table 4 |
|  |  |  |  |
| ISAR\_PRE\_REND\_MultiBinToSplitBinaural() | Process call to generate ISAR pre-renderer output on frame-by frame basis from the multi-binaural signal | SPLIT\_REND\_WRAPPER \*hSplitBin | * Input/output parameter
* ISAR pre-renderer handle
 |
| const IVAS\_QUATERNION headPosition | * Input parameter
* Quaternion for head rotation
 |
| const int32\_t SplitRendBitRate | * Input parameter
* Split rendering bitrate
 |
| ISAR\_SPLIT\_REND\_CODEC splitCodec | * Input parameter
* ISAR transport codec described in detail in Table 2
* Should be set as per ISAR\_SPLIT\_REND\_CONFIG\_DATA output of ISAR\_PRE\_REND\_open()
 |
| int16\_t codec\_frame\_size\_ms | * Input parameter
* Codec frame size if ISAR transport codec
* Should be set as per ISAR\_SPLIT\_REND\_CONFIG\_DATA output of ISAR\_PRE\_REND\_open()
 |
| ISAR\_SPLIT\_REND\_BITS\_HANDLE pBits | * Handle pBits should be allocated and initialized prior to calling function
* Output parameter
* ISAR bitstream handle described in detail in Table 5
 |
| float Cldfb\_In\_BinReal[][][] | * Input parameter
* CLDFB real input buffer
 |
| float Cldfb\_In\_BinImag[][][] | * Input parameter
* CLDFB imaginary input buffer
 |
| const int16\_t max\_bands | * Input parameter
* Number of bands in CLDFB input
 |
| float \*output[] | * Input/Output parameter
* PCM input/output buffer
 |
| const int16\_t low\_res\_pre\_rend\_rot | Reserved flag. Should always be set to 1 |
| const int16\_t cldfb\_in\_flag | Input parameter.1: if multi-binaural signal input is in CLDFB domain only0: Otherwise  |
| const int16\_t pcm\_out\_flag | Input parameter.1: if output binaural signal is in PCM format (BINAURAL\_SPLIT\_PCM config) 0: Otherwise |
| const int16\_t ro\_md\_flag | Input parameter.1: if the HRTFs used in the generation on multi-binaural signal can lead to ITD difference above 1.6 kHz0: Otherwise |
|  |  |
| ISAR\_PRE\_REND\_close() | Deallocates the ISAR handles/memory | SPLIT\_REND\_WRAPPER \*hSplitBinRend | * Input/Output parameter
* ISAR pre-renderer handle
 |
|  | IVAS\_REND\_AudioBuffer \*pSplitRendEncBuffer | reserved buffer field. Can be set to NULL |
|  |  |  |  |

Table 2: ISAR\_SPLIT\_REND\_CONFIG\_DATA description

|  |  |
| --- | --- |
| ISAR\_SPLIT\_REND\_CONFIG\_DATA fields | description |
| int32\_t splitRendBitRate | Split rendering bitrate |
| int16\_t hq\_mode | * High quality mode for 3 DOF
* Supported values: 0 or 1

NOTE: Adds complexity at pre-renderer |
| int16\_t dof | * Degree of Freedom
* Supported values: 0 to 3
 |
| int16\_t codec\_delay\_ms | look ahead delay of the external transport codec that is used to code BIN signal output of pre-renderer in BINAURAL\_SPLIT\_PCM mode |
| int16\_t codec\_frame\_size\_ms | Frame size of the ISAR pre-renderer transport codec |
| ISAR\_SPLIT\_REND\_POSE\_CORRECTION\_MODE poseCorrectionMode; | ISAR\_SPLIT\_REND\_POSE\_CORRECTION\_MODE\_NONE: No pose correction or 0-DOFISAR\_SPLIT\_REND\_POSE\_CORRECTION\_MODE\_CLDFB: Pose correction OR non-zero DOF |
| ISAR\_SPLIT\_REND\_CODEC codec | * ISAR transport codec
* Supported values: 0 to 3
* 0: LCLD
* 1: LC3plus
* 2: Default (internally set to LC3plus for time domain input, LCLD otherwise)
* 3: NONE (BINAURAL SPLIT PCM mode)
 |
| ISAR\_SPLIT\_REND\_RENDERER\_SELECTION rendererSelection; | Debug field. Default value: ISAR\_SPLIT\_REND\_RENDERER\_SELECTION\_DEFAULT |

Table 3: SPLIT\_REND\_WRAPPER description

|  |  |
| --- | --- |
| SPLIT\_REND\_WRAPPER fields | description |
| MULTI\_BIN\_REND\_POSE\_DATA multiBinPoseData | ISAR pre-renderer pose config data handle described in detail in Table 4 |
| ISAR\_BIN\_HR\_SPLIT\_PRE\_REND\_HANDLE hBinHrSplitPreRend | ISAR pre-renderer pose correction metadata handle |
| ISAR\_BIN\_HR\_SPLIT\_LCLD\_ENC\_HANDLE hSplitBinLCLDEnc | LCLD encoder handle |
| CLDFB\_HANDLES\_WRAPPER\_HANDLE hCldfbHandles | CLDFB handles |
| ISAR\_LC3PLUS\_ENC\_HANDLE hLc3plusEnc | LC3plus encoder handle |
| float \*lc3plusDelayBuffers | LC3plus delay buffer |
| int32\_t lc3plusDelaySamples | LC3plus delay in samples |

Table 4: MULTI\_BIN\_REND\_POSE\_DATA description

|  |  |
| --- | --- |
| MULTI\_BIN\_REND\_POSE\_DATA fields | **description** |
| int16\_t num\_poses; | Number of poses (including reference pose) for Metadata computation OR number of binaural signals in multi-binaural input to ISAR pre-renderer |
| float relative\_head\_poses[MAX\_HEAD\_ROT\_POSES][3]; | * Head poses (relative to reference head pose) along yaw, pitch and roll axis
* Binaural signals corresponding probing poses (poses other than reference head pose) in the multi-binaural input signal are computed based on this field.
 |
| int16\_t dof; | Same as described in ISAR\_SPLIT\_REND\_CONFIG\_DATA  |
| int16\_t hq\_mode; | Same as described in ISAR\_SPLIT\_REND\_CONFIG\_DATA |
| ISAR\_SPLIT\_REND\_ROT\_AXIS rot\_axis; | * Rotation axis along which pose correction is needed
* Supported values for 1 DOF (0, 1, 2, 3). 0: Default or YAW, 1: YAW, 2: Pitch, 3: Roll
* Supported values for 2 DOF (0, 4, 5, 6). 0: Default or YAW+PITCH, 4: YAW+PITCH, 5: YAW+ROLL, 6: PITCH+ROLL
* Supported values for 3 DOF (0 or Default)
 |
| ISAR\_SPLIT\_REND\_POSE\_CORRECTION\_MODE poseCorrectionMode | Same as described in ISAR\_SPLIT\_REND\_CONFIG\_DATA |

Table 5 ISAR\_SPLIT\_REND\_BITS\_HANDLE description

|  |  |
| --- | --- |
| ISAR\_SPLIT\_REND\_BITS\_HANDLE fields | Description |
|  |  |
| uint8\_t \*bits\_buf | Bitstream buffer (to be allocated by caller of ISAR pre-renderer) |
| int32\_t buf\_len | Length of bits\_buf buffer in bytes |
| int32\_t bits\_written | Number of bits written to bits\_buf buffer |
| int32\_t bits\_read | Number of bits read from bits\_buf buffer |
| int16\_t codec\_frame\_size\_ms | Transport Codec frame size as described in Table 2 |
| ISAR\_SPLIT\_REND\_CODEC codec | Transport Codec as described in Table 2 |
| ISAR\_SPLIT\_REND\_POSE\_CORRECTION\_MODE pose\_correction | Pose correction mode as described in Table 2 |
|  |  |

## A.3 ISAR post-renderer API

Table 6 provides an overview of ISAR pre-renderer API.

Table 6: ISAR post-renderer API functions

|  |  |  |  |
| --- | --- | --- | --- |
| Function names | Description of functions | Function arguments | Description of parameters |
|  |  |  |  |
| ISAR\_POST\_REND\_open() | Allocates and initializes ISAR handle phIsarRend | ISAR\_POST\_REND\_HANDLE \*phIsarRend | ISAR handle |
| const int32\_t outputSampleRate | Output sampling rate |
| const IVAS\_AUDIO\_CONFIG outConfig | Output configuration (Only BINAURAL output config supported) |
| const bool asHrtfBinary | Reserved/Unused parameter |
| const int16\_t nonDiegeticPan | Reserved/Unused parameter |
| const float nonDiegeticPanGain | Reserved/Unused parameter |
| const int16\_t num\_subframes | Number of 5ms subframes in the output. This specifies output frame size. Supported framesizes are 5, 10 and 20ms |
|  |  |  |  |
| ISAR\_POST\_REND\_InitConfig() | Initializes hIvasRend->splitRenderConfig | ISAR\_POST\_REND\_HANDLE hIsarRend | ISAR handle |
| const IVAS\_AUDIO\_CONFIG outAudioConfig | Output configuration (Only BINAURAL output config supported) |
|  |  |  |  |
| ISAR\_REND\_SetSplitRendBitstreamHeader() | Sets one time configuration. To be set during SDP negotiation | ISAR\_POST\_REND\_HANDLE hIsarRend | ISAR handle |
| const ISAR\_SPLIT\_REND\_CODEC codec | ISAR transport codec |
| const ISAR\_SPLIT\_REND\_POSE\_CORRECTION\_MODE poseCorrection | ISAR pose correction method |
| const int16\_t codec\_frame\_size\_ms | ISAR transport codec frame size |
|  |  |  |  |
|  |  |  |  |
| ISAR\_POST\_REND\_NumOutChannels() | Get number of output channels | ISAR\_POST\_REND\_HANDLE hIsarRend | ISAR handle |
| int16\_t \*numOutChannels | Number of output channels |
|  |  |  |  |
| ISAR\_POST\_REND\_AddInput () | Add input stream for processing | ISAR\_POST\_REND\_HANDLE hIsarRend | ISAR handle |
| const IVAS\_AUDIO\_CONFIG inConfig | Input configuration (Only BINAURAL\_SPLIT\_CODED and BINAURAL\_SPLIT\_PCM input configs supported) |
| ISAR\_POST\_REND\_InputId \*inputId | Input ID |
|  |  |  |  |
| ISAR\_POST\_REND\_GetInputNumChannels() | Get number of input channels | ISAR\_POST\_REND\_CONST\_HANDLE hIsarRend | ISAR handle |
| const ISAR\_POST\_REND\_InputId inputId | Input ID |
| int16\_t \*numChannels | number of input channels |
|  |  |  |  |
| ISAR\_POST\_REND\_GetDelay() | Get ISAR delay (this includes both pre-renderer and post renderer delay) | ISAR\_POST\_REND\_CONST\_HANDLE hIsarRend | ISAR handle |
| int16\_t \*nSamples | Delay in number of samples |
| int32\_t \*timeScale | Time scale of the delay, equal to renderer output sampling rate |
|  |  |  |  |
| ISAR\_POST\_REND\_FeedInputAudio() | Feed the audio PCM part of the input. Used in BINAURAL\_SPLIT\_PCM mode | ISAR\_POST\_REND\_HANDLE hIsarRend | ISAR handle |
| const ISAR\_POST\_REND\_InputId inputId | Input ID |
| const ISAR\_POST\_REND\_ReadOnlyAudioBuffer inputAudio | Input audio buffer |
|  |  |  |  |
| ISAR\_POST\_REND\_FeedSplitBinauralBitstream() | Feed ISAR bitstream | ISAR\_POST\_REND\_HANDLE hIsarRend | ISAR handle |
| const ISAR\_POST\_REND\_InputId inputId | Input ID |
| ISAR\_POST\_REND\_BitstreamBuffer \*hBits | Input bitstream buffer |
|  |  |
|  |  |
| ISAR\_POST\_REND\_GetSplitBinauralSamples() | Get ISAR output (BINAURAL pcm samples) | ISAR\_POST\_REND\_HANDLE hIsarRend | ISAR handle |
| IVAS\_REND\_AudioBuffer outAudio | Output audio buffer |
| bool\* needNewFrame | Flag to indicate the caller if a new bitstream frame is to be fed |
|  |  |  |  |
| ISAR\_POST\_REND\_SetHeadRotation() | Set head pose for each 5ms subframe | ISAR\_POST\_REND\_HANDLE hIsarRend | ISAR handle |
| const IVAS\_QUATERNION headRot | Head pose in quaternions |
| const IVAS\_VECTOR3 Pos | Position vector(reserved/unused field) |
| const ISAR\_SPLIT\_REND\_ROT\_AXIS rot\_axis | Rotation axis (reserved/unused field) |
| const int16\_t sf\_idx | Subframe index |
|  |  |  |  |
| ISAR\_POST\_REND\_SetSplitRendBFI() | Set Bad frame index (BFI) flag if the frame is lost | ISAR\_POST\_REND\_HANDLE hIsarRend | ISAR handle |
| const int16\_t bfi | BFI flag |
|  |  |  |  |
| ISAR\_POST\_REND\_Close() | Free ISAR memory and deallocate ISAR handle hIsarRend | ISAR\_POST\_REND\_HANDLE\* hIsarRend | ISAR handle |
|  |  |  |
|  |  |  |  |

Annex B (normative):
RTP Payload Format and SDP Parameters

The definition of the RTP payload format and SDP parameters for the ISAR feature of IVAS as part of the Annex A of TS 26.253 [4] is FFS.

A system agnostic RTP payload format and SDP parameters for the transport of the ISAR intermediate format and associated metadata is FFS.

Annex C (normative):
ISAR Reference Source Code

ISAR reference source code is available both as fixed-point code and as floating-point code.

The intermediate split renderer format encoder in floating-point code is defined in source code as part of the IVAS codec floating-point code specification [5] and can be built to an IVAS independent object library which functions are accessable by a pre-renderer through the corresponding APIs (see Annex A).

The post renderer including intermediate split renderer format decoder in floating-point code is defined as part of the IVAS codec floating-point code specification [5]. The post renderer reference source code can be built to an IVAS independent stand-alone executable.

While IVAS fixed-point code is not yet specified, ISAR fixed-point code is provided as a software patch to the IVAS floating-point code [5]; the software patch is provided as electronic attachment to this specification. Applying the patch allows running the ISAR split rendering feature in fixed-point code within the IVAS floating-point software framework.

Note: For future 3GPP releases that are expected to provide a IVAS fixed-point code specification, the ISAR fixed point-code will be provide as part of the IVAS fixed-point code specification. In that case, the electronic attachment of this specifications with the patch to the IVAS floating-point code will become obsolete and shall not be used.

Annex D (normative):
Test Vectors

Test vectors for the ISAR reference source code are defined as part of the IVAS test vector specification [6].

Annex E (informative):
Change history

|  |
| --- |
| **Change history** |
| **Date** | **Meeting** | **TDoc** | **CR** | **Rev** | **Cat** | **Subject/Comment** | **New version** |
| 2024-04 | SA4#127-bis-e | S4-240716 | - | - | - | Presented to 3GPP SA4 Audio SWG | 0.0.1 |
| 2024-04 | SA4#127-bis-e | S4-240809 | - | - | - | Presented to 3GPP SA4 plenary | 0.1.0 |
| 2024-05 | SA4#128 | S4-241051 |  |  |  | Rapporteur’s updates:Clause 5, Annexes A-D | 0.1.1 |
| 2024-05 | SA4#128 | S4-241183 |  |  |  | Rapporteur’s updates agreed by audio SWG | 0.2.0 |