ETSI TS 126 249 V18.0.0 (2024-07)

Immersive Audio for Split Rendering Scenarios (3GPP TS 26.249 version 18.0.0 Release 18)

3GPP
A GLOBAL INITIATIVE

Reference
RTS/TSGS-0426249vi00

Keywords

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™** and **LTE™** are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Legal Notice

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

The present document may refer to technical specifications or reports using their 3GPP identities. These shall be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between 3GPP and ETSI identities can be found under https://webapp.etsi.org/key/queryform.asp.

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Contents

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].

CLDFB Complex Low-delay Filter Bank

DoF/DOF Degree of Freedom

ISAR Immersive Audio for Split Rendering

IVAS	Immersive Voice and Audio Services
LCLD	Low Complexity Low Delay
LC3plus	Low Complexity Communication Codec Plus
MD	Medadata

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).

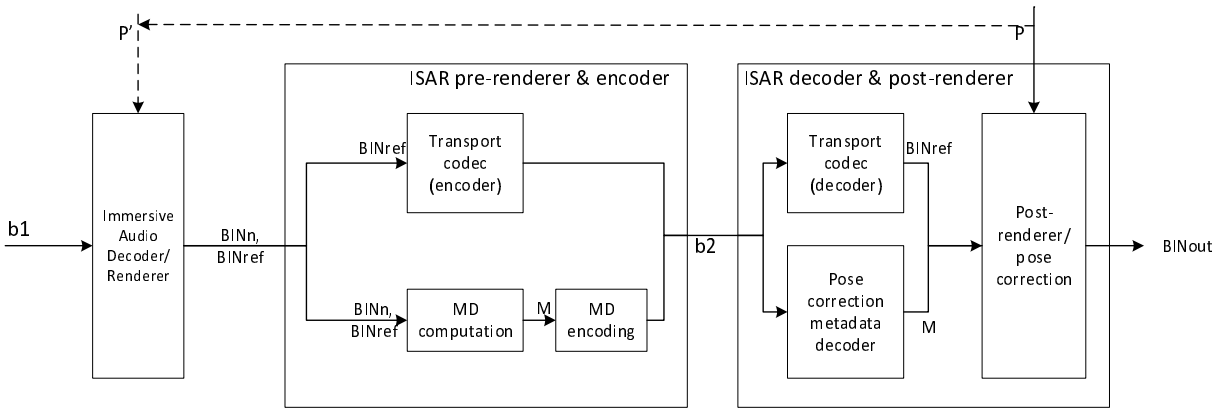

Figure 4.2-1: ISAR Split Rendering baseline system

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].

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].

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

The overview of ISAR pre-renderer API is provided in Table A.2-1.

Table A.2-1: 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 A.2-3
		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 A.2-2
		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 only 0: 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 cases 4: 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 domain 0: Otherwise
`ISAR_PRE_REND_GetMultiBinPoseData()`	Get the Pose correction metadata configuration. It 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 A.2-2
		`MULTI_BIN_REND_POSE_DATA *pMultiBinPoseData`	Output parameter. ISAR pre-renderer pose config data handle described in detail in Table A.2-4
		`const ISAR_SPLIT_REND_ROT_AXIS rot_axis`	Input parameter. ISAR rotation axis described in detail in Table A.2-4
`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 A-2-2. 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 A.2-5
		`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 only 0: 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 kHz 0: 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 A.2-2: 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-DOF` `ISAR_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 A.2-3: 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 A.2-4
`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 A.2-4: 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 A.2-5 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 A.2-2
ISAR_SPLIT_REND_CODEC codec	Transport Codec as described in Table A.2-2
ISAR_SPLIT_REND_POSE_CORRECTION_MODE pose_correction	Pose correction mode as described in Table A.2-2

A.3 ISAR post-renderer API

The overview of ISAR post-renderer API is provided in Table A.3-1.

Table A.3-1: 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 provided 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				Update to Annex C: inclusion of ISAR fixed-point code to this TS Removal of references to potential extensions over baseline Editorial updates	0.2.0
2024-05	SA4#128	S4-241345				Editorial updates	0.3.0
2024-06						Version 1.0.0 created by MCC	1.0.0
2024-06						Version 18.0.0 created by MCC upon TSG approval	18.0.0

History

Document history		
V18.0.0	July 2024	Publication