Qualcomm Technologies, Inc.

# Hexagon Multimedia: Audio Debug Guide

80-NF768-17 B

February 23, 2015

Confidential - May Contain Trade Secrets
2020-07-02 04:17:10 PDT
luo.yihua@flex.com

**Confidential and Proprietary – Qualcomm Technologies, Inc.**

© 2014-2015 Qualcomm Technologies, Inc. and/or its affiliated companies. All rights reserved.

**NO PUBLIC DISCLOSURE PERMITTED:** Please report postings of this document on public servers or websites to: DocCtrlAgent@qualcomm.com.

Not to be used, copied, reproduced, or modified in whole or in part, nor its contents revealed in any manner to others without the express written permission of Qualcomm Technologies, Inc.

QXDM Professional and Qualcomm Hexagon is a product of Qualcomm Technologies, Inc. Other Qualcomm products referenced herein are products of Qualcomm Technologies, Inc. or its subsidiaries.

Qualcomm
Confidential – May Contain Trade Secrets
2020-07-02 04:17:10 PDT
luo.yihua@flex.com

**Restricted Distribution:** Not to be distributed to anyone who is not an employee of either Qualcomm Technologies, Inc. or its affiliated companies without the express approval of Qualcomm Configuration Management.

Qualcomm, MSM,QXDM Professional, and Hexagon are trademarks of Qualcomm Incorporated, registered in the United States and other countries. All Qualcomm Incorporated trademarks are used with permission. Other product and brand names may be trademarks or registered trademarks of their respective owners.

This technical data may be subject to U.S. and international export, re-export, or transfer (“export”) laws. Diversion contrary to U.S. and international law is strictly prohibited.

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

# Revision history

| Revision | Date | Description |
|---|---|---|
| A | Jan 2014 | Initial release |
| B | Feb 2015 | Deleted product line from title; moved Section 1.4 References and Section 1.6 Acronyms to Appendix A and edited them |

Qualcomm
Confidential – May Contain Trade Secrets
2020-07-02 04:17:10 PDT
luo.yihua@flex.com

# Contents



Qualcomm Confidential - May Contain Trade Secrets


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

# Figures



# Tables



Qualcomm
Confidential – May Contain Trade Secrets
2020-07-02 04:17:10 PDT
luo.yihua@flex.com

# 1 Introduction

## 1.1 Purpose

This document describes how to debug frequently reported customer issues on the MSM™ Android™ platform.

## 1.2 Conventions

Function declarations, function names, type declarations, attributes, and code samples appear in a different font, for example, `#include`.

Code variables appear in angle brackets, for example, `<number>`.

Commands to be entered or code to be changed appear in a different font, for example, **`copy a:*.* b:`**.

Shading indicates content that has been added or changed in this revision of the document.

## 1.3 Technical assistance

For assistance or clarification on information in this document, submit a case to Qualcomm Technologies, Inc. (QTI) at https://support.cdmatech.com/.

If you do not have access to the CDMATech Support website, register for access or send email to support.cdmatech@qti.qualcomm.com.

Confidential – May Contain Trade Secrets
2020-07-02 04:17:10 PDT


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

# 2 DSP Overview

To debug DSP audio issues, the correct QXDM Professional™ (QXDM Pro) logs must be enabled. Using QXDM Pro logging, PCM data in the Hexagon™ processor and log messages related to LPASS can be captured. Logging helps isolate an issue in the Hexagon or APSS, and it can provide other useful information for the analysis of audio issues.

## 2.1 Hexagon Multimedia software overview

This section describes the Hexagon DSP audio firmware architecture and top-level static services that expose interfaces to client processors. For an overview of the static services that are required for audio voice features and enable clients to drive audio and voice use cases, refer to the *Hexagon Multimedia: Elite Firmware Architecture Description Document* (80-N0029-1).

Figure 2-1 illustrates this architecture.

Qualcomm
Confidential – May Contain Trade Secrets
2020-07-02 04:17:10 PDT
luo.yihua@flex.com

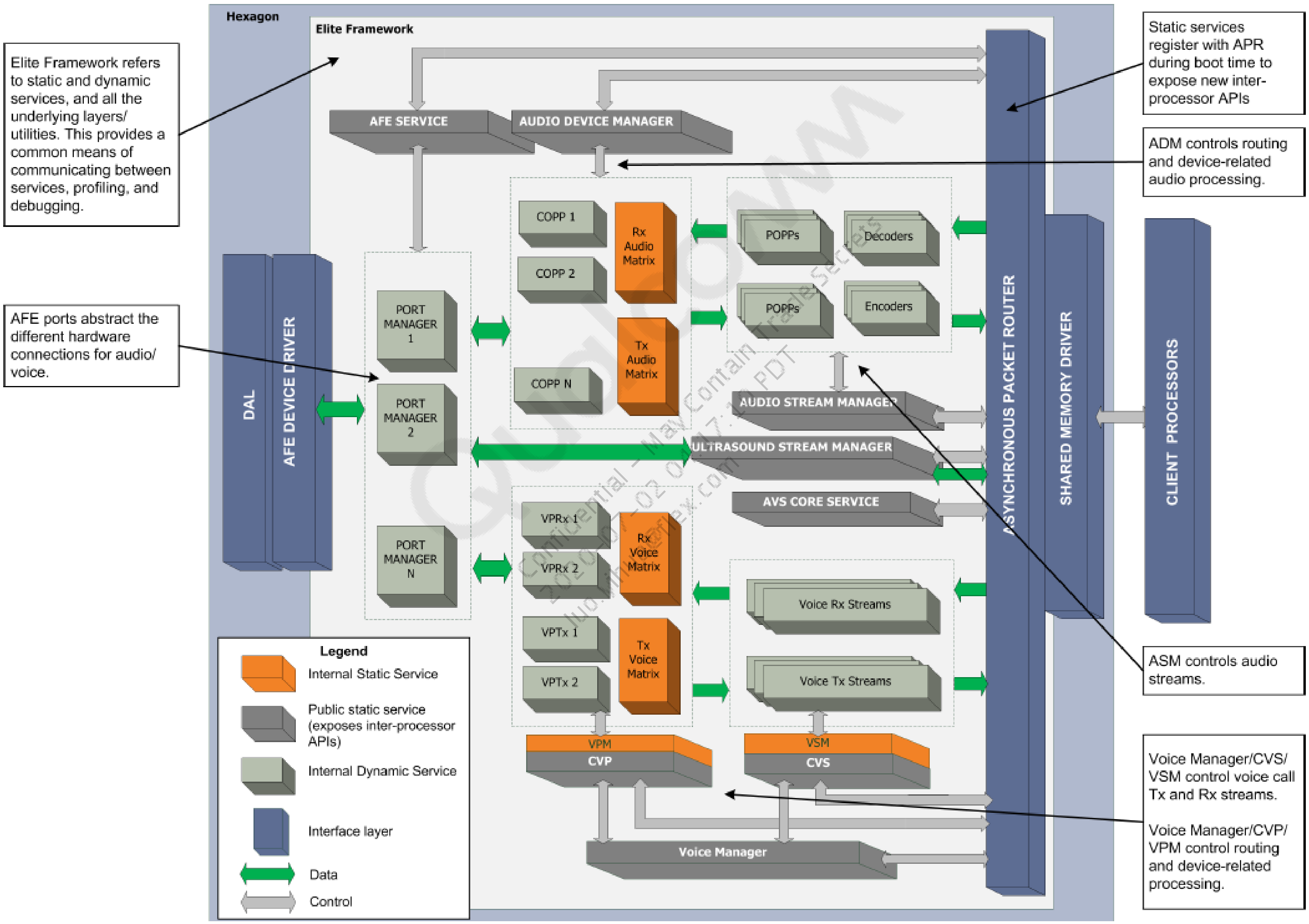


**Figure 2-1 Hexagon DSP architecture**


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

## 2.2 Audio Stream Manager (ASM)

The ASM controls audio streams for playback (Rx path) and record operations (Tx path). Supported use cases include file playback and record, streaming, and transcoding. The controls for the ASM are all stream-related, such as pause, resume, mute, and volume control.

A session is defined as a group of streams with the following properties:

- At most, one connection to a device matrix.
- All streams in the session are synchronized to the Pause, Run, and other session-level commands.
- The HLOS (remote client) is responsible for defining the session and Stream IDs for each stream it opens to give the ASM a simple way of assembling groups of streams into multi-stream sessions.
- The number of supported sessions is 8, and the number of streams per session is 1, except for gapless playback, where the number of streams per session can be either 1 or 2.

Figure 2-2 illustrates the setup for an ASM audio playback session.

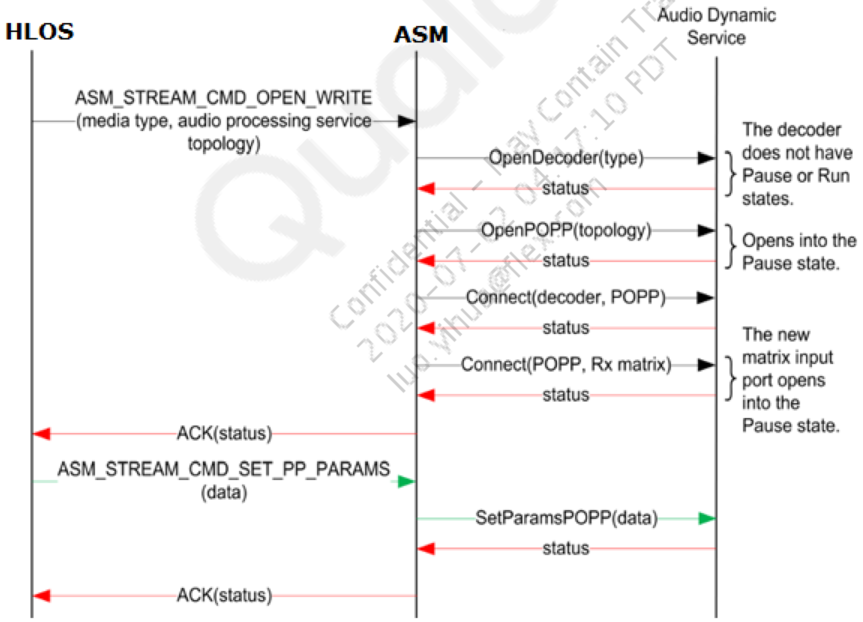


**Figure 2-2 Call flow: ASM audio playback session setup**

When setting up the ASM audio playback session:

- When opening a stream, two dynamic threads are created for the decoder function and PCM Postprocessor (POPP).
- The postprocessor is based on the topology ID provided by the HLOS's ACDB configuration.
- Before running a session, the HLOS must provide the audio calibration data to avoid latency introduced by modules during the module initialization sequence.
- Conversion to the PCM format is performed based upon the HLOS configuration. For example, PCM can be up-converted to 32-bit (Q27 format) even if the media content stream is 16-bit (Q13 format).


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

### 2.2.1 ASM playback session setup

Refer to Figure 2-2 for these steps:

1. The ASM receives the HLOS command, **ASM_STREAM_CMD_OPEN_WRITE_V3** (0x10db3), with parameters such as media type and topology.

   **NOTE:** The command names in Figure 2-2 inherently include the version (such as _V3).

```
00:02:48.917     AudioStreamMgr_AprIf.cpp  00085  AudioStreamMgr: Rec cmd
0x10db3 at [Addr=0x407, Port=0x201]
00:02:48.917     AudioStreamMgr.cpp  00356  Signal received = 10 !
00:02:48.917     AudioStreamMgr_SessionCmdHandler.cpp  00112
AudioStreamMgr [1,1]: receive apr command 0x10db3 at port [0x 201]
00:02:48.917     AudioStreamMgr_SessionCmdHandler.cpp  00727
AudioStreamMgr [1,1]: Bits per sample in open write command: 16
00:02:48.917     AudioStreamMgr_SessionCmdHandler.cpp  00760
AudioStreamMgr [1,1]: for playback, stream_perf_mode is 0 & pp buffer
duration is configured to: 10 msec
```

2. The input media format 0x10be9 is MP3, and it is associated with internal Session ID 1, Stream ID 1.

   **NOTE:** The internal Session ID starts at 0, and the external Session ID starts at 1.

```
00:02:48.917     AudioStreamMgr_Util.cpp  00298  AudioStreamMgr [1,1]:
Input Format 0x10be9
00:02:48.917     AudioStreamMgr_Util.cpp  00663  AudioStreamMgr [1, 1]:
Creating Decoder Service
00:02:48.918     CComboMp3DecoderLib.cpp  00053  Creating QCOM 16 bit MP3
decoder.
```

3. The MP3 decoder thread is created.

```
00:02:48.918     qurt_elite_thread.cpp  00165  THRD CREATE: Thread=0x1033
Name(Hex)= 41, 44, 65, 63, 0, 0, 0, 0
00:02:48.918     AudioStreamMgr_Util.cpp  00673  AudioStreamMgr [1,1]:
Decoder Service Created
```

4. Create a POPP thread with the name, **PAA13**, and associate the Downmix and Soft Volume Controls modules in the topology.

   For details on the topology and modules, refer to the *Hexagon Multimedia: Elite Audio Postprocessor API Interface Specification* for your build (see Section A.1).

```
00:02:48.918     AudioStreamMgr_Util.cpp  00754  AudioStreamMgr [1,1]:
Creating Post-Proc Service
00:02:48.920     audproc_appi_init.cpp  00155  PAA13 audproc_svc:
Initializing topo begin
00:02:48.921      appi_downmix.cpp  00157  APPI Downmix Init done with
outformat 2,16,48000,1,0
00:02:48.921      appi_downmix.cpp  00391  APPI Downmix Process check, 0
00:02:48.921      appi_downmix.cpp  00406  APPI Downmix Get param done
00:02:48.930     appi_softvolumecontrols.cpp  00183  APPI
SoftVolumeControls Init done with outformat 2,16,48000,1,0
```

```
00:02:48.930     appi_softvolumecontrols.cpp  00698  APPI
SoftVolumeControls Process check, 0
00:02:48.930     appi_softvolumecontrols.cpp  00742  APPI
SoftVolumeControls Get param done
00:02:48.931     audproc_appi_init.cpp  00529  PAA13 audproc_svc:
Initializing topology End
00:02:48.931     qurt_elite_thread.cpp  00165  THRD CREATE: Thread=0x1032
Name(Hex)= 41, 41, 31, 33, 0, 6a, 9, f0
00:02:48.931     AudioStreamMgr_Util.cpp  00802  AudioStreamMgr [1,1]:
Post-Proc Service Created
```

5. Connect the POPP of the internal Session ID 1 (aka external Session ID 2) to **Rx matrix input port 0**.

```
00:02:48.931     AudioStreamMgr_DevIF.cpp  00081  AudioStreamMgr [Session
Id = 1,Stream Id = 1]: Request RX Handle
00:02:48.931          AudDevMgr.cpp  00421  ADM: rcvd custom msg [opcode]
= [3145731]
00:02:48.931     AudDevMgr_MtMxIf.cpp  00052  ADM: Adm_MsgStreamConnect,
issuing RX ConnectMtMxInPort cmd, Live: 0, BM: 0
00:02:48.931           MixerSvc.cpp  00348  MtMx #0: rcvd custom msg
1572865
00:02:48.931     MixerSvc_MsgHandlers.cpp  00725  MtMx #0 creating new
i/p port 0 Live? 0 Burstmode? 0 MtMx Burstmode? 0
00:02:48.932     MixerSvc_MsgHandlers.cpp  00975  MtMx #0 new i/p port id
0 open success. Log ID: 537921024
00:02:48.932     AudDevMgr_MtMxIf.cpp  00074  ADM: Adm_MsgStreamConnect,
RX ConnectMtMxInPort cmd: ASM Session ID: 2, i/p port ID: 0
00:02:48.932     MixerSvc_MsgHandlers.cpp  01253  MtMx #0: Leaving cmd
[1572865] (CfgIpPort) handler with status 0
```

6. Connect the MP3 decoder to POPP **PAA13**.

```
00:02:48.932     AudioStreamMgr_Session.cpp  00719  AudioStreamMgr [1,1]:
connecting Svc [0x150000,0x1033] to Svc [0xe0000,0x1032]!
00:02:48.932     AudioStreamMgr_Session.cpp  00719  AudioStreamMgr [1,1]:
connecting Svc [0xe0000,0x1032] to Svc [0xf0000,0x0]!
00:02:48.932        AudioDecSvc.cpp  01865  Decoder service connecting to
down stream service. unCurrentBitfield=0x80000000
00:02:48.932        AudioDecSvc.cpp  01874  Connection Succeeded.
00:02:48.932        audproc_svc.cpp  00847  PAA13 audproc_svc: instance
0xF091C180 connecting to SvcID 0x   f0000
00:02:48.932        AudioDecSvc.cpp  01903  Decoder service done
connecting to down stream service. unCurrentBitfield=0xc0000000 with
result=0x0
00:02:48.932        audproc_svc.cpp  00878  PAA13 audproc_svc: POPP
Connection Succeeded.
```

7. Request the ADSPPM resource for CPU MIPS and system bus bandwidth configuration, based on the media type and module being enabled.

```
00:02:48.932     AudioStreamMgr_adsppm.cpp  00275
AudioStreamMgr_adsppm : Requesting resources from ADSP PM!
00:02:48.932     AudioStreamMgr_adsppm.cpp  00276
AudioStreamMgr_adsppm : CoreIds for ADSP: 7,9,10
```

```
00:02:48.932     AudioStreamMgr_adsppm.cpp  00738  ASM requesting ADSPPM
MIPs per thread = 21, total MIPs = 21
00:02:48.932     AudioStreamMgr_adsppm.cpp  00748  ADSPPM_Request for ASM
MIPS success
00:02:48.935     AudioStreamMgr_AprIf.cpp  00085  AudioStreamMgr: Rec cmd
0x10dab at [Addr=0x407, Port=0x101]
00:02:48.935     AudioStreamMgr_adsppm.cpp  00864  ASM requesting BW for
core id 7
00:02:48.935     AudioStreamMgr_adsppm.cpp  00875  ADSPPM_Request for ASM
BW success
00:02:48.935     AudioStreamMgr_adsppm.cpp  01043  ASM requesting ADSPPM
latency for core 7, sleep latency = 1000 us
00:02:48.935     AudioStreamMgr_adsppm.cpp  01054  ADSPPM_Request for
latency success
```

8. The ASM responds with the HLOS command, ASM_STREAM_CMD_OPEN_WRITE_V3 (0x10db3), with status 0x0 that indicates success.

```
00:02:48.935     AudioStreamMgr_AprIf.cpp  00405
AudioStreamMgr:Port=0x201: ISR status 0x0
Basic Ack: 0x10db3 [0x00]
```

## 2.2.2 Debugging check points for frequent issues

Verify the following:

- The input format and topology are supported by the ASM
- The ASM creates two dynamic threads correctly
- The ASM session is connected to correct matrix port
- The postprocessor module parameters are configured properly
- The CPU MIPS and system bus bandwidth are configured properly


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

## 2.3 Audio Device Manager (ADM)

The ADM establishes routings between audio streams and endpoints (AFE). Endpoints are either DMA channels to hardware input/output ports or pseudoports for enabling various stream loopback bridges.

**NOTE:** A typical pseudoport use case is routing an audio playback stream to a voice stream for in-call music delivery.

The ADM returns a COPP ID to the HLOS in the acknowledgment message to an open request. This differs from the ASM, where the HLOS defines the session and Stream IDs.

Figure 2-3 illustrates the setup for an ADM Rx COPP session.

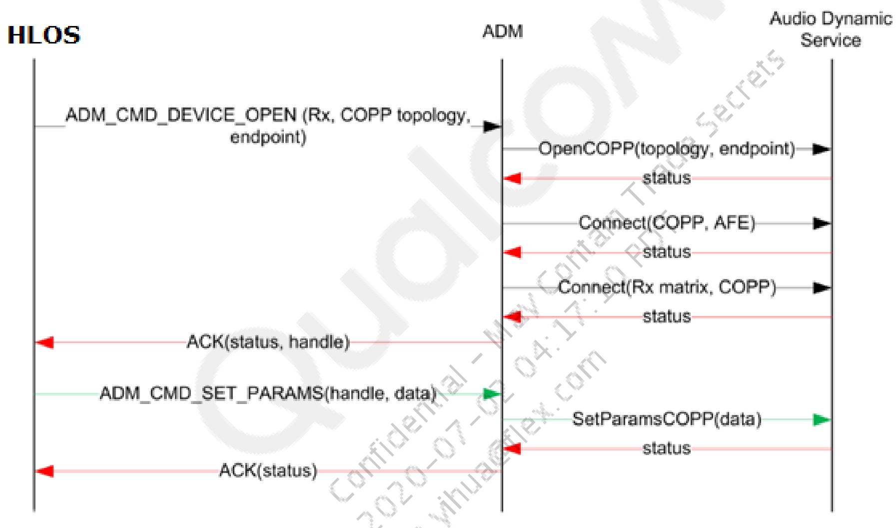


**Figure 2-3 Call flow: ADM Rx COPP session setup**

### 2.3.1 ADM playback session setup

Refer to Figure 2-3 for these steps:

1. The ADM receives the HLOS command, ADM_CMD_DEVICE_OPEN_V5, with topology ID 66323, and it creates a COPP thread.

   **NOTE:** The command names in Figure 2-3 inherently include the version (such as _V5).

```
00:02:48.936          AudDevMgr.cpp  01804  ADM processing
ADM_CMD_DEVICE_OPEN_V5
00:02:48.936          AudDevMgr.cpp  02178  ADM: ADM_CMD_DEVICE_OPEN_V5,
dev_num_ch = 1, dev_ch_mapping = 3, 0, 0, 0, 0, 0, 0, 0
00:02:48.936          AudDevMgr.cpp  02271  ADM: As part of Rx
ADM_CMD_DEVICE_OPEN_V5, issuing CREATE_COPP cmd, topologyID: 66323
00:02:48.936        audproc_svc.cpp  00208  audproc_svc: Creating
00:02:48.940     qurt_elite_thread.cpp  00165  THRD CREATE: Thread=0x1031
Name(Hex)= 43, 43, 31, 35, 0, 0, 0, 0
```


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

2. Connect the COPP to the AFE SLIMbus Rx interface (port 0x4000) with the sampling rate set to **48000 mono channel**.

```
00:02:48.940          AudDevMgr.cpp  02347  ADM: As part of Rx
ADM_CMD_DEVICE_OPEN_V5, # default ch: 1, EP1: 16384
00:02:48.940          AudDevMgr.cpp  02348  As part of Rx
ADM_CMD_DEVICE_OPEN_V5, issuing AFE_CONNECT_REQ cmd
00:02:48.940     AFEPortCustMsgHandler.cpp  00103  At <addr/instance id
0xf10fd568>  FADDCMD  <0x200001>
00:02:48.940     AFEPortCustMsgHandler.cpp  00141  AFESvc: executing
200001
00:02:48.940     AFEPortHandlers.cpp  00162  Rx Client 1, connect to port
dir = 0, intf = 4000, InterruptSamples=48, channels=1,
voice/audio=0,interleaved=0
00:02:48.940     AFEPortHandlers.cpp  01195  client 1 and port dir = 0,
intf = 4000 sample rates equal. returning.
00:02:48.940     AFEPortCustMsgHandler.cpp  00778   message SUCCESS
0
00:02:48.940          AudDevMgr.cpp  02368  ADM: As part of Rx
ADM_CMD_DEVICE_OPEN_V5, issuing CONNECT_COPP cmd
00:02:48.940        audproc_svc.cpp  00847  PCC15 audproc_svc: instance
0xF091F860 connecting to SvcID 0x   f0000
00:02:48.940        audproc_svc.cpp  00893  PCC15 audproc_svc: COPP
Connection Succeeded.
```

3. Connect the COPP to matrix Rx output port ID 0 with the following PCM configuration: **mono, 16 bit, 48K**.

```
00:02:48.940          AudDevMgr.cpp  02376  ADM: As part of Rx
ADM_CMD_DEVICE_OPEN_V5, issuing
CONNECT_MT_MX_OUT_PORT cmd
00:02:48.940          AudDevMgr.cpp  02400  ADM: As part of Rx
ADM_CMD_COPP_OPEN, # ch: 1, PTMode=0, NatMode=0,
Pull/Push Mode=1
00:02:48.940           MixerSvc.cpp  00348  MtMx #0: rcvd custom msg
1572866
00:02:48.940     MixerSvc_MsgHandlers.cpp  01276  MtMx #0: Processing cmd
1572866 (CfgOpPort) [port ID, port cfg] = [-1, 1]
00:02:48.940     MixerSvc_MsgHandlers.cpp  01344  MtMx #0 creating new
output port 0
00:02:48.940      MixerSvc_Util.cpp  05490  MtMx #0: [o/p port ID 0
num_chan, Bytes/sam, sample_rate] = [1, 2, 48000]
00:02:48.940      MixerSvc_Util.cpp  05490  MtMx #0: [o/p port ID 0
num_chan, Bytes/sam, sample_rate] = [1, 2, 48000]
00:02:48.940      MixerSvc_Util.cpp  05495  MtMx: Channel mapping = 3, 0,
0, 0,
```


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4. The ADM sets the COPP to the Run state.

   The input and output media formats are reconfigured to the correct state. The output PCM format is propagated to the downstream AFE.

   The following QXDM log shows the PCM format is set to mono, 16 bit, and 48000 sampling rate.

```
00:02:48.940         AudDevMgr.cpp  02424  ADM: As part of Rx
ADM_CMD_DEVICE_OPEN_V5, issuing  RUN_MT_MX_OUTPUT_PORT cmd
00:02:48.940         AudDevMgr.cpp  02435  ADM: As part of Rx
ADM_CMD_DEVICE_OPEN_V5, issuing RUN_COPP cmd
00:02:48.941       audproc_svc.cpp  01264  PCC15 audproc_svc: Run Begin
00:02:48.941       audproc_svc.cpp  01300  PCC15 audproc_svc: Run End,
sent ack
00:02:48.941         AudDevMgr.cpp  02607  ADM: Opened COPP [1]. COPP
requests 1 MCPS. LogID: 2
00:02:48.941    AudDevMgr_mmpm.cpp  00312  Sending ADM MMPM_Request_Ext
00:02:48.941    audproc_paramhandler.cpp  00055  PCC15: Setting media
format
00:02:48.941    audproc_paramhandler.cpp  00056  PCC15: Input media
format:
00:02:48.941    audproc_paramhandler.cpp  00261  PCC15: MediaFmt: Number
of channels: 1
00:02:48.941    audproc_paramhandler.cpp  00265  PCC15: MediaFmt:
Channel mapping: 3, 2, 0, 0, 0, 0, 0, 0
00:02:48.941    audproc_paramhandler.cpp  00266  PCC15: MediaFmt: Bits
per sample: 16
00:02:48.941    audproc_paramhandler.cpp  00267  PCC15: MediaFmt: Is
Interleaved: 0
00:02:48.941    audproc_paramhandler.cpp  00268  PCC15: MediaFmt: Is
Signed: 1
00:02:48.941    audproc_paramhandler.cpp  00269  PCC15: MediaFmt:
Sampling rate: 48000
00:02:48.942    audproc_appi_topo.cpp  00359  PCC15: reinit done, final
media type is as follows:
00:02:48.942    audproc_appi_topo.cpp  00360  PCC15: channels: 1
bitsPerSample: 16
00:02:48.942    audproc_appi_topo.cpp  00361  PCC15: sampleRate: 48000
isSigned: 1 isInterleaved: 0
00:02:48.942    audproc_appi_topo.cpp  00363  PCC15 audproc_svc: GenTopo
Change Mediatype End
00:02:48.942    audproc_paramhandler.cpp  00109  PCC15: Output media
format:
00:02:48.942    audproc_paramhandler.cpp  00261  PCC15: MediaFmt: Number
of channels: 1
00:02:48.942    audproc_paramhandler.cpp  00265  PCC15: MediaFmt:
Channel mapping: 3, 0, 0, 0, 0, 0, 0, 0
00:02:48.942    audproc_paramhandler.cpp  00266  PCC15: MediaFmt: Bits
per sample: 16
00:02:48.942    audproc_paramhandler.cpp  00267  PCC15: MediaFmt: Is
Interleaved: 0
00:02:48.942    audproc_paramhandler.cpp  00268  PCC15: MediaFmt: Is
Signed: 1
00:02:48.942    audproc_paramhandler.cpp  00269  PCC15: MediaFmt:
Sampling rate: 48000
```

5. The output PCM format is propagated to the downstream AFE, which is the playback use case.

```
00:02:48.942     audproc_msghandler.cpp  01071  PCC15 audproc_svc:
Sending MediaTypeMsg downstream.
00:02:48.943     AFEPortManager.cpp  02274  Media type msg
port_id=4000,client_id=1,sample_rate=48000,channels=1,bytes_p_ch=2,int_sa
mples_per_period=48
00:02:48.943     AFEPortHandlers.cpp  01195  client 1 and port dir = 0,
intf = 4000 sample rates equal. returning.
00:02:48.943     AudDevMgr_mmpm.cpp  00331  ADM MMPM_Request/Release_Ext
for total MIPS 31 is success
00:02:48.943     AudDevMgr_mmpm.cpp  00332  ADM MMPM_Request/Release_Ext
for AvgBW 31457280 is success
```

6. The ADM responds with the HLOS command, ADM_CMD_DEVICE_OPEN_V5, with status 0.

```
00:02:48.943     AudDevMgr_AprIf.cpp  00189  ADM APR ISR: status 0, ACK
opcode= 66345
```

### 2.3.2 Debugging check points for frequent issues

Verify the following:

- The topology ID is supported by the ADM with the correct channel mappings
- The ADM creates one dynamic thread correctly for the COPP thread
- The COPP is connected to the correct matrix output port
- The COPP is connected to the correct downstream AFE port HLOS use case
- The postprocessor module parameters are configured properly
- The CPU MIPS and system bus bandwidth are configured properly

Qualcomm Confidential – May Contain Trade Secrets 2020-07-02 04:17:10 PDT lu.wh@max.com

## 2.4 Audio matrix

The audio matrix is a dynamic service that acts as a bridge between device-management services and audio stream-management services. The audio matrix can:

- Receive PCM streams from multiple upstream services
- Split, reformat, and apply gains
- Mix and route gains to various downstream services
- Provide timestamps

Figure 2-4 illustrates the matrix routing in multiple sessions, and Figure 2-5 illustrates the routing sequence.

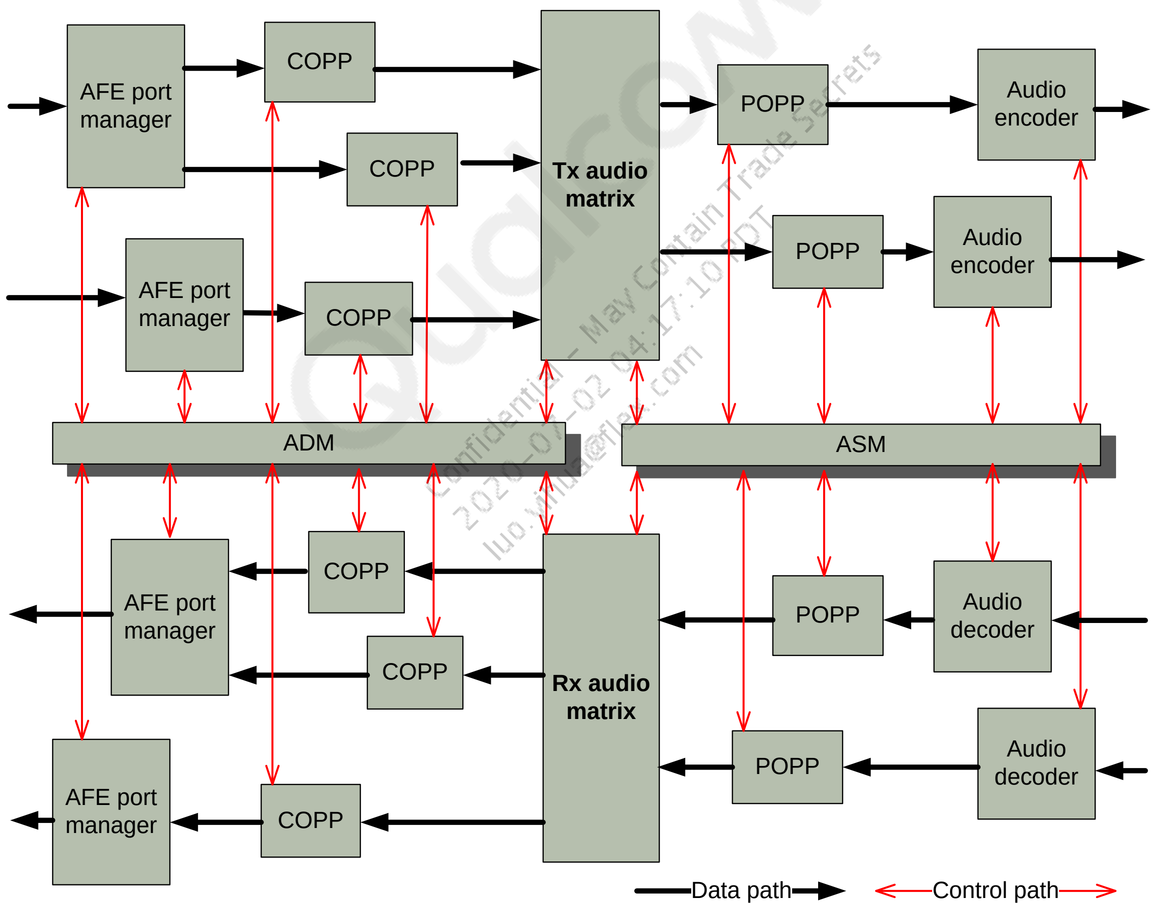


**Figure 2-4 Multiple session/device routing with the audio matrix**

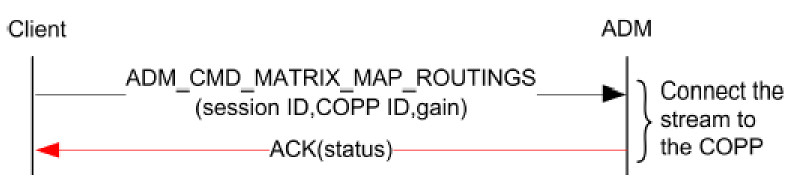


**Figure 2-5 Call flow: audio matrix routing**

Confidential – May Contain Trade Secrets
2020-07-02 04:17:10 PDT


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

### 2.4.1 Audio matrix routing setup

Refer to Figure 2-5 for these steps:

1. The ADM receive the HLOS command, ADM_CMD_MATRIX_MAP_ROUTINGS_V5, to rout Session ID 2 to COPP ID 1.

   **NOTE:** The command name in Figure 2-5 inherently includes the version (such as _V5).

```
00:02:48.943     AudDevMgr_AprIf.cpp  00189  ADM APR ISR: status 0, ACK
opcode= 66345
00:02:48.943     AudDevMgr_AprIf.cpp  00080  ADM_AprCallBackFct: At
0x4080063 APRCMD 66341 ADM rcvd payload
size 16
00:02:48.943          AudDevMgr.cpp  00337  ADM: rcvd APR msg [opcode,
dst port] = [66341, 0x63]
00:02:48.943          AudDevMgr.cpp  00994  ADM processing
ADM_CMD_MATRIX_MAP_ROUTINGS_V5, matrix ID = 0,
num sessions = 1
00:02:48.943          AudDevMgr.cpp  01015  ADM: Session ID = 2, num
copps = 1
00:02:48.943          AudDevMgr.cpp  02926  ADM_RSMM Entr: Mapping mask:
0 0 0
  00:02:48.943          AudDevMgr.cpp  02967  ADM_RSMM Exit: Mapping
mask: 0 0 0
```

2. External Session ID 2 (internal Session ID 1) is connected to matrix input port ID 0.

```
00:02:48.945          AudDevMgr.cpp  01059  ADM: Session ID translated to
MXAR i/p port ID = 0
00:02:48.945          AudDevMgr.cpp  02986  ADM_USMM Entr: Mapping mask:
0 0 0
00:02:48.945          AudDevMgr.cpp  03003  ADM_USMM: Inc coppID: 1 #
conn. sessions to 1
00:02:48.945          AudDevMgr.cpp  03007  ADM_USMM Exit: Mapping mask:
0 0 2
```

3. The COPP is connected to matrix output port ID 0.

```
00:02:48.945          AudDevMgr.cpp  01217  ADM: COPP ID 1 translating to
o/p port ID = 0
00:02:48.945          AudDevMgr.cpp  01238  ADM: RX Session ID [2] -->
Primary COPP ID [1]
00:02:48.945           MixerSvc.cpp  00348  MtMx #0: rcvd custom msg
1572872
```

4. The ADM responds with the HLOS command, ADM_CMD_MATRIX_MAP_ROUTINGS_V5, with result 0 indicating success.

```
00:02:48.945          AudDevMgr.cpp  01454  ADM:
ADM_CMD_MATRIX_MAP_ROUTINGS_V5 completed with success, result = 0
```

Qualcomm Confidential – May Contain Trade Secrets 2020-07-02 04:17:10 PDT luohuai@flex.com

### 2.4.2 Debugging check points for frequent issues

Verify the following:

- The stream is routed to the correct COPP per the HLOS use case
- The same session is routed to multiple devices
- The multiple session is routed to the same device

## 2.5 Audio Front End (AFE)

The AFE acts as an interface to the audio hardware from the aDSP. It mixes PCM samples from the audio and voice paths, converts the data to the format expected by the audio hardware (SRC/BIT covert/mixer), and matches rates to the audio clocks. It also routes the compressed data between hardware interfaces and aDSP internal clients.

Figure 2-6 illustrates the call flow for connecting the AFE port and enabling the AFE endpoint.

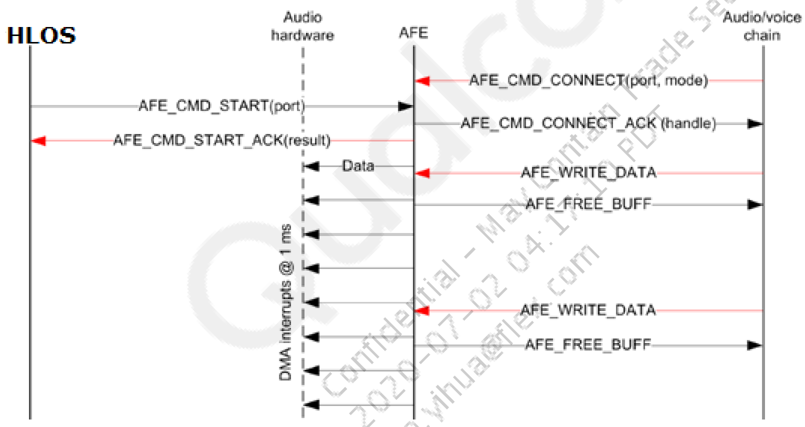


**Figure 2-6 Call flow: AFE startup**


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

## 2.6 Playback stream summary

To enable basic playback, the HLOS must perform the following steps:

1. Use the ADM to open a COPP session between the Rx device matrix and the AFE.
2. Use the ASM to open a write stream.
3. Use the AFE to turn on the endpoint DMA.
4. After performing steps 1 and 2, the client must also route the opened stream session to the opened device.
5. Once the playback session is established, the ASM/ADM moves the postprocessor and matrix input/output port from the Pause state to the Run state.

**NOTE:** Typically, the AFE port is started before ADM open and ADM matrix route to avoid intermittent silence data being inserted during device switch.

Figure 2-7 illustrates the topology for a typical tunneled playback session.

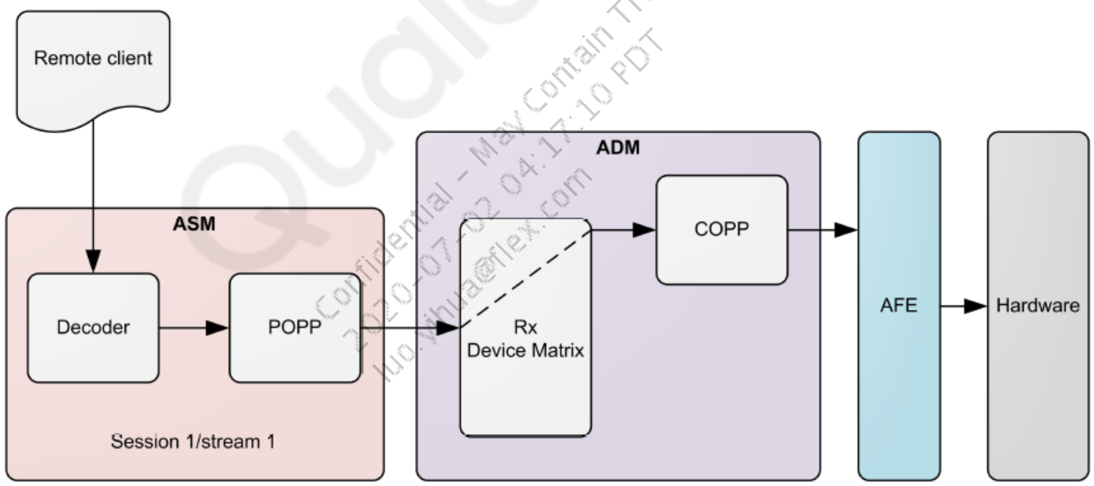


**Figure 2-7 Tunneled playback session topology**

Qualcomm Confidential – May Contain Trade Secrets 2020-07-02 04:17:10 PDT huo.yihua@flex.com

# 2.7 Flow control during stream playback

## 2.7.1 Stream pause

The ASM pauses the POPP, halting the flow at its input. The ASM also sends a pause to the matrix input port. While in the Pause state:

- The matrix continues to play out the remaining samples in its input queue to ensure that all PCM samples already post processed play out before resuming.
- Commands and parameters can be sent to the session and processed, and audible effects are perceived immediately after resuming due to the matrix pause behavior of playing out any samples that have already been post processed.

The log in the following steps shows the sequence of how the ASM handles a pause command from the HLOS.

1. The ASM receives an HLOS session pause command.

```
00:03:04.767     AudioStreamMgr_SessionCmdHandler.cpp  02056
AudioStreamMgr [2,1]: enter PAUSE command
handler
00:03:04.767     AudioStreamMgr_SessionCmdHandler.cpp  02105
AudioStreamMgr: leave PAUSE command handler2
00:03:04.768        audproc_svc.cpp  01176  PAA22 audproc_svc: Pause
Handler Begin
00:03:04.768     appi_softvolumecontrols.cpp  00675  APPI
SoftVolumeControls Get Soft Pause params, 30,0,0
00:03:04.768     appi_softvolumecontrols.cpp  00474  APPI
SoftVolumeControls Soft Pause Start
00:03:04.768     audproc_msghandler.cpp  00792  PAA22 audproc_svc: Soft
Pause started with 0
```

2. A soft pause is created to avoid a POP sound during the stream pause.

```
00:03:04.823     audproc_msghandler.cpp  00846  PAA22 audproc_svc: Soft
Pause Timer expired
00:03:04.823     audproc_msghandler.cpp  00871  PAA22 audproc_svc: Pause
Handler End with Timer expired, sent ack
00:03:04.823     AudioStreamMgr_SessionRespHandler.cpp  01379
AudioStreamMgr [2,1]: Pause ACK Tok 0xf11034c4,
Res 0x0
00:03:04.823     MixerSvc_MsgHandlers.cpp  00528  MtMx #0: Processing cmd
1572867 (Pause) [port ID, port dir] = [1, 1]
00:03:04.823     MixerSvc_MsgHandlers.cpp  02880  MtMx #0 cmd [1572867]
(Pause) handler: rcvd cmd in Active/Waiting
Active state [8]
00:03:04.823     MixerSvc_MsgHandlers.cpp  00548  MtMx #0: Leaving cmd
[1572867] (Pause) handler with status 0
```

3. The ASM responds with a message that the HLOS session has paused successfully.

```
00:03:04.823     AudioStreamMgr_SessionRespHandler.cpp  01379
AudioStreamMgr [2,1]: Pause ACK Tok 0xf11034f0,
Res 0x0
00:03:04.923     appi_softvolumecontrols.cpp  00483  APPI
SoftVolumeControls Soft Pause set ramp on resume
```

## 2.7.2 Stream flush

A stream flush during pause is the typical action during seek. A flush is acknowledged after all samples between the client and the device matrix are flushed, and all client buffers are returned.

- The ASM does not guarantee that an EOS marker in a data path, which is pending acknowledgment, is acknowledged when its stream is flushed.
- The client is responsible for tagging each EOS marker for each flush command.

ASM_STREAM_CMD_FLUSH can be called only when the stream's session is in the Pause state; otherwise, it returns an error.

Figure 2-8 illustrates pause, flush, and run during a playback session.

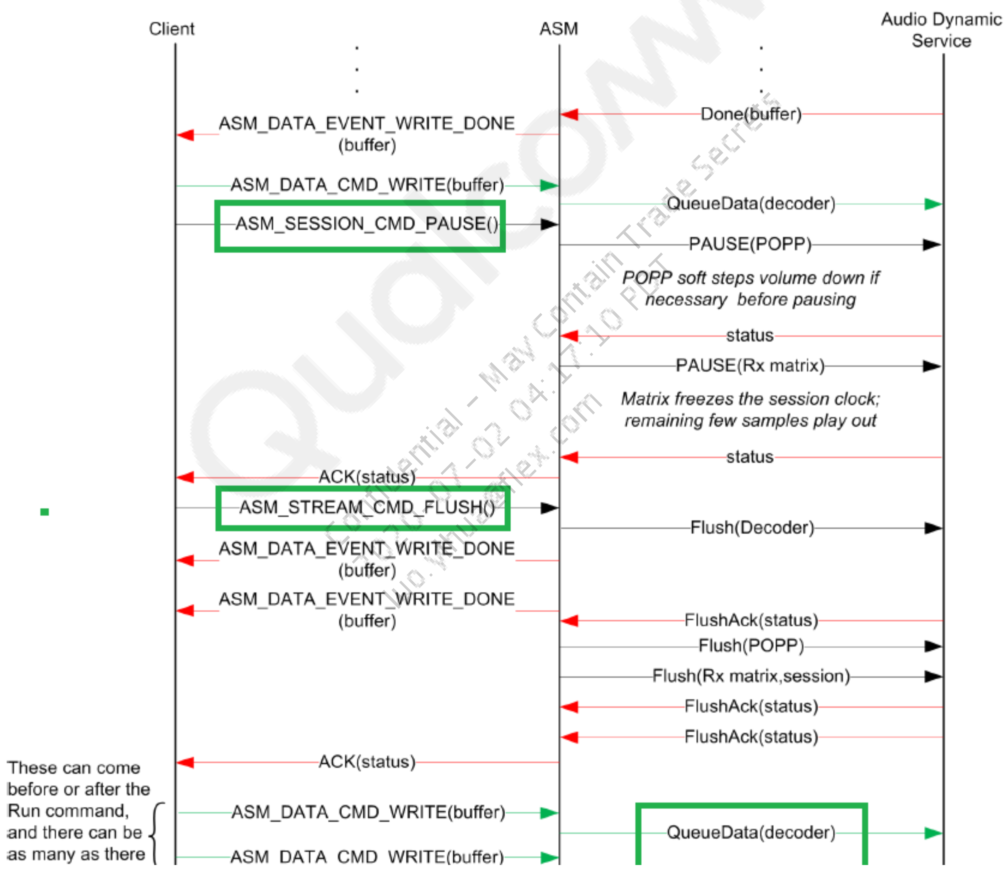


**Figure 2-8 Call flow: Pause, flush, and run sequence during playback**

Refer to Figure 2-8 for these steps:

1. The ASM receives an HLOS command, ASM_STREAM_CMD_FLUSH, with Session ID 2, Stream ID 1.

```
00:03:04.823     AudioStreamMgr_SessionCmdHandler.cpp  00112
AudioStreamMgr [2,1]: receive apr command
0x10bce at port [0x 301]
00:03:04.823     AudioStreamMgr_SessionCmdHandler.cpp  02139
AudioStreamMgr [2,1]: enter Flush Stream
command handler
00:03:04.823     AudioStreamMgr_Session.cpp  01087  AudioStreamMgr
[Session ID = 2, Stream ID = 1]:
Flushing/Pausing/Running decoder service
```

2. The ASM flushes the bitstream buffer Session ID 2's decoder.

```
00:03:04.823     AudioStreamMgr_Session.cpp  01141  AudioStreamMgr [2,1]:
Flushing Dec/Enc svc 0xf110330c
00:03:04.823        AudioDecSvc.cpp  01960  AudioDecSvc : Executing flush
Command, CurrentBitfield=0xa0000000
00:03:04.823     AudioStreamMgr_SessionCmdHandler.cpp  02183
AudioStreamMgr [2,1]: leave Flush command
Handler
00:03:04.825        AudioDecSvc.cpp  01980  AudioDecSvc : Done executing
flush Command,
CurrentBitfield=0xc0000000
00:03:04.825     AudioStreamMgr_SessionRespHandler.cpp  01260
AudioStreamMgr [Session ID = 2, Stream ID = 1]:
Flush ACK Tok 0xf110330c, Res 0x0
```

3. The ASM flushes the bitstream buffer to queue Session ID 2's POPP.

```
00:03:04.825     AudioStreamMgr_Session.cpp  00970  AudioStreamMgr [2,1]:
Flushing PP svc 0xf11034c4
00:03:04.825        audproc_svc.cpp  00749  PAA22 audproc_svc: Flush
Begin
00:03:04.825     audproc_appi_topo.cpp  00384  PAA22 audproc_svc: GenTopo
Reset Begin
00:03:04.827     audproc_appi_topo.cpp  00406  PAA22 audproc_svc: GenTopo
Reset End
00:03:04.827        audproc_svc.cpp  00796  PAA22 audproc_svc: Flush End,
sent ack
00:03:04.827     AudioStreamMgr_SessionRespHandler.cpp  01260
AudioStreamMgr [Session ID = 2, Stream ID = 1]:
lush ACK Tok 0xf11034c4, Res 0x0
```

4. The ASM flushes the bitstream buffer to queue the audio matrix.

```
00:03:04.827     AudioStreamMgr_DevIF.cpp  00405  AudioStreamMgr [Session
Id = 2,Stream Id = 1]: Flushing Matrix svc
0xf11034f0
00:03:04.827           MixerSvc.cpp  00348  MtMx #0: rcvd custom msg
1572869
```

```
00:03:04.827     MixerSvc_MsgHandlers.cpp  00618  MtMx #0: Processing cmd
1572869 (FLUSH) [port ID, port dir] = [1, 1]
00:03:04.827     MixerSvc_MsgHandlers.cpp  03044  MtMx #0 flush handler:
i/p port 1 reseting SessionTime to 0.
00:03:04.827     AudioStreamMgr.cpp  00356  Signal received = 80 !
00:03:04.827     MixerSvc_MsgHandlers.cpp  00638  MtMx #0: Leaving cmd
[1572869] (Flush) handler with status 0
```

5. The ASM responds with an HLOS command, ASM_STREAM_CMD_FLUSH, with status 0x0 indicating success.

```
00:03:04.827     AudioStreamMgr_SessionRespHandler.cpp  01260
AudioStreamMgr [Session ID = 2, Stream ID = 1]: Flush
ACK Tok 0xf11034f0, Res 0x0
00:03:04.827     AudioStreamMgr_AprIf.cpp  00405
AudioStreamMgr:Port=0x301: ISR status 0x0, Basic Ack: 0x10bce ,
[0x0,0]
```

### 2.7.3 End of Stream (EOS)

An EOS is acknowledged only after the final samples for the specified stream are rendered.

The ASM propagates an EOS marker through its data path and into the device matrix. The device matrix propagates received EOS markers to its output paths.

Eventually, the destination AFE ports acknowledge the EOS messages to the client after rendering all samples received before the EOS.

During EOS, the client gives the aDSP a unique client token each time to avoid EOS/flush race conditions (preferred method). The AFE's rendered event includes this client token, along with the original source and destination addresses.


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Figure 2-9 illustrates an EOS during typical playback.

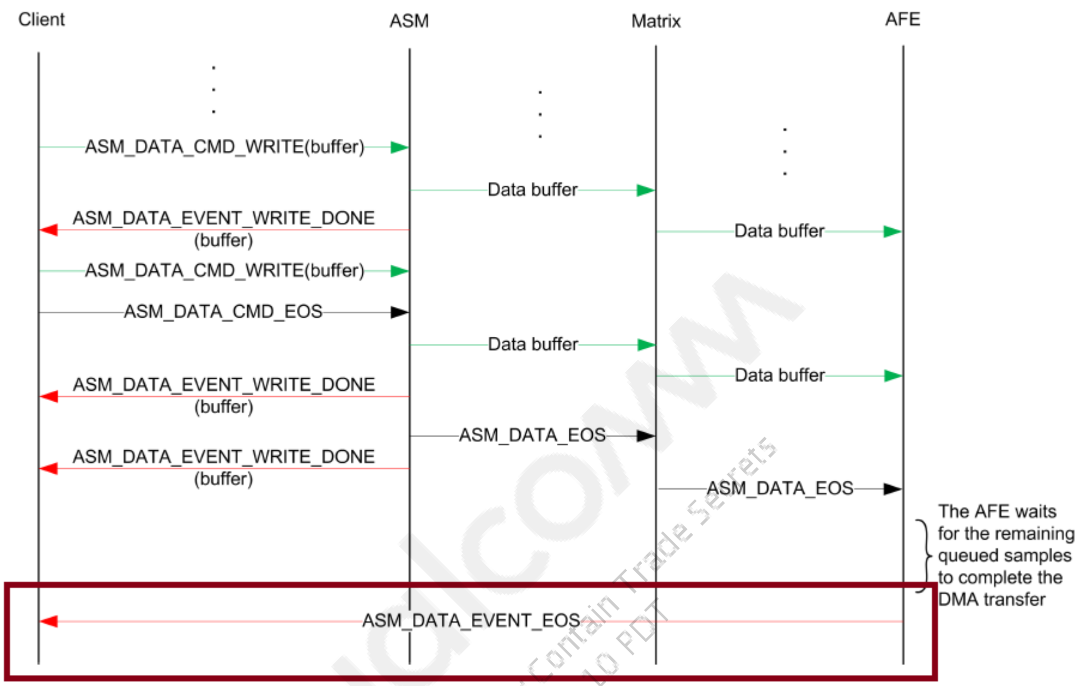


**Figure 2-9 Call flow: Typical playback EOS**

Confidential – May Contain Trade Secrets
2020-07-02 04:17:10 PDT

Refer to Figure 2-9 for these steps:

1. The ASM receives an HLOS EOS and propagates the command, ASM_DATA_CMD_EOS, to the decoder.

```
00:03:06.175        AudioDecSvc.cpp  00802  Decoder svc received EOS
cmd!!!!
00:03:06.195        AudioDecSvc.cpp  00942  DecSvc: sending EOS
downstream
```

2. The decoder sends the EOS command to the downstream matrix input port.

```
00:03:06.235     MixerSvc_InPortHandler.cpp  00330  MxAr #0 i/p port 0
rcvd EOS in ACTIVE/PAUSED state
00:03:06.235     MixerSvc_MsgHandlers.cpp  00359  MtMx #0 i/p port 0 rcvd
msg 12 (MtMx_MsgEos)
00:03:06.235     MixerSvc_MsgHandlers.cpp  00371  MtMx #0 i/p port 0 rcvd
regular EOS: EOSFormat=1
00:03:06.235     MixerSvc_MsgHandlers.cpp  00371  MtMx #0 i/p port 0 rcvd
regular EOS: EOSFormat=1
00:03:06.235     MixerSvc_MsgHandlers.cpp  00388  MtMx #0: i/p port 0
holding EOS, stop processing dataQ
00:03:06.235     MixerSvc_MsgHandlers.cpp  00460  MtMx #0: i/p port 0
trying to send EOS to connected active o/p port 0
00:03:06.235     MixerSvc_OutPortHandler.cpp  00757  MtMx #0 o/p port 0
sending EOS from i/p port 0 downstream, eos pending flag = 1
00:03:06.235     MixerSvc_OutPortHandler.cpp  00757  MtMx #0 o/p port 0
sending EOS from i/p port 0 downstream, eos pending flag = 1
```

```
00:03:06.235     MixerSvc_OutPortHandler.cpp  00781  MtMx #0 o/p port 0:
Sending EOS from i/p port 0 DS. EOSFormat: 1
00:03:06.235     MixerSvc_OutPortHandler.cpp  00781  MtMx #0 o/p port 0:
Sending EOS from i/p port 0 DS. EOSFormat: 1
```

3. The matrix mixer send the EOS command to the downstream AFE port.

   The AFE sends the EOS response instead of the ASM, which receives the EOS command from the HLOS.

```
00:03:06.256     AFEPortManager.cpp  02315  EoS msg
port_id=4000,client_id=1, EOSFormat:1
```

### 2.7.4 Get timestamps

The HLOS can periodically prompt for the session time to help adjust its progress bar or synchronize its local clock for audio/video synchronization with the ASM_SESSION_CMD_GET_SESSION_TIME command.

If the streams in the session do not have valid timestamps, only the samples delivered by the client and rendered cause the session clock to tick. Silence rendered during starvation does not advance the clock.

If the streams have valid timestamps, every tick of the rendering device increments the session clock.

The Rx device matrix gates the session render rate to synchronize it with the rendering device by delaying or dropping any received samples with non-current timestamps. Therefore, silence rendered during starvation advances the session clock.

During the Pause state, the session time does not advance.

When a flush is received, the session time reverts to zero.

The response to the ASM_SESSION_CMD_GET_SESSION_TIME command contains two values:

- The session time (s_t) of the approximate sample being rendered by the hardware.
- The absolute time at which that session time is actually sent to the hardware (for example, via DMA), which may be slightly in the future or in the past.

When the client receives these timestamps:

- If the client has access to an estimate of the delay between sending an audio sample to the connected hardware until the actual rendering, the client adds that delay estimate to the absolute time returned in ASM_SESSION_CMD_GET_SESSION_TIME. Let w_t be the resulting value of the absolute render time.
- To estimate the current session time being rendered in the connected audio hardware, the client reads the current wall clock (AV timer) (c_t) and then adds (c_t-w_t) to s_t.

In Single Stream Multiple Device (SSMD) scenarios, the device that is mentioned first in the ADM_CMD_MATRIX_MAP_ROUTINGS command is taken as the reference for AV sync purposes.


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Figure 2-10 illustrates the start sequence for audio/video synchronization.

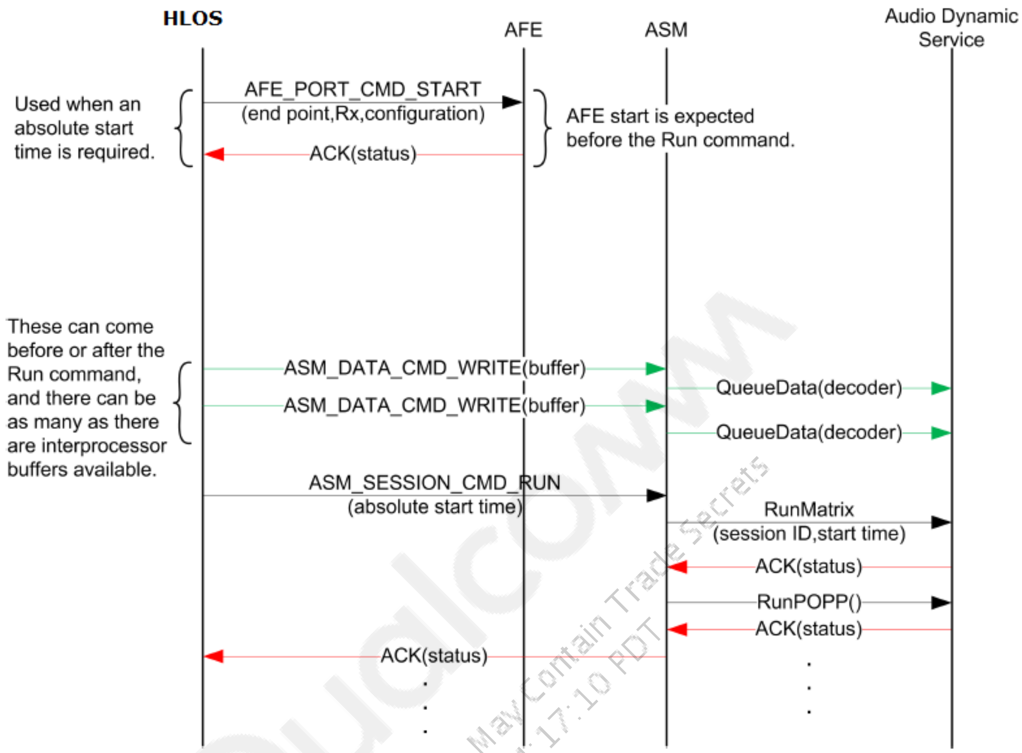


**Figure 2-10 Call flow: Start sequence for audio/video synchronization**

Figure 2-11 illustrates the start sequence for querying audio session time.

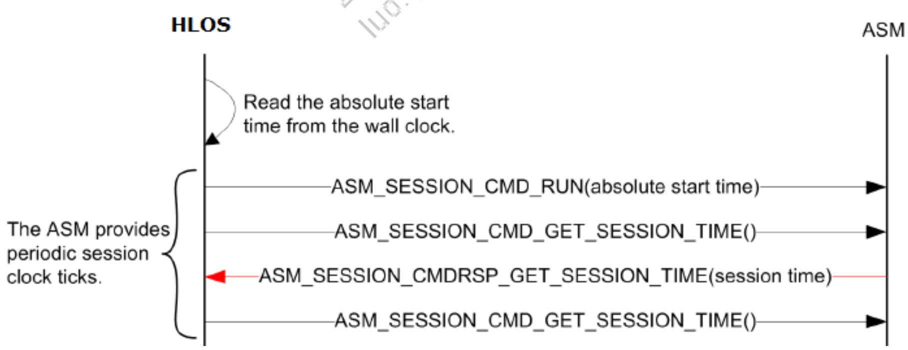


**Figure 2-11 Call flow: Start sequence for query audio session time**


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

### 2.7.5 Debugging check points for frequent issues

Verify the following:

- The HLOS sends pause/resume commands in the correct state
- The HLOS sends an EOS command when the ASM/ADM/AFE are still in the correct state, because the EOS command is propagated to the ADM/AFE, as described in Figure 2-10.
- The HLOS queries the session time when the ASM is in the Run state.
- The session time is reset to 0 due to a session flush command from the HLOS.

## 2.8 Record stream summary

To enable a basic recording session:

1. The HLOS uses the ADM to open and calibrate a COPP session between the AFE and Tx device matrix.
2. The HLOS uses the ASM to open a read stream.
3. The HLOS uses the AFE to turn on the endpoint DMA.
4. After performing steps 1 and 2, the client must also use the ADM_CMD_MATRIX_MAP_ROUTINGS command to route the opened COPP to the opened stream session.

**NOTE:** Steps 1 through 3 can be performed in any order to set up the device and stream session.

Qualcomm Confidential – May Contain Trade Secrets 2020-07-02 04:17:10 PDT luo.yihua@flex.com

Figure 2-12 illustrates the topology for a typical tunneled record session

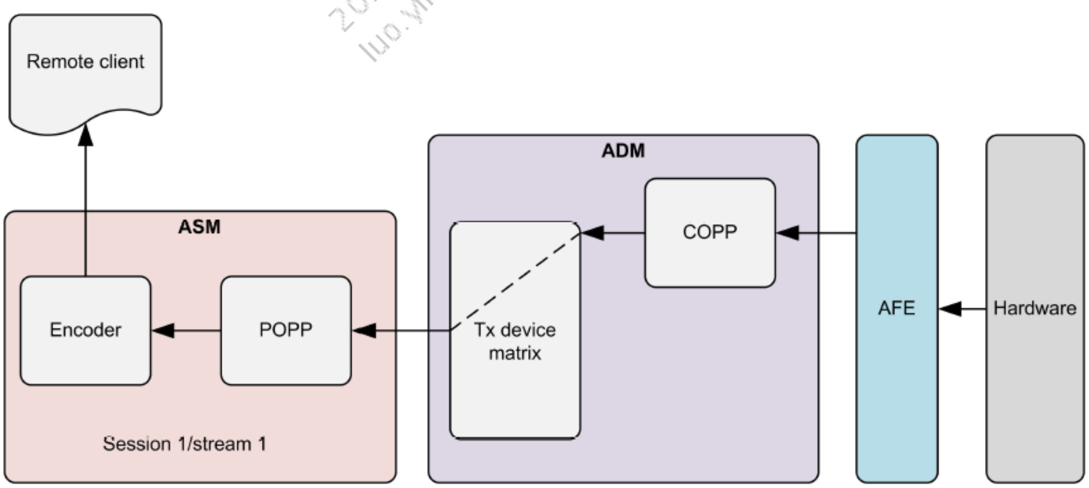


**Figure 2-12 Tunneled record session topology**

# 3 Audio Debugging Tips

Audio issues are typically complicated, and multiple points must be checked. This chapter explains how to approach debugging based on each use case.

**Table 3-1 FM use case**

| FM | Check point |
|---|---|
| Mute | Check the routing AIF configuration in AFE, such as the AFE Rx/Tx loopback port |
| Noise | Check the routing AIF configuration in AFE, such as the sampling rate |
| POP | N/A |
| Performance | N/A |
| Clipping | Check the correct gain configuration in the AFE loopback |

**Table 3-2 B2 A2DP use case**

| BT A2DP | Check point |
|---|---|
| Mute | Check if the AFE proxy session is created and is in the Run state |
| Noise | Check if any noise is found in ASM/ADM/AFE, or if an underrun is found in ADM/AFE |
| POP | Check if an underrun is found in AFE |
| Performance | N/A |
| Clipping | N/A |

**Table 3-3 Playback/recording use case**

| Playback | Check point |
|---|---|
| Mute | Check if session and device are created and connected correctly<br>Check if the stream is being muted by the HLOS<br>Check the AFE AIF port |
| Noise | Check if any noise is found in decoder/POPP/COPP |
| POP | Check if any underrun is found in decoder/POPP/matrix/COPP/AFE |
| Performance | Check if enough CPU MIPS and system bus bandwidth is being set |
| Clipping | Check the correct gain configuration in POPP/Matrix/COPP/AFE |

## 3.1 Log collection

Enable logs by following the steps described in *Hexagon Multimedia – Audio PCM/Bitstream Logging Through QXDM Professional* (80-N3470-4), and identify the issue in the ASM/ADM/AFE or HLOS. Figure 3-1 illustrates the Audio PCM logging tap point.

Qualcomm
Confidential – May Contain Trade Secrets
2020-07-02 04:17:10 PDT
luo.hhuadfuj.com

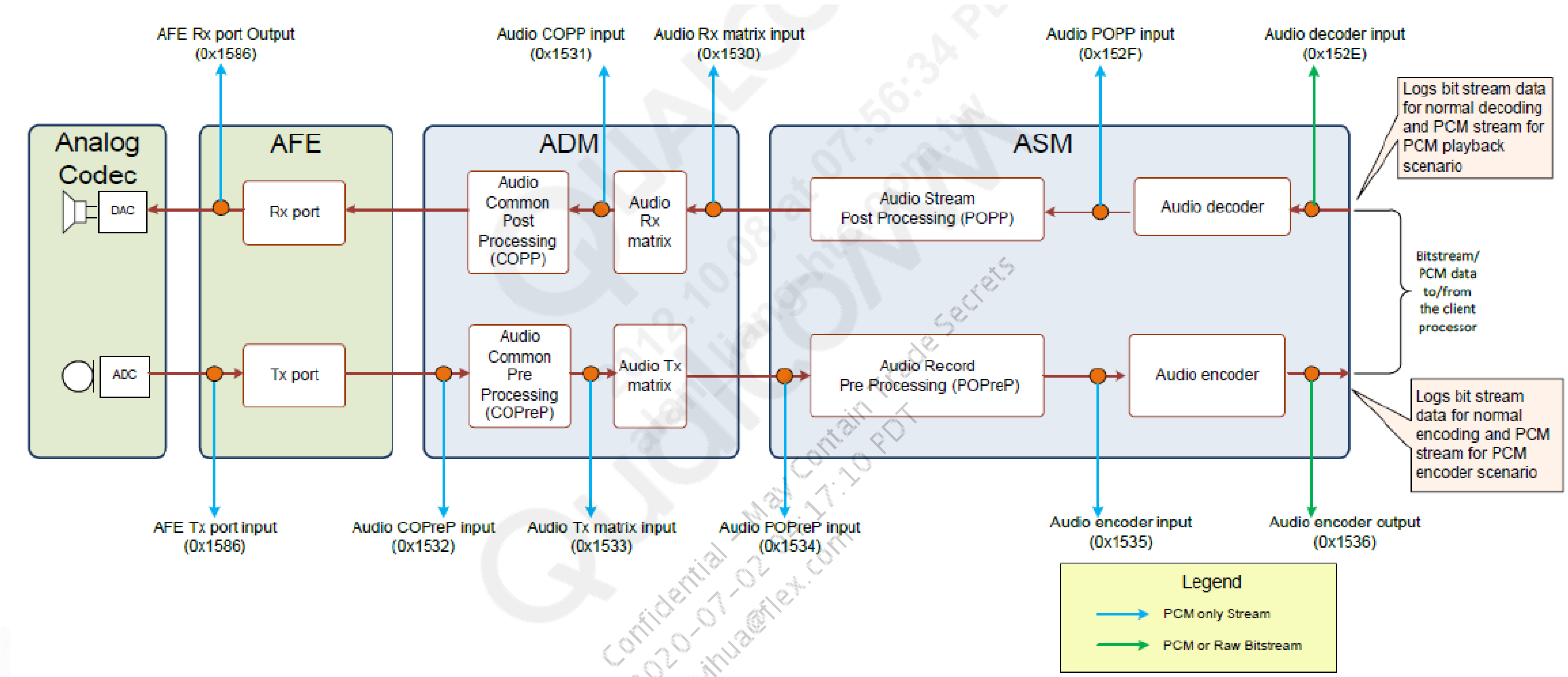


**Figure 3-1 Audio PCM logging tap point**

# 3.2 Sample log analysis for the issue

## 3.2.1 Sound quality

### 3.2.1.1 Serious noise

Serious noise is recorded if the recording is made after a voice call. This issue is not seen if a call is not made before FM recording (since power on).

Per the log shown below, matrix input and matrix output formats are the same, so the matrix should not introduce the noise.

Noise is not observed in AFE Tx port input (0x1586 with port ID 0x3005) or CoPreP Input (0x1532).

Noise is observed in 0x1533, matrix Tx input port, which means the noise is introduced by CoPreP.

MSIIR is enabled in CoPreP, so try disabling the module in the ACDB file.

To fix the issue, add a new device ID for the FM use case. To disable MSIIR for the FM recording use case, do not apply the same MSIIR coefficient.

1. The matrix input port sampling rate and channel number configuration are the same as for the output port.

```
00:25:41.561       ./src/MixerSvc_Util.cpp                5497       H
MtMx #1: [o/p port ID 0 num_chan, Bytes/s, sample_rate] = [2, 2, 48000]
00:25:41.561       ./src/MixerSvc_Util.cpp                3554       H
MtMx #1: O/p port 0 sending media_format msg downstream
00:25:41.561       ./src/MixerSvc_Util.cpp                5497       H
MtMx #1: [o/p port ID 0 num_chan, Bytes/s, sample_rate] = [2, 2, 48000]
00:25:41.561       ./src/MixerSvc_Util.cpp                5934       H
MtMx #1: i/p port ID 0, o/p port ID 0, ChannelMixerLib will not be used
(optimization). Cont w/o lib.
00:25:41.561       ./src/MixerSvc_Util.cpp                5937       H
MtMx i/p #ch 2, o/p #ch 2, i/p (ch[0],ch[1]) = (1,2), o/p (ch[0],ch[1]) =
(1,2)
```

2. MSIIR is enabled because the process check is set to 1.

```
00:25:41.511  ./src/appi_multistageiir.cpp       818         H
APPI MSIIR Process check, 1
00:25:41.511  ./src/appi_multistageiir.cpp       818         H
APPI MSIIR Process check, 1
00:25:41.512  ./src/appi_multistageiir.cpp       818         H
APPI MSIIR Process check, 1
00:25:41.560  ./src/appi_multistageiir.cpp       818         H
APPI MSIIR Process check, 1
```


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

## 3.2.2 No sound

### 3.2.2.1 Intermittent sound is lost when disabling an AFE proxy port

If intermittent sound is lost during the following sequence:

1. AFE proxy Tx port is enabled.
2. AFE proxy Rx port is enabled.
3. Start proxy port read with pcm_read().
4. Start tunnel playback is enabled.
5. Stop the proxy Tx port.
6. Stop the proxy Rx port.

In Step 6, observe the intermittent sound lost in the AFE port with a 10% out of 20% testing reproduction rate.

Underrun messages are printed because the Mi2S AFE port is open, and the COPP connected to Mi2s is closed and opened multiple times.

The AFE sends zeros when no AFE client is connected to it in time.

Explanation control sequence from the HLOS:

1. The Mi2s COPP is connected to the stream.
2. The COPP connected to Mi2S is opened after 400 milliseconds.
3. The Mi2s COPP is connected to the stream after 400 milliseconds.
4. A huge gap in the routing stream to device results in the AFE failing to get client data in time.

```
18:46:08.827        ./src/AFEPortAprHandler.cpp  00091  AFECmdDmaStart:
port_id=0x6, sample_rate=48000, nChannels=2, Gain=0x2000.
18:46:09.288        ./src/AudDevMgr.cpp  03144  ADM: As part of Rx
ADM_CMD_COPP_OPEN, # default ch: 2, EP1: 6
18:46:09.290        ./src/AudDevMgr.cpp  01044  ADM processing
ADM_CMD_MATRIX_MAP_ROUTINGS, matrix ID = 0, num sessions = 1
```

5. An AFE underrun message is printed if the AFE upstream client does not send PCM data in 10 ms.

```
18:46:09.288        ./src/AFEPortManager.cpp  01108  AfePort intf=6,
dir=0: Audio data not available in Rx path from client 0. CurrTime-
PrevTime = -1150186942 us. Num Underrun = 1!!
```

6. The COPP connected to Mi2S is closed, and then the COPP connected to Mi2S is opened after ~100 ms gap.

```
18:46:09.288        ./src/AFEPortManager.cpp  01108  AfePort intf=6,
dir=0: Audio data not available in Rx path from client 0. CurrTime-
PrevTime = -1150186942 us. Num Underrun = 1!!
```

Furthermore, matrix accumulation is driven by the primary output port. Because there is no data consumption on the primary port, the secondary port (which is connected to MI2s) is waiting for data accumulation.

- To avoid the gaps, close the complete device session related to the proxy Rx port immediately after closing the AFE proxy Rx port so that the primary output port is switched to the secondary port connected to Mi2S.
- The device enable sequence must be AFE Port Start > ADM Open > ADM Map Routing.
- The device disable sequence must be ADM COPP Close > AFE Port Stop.

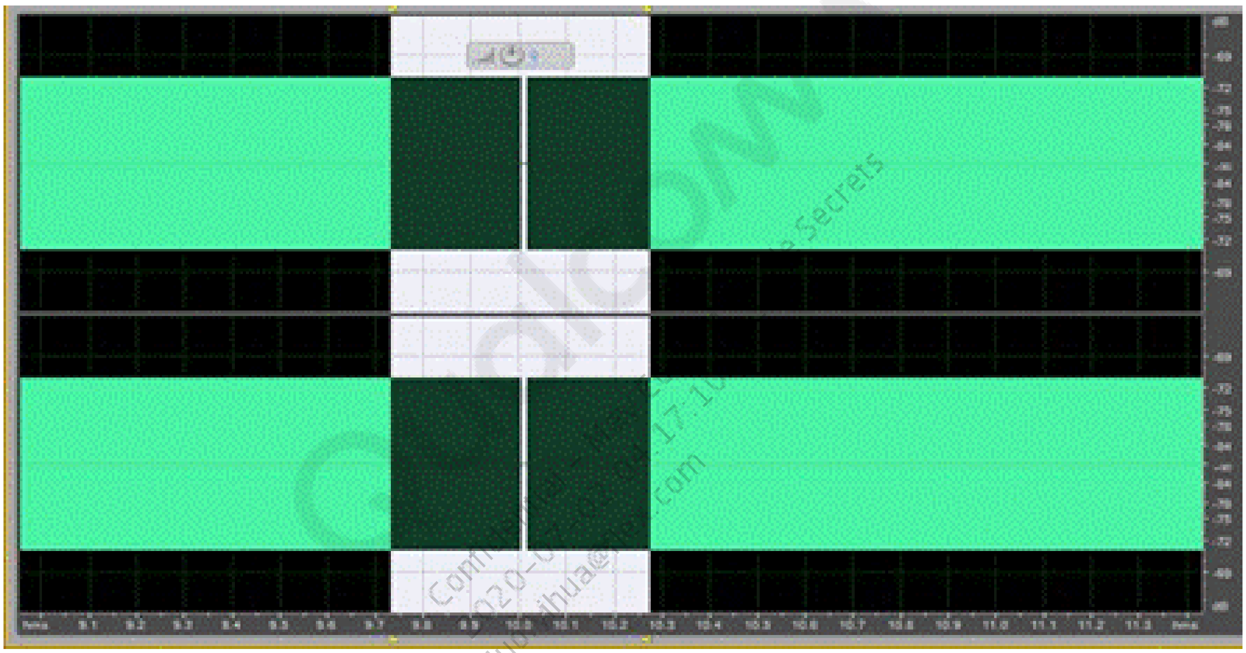

**Figure 3-2 0x1586 PCM log shows 10 milliseconds of silence in the PCM data**

#### 3.2.2.2 Audio mute with q6asm_mmapcallback error

From the HLOS log, continue observing the ASM memory map/unmap error.

```
<7>[ 3655.290868 / 06-21 11:19:05.208] msm-pcm-routing msm-pcm-routing:
reg 0
<7>[ 3655.262919 / 06-21 11:19:05.178] adm_callback: Basic callback
received, wake up.
<7>[ 3655.262968 / 06-21 11:19:05.178] send_adm_cal: Audvol cal sent for
port id: 0x4000, path 0
<7>[ 3655.262986 / 06-21 11:19:05.178] adm_matrix_map, copp_id: 0
<7>[ 3655.290868 / 06-21 11:19:05.208] msm-pcm-routing msm-pcm-routing:
reg 0
<7>[ 3655.290887 / 06-21 11:19:05.208] msm-pcm-routing msm-pcm-routing:
reg 0 val 1
<3>[ 3665.079578 / 06-21 11:19:15.008] q6asm_mmapcallback: cmd = 0x10d94
returned error = 0x2 sid:1
<7>[ 3665.100159 / 06-21 11:19:15.028] adm_close port_id=0x4000 index 15
perf_mode: 0
<7>[ 3665.100172 / 06-21 11:19:15.028] adm_close:Closing ADM: perf_mode:
0
<7>[ 3665.100184 / 06-21 11:19:15.028] adm_close:coppid 0 portid=0x4000
index=15 coppcnt=0
<7>[ 3665.106072 / 06-21 11:19:15.028] adm_callback_debug_print: code =
0x110e8 PL#0[10327], PL#1[0], size = 8
<7>[ 3665.106095 / 06-21 11:19:15.028] adm_callback: Basic callback
received, wake up.
<7>[ 3665.106132 / 06-21 11:19:15.028] adm_close: remove adm device from
rtac
<7>[ 3665.131157 / 06-21 11:19:15.048] msm-pcm-routing msm-pcm-routing:
reg 0
```

From the QXDM log, the memory map failure is due to the same physical address being mapped in a previous session.

```
01:28:18.078       76   AudioStreamMgr_SysAprCmdHandler.cpp
AudioStreamMgr: Processing ASM_CMD_SHARED_MEM_MAP_REGIONS Command
01:28:18.078      801   qurt_elite_memorymap.cpp   qurt_elite_memorymap
qurt_elite_memorymap_region_create qurt_mem_region_create fails,
qurt_mem_result:(2)
01:28:18.078      369   qurt_elite_memorymap.cpp   qurt_elite_memorymap
Create shared mem region Failed (Status,0x1)
01:28:18.078      415   qurt_elite_memorymap.cpp   qurt_elite_memorymap
qurt_elite_memorymap_shm_phymem_map failed
01:28:18.078      348   EliteMem_Util.cpp   EliteMem: Failed to map the
physical memory, error code is 0x1
```

For instructions on how to load an aDSP dump, refer to the *Hexagon Multimedia: Elite Audio Postprocessor API Interface Specification* for your build (see Section A.1). Per the aDSP dump/APR log, physical address 7E005000 was mapped for an MVM voice use case. The ASM runs the error because the same physical address must not be mapped twice.

```
DYNAMICALLY ALLOCATED MEMORY REGION LIST
  -----------------------------------------------------------------
DYNAMICALLY ALLOCATED MEMORY REGION LIST
  -----------------------------------------------------------------
Virtual_addr  Physical_addr    Size     Owner thread    ASID

FE0A1000      7E068000      0x1000        00000061       0
FE180000      7E005000      0x5A000        00000043       0/// 0x43 is
MVM thrad
FE0FF000      7E066000      0x1000        0000005B       0

 00000000 |  03 00 01 00   00 00 00 00  |  00 50 00 7E   00 00 00 00
0 00 00  //// 00 50 00 7E is address 0x7E005000
```

The following notes pertain to the previous log:

- When the media server crashes and relaunches, the ACDB loader can acquire ion memory again to hold calibration data.
- However, the native voice driver is not aware of the change of ion buffer for calibration data. Therefore, it never performs a memory unmap. The actual memory was deallocated because when the media server crashed, the ion file descriptor is closed, which results in an original ACDB ion buffer.

Qualcomm
Confidential - May Contain Trade Secrets
2020-07-02 04:17:10 PDT
luo.yihua@flex.com

### 3.2.2.3 No sound in MP3, but can hear system notification tone

Per the description of the issue, there is no problem with PCM playback (aka the touch sound use case).

1. The ASM received an HLOS stream open command and started to create two dynamic threads for the decoder and postprocessor.

```
00:06:36.567   AudioStreamMgr_AprIf.cpp   89   L   AudioStreamMgr: Rec
cmd 0x10db3 at [Addr=0x407, Port=0x101]
00:06:36.567   AudioStreamMgr.cpp   357   L   Signal received = 4 !
```

2. The ASM started to create a topology, but failed during the module initialization process.

```
00:06:36.568   audproc_appi_init.cpp   159   H   PAA0C audproc_svc:
Initializing topo begin
00:06:36.568   audproc_appi_topo.cpp   1406   H   Requesting 3840 bytes
default memory for re-allocating temp buffers
00:06:36.568   appi_downmix.cpp   94   H   APPI Downmix Getsize done,
requires 280 bytes
00:06:36.568   appi_genericresampler24src.c   93   H   APPI Resampler
Getsize done, requires 152 bytes
00:06:36.568   appi_softvolumecontrols.cpp   102   H   APPI
SoftVolumeControls Getsize done, requires 1224 bytes
00:06:36.568   appi_combopp.cpp   106   H   APPI ComboPP Getsize done,
requires 4680 bytes
00:06:36.568   audproc_appi_init.cpp   473   F   PAA0C audproc_svc:
Module with ID 0x10C39 failed in getsize.
```

3. Again, the ASM started to create a topology, but failed during the module initialization process.

```
00:06:36.568   audproc_appi_topo.cpp   574   H   PAA0C audproc_svc:
GenTopo Destroy Begin.
00:06:36.568   audproc_appi_topo.cpp   629   H   PAA0C audproc_svc:
GenTopo Destroy End, now freeing GenTopo.
00:06:36.568   audproc_svc.cpp   332   F   PAA0C audproc_svc: Failed to
initialize features!
00:06:36.568   audproc_svc.cpp   449   H   PAA0C audproc_svc: Destroy Svc
Begin
00:06:36.568   audproc_svc.cpp   484   H   audproc_svc: Destroy Svc End
00:06:36.568   AudioStreamMgr_Util.cpp   343   E   AudioStreamMgr [0,1]:
Failed to create Post=proc service
00:06:36.568   AudioStreamMgr_Session.cpp   1252   E   AudioStreamMgr
[0,1]: destroying module 1376256 due to unexpected error
00:06:36.568   AudioStreamMgr_Session.cpp   1277   H   AudioStreamMgr
[0,1]: send destroy to Svc [0x150000]
00:06:36.568   AudioStreamMgr_Session.cpp   1295   E   AudioStreamMgr
[0,1]: module 1376256 is destroyed due to unexpected error
```


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4. The ASM responded that the HLOS stream open command failed.

```
00:06:36.568   AudioStreamMgr_AprIf.cpp   410   L
AudioStreamMgr:Port=0x101: ISR status 0x3, Basic Ack: 0x10db3 , [0x0,0]
```

### Suggestion (from the DSP)

Remove the unsupported module from the ACDB.

#### 3.2.2.4 No sound when connecting an HDMI cable to a specific TV

During HDCP, the HDMI receiver on the TV side reported support for up to six channels.

1. The ADM received an HLOS device open command.

```
00:09:02.976          AudDevMgr.cpp  01803  ADM processing
ADM_CMD_DEVICE_OPEN_V5
00:09:02.976          AudDevMgr.cpp  01879  ADM: device_perf_mode is [1]
& COPP Buffer duration in ms is [1]
```

2. The channel mapping is wrong; you can see that only two valid channel are mapped. A zero means no mapping.

```
00:09:02.976          AudDevMgr.cpp  02190  ADM: ADM_CMD_DEVICE_OPEN_V5,
dev_num_ch = 6, dev_ch_mapping = 1, 2, 0, 0, 0, 0, 0, 0
00:09:02.976          AudDevMgr.cpp  02220  ADM: Invalid device channel
mapping 0 for channel #3
```

### Suggestion

The HLOS should provide channel mapping during device open.


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

#### 3.2.2.5 Audio has no sound when switching immediately to speaker after picking up a call

In the HLOS log, you can see the following events: stream close timeout, memory map error, and invalid session.

```
<3>[ 1388.791330,0] timeout. waited for response opcode[0x10bcd] //
ASM_STREAM_CMD_CLOSE
<3>[ 1388.791561,0] q6asm_mmapcallback: cmd = 0x10d94 returned error =
0xa sid:2
<3>[ 1388.792133,0] q6asm_callback:Session ID is invalid, session =
1802201963
<3>[ 1388.792235,0] q6asm_callback:Session ID is invalid, session =
1802201963
```

The ASM response memory map failed because the same address was mapped in a previous session.

```
00:06:44.588   qurt_elite_memorymap.cpp   802   E   qurt_elite_memorymap
qurt_elite_memorymap_region_create qurt_mem_region_create fails,
qurt_mem_result:(2)
00:06:44.588   qurt_elite_memorymap.cpp   370   F   qurt_elite_memorymap
Create shared mem region Failed (Status,0x1)
00:06:44.588   qurt_elite_memorymap.cpp   416   H   qurt_elite_memorymap
qurt_elite_memorymap_shm_phymem_map failed
00:06:44.588   EliteMem_Util.cpp   344   E   EliteMem: Failed to map the
physical memory, error code is 0x1
```

Observe the same error because the HLOS and aDSP are out of sync in the memory map.

```
00:06:44.932   AudioDecSvc.cpp   744   E   AudioDecSvc:Phy to Virt
Failed(paddr,vaddr)-->(00,0)
00:06:44.932   EliteMem_Util.cpp   93   E   elite_mem_map_get_shm_attrib
failed, & input is [mapclient, maphandle, PhyMsw,PhyLsw]=[f08bc040,
0,       0,1060605952],error 0x1
00:06:44.932   EliteMem_Util.cpp   100   E   elite_mem_map_get_shm_attrib
failed & sharednode is [PhyMsw,PhyLsw,Virt,Size]=[       0,       0,
0,0],error 0x1
00:06:44.932   AudioDecSvc.cpp   744   E   AudioDecSvc:Phy to Virt
Failed(paddr,vaddr)-->(00,0)
```

### Suggestion

The HLOS must ensure that the same physical address is not mapped more than once without being unmapped.

### 3.2.2.6 Audio has no sound and SLIMbus warning log is produced

Observe that the SLIMbus warning log is produced because an LPASS AHB is due to system bus bandwidth shortage.

```
00:07:04.768   SlimBus.c   2209   H   [WARN] disabled port interrupt due to
overflow/underflow (client: 0xF08B9640) (resource: 0x16200) (port: 0)
00:07:04.770   SlimBusBamLib.c   1481   H   [WARN] enabled port interrupt
after overflow/underflow resolved (client: 0xF08B9640) (resource: 0x16200)
(port: 0)
00:07:04.770   SlimBus.c   2209   H   [WARN] disabled port interrupt due to
overflow/underflow (client: 0xF08B9640) (resource: 0x16200) (port: 0)
00:07:04.770   SlimBusBamLib.c   1481   H   [WARN] enabled port interrupt
after overflow/underflow resolved (client: 0xF08B9640) (resource: 0x16200)
(port: 0)
00:07:04.770   SlimBus.c   2209   H   [WARN] disabled port interrupt due to
overflow/underflow (client: 0xF08B9640) (resource: 0x16200) (port: 0)
00:07:04.771   SlimBusBamLib.c   1481   H   [WARN] enabled port interrupt
after overflow/underflow resolved (client: 0xF08B9640) (resource: 0x16200)
(port: 0)
00:07:04.771   SlimBus.c   2209   H   [WARN] disabled port interrupt due to
overflow/underflow (client: 0xF08B9640) (resource: 0x16200) (port: 0)
00:07:04.772   SlimBusBamLib.c   1481   H   [WARN] enabled port interrupt
after overflow/underflow resolved (client: 0xF08B9640) (resource: 0x16200)
(port: 0)
00:07:04.772   SlimBus.c   2209   H   [WARN] disabled port interrupt due to
overflow/underflow (client: 0xF08B9640) (resource: 0x16200) (port: 0)
00:07:04.773   SlimBusBamLib.c   1481   H   [WARN] enabled port interrupt
after overflow/underflow resolved (client: 0xF08B9640) (resource: 0x16200)
(port: 0)
00:07:04.773   SlimBus.c   2209   H   [WARN] disabled port interrupt due to
overflow/underflow (client: 0xF08B9640) (resource: 0x16200) (port: 0)
00:07:04.775   SlimBusBamLib.c   1481   H   [WARN] enabled port interrupt
after overflow/underflow resolved (client: 0xF08B9640) (resource: 0x16200)
(port: 0)
00:07:04.775   SlimBus.c   2209   H   [WARN] disabled port interrupt due to
overflow/underflow (client: 0xF08B9640) (resource: 0x16200) (port: 0)
00:07:04.775   SlimBusBamLib.c   1481   H   [WARN] enabled port interrupt
after overflow/underflow resolved (client: 0xF08B9640) (resource: 0x16200)
(port: 0)
00:07:04.775   SlimBus.c   2209   H   [WARN] disabled port interrupt due to
overflow/underflow (client: 0xF08B9640) (resource: 0x16200) (port: 0)
```

## Suggestion

The AFE driver should vote enough AHB bandwidth for the SLIMbus use case.


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

## 3.2.3 Logical errors

### 3.2.3.1 ASM memory map callback error

The ASM_CMD_SHARED_MEM_UNMAP_REGIONS command failed because the aDSP is still in the process of closing a stream.

1. The ASM received an ASM_STREAM_CMD_CLOSE command from the HLOS and started to destroy the decoder and postprocessor.

```
00:24:31.828       85   AudioStreamMgr_AprIf.cpp   AudioStreamMgr: Rec
cmd 0x10bcd at [Addr=0x407, Port=0x101]
00:24:31.828      356   AudioStreamMgr.cpp   Signal received = 4 !
00:24:31.828      112   AudioStreamMgr_SessionCmdHandler.cpp
AudioStreamMgr [0,1]: receive apr command 0x10bcd at port [0x 101]
```

2. The ASM disconnected the decoder service successfully.

```
00:24:31.828      854   AudioStreamMgr_Session.cpp   AudioStreamMgr
[0,1]: disconnecting Svc [0x150000,0x6034] to Svc [0xf0000,0x0]!
00:24:31.828     1901   AudioDecSvc.cpp   Decoder service disconnecting
to down stream service. unCurrentBitfield=0xa0000000
00:24:31.828     1915   AudioDecSvc.cpp   Disonnection Succeeded.
00:24:31.828      356   AudioStreamMgr.cpp   Signal received = 8 !
00:24:31.828     1930   AudioDecSvc.cpp   Decoder service done
disconnecting to down stream service. unCurrentBitfield=0x80000000
00:24:31.828      423   AudioStreamMgr_SessionRespHandler.cpp
AudioStreamMgr [0,1]:svc ID [0x150000] disconnected Res0x0
00:24:31.828      421   AudDevMgr.cpp   ADM: rcvd custom msg [opcode] =
[3145732]
00:24:31.828     2926   AudDevMgr.cpp   ADM_RSMM Entr: Mapping mask: 0 0
1
00:24:31.828     2936   AudDevMgr.cpp   ADM_RSMM: Dec CoppID: 0 # conn.
sessions to 0
00:24:31.828     2967   AudDevMgr.cpp   ADM_RSMM Exit: Mapping mask: 0 0
0
00:24:31.828      356   AudioStreamMgr.cpp   Signal received = 8 !
00:24:31.828      520   AudioStreamMgr_SessionRespHandler.cpp   Entered
disconnect handler !!!
00:24:31.828      668   AudioStreamMgr_Session.cpp   AudioStreamMgr
[0,1]: Sent destroy to Svc [0x150000]
```


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

3. The HLOS sent a memory unmap command, but the ASM is still processing the stream close command. The unmap failed because the decoder/postprocessor still refers to the memory region.

```
00:24:36.828       85   AudioStreamMgr_AprIf.cpp   AudioStreamMgr: Rec
cmd 0x10d94 at [Addr=0x407, Port=0x0]
00:24:36.828      356   AudioStreamMgr.cpp   Signal received = 1 !
00:24:36.828       83   AudioStreamMgr_SysAprCmdHandler.cpp
AudioStreamMgr: Processing ASM_CMD_SHARED_MEM_UNMAP_REGIONS Command
00:24:36.828      936   qurt_elite_memorymap.cpp   qurt_elite_memorymap
non zero ref count detected, cannot unmap the node  (client
token,mmhandle,ref count)->(0xf09d2aa0,0xf0a178c8,0x1)
00:24:36.828      420   EliteMem_Util.cpp   EliteMem: Failed to unmap the
phyiscal memory, error code is 0xa
```

### Suggestion

The HLOS should wait for the response from the ASM_STREAM_CMD_CLOSE (0x10bcd) command, and then send the ASM_CMD_SHARED_MEM_UNMAP_REGIONS command.

## 3.2.3.2 EOS failed because the aDSP is not in the run state

The following message is in the HLOS log:

```
[  382.295259 / 08-21 10:59:41.433]/CPU:0 msm_pcm_playback_close: CMD_EOS
failed, cmd_pending 0x8
```

1. The AFE proxy Rx port was disabled and enabled, and HLOS sent the EOS command later. The AFE proxy Rx port responded with the EOS result to the HLOS after the proxy port start.

```
00:06:00.580      328   AFEPortAprHandler.cpp   AFE DMA STOP CMD
Response: port id: 0x2000, result: 0x0
00:06:00.590      599   AFEDeviceDriver.cpp   AFECmdDmaStart: Dma Start
Success: dir: 0, intf: 0x2000
00:06:01.890      791   AudioDecSvc.cpp   Decoder svc received EOS
cmd!!!!
00:06:01.890      931   AudioDecSvc.cpp   DecSvc: sending EOS downstream
00:06:01.891      236   audproc_msghandler.cpp   PAA99 audproc_svc: EOS
Delivering out f091bf60
00:06:01.900     2315   AFEPortManager.cpp   EoS msg
port_id=2000,client_id=0, EOSFormat:1
```

2. The AFE proxy Rx port is disabled, and the AFE SLIMbus Rx port is enabled and disabled.

```
00:06:01.917      328   AFEPortAprHandler.cpp   AFE DMA STOP CMD
Response: port id: 0x2000, result: 0x0
00:06:08.852      599   AFEDeviceDriver.cpp   AFECmdDmaStart: Dma Start
Success: dir: 0, intf: 0x4000
00:06:11.223      328   AFEPortAprHandler.cpp   AFE DMA STOP CMD
Response: port id: 0x4000, result: 0x0
```


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

3. The HLOS sent the EOS command to the aDSP. The aDSP failed to respond at this point because there are still unfinished data buffers that are part of dataQ and are yet to be serviced.

   Before the EOS is issued to the stream session, the device session is closed and reopened with endpoint 0x2000 (RX_proxy port).

```
00:06:21.206       85   AudioStreamMgr_AprIf.cpp   AudioStreamMgr: Rec
cmd 0x10bdb at [Addr=0x407, Port=0x101]
```

### Sequence for processing an EOS command in the aDSP

The EOS is acknowledged only after the final samples for the specified stream are rendered. To achieve this:

1. The ASM propagates an EOS marker through its data path and into the device matrix.
2. The device matrix propagates received EOS markers to its output paths.
3. Eventually, the destination AFE ports acknowledge the EOS messages to the client (by raising ASM_DATA_EVENT_RENDERED_EOS events) after rendering all samples received before the EOS.
4. The client can handle the scenario of the AFE port acknowledging the EOS message to a closed stream if the client closes the stream before receiving ASM_DATA_EVENT_RENDERED_EOS.
5. During EOS, the client gives the aDSP a unique client token each time to avoid EOS/flush race conditions (preferred method).
6. The AFE's rendered event includes this client token, along with the original source and destination addresses.

In this situation:

- The EOS has been issued to the stream, and the decoder processes it after the existing data buffers are completely rendered downstream.
- To do so, the entire chain (decoder->matrix->device) must be in the Run state. But, as the log shows, the device is attached to the stream, but it is in the Run state.
- Therefore, there is no way the existing data buffers are consumed at the decoder service. Because the AFE port is in the Stop state, the chain is stalled, and the EOS buffer is not processed at the decoder, along with data buffers that have been issued before the EOS.

### Suggestion

The HLOS must ensure that the entire chain (decoder > matrix > device > AFE) is in the Run state before sending an EOS command.

Confidential – May Contain Trade Secrets
2020-07-03 04:10 PDT
luyihua@flex.com

#### 3.2.3.3 Audio mute/progress bar has stopped

The audio mute/progress bar has stopped during the following sequence:

1. Play music in LPA mode over a speaker.
2. Insert a headset.
3. In 10 to 20% of instances, music playback and the progress bar on the music app stop. The app behaves as though it is still playing the song but no output is heard.

The log does not show `buf_done_cnt is 7484`, which implies the buffer is still in the downstream decoder.

The issue is due to the shortage of LPASS AHB, which resulted in the AFE/SLIMbus failing to send data.

```
02:00:27.401   AudioDecSvc.cpp   639   H   buf_recv_cnt is 7482
02:00:27.406   AudioDecSvc_Util.cpp   1047   H   buf_done_cnt is 7482
02:00:27.406   AudioDecSvc_Util.cpp   1047   H   buf_done_cnt is 6781
02:00:27.406   AudioDecSvc.cpp   639   H   buf_recv_cnt is 7483
02:00:27.406   AudioDecSvc.cpp   639   H   buf_recv_cnt is 6782
02:00:27.411   AudioDecSvc_Util.cpp   1047   H   buf_done_cnt is 7483
02:00:27.411   AudioDecSvc.cpp   639   H   buf_recv_cnt is 7484
```

#### 3.2.3.4 Wrong music during playback session time

The wrong music is played during the following sequence:

1. Music playback is in LPA mode over the speaker.
2. Press the home screen, and make a mobile-originated call.
3. The music pauses at Time Y seconds.
4. When the call is ended, the music resumes.
5. Go to the Music app to check the progress bar.
6. The progress bar does not show Y seconds because the music does not resume from the time when it was paused.

The session time is reset to 0 because of a flush command. In this case, the HLOS should keep the last session time in Step 3, and accumulate the session time in Step 4 if the flush command was issued between Step 3 and Step 4.

```
01:16:47.263  AudioStreamMgr_SessionCmdHandler.cpp        2060       M
AudioStreamMgr [2,1]: enter Flush Stream command handler
01:18:19.158  MixerSvc_MsgHandlers.cpp      2881       H             MtMx
#0 flush handler: i/p port 1 reseting SessionTime to 0.
01:18:23.940  AudioStreamMgr_SessionCmdHandler.cpp        2060       M
AudioStreamMgr [1,1]: enter Flush Stream command handler
01:18:23.945  MixerSvc_MsgHandlers.cpp      2881       H             MtMx
#0 flush handler: i/p port 1 reseting SessionTime to 0.
01:18:23.955  AudioStreamMgr_SessionCmdHandler.cpp        2060       M
AudioStreamMgr [1,1]: enter Flush Stream command handler
01:18:23.955  MixerSvc_MsgHandlers.cpp      2881       H             MtMx
#0 flush handler: i/p port 1 reseting SessionTime to 0.
```


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

# A References

## A.1 Related documents

| Title | Number |
|---|---|
| **Qualcomm Technologies, Inc.** | |
| *Hexagon Multimedia: Elite Firmware Architecture Description Document* | 80-N0029-1 |
| *Hexagon Multimedia: Elite Audio Postprocessor Interface API Interface Specification*<br>▪ For ADSP.BF.2.0/2.2<br>▪ For ADSP.BF.2.4<br>▪ For MPSS.JO.1.0 and MPSS.DPM.1.0 | <br><br>80-N3226-1<br>80-NF769-8<br>80-NF900-8 |
| *Hexagon Multimedia – Audio PCM/Bitstream Logging Through QXDM Professional* | 80-N3470-4 |

Confidential – May Contain Trade Secrets
2020-07-02 04:17:10 PDT
yinyihua@flex.com

## A.2 Acronyms and terms

| Acronym/term | Definition |
|---|---|
| ADM | Audio Device Manager |
| ADSPPM | Audio (aDSP) Power Management |
| AFE | Audio Front End |
| AHB | Advanced High Performance Bus |
| APSS | Applications Processor Sub-System |
| ASM | Audio Stream Manager |
| COPP | Common Object Postprocessor |
| CoPreP | Common Object Preprocessor |
| DMA | Direct Memory Access |
| EOS | End Of Stream |
| HDCP | High-bandwidth Digital Content Protection |
| HLOS | High-Level Operating System |
| LPA | Low Power Audio |
| LPASS | Low Power Audio Subsystem |
| MSIIR | Multi-Stage IIR |
| MVM | Multimode Voice Manager |
| PCM | Pulse Coded Modulation |
| POPP | Per-Object Postprocessor |

| Acronym/term | Definition |
|---|---|
| QXDM | Qualcomm eXtensible Diagnostic Monitor |
| SSMD | Single Stream Multiple Device |

Qualcomm
Confidential – May Contain Trade Secrets
2020-07-02 04:17:10 PDT
luo.yihua@flex.com