**Source: Editor1**

**Title:** **IVAS-7a: Processing plan for selection phase**

**Version: v.0.10.0**

**Agenda Item: 14.2**

Document History:

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Detail of changes** |
| 0.1.0 | 25 Oct 2019 | Initial version |
| 0.1.1 | 22 Jan 2020 | Implementation of S4-200119 according to agreements in EVS SWG |
| 0.2.0 | 5 Feb 2021 | Changed scope to selection phase |
| 0.3.0 | 23 Feb 2022 | Implementation of S4-220164, S4-220165, S4-220166 with agreed amendments of the EVS SWG at SA4#117-e. |
| 0.4.0 | 13 Apr 2022 | Implementation of S4-220432 |
| 0.5.0 | 13 Apr 2022 | Agreement of clause 3.5-3.7 |
| 0.6.0 | 16 May 2022 | Implementation of S4-220733 |
| 0.7.0 | 17 Nov 2022 | Implementation of S4-221464 |
| 0.8.0 | 23 Feb 2023 | Agreements based on S4-230221, S4-230239, S4-230261, S4-230262 implemented at SA4#122. |
| 0.8.1 | 3 Apr 2023 | Agreements based on S4aA220031. |
| 0.9.0 | 20 Apr 2023 | Implementation of S4-230579. |
| 0.10.0 | 26 May 2023 | Implementation of agreed parts of S4-230919, S4-230936, and further edits at SA4#124. |

**CONTENTS**

[3.1 Acronyms 3](#_Toc135872429)

[3.2 Definitions 4](#_Toc135872430)

[3.3 Experiments 4](#_Toc135872431)

[3.4 Listening lab designators 5](#_Toc135872432)

[3.5 Bitstream format 5](#_Toc135872433)

[3.6 Audio format 5](#_Toc135872434)

[3.7 Audio track designators 5](#_Toc135872435)

[3.8 Audio track configurations 6](#_Toc135872436)

[3.9 Preamble definition 8](#_Toc135872437)

[3.10 File naming 8](#_Toc135872441)

[3.10.1 Generic definitions 8](#_Toc135872442)

[3.10.2 Naming examples of the input files and processed output files 8](#_Toc135872443)

[3.10.3 File naming error patterns 10](#_Toc135872445)

[3.10.4 File naming jitter profiles 10](#_Toc135872446)

[3.10.5 File naming jitter derived error patterns 10](#_Toc135872447)

[4.1 General considerations for processing 10](#_Toc135872451)

[4.1.1 Source material requirements 10](#_Toc135872452)

[4.1.2 Concatenated sequences processing 10](#_Toc135872453)

[4.1.3 Frame error application 10](#_Toc135872457)

[4.1.4 Concatenation setup 10](#_Toc135872459)

[4.2 Generation of test samples 11](#_Toc135872460)

[4.2.1 P.SUPPL800 speech samples 11](#_Toc135872461)

[4.2.2 General processing for background noise 13](#_Toc135872462)

[4.3 Pre-processing 13](#_Toc135872464)

[4.3.1 Pre-processing stages 13](#_Toc135872465)

[4.3.2 Pre-processing for experiments p03, p06, p07 and m1a-m7b 13](#_Toc135872466)

[4.3.3 Pre-processing for experiments p01-p02, p04-p05, p08-p09 14](#_Toc135872467)

[4.4 Processing for anchor conditions 14](#_Toc135872468)

[4.4.1 MNRU/ESDRU 14](#_Toc135872469)

[4.4.2 7 kHz low-pass anchor 14](#_Toc135872470)

[4.5 Processing for stereo inputs 14](#_Toc135872473)

[4.5.1 Reference conditions 15](#_Toc135872474)

[4.5.2 CuT conditions 15](#_Toc135872475)

[4.5.3 Reference codec conditions 15](#_Toc135872478)

[4.6 Processing for multi-channel audio inputs 15](#_Toc135872479)

[4.6.1 Reference conditions 15](#_Toc135872482)

[4.6.2 CuT conditions 16](#_Toc135872483)

[4.6.3 Reference codec conditions 16](#_Toc135872484)

[4.7 Processing for scene-based audio inputs 16](#_Toc135872485)

[4.7.1 Reference conditions 16](#_Toc135872486)

[4.7.2 CuT conditions 17](#_Toc135872487)

[4.7.3 Reference codec conditions 17](#_Toc135872491)

[4.8 Processing for metadata-assisted spatial audio (MASA) inputs 17](#_Toc135872492)

[4.8.1 Reference conditions 17](#_Toc135872493)

[4.8.2 CuT conditions 18](#_Toc135872494)

[4.8.3 Reference codec conditions 18](#_Toc135872495)

[4.9 Processing for object-based audio inputs 20](#_Toc135872496)

[4.9.1 Reference conditions 20](#_Toc135872497)

[4.9.2 CuT conditions 20](#_Toc135872501)

[4.9.3 Reference codec conditions 21](#_Toc135872502)

[4.10 Post-processing 22](#_Toc135872503)

[4.10.1 Segmentation and windowing of processed conditions 22](#_Toc135872505)

[4.10.2 Level normalization of processed conditions 22](#_Toc135872506)

[5.1 Pre- and post-processing operations 22](#_Toc135872507)

[5.1.1 General delay compensation for the STL filter tool 22](#_Toc135872509)

[5.1.2 Filtering operations 22](#_Toc135872510)

[5.1.3 Level adjustment 22](#_Toc135872511)

[5.1.4 Summation of audio signals 23](#_Toc135872532)

[5.1.5 Concatenation 23](#_Toc135872533)

[5.1.6 Sampling rate changes (resampling) 23](#_Toc135872546)

[5.1.7 Windowing and segmentation 24](#_Toc135872556)

[5.1.8 Truncation of HOA3 to HOA2 / FOA / planar FOA 24](#_Toc135872574)

[5.1.9 Rendering to Binaural (HRIRs) 24](#_Toc135872575)

[5.1.10 Rendering to Binaural Room (BRIRs) 25](#_Toc135872576)

[5.1.11 Rendering to Loudspeaker 26](#_Toc135872577)

[5.1.12 MASA parameters analysis and downmix 27](#_Toc135872578)

[5.2 Condition processing 27](#_Toc135872579)

[5.2.1 MNRU conditions 27](#_Toc135872581)

[5.2.2 ESDRU conditions 27](#_Toc135872582)

[5.2.3 LP7 conditions 27](#_Toc135872584)

[5.2.4 EVS operation 27](#_Toc135872585)

[5.2.5 CuT operation (Encoding) 28](#_Toc135872586)

[5.2.6 CuT operation (Decoding, non-JBM case) 29](#_Toc135872587)

[5.2.7 CuT operation (Decoding, JBM case) 29](#_Toc135872588)

[5.3 Encoder and decoder CuT executable requirements 30](#_Toc135872589)

[5.4 Error Insertion (EID) 30](#_Toc135872590)

[Frame error tool 31](#_Toc135872591)

[5.4.1 31](#_Toc135872606)

[5.4.2 Pattern generation 31](#_Toc135872607)

[5.4.3 Derive EPFs from jitter profiles 31](#_Toc135872608)

[5.4.4 Network simulator for packet jitter generation 31](#_Toc135872609)

[5.4.5 Cutting Tool for JBM tests 32](#_Toc135872610)

[5.5 Automatic complexity and memory instrumentation (WMC tool) 32](#_Toc135872611)

[5.5.1 Preparation of the source code 32](#_Toc135872612)

[5.5.2 Instrumentation with the WMC tool 32](#_Toc135872613)

[5.5.3 Running the instrumented code 32](#_Toc135872614)

[A.1 Delay compensation for filter operations and reference codecs 34](#_Toc135872615)

[A.2 Tools used 34](#_Toc135872656)

[A.2.1 ITU-T STL processing tools 34](#_Toc135872658)

[A.2.2 Reference codecs 35](#_Toc135872659)

[A.2.3 Other tools 35](#_Toc135872660)

[B.1 Delay and error profile offset 37](#_Toc135872661)

[B.2 Seeds for rate switching 37](#_Toc135872662)

[B.3 Initialization of gen-patt tool 37](#_Toc135872663)

# Introduction

This document describes the processing for the selection tests of the 3GPP IVAS codec. It should be read in conjunction with the associated IVAS selection phase test plan, IVAS-8a [1]. The corresponding processing scripts, which shall be used for the selection test processing, are available at 3GPP Forge [11]. The latest available version of the scripts shall be used. Although the descriptions and the processing scripts should match, in case of any discrepancy, the processing scripts prevail over the descriptions in this document. Unless otherwise specified, processing steps are performed using built-in Python functions or as available in common Python libraries.

# Responsibilities

The host lab (HL) is responsible for the processing of the experiments while the cross-check lab (CL) verifies the correctness of the processing. The various roles are further specified in IVAS-8a.

The listening tests will be conducted by the listening labs (LL) in accordance with IVAS-8a.

# Definitions and formats

## Acronyms

CL Cross-check Laboratory

CuT Codec under Test

DCR Degradation Category Rating

DTX Discontinuous transmission

ESDRU Energy-based Spatial Distortion Reference Unit

EVS Enhanced Voice Services

FB Fullband

FE Frame Erasure

FOA First-Order Ambisonics

GAL Global Analysis Laboratory

HL Host Laboratory

HOA3 Higher-Order Ambisonics, 3rd order

IVAS Immersive Voice and Audio Services

LKFS Loudness, K-weighted, relative to Full Scale

LL Listening Laboratory

MASA Metadata-Assisted Spatial Audio

MNRU Modulated Noise Reference Unit

MUSHRA Multi Stimulus test with Hidden Reference and Anchor

NB Narrowband

SWB Super-wideband

WB Wideband

## Definitions

Anchor condition Controlled impairment, e.g., using MNRU, ESDRU, low-pass filtered signal

CuT condition Codec under test processed for testing

Reference condition Quality target for comparative tests, sometimes also called Direct condition

Reference codec condition Quality comparison to reference codec, e.g. EVS, for evaluation of performance requirements

## Experiments

The selection test plan‎ (IVAS-8a) [1] describes the listening experiments. For the processing of each experiment, a three-letter experiment designator shall be used for the file naming convention. An overview of all experiments is presented in Table 1.

Table 1: Overview of experiments

| Experiment Designator | Experiment Name (IVAS-8a) | Input format | Source Material |
| --- | --- | --- | --- |
| p01 | P800-1 | Stereo | Clean speech |
| p02 | P800-2 | Stereo | Speech+Background |
| p03 | P800-3 | Stereo | Mixed & Music |
| p04 | P800-4 | FOA | Clean speech |
| p05 | P800-5 | FOA | Speech+Background |
| p06 | P800-6 | 1 Object | Clean speech |
| p07 | P800-7 | 2 Objects | Clean speech |
| p08 | P800-8 | MASA | Clean speech |
| p09 | P800-9 | MASA | Speech+Background |
| m1a | BS1534-1a | Stereo | Generic Audio |
| m1b | BS1534-1b | Stereo | Generic Audio |
| m2a | BS1534-2a | 5.1 | Generic Audio |
| m2b | BS1534-2b | 5.1 | Generic Audio |
| m3a | BS1534-3a | 7.1.4 | Generic Audio |
| m3b | BS1534-3b | 7.1.4 | Generic Audio |
| m4a | BS1534-4a | FOA | Generic Audio |
| m4b | BS1534-4b | HOA2 | Generic Audio |
| m5a | BS1534-5a | HOA3 | Generic Audio |
| m5b | BS1534-5b | HOA3 | Generic Audio |
| m6a | BS1534-6a | Objects | Generic Audio |
| m6b | BS1534-6b | Objects | Generic Audio |
| m7a | BS1534-7a | MASA | Generic Audio |
| m7b | BS1534-7b | MASA | Generic Audio |

## Listening lab designators

The following table lists the responsible listening labs (LL) with their dedicated LL designator to be used for the file naming convention.

Table 2: List of listening lab designators

|  |  |
| --- | --- |
| **LL designator** | **Listening Lab Company** |
| **a** | Force Technology |
| **b** | Head Acoustics/ IKS |
| **c** | MQ University |
| **d** | Mesaqin.com |

## Bitstream format

The candidate codec shall be based on ITU-T G.192 [2] format as a common bit-stream interface.

## Audio format

The format of input/output audio files shall be WAVE, 16-bit little endian format. For multi-track audio, the audio tracks shall be ordered according to Table 5.

## Audio track designators

Audio tracks are designated according to Table 3 with further specification according to Table 4.

Table 3: Track designators

| Track designator | Definition | |
| --- | --- | --- |
|  | Index: 1, 2, … |
| M | Mono channel | |
| L | Left channel | |
| R | Right channel | |
| CH\_A\_E | Channel at nominal azimuth angle  and nominal elevation angle | |
| LFE | LFE channel | |

Table 4: Channel specification

| Specification variable | Definition |
| --- | --- |
|  | Nominal azimuth angle in  range: degrees  Represented by three-digit number always including a sign (padded from the left with zeros if necessary).  Positive values indicate positions left of the frontal direction. The sign assigned to azimuth value zero shall be +. |
|  | Nominal elevation angle in  range: degrees  Represented by two-digit number always including a sign (padded from the left with zeros if necessary).  Positive values indicate positions above the horizontal plane. The sign assigned to elevation value zero  shall be +. |

## Audio track configurations

Input/output audio shall follow configurations as specified in Table 5. Ambisonics components follow the ACN ordering where for real-valued spherical harmonics components of order and degree , where .

Table 5: Audio track configurations

| Audio format (designator) | Number of tracks | Index | Configuration  (incl. ordering) | Azimuth Range | Elevation Range |
| --- | --- | --- | --- | --- | --- |
| Mono (M) | 1 | 1 | M | - | - |
| Stereo (ST) | 2 | 1,2 | L, R | - | - |
| Binaural (BIN) | 2 | 1,2 | L, R | - | - |
| Multi-channel 5.1 (MC51) | 6 | 1 | CH\_A+030\_E+00 | +30 | 0 |
| 2 | CH\_A-030\_E+00 | -30 | 0 |
| 3 | CH\_A+000\_E+00 | 0 | 0 |
| 4 | LFE | - | - |
| 5 | CH\_A+110\_E+00 | +100 … +120 | 0 … +15 |
| 6 | CH\_A-110\_E+00 | -100 … -120 | 0 ... +15 |
| Multi-channel 7.1 (MC71) | 8 | 1 | CH\_A+030\_E+00 | +30 ... +45 | 0 |
| 2 | CH\_A-030\_E+00 | -30 … -45 | 0 |
| 3 | CH\_A+000\_E+00 | 0 | 0 |
| 4 | LFE | - | - |
| 5 | CH\_A+110\_E+00 | +85 … +110 | 0 |
| 6 | CH\_A-110\_E+00 | -85 … -110 | 0 |
| 7 | CH\_A+135\_E+00 | +120 … +150 | 0 |
| 8 | CH\_A-135\_E+00 | -120 … -150 | 0 |
| Multi-channel 5.1+4 (MC514) | 10 | 1 | CH\_A+030\_E+00 | +30 | 0 |
| 2 | CH\_A-030\_E+00 | -30 | 0 |
| 3 | CH\_A+000\_E+00 | 0 | 0 |
| 4 | LFE | - | - |
| 5 | CH\_A+110\_E+00 | +100 … +120 | 0 … +15 |
| 6 | CH\_A-110\_E+00 | -100 … -120 | 0 … +15 |
| 7 | CH\_A+030\_E+35 | +30 … +45 | +30 … +55 |
| 8 | CH\_A-030\_E+35 | -30 … -45 | +30 … +55 |
| 9 | CH\_A+110\_E+35 | +100 … +135 | +30 … +55 |
| 10 | CH\_A-110\_E+35 | -100 … -135 | +30 … +55 |
| Multi-channel 7.1+4 (MC714) | 12 | 1 | CH\_A+030\_E+00 | +30 … +45 | 0 |
| 2 | CH\_A-030\_E+00 | -30 … -45 | 0 |
| 3 | CH\_A+000\_E+00 | 0 | 0 |
| 4 | LFE | - | - |
| 5 | CH\_A+135\_E+00 | +120 … +150 | 0 |
| 6 | CH\_A-135\_E+00 | -120 … -150 | 0 |
| 7 | CH\_A+090\_E+00 | +85 … +110 | 0 |
| 8 | CH\_A-090\_E+00 | -85 … -110 | 0 |
| 9 | CH\_A+030\_E+35 | +30 … +45 | +30 … +55 |
| 10 | CH\_A-030\_E+35 | -30 … -45 | +30 … +55 |
| 11 | CH\_A+135\_E+35 | +100 … +150 | +30 … +55 |
| 12 | CH\_A-135\_E+35 | -100 … -150 | +30 … +55 |
| FOA (SBA1) | 4 | 1…4 | Ambisonics components with 0,1,2,3 | - | - |
| HOA\*  (SBA) |  | 1… | Ambisonics components with 0,1, 2,… -1 | - | - |
| Mono objects (OBA) | 1…4 | 1…4 | Object(s) with ID 1…4 | - | - |
| Metadata-assisted spatial audio, mono (MASA1) | 1 | 1 | M | - | - |
| Metadata-assisted spatial audio, stereo (MASA2) | 2 | 1,2 | L, R | - | - |

\* = Ambisonics order

## Preamble definition

The purpose of the preamble is to ensure that the CuT and reference systems are in proper warm-up state when processing the file portion comprising the samples to be evaluated. A preamble shall be uniformly distributed low-level noise obtained as 16-bit values with amplitudes between +4 and -4. Uncorrelated noise shall be used across audio tracks.

## File naming

### Generic definitions

The following generic naming convention is used for input/output of processing modules:

*input* Input to a processing module (or sequence of modules)

*output* Output from a processing module (or sequence of modules)

Table 6 specifies suffixes to be used whenever needed for differentiation and may otherwise be left out.

Table 6: File name suffixes

| Suffix | Specification |
| --- | --- |
|  | Track designator according to Table 3 or track index according to Table 5 |
|  | Sampling frequency in kHz being .8k, .16k, .32k or .48k |

### Naming examples of the input files and processed output files

The following sections describe the file naming conventions where all letters in blue indicate variables in the filename while black letters indicate constants.

#### P.SUPPL800 input content files

The filenames of the input content samples are represented by:

*leeeayszz*.wav

The filenames of the accompanying metadata files (applicable to metadata-assisted spatial audio, object-based audio) are represented by:

*leeeayszz*.met for metadata-assisted spatial audio

*leeeayszz*.wav.*o*.csv for object-based audio

where:

* *l* stands for the listening lab designator (a through d according to Table 2)
* *eee* stands for the experiment designator, e.g. p01 (see Table 1)
* *a* stands audio, and *y* is the per experiment category according to IVAS-8a
* *s* stands for sample and *zz* is the sample number; 01, 02, 03, 04, 05, 06, 07, where 07 is the preliminary
* *o* stands for object number; 0, 1, 2, 3

#### BS.1534 input content files

The filenames of the input content samples are represented by:

*leeeayszz*.wav

The filenames of the accompanying metadata files (applicable to metadata-assisted spatial audio, object-based audio) are represented by:

*leeeayszz*.met for metadata-assisted spatial audio

*leeeayszz*.wav.*o*.csv for object-based audio

where:

* *l* stands for the listening lab designator (a through d according to Table 2)
* *eee* stands for the experiment designator, e.g. m1a (see Table 1)
* *a* stands audio, and *y* is set to 1, *s* stands for sample and *zz* is the sample number; 01 … 12
* *o* stands for object number; 0, 1, 2, 3

#### Noise files

The filenames of the input noise samples are represented by:

*leeeny*.wav

where:

* *l* stands for the listening lab designator (a through d according to Table 2)
* *eee* stands for the experiment designator, e.g. m1a (see Table 1)
* *n* stands for noise and *y* is the per experiment category according to IVAS-8a

#### Processed P.SUPPL800 content files

The filenames of the processed content samples are represented by:

*leeeayszz*.c*nn*.wav

The filenames of the accompanying metadata files (applicable to metadata-assisted spatial audio, object-based audio) are represented by:

*leeeayszz*.c*nn*.met for metadata-assisted spatial audio

*leeeayszz*.c*nn*.wav.*o*.csv for object-based audio

where:

* *l* stands for the listening lab designator (a through d according to Table 2)
* *eee* stands for the experiment designator, e.g. p01 (see Table 1)
* *a* stands audio, and *y* is the per experiment category according to IVAS-8a
* *s* stands for sample and zz is the sample number; 01, 02, 03, 04, 05, 06, 07, where 07 is the preliminary
* *c* stands for the condition with the number *nn* = 01, 02, .. as specified in IVAS-8a
* *o* stands for object number; 0, 1, 2, 3

#### Processed BS.1534-4 content files

The filenames of the processed content samples are represented by:

*leeeayszz*.c*nn*.wav

The filenames of the accompanying metadata files (applicable to metadata-assisted spatial audio, object-based audio) are represented by:

*leeeayszz*.c*nn*.met for metadata-assisted spatial audio

*leeeayszz*.c*nn*.wav.*o*.csv for object-based audio

where:

* *l* stands for the listening lab designator (a through d according to Table 2)
* *eee* stands for the experiment designator, e.g. m1a (see Table 1)
* *a* stands audio, and *y* is set to 1
* *s* stands for sample and *zz* is the sample number; 01 … 12
* *c* stands for the condition with the number *nn* = 01, 02, .. as specified in IVAS-8a
* *o* stands for object number; 0, 1, 2, 3

### File naming error patterns

Frame erasure files will have the name “eeefrr.g192”, where:

* eee stands for the experiment designator, e.g. p01 (see Table 1)
* f stands for frame loss pattern file
* rr stands for the frame loss rate in percent.

### File naming jitter profiles

The MTSI jitter profiles have the file names “dly\_error\_profile\_x.dat” where x is the profile number.

### File naming jitter derived error patterns

The error patterns which are derived from the MTSI jitter profiles have the file names “f\_profile\_x.g192”, where:

* f stands for frame loss pattern file
* profile\_x stands for the number of the MTSI jitter profile, see 3.10.4.

# Processing stages

This clause defines, in the form of diagrams, the processing stages that shall be supported by the IVAS candidate codec under test (CuT).

## General considerations for processing

### Source material requirements

The input source material shall follow the format specification in clause 3.6. All input source material files shall be 20 ms block aligned.

Editor’s note: For BS.1534 experiments, identical file lengths of processed items may be required, TBC.

### Concatenated sequences processing

In all experiments, the pre-processed material will be processed concatenated, comprising a preamble followed by the *samples* being evaluated. The post processing stage will discard the preamble and extract the *samples* to be evaluated (see below). For all P.SUPPL800 tests, the complete sequence of preamble, samples for evaluation and preliminaries will be processed twice in concatenation. The *samples* to be used for the tests are extracted from the second part of the processing.

### Frame error application

In frame erasure conditions, the erasures shall affect the same segments of input signal for the CuT and reference codecs. This shall be done by compensating for all encoder-side delays. As for the EVS reference codec (see clause A.1), CuT encoder and decoder executables shall compensate the delay of the output files.

Note: No delay compensation is applied for jitter related delay.

Note: Exact delay compensation might not be possible.

### Concatenation setup

The *samples* are concatenated after level adjustment. If the format includes metadata (object-based audio), the same concatenation is applied to the corresponding metadata-files.

The concatenation for P.SUPPL800 involves concatenation of clean speech *samples* followed by superposition with noise. For the clean-speech P.SUPPL800 tests, the noise may represent very low-level recording noise. In the P.SUPPL800 tests with background noise, the noise used depends on the audio content category specified in the respective test plan.

A consequence of using different background noises for different audio content categories is that the concatenation (and subsequent processing) can only be done for the subset of *samples* of the same category. Since the test plan defines 6 categories, 6 subsets of clean speech *samples* are concatenated followed by superimposing the noise of that category. To align the processing for the clean-speech tests, even in that case, the concatenation is done in subsets corresponding to the 6 categories.

The concatenation for P.SUPPL800 is illustrated in Table 7 below where all *samples* belonging to a category are concatenated from left to right and line by line. Different lines represent different categories. The concatenation comprises firstly the low-level noise preamble in accordance with clause 3.9. Then the *samples* for the actual evaluation and one *sample* for the preliminary (listener familiarization) test follow. The *sample* order is therefore for a given category *y:* *ays6, ays7, ays1*, *ays2*, *ays3*, *ays4*, *ays5*, *ays6, ays7.* The total length of the final concatenated clean speech sample of a category (name *eee*a*y*\_cl) may vary depending on the length of the *samples* being concatenated.

Table 7: Concatenation order of P.SUPPL800 files

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Concatenated sample** | **Category** | **Item sequence in concatenated files** | | | | | | | |
| Preamble | Samples for evaluation | | | | | | Preliminary |
| ***eee*a1\_cl** | **1** | low-level noise | a1s1 | a1s2 | a1s3 | a1s4 | a1s5 | a1s6 | a1s7 |
| ***eee*a2\_cl** | **2** | low-level noise | a2s1 | a2s2 | a2s3 | a2s4 | a2s5 | a2s6 | a2s7 |
| ***eee*a3\_cl** | **3** | low-level noise | a3s1 | a3s2 | a3s3 | a3s4 | a3s5 | a3s6 | a3s7 |
| ***eee*a4\_cl** | **4** | low-level noise | a4s1 | a4s2 | a4s3 | a4s4 | a4s5 | a4s6 | a4s7 |
| ***eee*a5\_cl** | **5** | low-level noise | a5s1 | a5s2 | a5s3 | a5s4 | a5s5 | a5s6 | a5s7 |
| ***eee*a6\_cl** | **6** | low-level noise | a6s1 | a6s2 | a6s3 | a6s4 | a6s5 | a6s6 | a6s7 |

Note: For speech with background the complete sequence of preamble, samples for evaluation and preliminaries shall be repeated once for the processing, and the second part shall be used for the testing.

Note: For all P.SUPPL800 tests, the complete sequence of preamble, samples for evaluation and preliminaries will be processed twice in concatenation. The *samples* to be used for the tests are extracted from the second part of the processing.

For all BS.1534 tests, the *samples* are individually processed twice in concatenation. The second part shall be used for the testing. There shall be no low-level noise preamble for the BS.1534 tests.

For all experiments, after the main processing, the processed concatenated samples for evaluation are divided into separate samples. A half Hann window of 100 ms tapering shall be applied to the start and end of the separated samples. The procedure for splitting and windowing concatenated files is described in clause 5.1.7.

Note: Metadata-assisted spatial audio inputs are created from concatenated FOA/HOA2 audio input files.

## Generation of test samples

### P.SUPPL800 speech samples

Except for the experiment p03, p06 and p07, clean speech *samples* for the P.SUPPL800 tests are created by filtering clean mono sentences with room impulse responses generating the clean stereo/spatial speech samples. Note that this clean stereo/spatial speech *sample* procedure applied regardless of whether the experiment is a clean speech or a speech with background experiment.

The main steps for clean speech *sample* creation prior to mixing with background sounds are:

1. Convolve a raw mono clean speech sentence of a first talker with a set of room impulse responses respective the position of that talker relative to a capture point to get a stereo or spatial representation of that sentence.
2. Level normalize first convolved sentence according to test plan requirements using the BS.1770 tool.
3. Convolve a raw mono clean speech sentence of a second talker with a set of room impulse responses respective the position of that talker relative to a capture point to get a stereo or spatial representation of that sentence.
4. Level normalize second convolved sentence according to test plan requirements using the BS.1770 tool.
5. Superpose both level normalized sentences with defined time offset. The superposition of the level normalized sentences with defined time offset is done as follows:
   1. The first sentence is inserted in an empty buffer after a leading silence period (zeros) of X ms.
   2. The second sentence is added to that buffer with a time offset of Y ms.
      1. Y is calculated as: Y = X + length(raw mono sentence 1) + GAP, wherein ‘GAP’ is the duration of the silence period between the two sentences as specified in the test plan. In case of overtalk, this parameter may be negative.
   3. Finally, a trailing silence period of Z ms (zeros) is appended.

Note: This procedure creates a two-sentence stereo or spatialized test item of length   
X + length(raw mono sentence 1) + GAP + length(filtered sentence 2) + Z.

The total length of the generated samples shall not exceed 10s. The leading silence shall be X=500 ms, and the trailing silence shall be Z=1000 ms. The GAP is specified in IVAS-8a for each experiment separately.

This procedure is visualized in the following figure:



Figure 1: P.SUPPL800 clean speech *sample* creation

“Level adjustment” is done using BS.1770 and scaling using Python. Time shift and appending trailing silence is also performed using Python. The optional resampling is done in accordance with clause 5.1.6 whenever needed for the convolution and the subsequently resampled back to 48 kHz prior to level adjustment. The digital low-level noise is an artificial random sequence with amplitude [-4, +4], uncorrelated between the audio tracks.

The convolution with room impulse responses, is done for each track *input* separately using:

reverb.exe input IR output

where IR corresponds to the applicable spatial room impulse response in accordance with IVAS-8a [1]. The sampling rate for the audio tracks and the impulse responses need to match. Splitting/joining of the audio tracks is performed using Python.

### General processing for background noise

The background noise shall be trimmed to the length of the concatenated clean speech sample to which the noise shall be added.



Figure 2: Trimming of background noise.

The resulting 48 kHz trimmed noise file is further on also referenced as “48 kHz input background”.

## Pre-processing

### Pre-processing stages

#### Filtering

Filtering is performed using either

* HP50\_48KHZ high pass filter at 48 kHz, see 5.1.2.1, or
* 20KBP band pass filter at 48 kHz, see 5.1.2.2.

The test plan [1] specifies which filter to use dependent on the experiment.

#### Level adjustment

The signals are iteratively level adjusted based on BS.1770 measures until or the number of iterations exceeds 10.



Figure 3: Level adjustment using BS.1770

### Pre-processing for experiments p03, p06, p07 and m1a-m7b

The following figure illustrates the pre-processing for the P.SUPPL800 music and mixed content and object-based audio experiments p03, p06 and p07, and all the BS.1534 generic audio experiments m1a-m7b. It is not applicable to P.SUPPL800 experiments relying on sample generation according to clause 4.2.1.



Figure 4: Pre-processing for experiments p03, p06, p07 and m1a-m7b.

### Pre-processing for experiments p01-p02, p04-p05, p08-p09

The following figure illustrates the pre-processing for the clean speech and speech with background P.SUPPL800 experiments p01-02, p04-05 and p08-09, relying on sample generation according to clause 4.2.1.



Figure 5: Pre-processing for P.SUPPL800 experiments p01-02, p04-05, p08-09

For the 48 kHz input speech, after initial windowing, a high pass filtering using the HP50\_48KHZ filter is performed. The level adjustment is done as described inv clause 4.3.1.2 using BS.1770 targeting the playout level. Subsequently, the filtered and loudness corrected files are concatenated.

For the 48 kHz input background, the level adjustment is done as described in clause 4.3.1.2 using BS.1770 with gating disabled (‘rms’ mode). The target loudness is adjusted according to (playout loudness – target SNR). Subsequently, both signals are mixed.

## Processing for anchor conditions

### MNRU/ESDRU

Note: ESDRU is only available for stereo/binaural audio.



Figure 6: Processing for MNRU/ESDRU conditions. Rendering is bypassed when audio format is already stereo/binaural/5.1/7.1+4.

### 7 kHz low-pass anchor



Figure 7: Processing for LP7 conditions. Rendering is bypassed when audio format is already stereo/binaural/5.1/7.1+4.

## Processing for stereo inputs

### Reference conditions

Level normalization for stereo is assuming (virtual) loudspeakers at azimuth for both loudspeaker and binaural rendering.



Figure 8: Processing for stereo reference conditions.

### CuT conditions



Figure 9: Processing for stereo CuT conditions. Error insertion / Network simulation is bypassed for error-free/non-JBM conditions.

### Reference codec conditions



Figure 10: Processing for stereo reference codec conditions. Error insertion / Network simulation is bypassed for error-free/non-JBM conditions.

Processing for stereo reference codec conditions is performed using 2x EVS encoding/decoding. In case of error insertion/network simulation, the same error pattern is applied to both EVS bitstreams.

## Processing for multi-channel audio inputs

### Reference conditions



Figure 11: Processing for multi-channel audio reference conditions

### CuT conditions



Figure 12: Processing for multi-channel audio CuT conditions. Error insertion / Network simulation is bypassed for error-free/non-JBM conditions.

### Reference codec conditions



Figure 13: Processing for multi-channel audio reference codec conditions. Error insertion / Network simulation is bypassed for error-free/non-JBM conditions.

Processing for multi-channel reference codec conditions is performed using Nx EVS encoding/decoding. The LFE channel is coded using EVS NB 9.6 kbps mode. In case of error insertion/network simulation, the same error pattern is applied to all EVS bitstreams.

## Processing for scene-based audio inputs

### Reference conditions



Figure 14: Processing for scene-based audio reference conditions

### CuT conditions



Figure 15: Processing for scene-based audio CuT conditions. Error insertion / Network simulation is bypassed for error-free/non-JBM conditions.

### Reference codec conditions



Figure 16: Processing for scene-based audio reference codec conditions. Error insertion / Network simulation is bypassed for error-free/non-JBM conditions.

Processing for scene-based audio reference codec conditions is performed using Nx EVS encoding/decoding. Before encoding, a truncation of the scene-based audio signal to FOA or planar FOA is performed. In case of error insertion/network simulation, the same error pattern is applied to all EVS bitstreams.

## Processing for metadata-assisted spatial audio (MASA) inputs

### Reference conditions

Note: 48 kHz input in FOA/HOA2 format is expected. Stereo-MASA with 1-dir can be obtained from FOA. Stereo-MASA with 2-dir requires HOA2.



Figure 17: Processing for MASA reference conditions

### CuT conditions

Note: 48 kHz input in FOA/HOA2 format is expected. Stereo-MASA with 1-dir can be obtained from FOA. Stereo-MASA with 2-dir requires HOA2.



Figure 18: Processing for MASA CuT conditions

### Reference codec conditions

#### MASA reference codec conditions based on multi-EVS and UQ metadata

Note: 48 kHz input in FOA/HOA2 format is expected. Stereo-MASA with 1-dir can be obtained from FOA. Stereo-MASA with 2-dir requires HOA2.



Figure 19: Processing for MASA reference codec conditions based on multi-EVS and UQ metadata

#### MASA reference codec conditions based on multi-EVS FOA

Note: 48 kHz input in FOA/HOA2 format is expected. Stereo-MASA with 1-dir can be obtained from FOA. Stereo-MASA with 2-dir requires HOA2.



Figure 20: Processing for MASA reference codec conditions based on multi-EVS FOA. All-zero Z component is added in the cases this component is not encoded.

## Processing for object-based audio inputs

### Reference conditions



Figure 21: Processing for object-based audio reference conditions



### CuT conditions



Figure 22: Processing for object-based audio CuT conditions. Error insertion / Network simulation is bypassed for error-free/non-JBM conditions.

### Reference codec conditions



Figure 23: Processing for object-based audio reference codec conditions. Error insertion / Network simulation is bypassed for error-free/non-JBM conditions.

Processing for object-based audio reference codec conditions is performed using Nx EVS encoding/decoding. In case of error insertion/network simulation, the same error pattern is applied to all EVS bitstreams.

## Post-processing

### Segmentation and windowing of processed conditions



Figure 24: Segmentation and windowing of processed concatenated samples

### Level normalization of processed conditions



Figure 25: Level adjustment of processed reference, reference codec, anchor and CuT conditions using BS.1770

Note: For FER conditions the post level adjustment should be based on corresponding samples processed without frame losses.

# Processing modules

This section specifies the operation of pre- and post-processing modules being utilized by the processing stages. Certain tools expect mono PCM input. In this case, removal of the WAVE header and splitting of multi-track inputs into mono PCM is required.

## Pre- and post-processing operations

### General delay compensation for the STL filter tool

All filtering steps include a delay compensation step in accordance with clause A.1. For preparing the delay compensation, samples of the preamble are added to the end of the input file before applying the filter step. After completion of the filtering step, the samples are to be removed from the beginning of the filtered file.

### Filtering operations

#### HP50\_48KHZ filtering

To produce a 50Hz high pass filtered 48kHz sampling file use:

filter.exe HP50\_48KHZ *input.48k* *output.48k* 960

#### 20KBP filtering

To produce a 20KBP band pass filtered 48 kHz sampling file use:

filter.exe 20KBP *input.48k* *output.48k* 960

### Level adjustment

#### BS.1770 level adjustment

To obtain the scaling factor for normalization to the level of a 48 kHz sampled file, use:

bs1770demo.exe –nchan N –lev L -conf xxxx *input.48k*

where N is the number of channels, L the target level in LKFS (default: -26), and xxxx a configuration string with one value per channel specifying channel weighting according to:

x = ’1’ loudspeaker position within |elevation| < 30 deg, 60 deg <= |azimuth| <= 120 deg,   
(weighted by 1.41)

x = ’L’ LFE channel (weighted by 0)

x = ’0’ otherwise (weighted by 1)

To obtain the scaling factor for normalization for stereo or binaural to -26 LKFS:

bs1770demo.exe –nchan 2 –lev -26 -conf 00 *input.48k*

To obtain the scaling factor for normalization for 7.1+4 multi-channel configuration, as specified in Table 5, to -26 LKFS:

bs1770demo.exe –nchan 12 –lev -26 -conf 000L00110000 *input.48k*

To obtain the scaling factor for normalization for 5.1 multi-channel configuration, as specified in Table 5, to   
-26 LKFS:

bs1770demo.exe –nchan 6 –lev -26 -conf 000L11 *input.48k*

To normalize the level of a 48 kHz sampled background noise file, in addition the option *“*-rms*”* shall be used.

#### BS.1770 level measuring

To measure the level of a 48 kHz sampled file, use:

bs1770demo.exe –nchan N -conf xxxx *input.48k*

where N is the number of channels, and xxxx a configuration string with one value per channel specifying channel weighting according to:

x = ’1’ loudspeaker position within |elevation| < 30 deg, 60 deg <= |azimuth| <= 120 deg,   
(weighted by 1.41)

x = ’L’ LFE channel (weighted by 0)

x = ’0’ otherwise (weighted by 1)

To measure the loudness for stereo or binaural to -26 LKFS:

bs1770demo.exe –nchan 2 –lev -26 -conf 00 *input.48k*

To measure the loudness for 7.1+4 multi-channel configuration, as specified in Table 5, to -26 LKFS:

bs1770demo.exe –nchan 12 –lev -26 -conf 000L00110000 *input.48k*

To measure the loudness for 5.1 multi-channel configuration, as specified in Table 5, to -26 LKFS:

bs1770demo.exe –nchan 6 –lev -26 -conf 000L11 *input.48k*

To measure the level of a 48 kHz sampled background noise file, in addition the option *“*-rms*”* shall be used.

### Summation of audio signals

The addition of audio signals (samples) in the same sampling frequency (16, 32 or 48 kHz) is performed using Python.

### Concatenation

The concatenation of samples is performed using Python. The undo\_concat.txt contains the parameters for segmentation.

### Sampling rate changes (resampling)

#### Rate-change from 48- to 32-kHz sampling

To produce a 32 kHz sampling file from a 48 kHz sampling file, use:

filter.exe –up SHQ2 input.48k tmp.96k 960

filter.exe –down SHQ3 tmp.96k output.32k 1920

#### Rate-change from 32- to 48-kHz sampling

To produce a 48 kHz sampling file from a 32 kHz sampling file, use:

filter.exe –up SHQ3 input.32k tmp.96k 640

filter.exe –down SHQ2 tmp.96k output.48k 1920

### Windowing and segmentation

#### Initial windowing

Initial windowing using a half Hann window of 100 ms tapering is performed using Python.

#### Segmentation

The segmentation of samples is performed using Python. A half Hann window of 100 ms tapering shall be applied to the start and end of the separated samples.

### Truncation of HOA3 to HOA2 / FOA / planar FOA

Truncation of HOA3 input signals to HOA2, FOA or planar FOA (removing higher-order components or Z component) is performed using Python.

### Rendering to Binaural (HRIRs)

#### Input format Channel-based audio, including mono (1.0), stereo (2.0), surround (5.1 and 7.1), surround + height (5.1+4 and 7.1+4)

For surround (5.1, 7.1) and surround + height (5.1+4, 7.1+4) input formats, direct convolution using the Default HRIR set as specified in [5], Annex B.1 is used. The subset of the HRIRs corresponding to the loudspeaker positions is selected.

Binaural rendering of stereo is implemented as a pass-through. Mono input is first rendered to stereo and then considered as binaural.

Example commandline (5.1 Input Format) [11]:

python3 -m ivas\_processing\_scripts.audiotools -if 5\_1 -of BINAURAL  
-i path/to/input\_file -o path/to/output\_file

#### Input format Binaural audio

For binaural audio as input and output format, no extra rendering is applied (pass-through).

#### Input format Scene-based audio (Ambisonics): FOA, HOA2 and HOA3

A computationally efficient and therefore preferred signal domain for binaural rendering of scene-based audio is the spherical harmonics domain. This relies on the HRIRs transformed from time to spherical harmonics domain in which the Ambisonics component signals can directly be convolved with the transformed HRIRs. There are various techniques for the transformation that all rely on least square optimization techniques under certain optimization constraints. The transformed HRIRs shall be obtained from the default HRIRs (See Annex B of Pdoc IVAS-4) using a method keeping good balance between spatial resolution of especially higher order Ambisonics and low degree of timbral artifacts, as provided by the codec proponents. A specification of the transformation method will be made available as part of the IVAS codec deliverables.

Specifically, the sets of HRIRs in spherical harmonics domain are unique for each Ambisonics order. Their length is 128 samples. The transformed HRIRs will be made available along with the reference renderer.

Example command line (HOA3 input format) [11]:

python3 -m ivas\_processing\_scripts.audiotools -if HOA3 -of BINAURAL  
-i path/to/input\_file -o path/to/output\_file

#### Input format Metadata-assisted spatial audio

The rendering of Metadata-assisted spatial audio (MASA) is performed using the masaRenderer as provided in [10]. The masaRenderer supports rendering to binaural audio using the Default HRIR ser as specified in [5], Annex B.1. In the scripts, a simple wrapper around masaRenderer is provided.

Example command line (MASA 1 transport channel) [11]:

python3 -m ivas\_processing\_scripts.audiotools -if MASA1 -of BINAURAL  
-i path/to/input\_file -o path/to/output\_file –im metadata.met

#### Input format Object-based audio

For object-based audio as an input format, rendering for the minimal set of object metadata as defined in [5] Annex C using the Default HRIR ([5], Annex B.1) shall be done. Object position by means of azimuth and elevation on a 20 ms frame basis shall be taken into account. A convolution of object audio with interpolated HRIRs corresponding to the object position is performed. The interpolation is subsequently performed as follows:

* Frame-wise dynamic search of suitable triangle on sphere
* Triangle has to contain target position and has to be as small as possible 🡪 proximity search starting with nearest three points on sphere (great circle distance) and subsequently increasing search distance if triangle is not suitable
* Test if target position lies in current triangle is done by checking the signs of the spherical Barycentric coordinates [8]
* When suitable triangle is found, compute the spherical barycentric coordinates and apply these weights to the respective HRIRs at the triangle vertices (similar to the bilinear interpolation in [9])
* A crossfade is performed between frames
* The interpolated HRIR is used for convolution

Example command line (2 objects) [11]:

python3 -m ivas\_processing\_scripts.audiotools -if ISM2 -of BINAURAL  
-i path/to/input\_file -o path/to/output\_file –im meta\_1.csv meta\_2.csv

### Rendering to Binaural Room (BRIRs)

#### Input format Channel-based audio, including mono (1.0), stereo (2.0), surround (5.1 and 7.1), surround + height (5.1+4 and 7.1+4)

For surround (5.1, 7.1) and surround + height (5.1+4, 7.1+4) input formats, direct convolution using the Default BRIR set as specified in [5], Annex B.2 is used. The subset of the BRIRs corresponding to the loudspeaker positions is selected.

Binaural room rendering of stereo is implemented as a pass-through. Mono input is first rendered to stereo and then considered as binaural room.

Example command line (5.1 Input Format) [11]:

python3 -m ivas\_processing\_scripts.audiotools -if 5\_1 -of BINAURAL\_ROOM  
-i path/to/input\_file -o path/to/output\_file

#### Input format Binaural audio

For binaural audio as input and output format, no extra rendering is applied (pass-through).

#### Input format Scene-based audio (Ambisonics): FOA, HOA2 and HOA3

For scene-based audio, binaural room rendering using the sparse BRIR set as specified in [5], Annex B.2 is performed in two steps:

1. (Pre-) rendering from Ambisonics to 7.1+4 loudspeaker layout
2. Binaural room rendereding from 7.1+4 with BRIRs (see above)

Example command line (HOA3 Input Format) [11]:

python3 -m ivas\_processing\_scripts.audiotools -if HOA3 -of BINAURAL\_ROOM  
-i path/to/input\_file -o path/to/output\_file

#### Input format Metadata-assisted spatial audio

For Metadata-assisted spatial audio, binaural room rendering using the sparse BRIR set as specified in [5], Annex B.2 is performed in two steps:

1. (Pre-) rendering to 7.1+4 using the masaRenderer as defined in the MASA reference software [10].
2. Further rendered from 7.1+4 with BRIRs (see above)

Example command line (MASA 1 transport channel) [11]:

python3 -m ivas\_processing\_scripts.audiotools -if MASA1 -of BINAURAL\_ROOM  
-i path/to/input\_file -o path/to/output\_file –im metadata.met

#### Input format Object-based audio

For Object-based audio, binaural room rendering using the sparse BRIR set as specified in [5], Annex B.2 is performed in two steps

1. (Pre-) rendering from Object based audio to 7.1+4 loudspeaker layout by EFAP panning
2. Further rendered from 7.1+4 with BRIRs (see above)

Example command line (2 objects) [11]:

python3 -m ivas\_processing\_scripts.audiotools -if ISM2 -of BINAURAL\_ROOM  
-i path/to/input\_file -o path/to/output\_file –im meta\_1.csv meta\_2.csv

### Rendering to Loudspeaker

#### Input format Channel-based audio, including mono (1.0), stereo (2.0), surround (5.1 and 7.1), surround + height (5.1+4 and 7.1+4)

The channel-based audio input formats mono (1.0 - CICP1), stereo (2.0 – CICP2), surround (5.1 – CICP6 and 7.1 – CICP12), surround + height (5.1+4 – CICP16 and 7.1+4 – CICP19) are expected to be routed directly to the corresponding loudspeaker format. No additional rendering is expected.

#### Input format Binaural audio

Playback of binaural audio on loudspeaker is currently not foreseen in the selection tests and thus currently not supported.

#### Input format Scene-based audio (Ambisonics): FOA, HOA2 and HOA3

Loudspeaker rendering for scene-based audio uses the all-round Ambisonic panning and decoding (ALLRAD) [7]. ALLRAD is implemented with EFAP [6] in intensity panning mode (a.k.a. EFIP) as the panning method after subjective listening showed benefits for intensity panning over amplitude panning and EFAP over VBAP. A t-design of order 11 was chosen according to the recommendation to use a design of order ≥ 2N+1. The following output configurations are supported: Surround (5.1 – CICP6 and 7.1 – CICP12), surround + height (5.1+4 – CICP16 and 7.1+4 – CICP19).

Example command line (HOA3 input format) [11]:

python3 -m ivas\_processing\_scripts.audiotools -if HOA3 -of 7\_1\_4  
-i path/to/input\_file -o path/to/output\_file

#### Input format Metadata-assisted spatial audio

The rendering of Metadata-assisted spatial audio (MASA) is performed using the masaRenderer as provided in [10]. The masaRenderer supports rendering to 5.1 (CICP6) and 7.1.4 (CICP19) loudspeaker format. In the scripts, a simple wrapper around masaRenderer is provided.

Example command line (MASA 1 transport channel) [11]:

python3 -m ivas\_processing\_scripts.audiotools -if MASA1 -of 7\_1\_4  
-i path/to/input\_file -o path/to/output\_file –im metadata.met

#### Input format Object-based audio

Object-based audio is rendered using EFAP [6] to loudspeakers. EFAP was chosen due to benefits shown in subjective tests for sparse loudspeaker layouts. The use of polygons as opposed to triangles helps avoid inconsistencies especially when panning a virtual object e.g., in the rear trapezoid formed by the middle and upper rear speakers of most loudspeaker configurations where VBAP performs an arbitrary triangulation of the trapezoid.

The following output configurations are supported: Surround (5.1 – CICP6 and 7.1 – CICP12), surround + height (5.1+4 – CICP16 and 7.1+4 – CICP19).

Example command line (2 objects) [11]:

python3 -m ivas\_processing\_scripts.audiotools -if ISM2 -of 7\_1\_4  
-i path/to/input\_file -o path/to/output\_file –im meta\_1.csv meta\_2.csv

### MASA parameters analysis and downmix

To create stereo MASA input format from 1st order ambisonics, where only 1Dir is possible, use:

masaAnalyzer -stereo -1dir -foa input\_4chn.48k masa\_stereo.48k masa.met

Note. Input file shall be raw PCM, 16-bit, 48 kHz, 4-channel interleaved audio.

To create stereo MASA input format from 2nd order ambisonics, where both 1Dir / 2Dir are possible, use:

masaAnalyzer -stereo -2dir -hoa2 input\_9chn.48k masa\_stereo.48k masa.met

Note: Input file shall be raw PCM, 16-bit, 48 kHz, 9-channel interleaved audio.

## Condition processing

This section describes the main processing modules utilized by the processing stages for the test conditions.

### MNRU conditions

To generate a P.50 FB MNRU anchor at XXX dB for a file *input.48k*, use:

p50fbmnru.exe input.48k output.48k XXX M

Note that the P.50 MNRU processing for SWB conditions requires rate-changes between 32 kHz and 48 kHz.

### ESDRU conditions

To generate an ESDRU anchor at distortion level alpha for a file *input*, use:

esdru.exe -e\_step 0.5 -seed 1 -sf fs alpha input output

where fs is the sample frequency, default 48 kHz, where and corresponds to the highest distortion level.

### LP7 conditions

To generate an LP7 anchor use:

filter.exe LP7 *input.48k* *output.48k* 960

where LP7 denotes 7 kHz low-pass filtering.

### EVS operation

To process a file *input.48k* through the EVS codec at XXX bit/s, use:

EVS\_cod.exe -max\_band SWB/FB [-dtx] XXX 48 *input.48k* bitstream

EVS\_dec.exe 48 bitstream *output.48k*

where XXX is one of 7200, 8000, 9600, 13200, 16400, 24400, 32000, 48000, 64000, 96000, 128000.

To process the LFE channel *input.48k* through the EVS codec, at 9600 bit/s, use:

EVS\_cod.exe -max\_band NB 9600 48 *input.48k* bitstream

EVS\_dec.exe 48 bitstream *output.48k*

Note: EVS encoder and decoder provide delay compensated output files.

### CuT operation (Encoding)

#### Input format stereo

To encode a stereo file *input.48k* through the IVAS codec at XXX bit/s, use:

IVAS\_cod.exe -stereo -max\_band SWB/FB [-dtx] XXX 48 *input.48k* bitstream

where XXX is one of 13200, 16400, 24400, 32000, 48000, 64000, 80000, 96000, 128000, 160000, 192000, 256000.

#### Input format channel-based audio 5.1 and 7.1.4

To encode a multi-channel file *input.48k* through the IVAS codec at XXX bit/s, use:

IVAS\_cod.exe -mc MC\_CONF XXX 48 *input.48k* bitstream

where   
XXX is one of 13200, 16400, 24400, 32000, 48000, 64000, 80000, 96000, 128000, 160000, 192000, 256000, 384000, 512000 and  
MC\_CONF is 5\_1 for 5.1 multi-channel and 7\_1\_4 for 7.1.4 multi-channel audio files.

#### Input format scene-based audio (FOA, HOA2, HOA3)

To encode a scene-based file *input.48k* through the IVAS codec at XXX bit/s, use:

IVAS\_cod.exe -sba SBA\_ORDER [-dtx] XXX 48 *input.48k* bitstream

where   
XXX is one of 13200, 16400, 24400, 32000, 48000, 64000, 80000, 96000, 128000, 160000, 192000, 256000, 384000, 512000   
and  
SBA\_ORDER is 1 for FOA, 2 for HOA2 and 3 for HOA3 Ambisonics audio files.

#### Input format Metadata-assisted spatial audio

To encode a metadata-assisted spatial audio file *input.48k* through the IVAS codec at XXX bit/s, use:

IVAS\_cod.exe -masa MASA\_CH MASAFILE [-dtx] XXX 48 *input.48k* bitstream

where   
XXX is one of 13200, 16400, 24400, 32000, 48000, 64000, 80000, 96000, 128000, 160000, 192000, 256000, 384000, 512000   
and  
MASA\_CH specifies the number of input/transport channels (1 or 2)

and

MASAFILE specifies the input file containing parametric MASA metadata.

#### Input format Object-based audio

To encode an object-based audio file *input.48k* through the IVAS codec at XXX bit/s, use:

IVAS\_cod.exe -ism OBJ\_NUM OBJFILE0 [OBJFILE1 [OBJFILE2 [OBJFILE3]]] [-dtx] XXX 48 *input.48k* bitstream

where   
XXX is one of 13200, 16400, 24400, 32000, 48000, 64000, 80000, 96000, 128000, 160000, 192000, 256000, 384000, 512000  
and  
OBJ\_NUM specifies the number of objects

and

OBJFILE0 specifies input file containing object metadata belonging to object 0,  
OBJFILE1 specifies input file containing object metadata belonging to object 1,  
OBJFILE2 specifies input file containing object metadata belonging to object 2,  
OBJFILE3 specifies input file containing object metadata belonging to object 3.

### CuT operation (Decoding, non-JBM case)

#### Input format stereo

To decode a stereo bitstream *bitstream* through the IVAS codec, use:

IVAS\_dec.exe stereo 48 bitstream output.48k

#### Input format channel-based audio 5.1 and 7.1.4

To decode a multi-channel bitstream *bitstream* through the IVAS codec, use:

IVAS\_dec.exe MC\_CONF 48 bitstream output.48k

where   
MC\_CONF is 5\_1 for 5.1 multi-channel and 7\_1\_4 for 7.1.4 multi-channel audio files.

#### Input format scene-based audio (FOA, HOA2, HOA3)

To decode a scene-based bitstream *bitstream* through the IVAS codec, use:

IVAS\_dec.exe SBA\_CONF 48 bitstream output.48k

where   
SBA\_CONF is FOA for FOA, HOA2 for HOA2 and HOA3 for HOA3 Ambisonics audio files.

#### Input format Metadata-assisted spatial audio

To decode a metadata-assisted spatial audio bitstream *bitstream* through the IVAS codec, use:

IVAS\_dec.exe EXT 48 bitstream output.48k

The decoder will write out an additional metadata file output.48k.met which contains the parametric MASA metadata.

#### Input format Object-based audio

To decode an object-based bitstream *bitstream* through the IVAS codec, use:

IVAS\_dec.exe EXT 48 bitstream output.48k

The decoder will write out an additional metadata files output.48k.[0-3].csv which contains the object metadata as follows:

output.48k.0.csv specifies input file containing object metadata belonging to object 0,  
output.48k.1.csv specifies input file containing object metadata belonging to object 1,  
output.48k.2.csv specifies input file containing object metadata belonging to object 2,  
output.48k.3.csv specifies input file containing object metadata belonging to object 3.

### CuT operation (Decoding, JBM case)

#### Input format stereo

To decode a stereo RTP+G.192 file *netsimoutput* written by the network simulator through the IVAS codec, use:

IVAS\_dec.exe –Tracefile tracefile\_dec –VOIP stereo 48 netsimoutput output.48k

#### Input format channel-based audio 5.1 and 7.1.4

To decode a multi-channel RTP+G.192 file *netsimoutput* written by the network simulator through the IVAS codec, use:

IVAS\_dec.exe –Tracefile tracefile\_dec –VOIP MC\_CONF 48 netsimoutput output.48k

where   
MC\_CONF is 5\_1 for 5.1 multi-channel and 7\_1\_4 for 7.1.4 multi-channel audio files.

#### Input format scene-based audio (FOA, HOA2, HOA3)

To decode a scene-based audio RTP+G.192 file *netsimoutput* written by the network simulator through the IVAS codec, use:

IVAS\_dec.exe –Tracefile tracefile\_dec –VOIP SBA\_CONF 48 netsimoutput output.48k

where   
SBA\_CONF is FOA for FOA, HOA2 for HOA2 and HOA3 for HOA3 Ambisonics audio files.

#### Input format Metadata-assisted spatial audio

To decode a metadata-assisted spatial audio RTP+G.192 file *netsimoutput* written by the network simulator through the IVAS codec, use:

IVAS\_dec.exe –Tracefile tracefile\_dec –VOIP EXT 48 netsimoutput output.48k

The decoder will write out an additional metadata file output.48k.met which contains the parametric MASA metadata.

#### Input format Object-based audio

To decode an object-based RTP+G.192 file *netsimoutput* written by the network simulator through the IVAS codec, use:

IVAS\_dec.exe –Tracefile tracefile\_dec –VOIP EXT 48 netsimoutput output.48k

The decoder will write out an additional metadata files output.48k.[0-3].csv which contains the object metadata as follows:

output.48k.0.csv specifies input file containing object metadata belonging to object 0,  
output.48k.1.csv specifies input file containing object metadata belonging to object 1,  
output.48k.2.csv specifies input file containing object metadata belonging to object 2,  
output.48k.3.csv specifies input file containing object metadata belonging to object 3.

## Encoder and decoder CuT executable requirements

The executables shall be compatible to WIN32.

## Error Insertion (EID)

For the conditions where random frame erasures are desired, frame erasure patterns are applied to the bitstream using tools from the [ITU-T STL2009 library].

### Frame error tool

For all reference codecs and CuTs, the following processing shall be used:

eid-xor.exe –vbr –fer g192bsin ep.g192 g192bsout

where:

g192bsin is the input bit stream

ep.g192 is the error pattern file

g192bsout is the output bit stream

### Pattern generation

Before generating error patterns, the sta file of the gen-patt tool needs to be initialized as specified in Annex B.3.

The error patterns used are generated using the gen-patt tool as follows:

gen-patt.exe -tailstat -fer -g192 -gamma 0 -rate XXX -tol 0.001 -reset -n LENGTH -start 501 ep.g192

where XXX is the required erasure rate, i.e. 0.03 for 3% and 0.06 for 6% FER.

Different error patterns should be generated for each experiment. For all experiments, this parameter needs to be adapted to the length of the concatenated sequence.

### Derive EPFs from jitter profiles

To derive an error pattern file from one of the delay and error profiles 1, 2, or 3 given in the MTSI specification, use:

dlyerr\_2\_errpat.exe –d 200 –f 1 –w –s YYY –i *dly\_err\_profile\_XXX.dat -o* f\_profile\_*XXX*.g192

where XXX is one of 1, 2, or 3, and YYY is the random offset into the profile.

To derive an error pattern file from one of the delay and error profiles 4 or 6 given in the MTSI specification, use:

dlyerr\_2\_errpat.exe –l 1 –f 1 –w –s YYY –i *dly\_err\_profile\_XXX.dat -o* f\_profile\_*XXX*.g192

where XXX is one of 4, or 6, and YYY is the random offset into the profile.

To derive an error pattern file from one of the delay and error profile 5 given in the MTSI specification, use:

dlyerr\_2\_errpat.exe –l 1 –f 2 –w –s YYY –i *dly\_err\_profile\_5.dat -o* f\_profile\_5.g192

where YYY is the random offset into the profile.

To derive an error pattern file, assuming one speech frame per packet and a maximum permitted late loss rate of 1% from any other delay and error profile DLY\_ERR\_PROFILE, use:

dlyerr\_2\_errpat.exe –l 1 –f 2 –w –s YYY –i *DLY\_ERR\_PROFILE -o* f\_DLY\_ERR\_PROFILE.g192

where YYY is the random offset into the profile.

### Network simulator for packet jitter generation

To simulate packet jitter for CuT conditions, use

network\_simulator.exe dly\_error\_profile\_x.dat bitstream netsimoutput tracefile\_sim nFramesPerPacket [offset]

where

* dly\_error\_profile\_x.dat is the MTSI delay and error profile number x
* bitstream is the G.192 input bitstream file name
* netsimoutput is the RTP+G.192 output file name
* tracefile\_sim is the trace output file name of the network simulator
* nFramesPerPacket is the number of frames per packet (1, 2)
* offset is the shift/offset in delay and error profile in frames (default: 0)

### Cutting Tool for JBM tests

To trim parameters for the file segmentation after de-jittering, use

jbmtrim.exe –file Fs tracefile\_dec undo\_concat.txt undo\_concat\_trim.txt

where

* file is to enable the undo\_concat.txt support
* Fs is the operating sampling rate
* tracefile\_dec is the trace output file of the CuT decoder
* undo\_concat.txt is the file name where parameters for segmentation are stored
* undo\_concat\_trim.txt is the file name containing the corrected parameters for segmentation

## Automatic complexity and memory instrumentation (WMC tool)

Automatic instrumentation of floating-point C source code to measure the computational complexity and memory consumption is done using the STL WMC tool (wmc\_tool). It is an extension of the “Complexity evaluation tool for floating-point C Code” described in the OpenITU/STL library, Chapter 18.12 [12].

The WMC tool inserts instrumentation information directly into .c source files. By compiling the instrumented code and running the executables, it is possible to get the estimated complexity and memory consumption of the codec being evaluated.

### Preparation of the source code

Before instrumenting the codec with the WMC tool is recommended to remove the pre-processor switches (#if, #ifdef, #else, ...) from the source code. This is due to the fact that the WMC tool is sensitive to unbalanced opening/closing curly braces {} that may be introduced by #ifdef ... #endif instructions.

### Instrumentation with the WMC tool

After removing the pre-processor switches the WMC tool is invoked, e.g., for an encoder using:

wmc\_tool -m encoder.c lib\_enc/\*.c

where encoder.c includes the necessary functions for printing statistics about the memory consumption and lib\_enc includes the encoder source code files, see [12] for further details. The WMC tool will instrument the .c source files in-place. Note, that utility and debug functions of the codec simulation software shall not be instrumented.

### Running the instrumented code

After instrumenting the source code with the WMC tool, the code shall be compiled (without any debugging code). By running the codec, the estimated complexity and memory consumption for the processing will be printed (given the necessary printing functions are implemented as described in clause 5.5.2).

Details about the memory printout and how to read it can be found in the ITU-T STL manual [12].

# References

1. Pdoc IVAS-8a: “Test plan for selection phase”
2. Recommendation ITU-T G.192: “A common digital parallel interface for speech standardization activities”, March 1996
3. Recommendation ITU-T G.191 “Software tools for speech and audio coding standardization”, March 2010
4. TS 26.114, “IP Multimedia Subsystem (IMS); Multimedia Telephony; Media handling and interaction”
5. Pdoc IVAS-4: “IVAS Design Constraints”
6. C. Borß, "A Polygon-Based Panning Method for 3D Loudspeaker Setups," Paper 9106, (2014 October.)
7. F. Zotter, and M. Frank, "All-Round Ambisonic Panning and Decoding," J. Audio Eng. Soc., vol. 60, no. 10, pp. 807-820, (2012 October.)
8. T. Langer, A. Belyaev and H. Seidel, "Spherical Barycentric Coordinates," In Proceedings of the fourth Eurographics symposium on Geometry processing (SGP '06). Eurographics Association, Goslar, DEU, 81–88, (2006)
9. F.P. Freeland, L.W.P. Biscainho and P.S.R. Diniz, "Interpolation of Head-Related Transfer Functions (HRTFS): A multi-source approach." 2004 12th European Signal Processing Conference. Vienna, 2004, pp. 1761–1764.
10. S4-230221 - Processing updates for IVAS MASA C Reference Software
11. <https://forge.3gpp.org/rep/ivas-codec-pc/ivas-processing-scripts.git>
12. OpenITU/STL, <https://github.com/openitu/STL>, May 2023

1. External Resources
   1. Delay compensation for filter operations and reference codecs

The processing steps are delay-compensated in order to apply error insertion on the same parts of the audio signal and to be able to extract the original length and offset for each audio sample used in the tests.

The delay compensation is initialized by concatenating the file to be filtered and the first 960 samples of the preamble. After processing, the delay of the processing operation is compensated, and the original file length is restored.

To compensate the delay for filter operations and reference conditions for encoder and decoder the delay compensation values in in Table 87 and Table 98 shall be used.

Table 8: Delay compensation values for filter operations

|  |  |
| --- | --- |
| **Filter operation** | **Value for delay compensation after filtering operation** |
| -up SHQ2 | 436 |
| -up SHQ3 | 436 |
| -down SHQ2 | 218 |
| -down SHQ3 | 145 |
| HP50\_48KHZ | 839 |
| 20KBP | 200 |

Table 9: Delay compensation values for reference codecs

| **Codec** | **Value for encoder delay compensation** | **Value for decoder delay compensation** |
| --- | --- | --- |
| EVS | 0 | 0 |

* 1. Tools used

The following clause documents the origin of the tools.

* + 1. ITU-T STL processing tools

|  |  |
| --- | --- |
| Source | * ITU-T G.191 * S4-120344 “Filter masks for EVS testing” |
| URL | * https://github.com/openitu/STL * <https://www.itu.int/rec/T-REC-G.191-202303-I> * <http://ftp.3gpp.org/tsg_sa/WG4_CODEC/TSGS4_68/Docs/S4-120344.zip> * <https://github.com/ErikNorvell-Ericsson/STL.git> |
| Version / Release | * G.191 / STL2023 |
| Description | Software tools for speech and audio coding standardization |
| Comments | * G.191 filter tool patched with S4-120344 to enable support for HP50 and SHQ filter * bs1770demo is patched for support of “rms” option, see <https://github.com/ErikNorvell-Ericsson/STL.git>, dev branch * A bugfixed version of the gen-patt tool is available at <https://github.com/openitu/STL>, dev branch |
| Executables | filter, eid-xor, gen-patt, bs1770demo, esdru, p50fbmnru, wmc\_tool |
| Status | Available |

* + 1. Reference codecs
       1. EVS

|  |  |
| --- | --- |
| Source | TS 26.443: Codec for Enhanced Voice Services (EVS) |
| URL | https://www.3gpp.org/ftp/Specs/archive/26\_series/26.443/26443-h00.zip |
| Version / Release | 17.0.0 |
| Description | EVS encoder and decoder software |
| Comments |  |
| Executables | EVS\_cod.exe, EVS\_dec.exe |
| Status | Available |

* + 1. Other tools
       1. Network simulator

|  |  |
| --- | --- |
| Source | Fraunhofer |
| URL | <http://ftp.3gpp.org/tsg_sa/WG4_CODEC/TSGS4_76/Docs/S4-131277.zip> |
| Version / Release | - |
| Description | Jitter Simulator |
| Comments |  |
| Executables | network\_simulator.exe |
| Status | Available |

* + - 1. Jitter profile to EPF converter

|  |  |
| --- | --- |
| Source | Fraunhofer: S4-121077 |
| URL | http://ftp.3gpp.org/tsg\_sa/WG4\_CODEC/TSGS4\_70/Docs/S4-121077.zip |
| Version / Release | - |
| Description | Converts MTSI jitter profiles to error pattern for reference codecs |
| Comments |  |
| Executables | dlyerr\_2\_errpat.exe |
| Status | Available |

* + - 1. JBM trim tool

|  |  |
| --- | --- |
| Source | Fraunhofer: AHEVS-181 |
| URL | http://ftp.3gpp.org/tsg\_sa/WG4\_CODEC/Ad-hoc\_EVS/Docs/AHEVS-181.zip |
| Version / Release | - |
| Description | Tool for trim parameters for segmentation of samples |
| Comments |  |
| Executables | jbmtrim.exe |
| Status | Available |

* + - 1. Randomization tool

|  |  |
| --- | --- |
| Source | S4-121078 |
| URL | http://ftp.3gpp.org/tsg\_sa/WG4\_CODEC/TSGS4\_70/Docs/S4-121078.zip |
| Version / Release | - |
| Description | Tool for providing all randomizations depending on a master seed |
| Comments |  |
| Executables | random.exe |
| Status | Available |

* + - 1. IVAS MASA C Reference Software

|  |  |
| --- | --- |
| Source | Nokia, Orange: S4-230221 |
| URL | https://www.3gpp.org/ftp/TSG\_SA/WG4\_CODEC/TSGS4\_122\_Athens/Docs/S4-230221.zip |
| Version / Release | - |
| Description | masaAnalyzer is the tool that takes in input signal and processes it into transport signals and metadata specified in MASA format.  masaRenderer is the tool that takes in the created MASA format signals and renders it to various listenable output configurations. |
| Comments |  |
| Executables | masaAnalyzer.exe, masaRenderer.exe |
| Status | Available |

1. Randomization scripts

The randomization scripts require one master seed, which will be provided by the HL. The master seed (MASTER\_SEED) determines all parameters for noise files, EPFs, jitter and rate switching.

In addition to the master seed, a pre-run number P is chosen per experiment and lab, which is used in combination with the master seed to determine error patterns and noise file offsets.

Table 10: Definition of Pre-run numbers P

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Experiment** | **LL a**  **Pre-run number P** | **LL b**  **Pre-run number P** | **LL c**  **Pre-run number P** | **LL d**  **Pre-run number P** |
| P800-1 | 101 | 102 | 103 | 104 |
| P800-2 | 105 | 106 | 107 | 108 |
| P800-3 | 109 | 110 | 111 | 112 |
| P800-4 | 113 | 114 | 115 | 116 |
| P800-5 | 117 | 118 | 119 | 120 |
| P800-6 | 121 | 122 | 123 | 124 |
| P800-7 | 125 | 126 | 127 | 128 |
| P800-8 | 129 | 130 | 131 | 132 |
| P800-9 | 133 | 134 | 135 | 136 |
| BS1534-1a | 137 | 138 | 139 | 140 |
| BS1534-1b | 141 | 142 | 143 | 144 |
| BS1534-2a | 145 | 146 | 147 | 148 |
| BS1534-2b | 149 | 150 | 151 | 152 |
| BS1534-3a | 153 | 154 | 155 | 156 |
| BS1534-3b | 157 | 158 | 159 | 160 |
| BS1534-4a | 161 | 162 | 163 | 164 |
| BS1534-4b | 165 | 166 | 167 | 168 |
| BS1534-5a | 169 | 170 | 171 | 172 |
| BS1534-5b | 173 | 174 | 175 | 176 |
| BS1534-6a | 177 | 178 | 179 | 180 |
| BS1534-6b | 181 | 182 | 183 | 184 |
| BS1534-7a | 185 | 186 | 187 | 188 |
| BS1534-7b | 189 | 190 | 191 | 192 |

* 1. Delay and error profile offset

As the JBM profiles can be looped, the range should be the entire profile.

To randomly select an JBM offset for a profile with NUM\_ENTRIES delay values:

random -n 1 -s MASTER\_SEED –d PROFILE\_NUMBER –r 0 (NUM\_ENTRIES-1)

where PROFILE\_NUMBER is the number of the profile.

* 1. Seeds for rate switching

The seeds to generating the rate switching profiles (Section 0) can be obtained by

SEEDx = MASTER\_SEED

* 1. Initialization of gen-patt tool

In order to initialize the gen-patt tool, a *sta file* needs to written. Therefore, two templates are provided:

Template for 6% frame error rate:

EID

BER = 0.060000

GAMMA = 0.000000

RAN-seed = 0x176d71ac

Current State = G

GOOD->GOOD = 0.880000

GOOD->BAD = 1.000000

BAD ->GOOD = 0.880000

BAD ->BAD = 1.000000

Template for 3% frame error rate:

EID

BER = 0.030000

GAMMA = 0.000000

RAN-seed = 0xab5d4825

Current State = G

GOOD->GOOD = 0.940000

GOOD->BAD = 1.000000

BAD ->GOOD = 0.940000

BAD ->BAD = 1.000000

where the RAN-seed needs to be replaced in HEX format. To generate these numbers, use

random -n 1 -s MASTER\_SEED –d P –r 0 99999999

where P is the pre-run number according to Table 10.