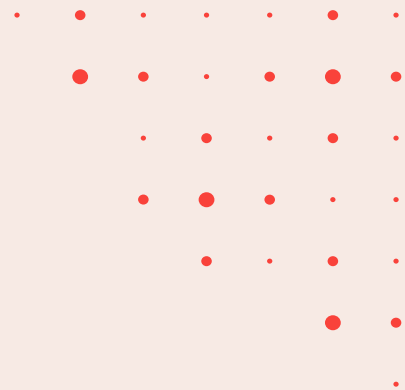


How to Make Android Automotive Audio System Play by Your Rules



Agenda

- 
1. Why should OEMs / Tier 1 suppliers invest in Android Automotive today?
 2. Android OS: from basic to more advanced features
 3. Current state of Android Automotive audio
 4. Android Automotive OS limitations and how to address them
 5. Difficulties with effective multi-zoning & audio differentiation for the driver and passengers
 6. What's next? Future of AAOS and audio

Webinar Structure & Guidelines

- Webinar duration – 1 hour
- All participants are in mute mode
- Feel free to ask questions in the chat
- Q&A session at the end
- Webinar recording and presentation will be shared



Host:

Yevgen Mikhnenko

Head of Digital Transformation

- Strategic visionary & digitalization evangelist
- Building long-term relationships with the customers for 12+ years
- Keen on business development in DACH, UK, Nordics and North America



Piotr Krawczyk

Lead Software Engineer at Tietoevry

- 17 years of experience in software development
- Engaged in various automotive R&D projects: AUTOSAR, Android OS, audio routing, voice and audio processing for voice assistants, specific Android Automotive OS requirements and solutions
- Worked with major OEMs and Tier 1s on production and R&D projects



Stefan Wysocki

Lead Software Engineer at Tietoevry

- 12 years of experience
- During the last 6 years was focused on Android Automotive: secure access to vehicle data and diagnostics, as well as other Android Audio solutions
- Worked with major Automotive and Mobile OEMs, TIER-1s



Why Should OEMs / Tier 1 Suppliers Invest in Android Automotive Today?

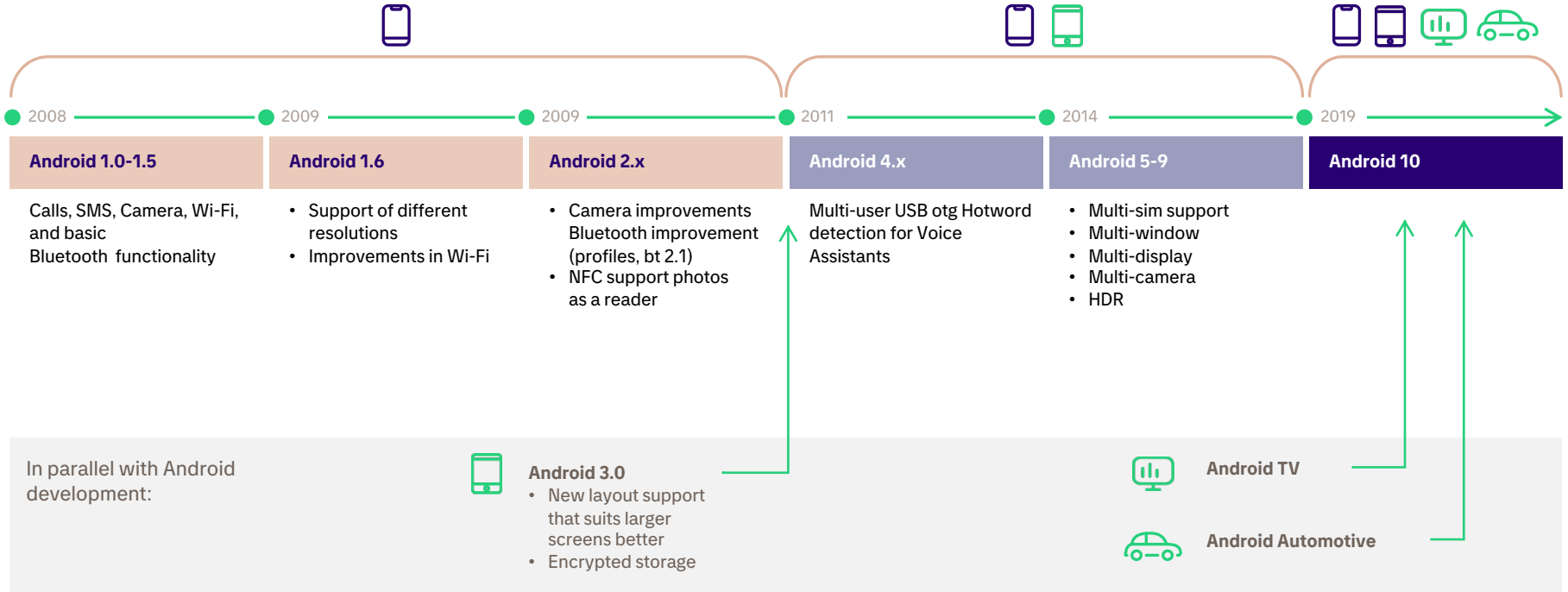
Customers tend to use their personal devices in certain manner

They expect the same capabilities from their cars:

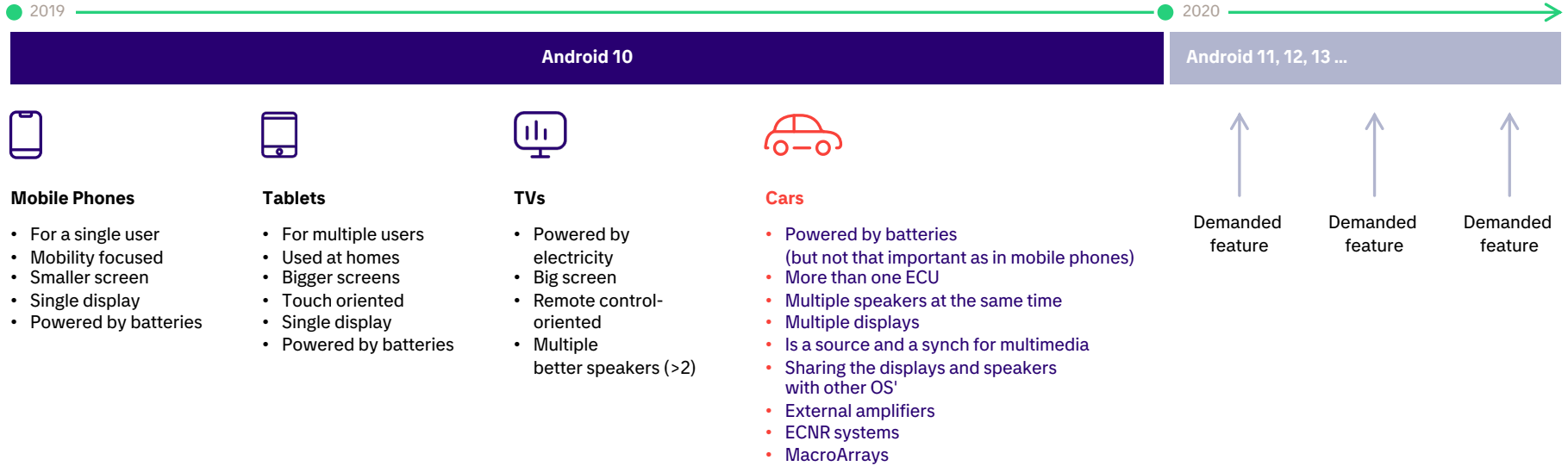
- Android gives a familiar UX to the users: swipes, back button, task management
- Large set of applications
- Big group of developers of the applications



Android OS: From Basic to More Advanced Features



Android OS: From Basic to More Advanced Features



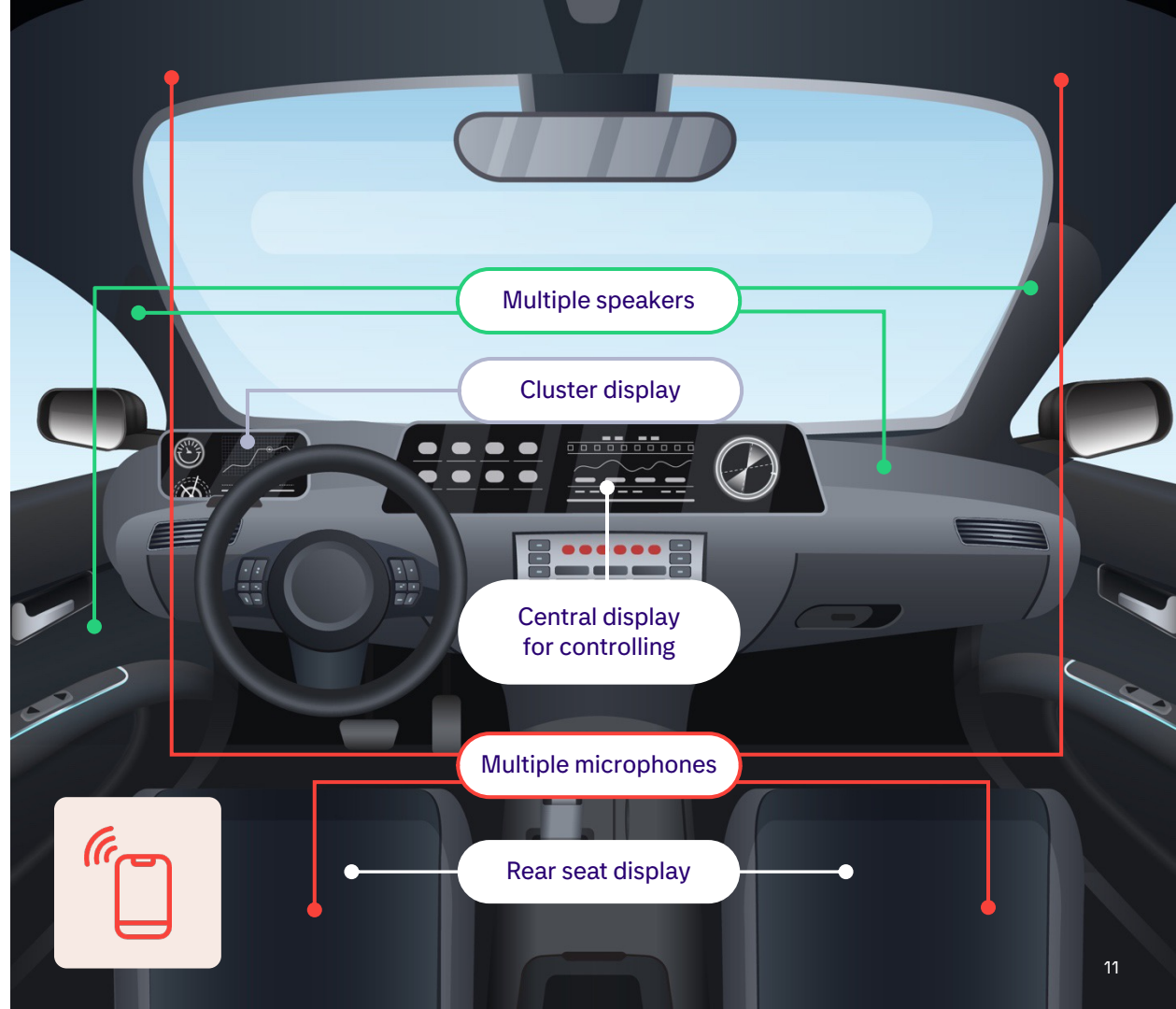
Diving Deeper into the Vehicle Setup

Infotainment Systems –
Complex & Implicit

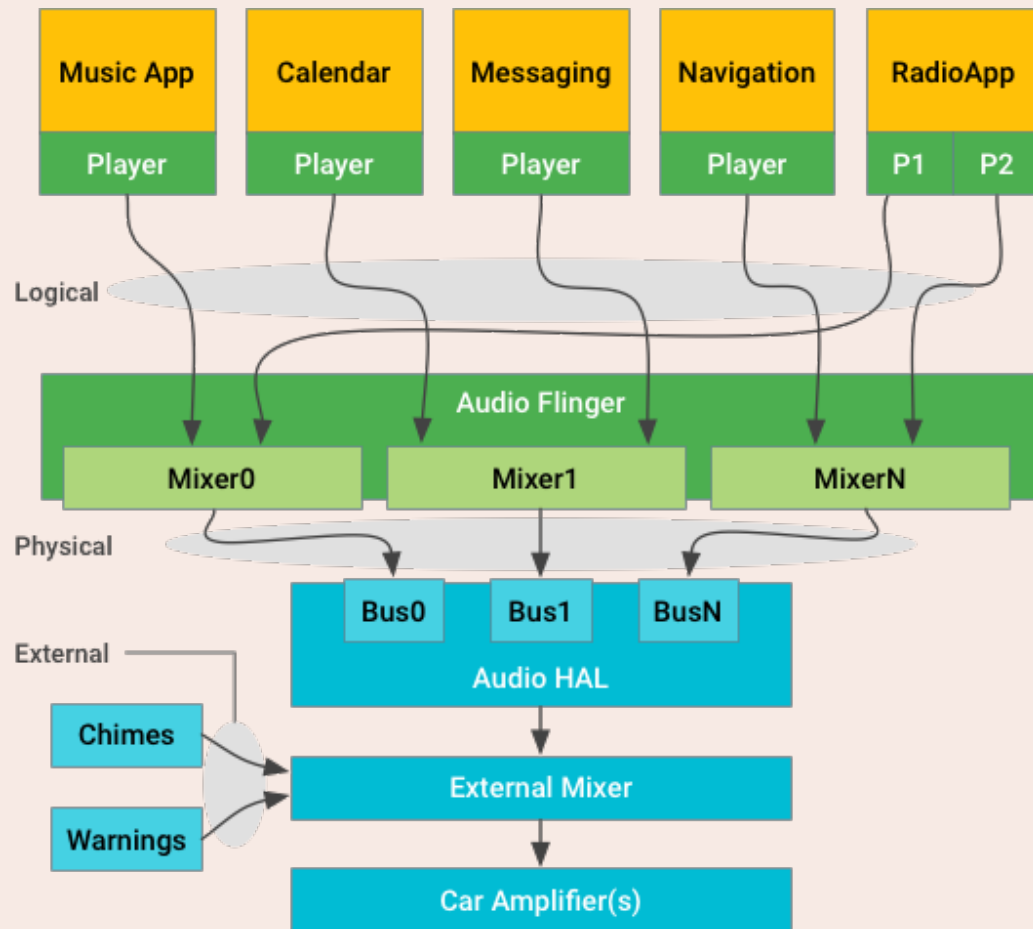


Modern Car Setup

- Multiple speakers
- Cluster display
- Multiple microphones
- Central display for controlling
- Rear seat display



The Current State of Android Automotive Audio



OTTB Limits of Android Automotive & How to Address Them

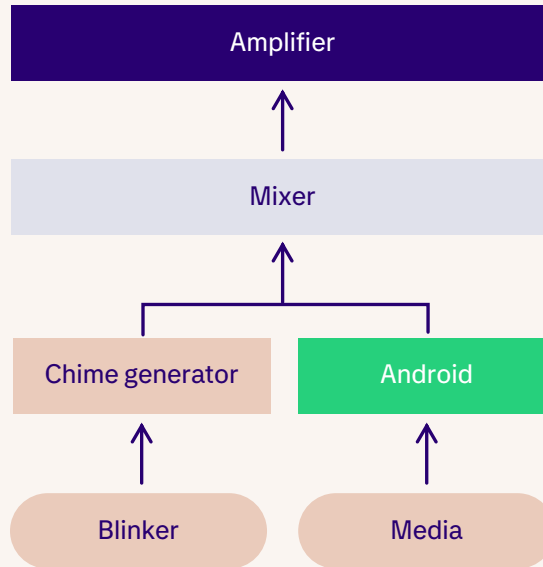


OTTB Limits of Android Automotive & How to Address Them:

What Needs to be Added to Android:



External mixing with safe sources – infotainment systems like Android is the most noticeable source of sounds, but not the only one. Others like warning beeps or turn signals are extremely important and are not managed by Android. These sounds are related to safety features in the car and mixing them with infotainment sounds is not in the scope of Android.



Additional volume management – managing volume can be a complex task in car. The volume should be monitored to not exceed the safe level. It can also be adopted to the current speed or environment (opened/closed windows or roof). Volume can also be changed from different interfaces (UI, manual knob) or even different consoles (volume control on each door). Additionally, OEMs could implement the saving of the preferred volume level based on the connected headphones or the driver

Checklist for Selecting the Right Audio Architecture

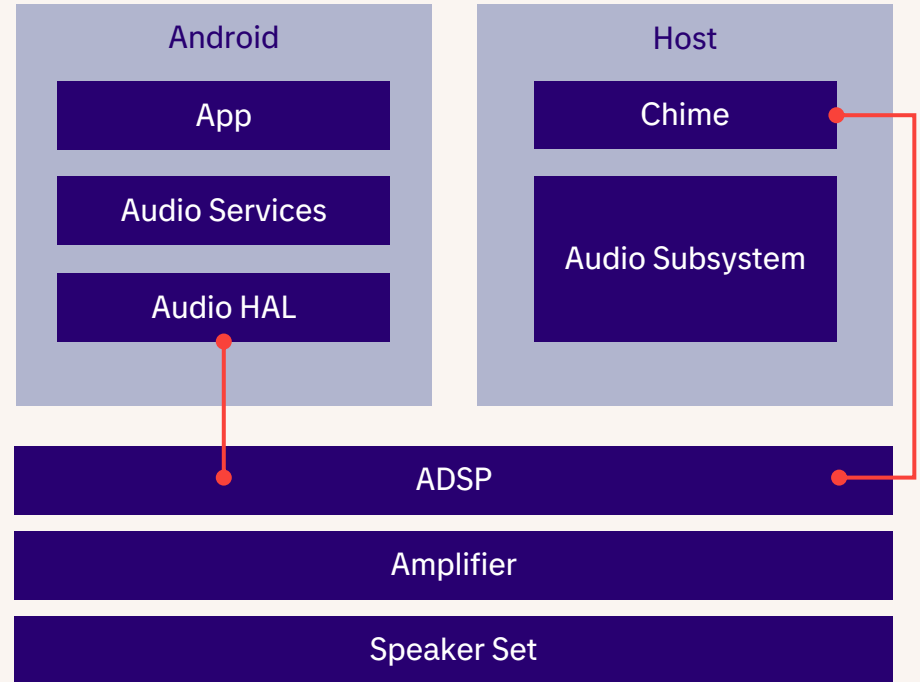
- Audio quality
- Complexity of features
- Project timeline
- Design maintenance costs



DSP Support (Scenario 1)



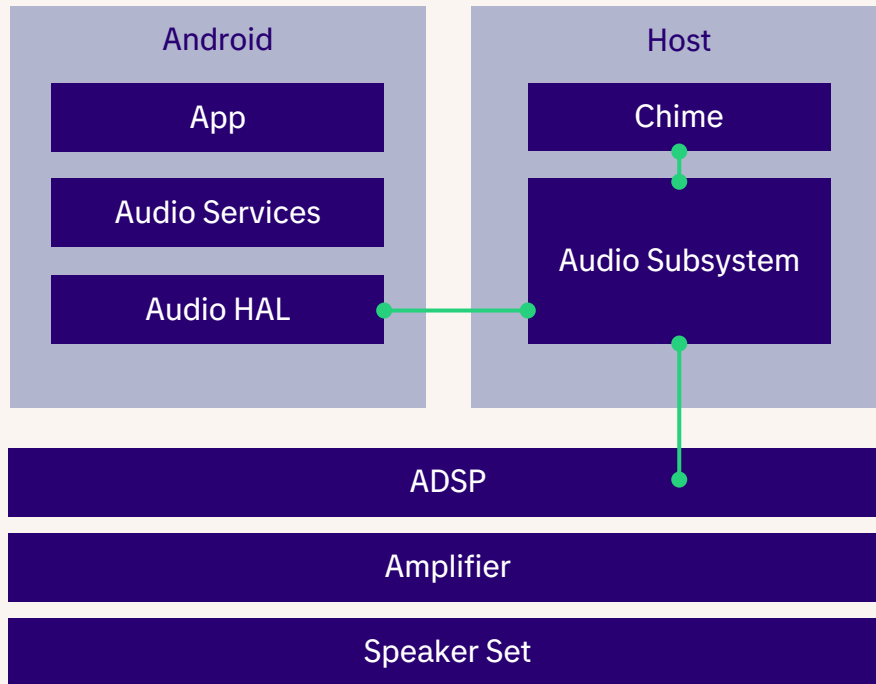
Using DSP decreases power consumption and CPU load but increases complexity and maintenance efforts, while also reducing portability. SoCs are getting more powerful so the processing can as well be made on the CPU. There are pros and there are cons, depends on platform and goals.



DSP Support (Scenario 2)



Another option is to utilize the audio subsystem on a safe partition to manage the routing of audio streams. Meanwhile, an additional mechanism for inter-partition transport (like virito-snd) appears to be a more portable and flexible solution.



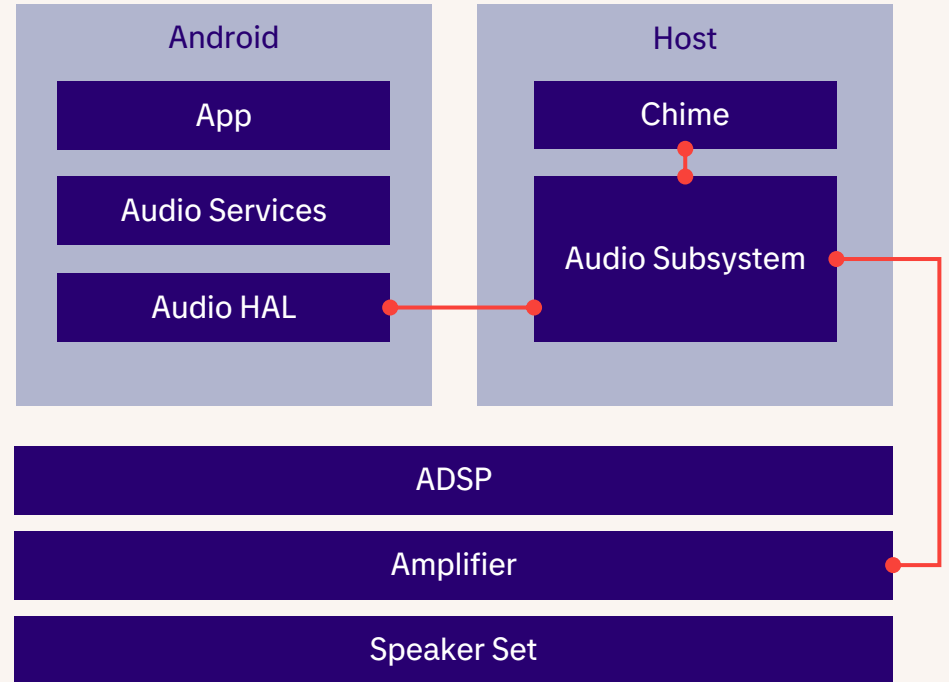
DSP-less Solution



The DSP role can be taken by a powerful enough SoC CPU. There are some pros of this approach:

- It is possible to utilize more opensource solutions for audio routing and processing.
- Easier to maintain (no DSP firmware)
- Less HW dependency

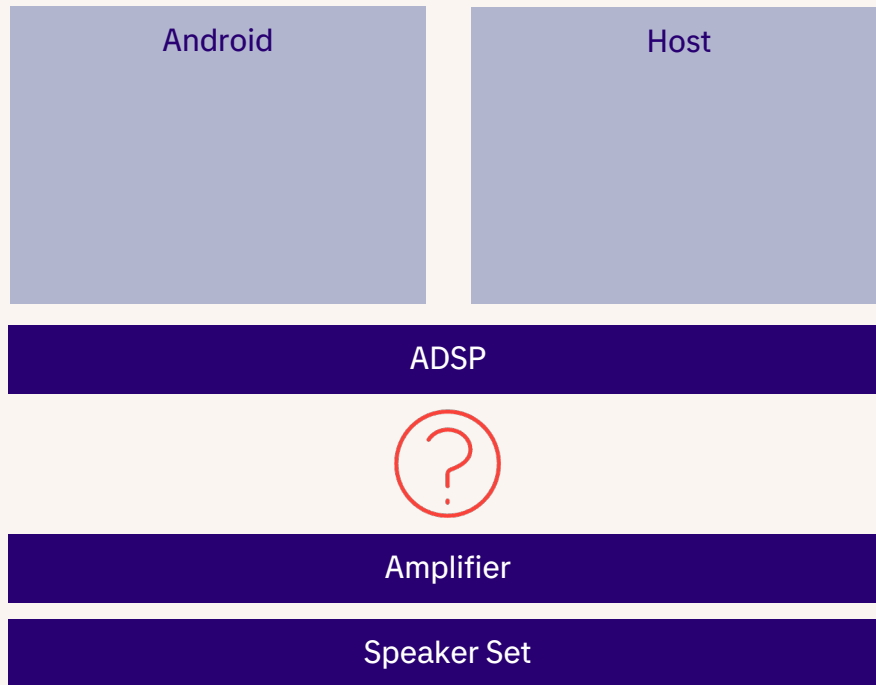
Audio processing with the CPU generates constant load.



Connection with the External Amplifier



There are plenty of options for how to transport the audio. One of the most flexible options on the market is the A2B, which allows transporting audio while communicating with the connected devices using just the twisted pair.



Radio Tuner Connection



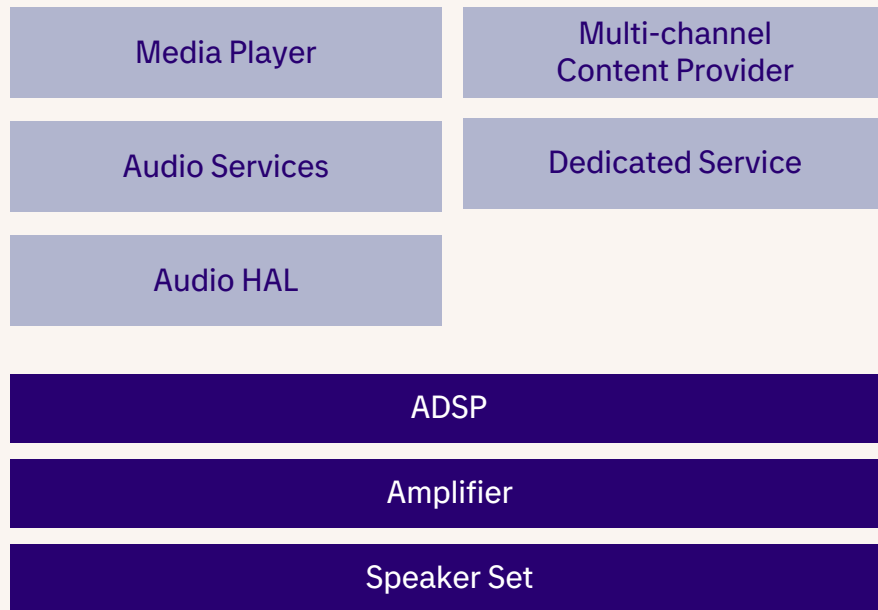
While streaming services have almost completely replaced other media on the market, the broadcast radio is still the most fundamental use case for in-car audio



Additional Codecs, Multi-channel Content (7.1, 5.1)



Modern cars can have high-end amplifiers and tons of speakers. Sophisticated surrounding content can bring astounding experiences for the driver and passengers, but it's not a standard feature of Android.

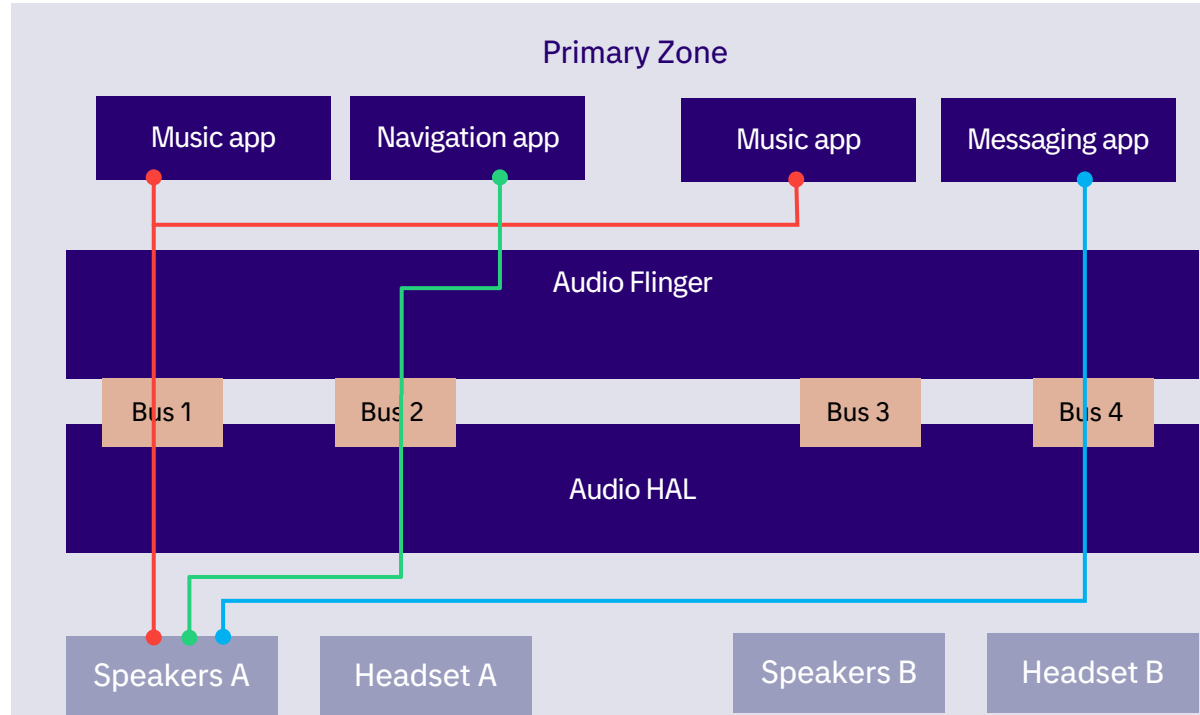


Difficulties with Effective Multi-zoning & Audio Differentiation for the Driver and Passengers



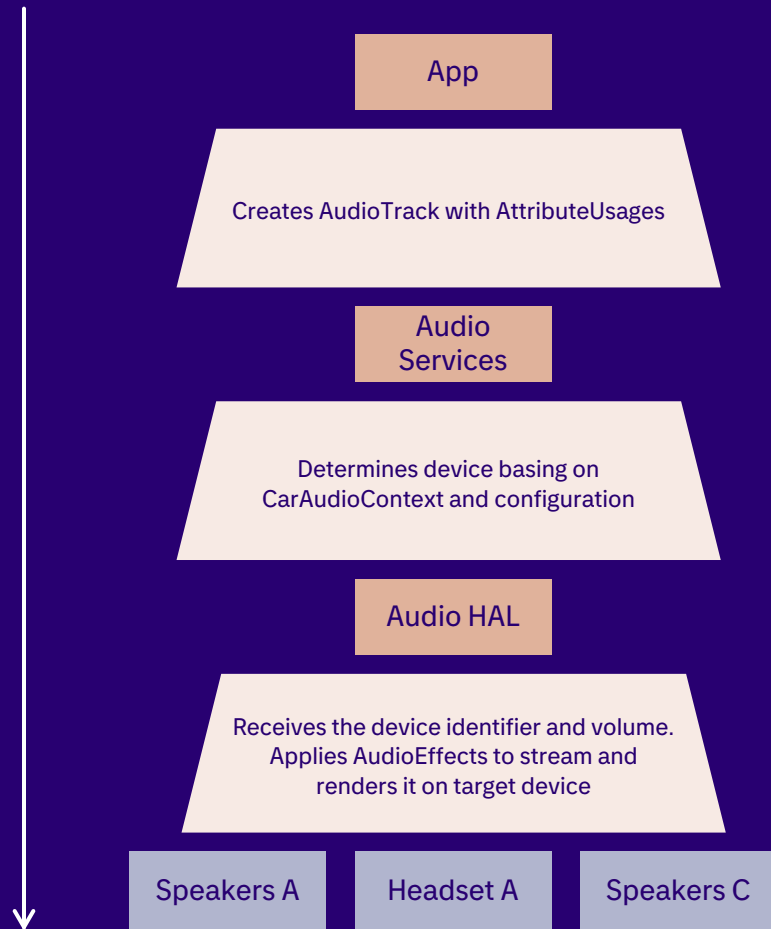
Multizone Enablement – How Android Operates Without Multi-zoning

- Assign the new user with Occupant Zone
- CarOccupantZoneService maps touch inputs, volume knobs and displays
- CarAudioService maps occupantZoneId with audioZoneId



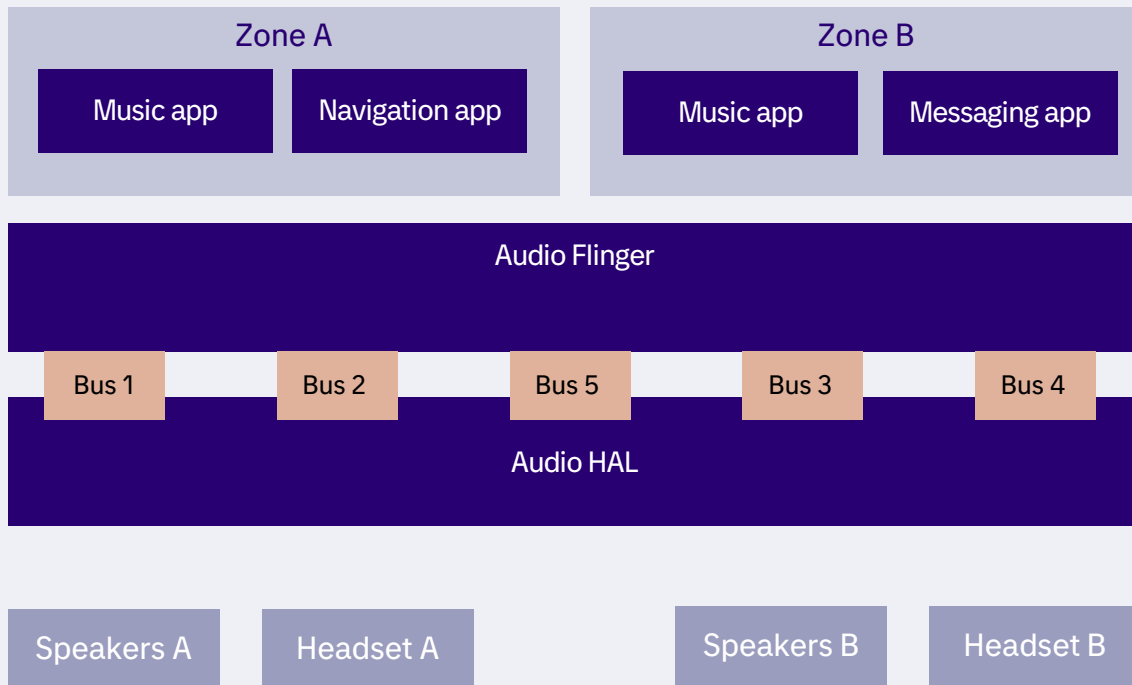
Multizone – How HAL Sees It

- Audio zone is an abstraction layer offered by the framework.
- It is configured in `car_audio_configuration.xml`
- Audio sources grouped in `CarAudioContexts` are connected in configuration to the bus device
- The device is attached to a single zone
- Each device has a separate volume control.



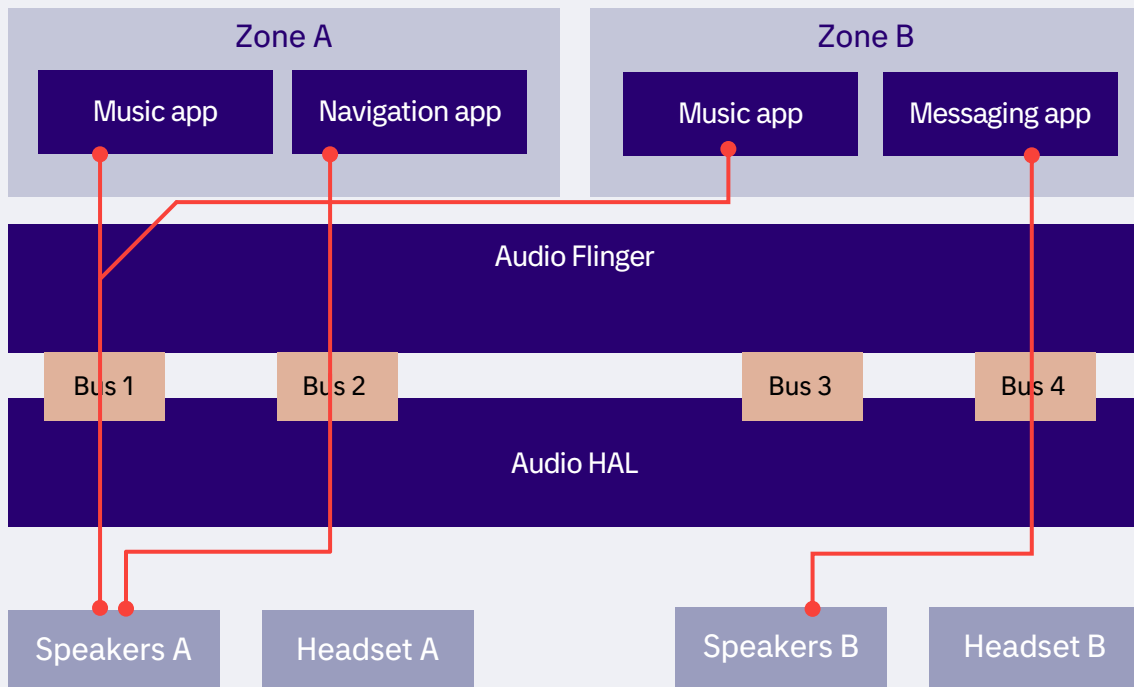
Multizone Architecture

```
<zone name="ZoneA" audioZoneld="0" occupantZoneld="0">
<zoneConfigs>
<zoneConfig name="Default config for Zone A" isDefault="true">
<volumeGroups>
<group>
<device address="bus 1">
<context context="music"/>
</device>
</group>
</group>
<group>
<device address="bus 2">
<context context="navigation"/>
</device>
</group>
</volumeGroups>
</zoneConfig>
</zoneConfigs>
</zone>
<zone name="ZoneB" audioZoneld="1" occupantZoneld="1">
<zoneConfigs>
<zoneConfig name="Default config for Zone B" isDefault="true">
<volumeGroups>
<group>
<device address="bus 3">
<context context="music"/>
</device>
</group>
<group>
<device address="bus 4">
<context context="notification"/>
</device>
</group>
</volumeGroups>
</zoneConfig>
</zoneConfigs>
</zone>
```



Dynamic Zone Configurations – Passenger Audio Cast

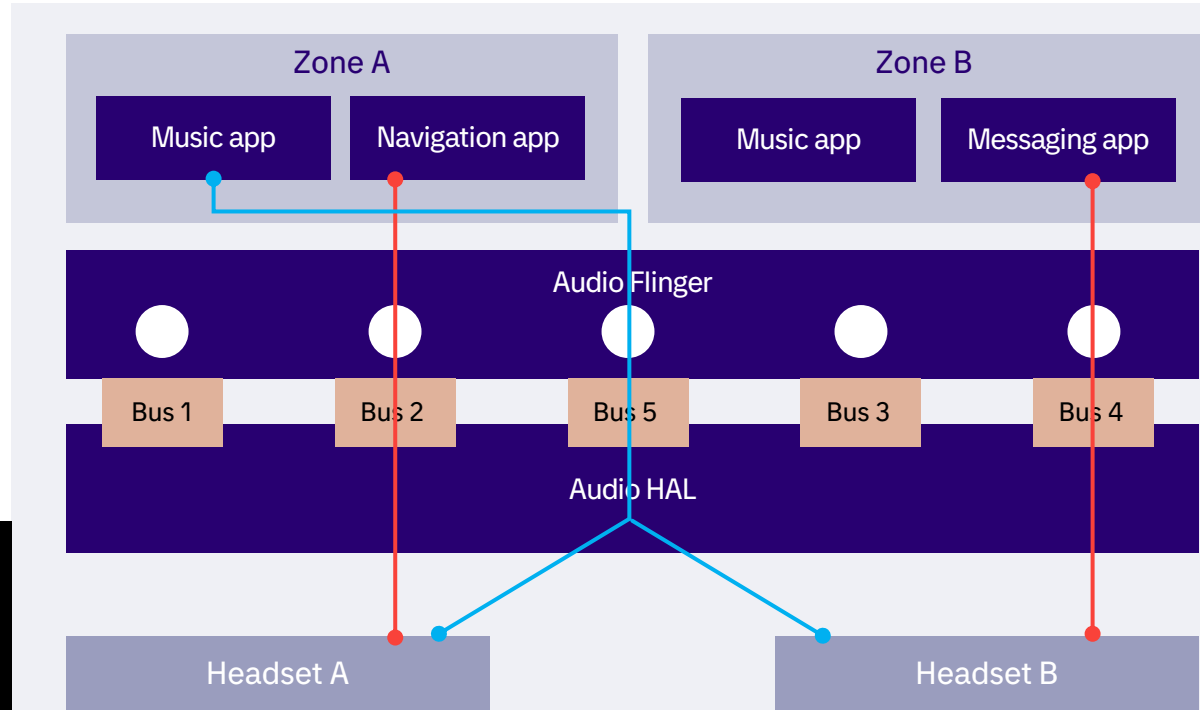
```
</zone>  
<zone name="Zone1" audioZoneId="1" occupantZoneId="1">  
  <zoneConfigs>  
    <zoneConfig name="Default config for Zone B" isDefault="true">  
      <volumeGroups>  
        <group>  
          <device address="bus 3">  
            <context context="music"/>  
          </device>  
        </group>  
        <group>  
          <device address="bus 4">  
            <context context="notification"/>  
          </device>  
        </group>  
      </volumeGroups>  
    </zoneConfig>  
    <zoneConfig name="Audio cast config">  
      <volumeGroups>  
        <group>  
          <device address="bus 1">  
            <context context="music"/>  
          </device>  
        </group>  
        <group>  
          <device address="bus 4">  
            <context context="notification"/>  
          </device>  
        </group>  
      </volumeGroups>  
    </zoneConfig>  
  </zoneConfigs>  
</zone>
```



Dynamic Zone Configurations – Passenger's Audio Mirroring

HAL receives information which buses are the destination of the mixed source
bus: `setParameter(mirroring_src=bus_1000;mirroring_dest=bus_10, bus_20)` when `carAudioManager.enableMirrorForAudioZones(List<Integer> audioZonesToMirror)` is called

```
<mirroringDevices>  
  <mirroringDevice address="bus_1000"/>  
</mirroringDevices>
```

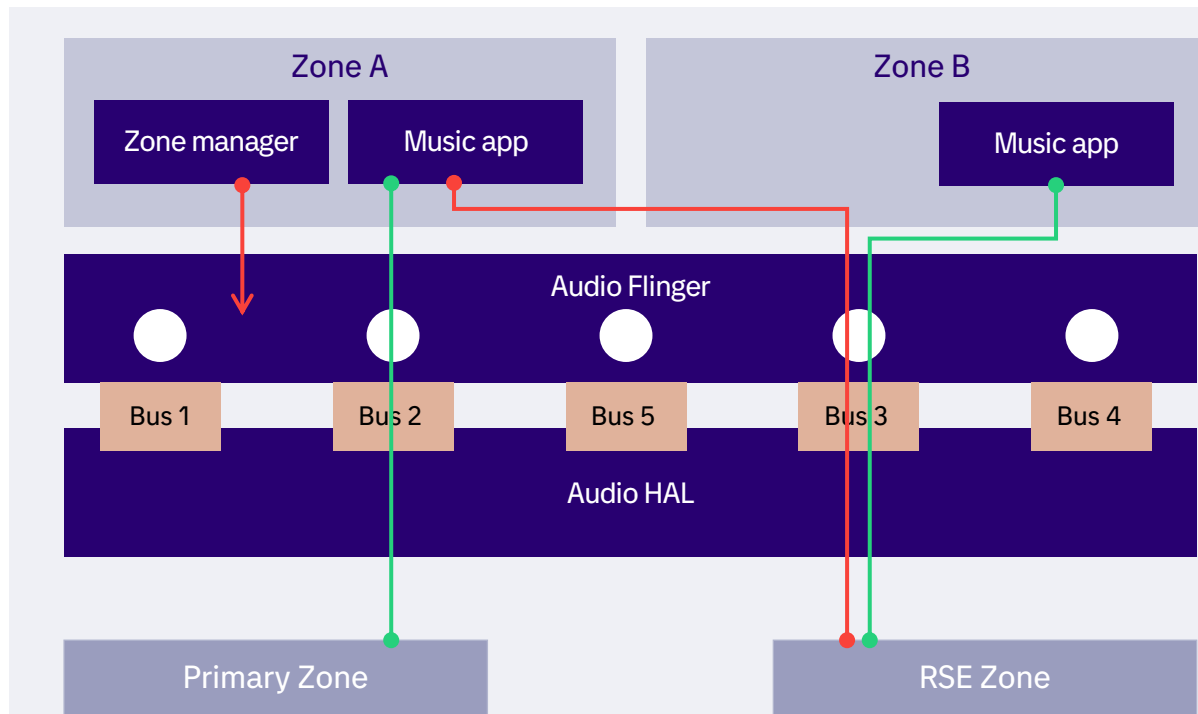


Potential Use Case: Zone Manager

By using the system's APIs it is possible to create an application that can change the output. For example, a music app that switches from one zone to another.

Examples:

- Driver can change output of RSE zone to the Primary Zone
- Driver can start playback of media in the Primary Zone and redirect it to the RSE zone



What you need to have Android on Board

- **Audio HAL**
 - Silicon vendors solution
 - OS providers solution
 - Proprietary/custom solution
- **Mixing with external / safe sources**
 - Software or hardware
- **Configuration**
 - Android audio configuration
 - Audio Zones configuration
 - Architecture specific configuration



What's next? Future of AAOs and Audio

- Trends
- Features
- Use Cases



Q&A

Thank you

 tietoevry

