20200204-05-AASIG-F2F-meeting-minutes

AASIG-F2F-meeting-minutes

09.00	• Welcome, logistics (e.g. IT) and agenda review • Setting up IT related stuff, getting everyone connected • Announcement: GENIVI Spring AMM will take place on May 12-14 in Leipzig, Germany. We will have AASIG face to face meeting(s) there	TODO
09:20	Agenda Review • Philippe: reminds the objective of the meeting which is to agree on a couple of design scenarios and move to detailed design and implementation, target is to have PoCs available at the end of Q2 and be able to contact Google • Gunnar: Contacting Google -> we need to prepare before the time window closes • Going though the agenda, topic by topic so that everyone is in-sync	
09:30	Roundtable • Alexander: BMW, connectivity, now looking into boardnet abstraction and how we can use it on different platform • Gunnar: Genivi project, development/technical lead of Genivi • Piotr: Tieto, AA for 3 years, audio routing, helping with Genivi • Stefan: Tieto, AA for 8 years, mobile, embedded, architecture • Philippe: Genivi 10 years, Autosar in parallel, working with German OEMs & tier1, before was managing a SW consulting company working with Continental in particular, Telecom before that and Aerospace • Bartosz: Tieto, wroslaw, 8+ experience with Android, multimedia domain • Nadim: esolutions, elektrobit, Mobis, SW developer, how Mobis can contribute • Justin: hardware engineer, connectivity, audio, how Mobis can contribute • Kevin: CTO high mobility, data model of the car, and other topics, see where Genivi is going • Gururaja: Bosch, 6 years with Genivi, connectivity, HMI SW development, Infotainment systems • SKYPE: Johan: Mitsubishi electric tier1, 6 years, Genivi quite new, but would like to get back • Sachin: Mercedes Benz Research North America	
	Vehicle HAL Security design	
09:45-10:45	Access control and permissions in Android (Stefan)	

	• Stefan shows slide deck : Build connection VSS to Android permissions model.pdf
	• Stefan shows also :
	  ◦ The doc on Custom App Permission: https://developer.android.com/guide/topics/permissions/defining
	  ◦ an example of permissions on google git, one called vehicle energy
	• custom permissions: only one is available, it is CAR_VENDOR_EXTENSION, this is a problem
	• we need to invent our own version of this extension for additional permissions we might need
	• Android have already a group of permissions available via API (manifest file for examples)
	
	How to verify permissions
	
	• For verifying the permissions, there are different strategies and security concepts
	• Even for new applications, we can assess how harmful an application is
	• Gunnar: question on the key management and the signature verification process
	  ◦ Piotr: the playstore is considered as a trusted source of applications
	• Stefan: shows the verification process as set up by Google, link: https://source.android.com/security/apksigning/v2
	• OEMs need to have a secret key service management inside their process in order to support the permission verification
	• the package can be signed with the platform key and released to GooglePlay
	• there will be for instance different instantiations of the same code with different signing keys for the different vendors (OEMs) who want to deploy the same app
	  ◦ currently there is no shared signing key among OEMs
	• when Stefan says GooglePlay, this is GooglePay in GAS
	• Guru: my original thought was that both the app developer and the OEM should approve the deployment of an app with their own key
	  ◦ Stefan: this is not possible, there is only one key
	• Gunnar: can an App be signed by 2 different keys ? The system would check the signature of both keys
	  ◦ Stefan: Samsung, Sony do this, they use additional APIs for this purpose
	• Guru: This restricts how third parties can create application for our systems
	  ◦ Stefan: no, it doesn't
	• Sachin: what do you mean ?
	  ◦ Stefan: We can have our own version of stores
	• Gunnar: this needs more thought, the idea is to have two signing keys, *developer* that I made this, *system* that we agree to this.
	  ◦ Stefan: But it's not possible
	• Alex: let us assume every OEM has its own playstore, the key is developer key, but the OEM would add it or not
	  ◦ Side load an application (APK download and application)
	  ◦ So the model is that everything that is downloaded is already accepted by the system
	  ◦ The store is not the only source of applications
	• Gunnar: This is more a question about open/close system: it's for the user to accept the permissions and the OEM should not restrict or regulate them unless it is a sensitive permission. We don't need to sign it from the OEM
	  ◦ We already showed that the properties are available and some not (protection level), a propert can be: normal, dangerous (needs to be asked, user should agree), signed, privileged
	• Sachin: each display will have their permissions, for OEM how can we configure these displays
	  ◦ This might be treated in the next presentation by Piotr
	• Gunnar: the normal model is that the system does verify the signature
	  ◦ Each developer sign with their keys and put them on the app store
	  ◦ The system checks if the signature is valid and the key format is correct
	  ◦ There is no cross checking (like https and playstore)
	  ◦ Playstore is only certifying that the application was downloaded using the playstore (and not a side download).
	• Gunnar: We need to check the open/semi-open model, and understand how we will manage it from technical point of view.
	  ◦ Analyzing the need and the solutions
	  ◦ The solutions would be too much for this group
	  ◦ Probably these discussions are being done already between single OEM and Google but not with GENIVI
	  ◦ Or we need to have our own service.
	• Side note: The Android deficiencies come from the history of Android, from phone to car, there are gaps
	
	Next steps
	
	• Gunnar. there is some work to figure out the different models
	• Sachin: Google has decided beforehand for instance that controlling the HVAC is only possible for an App signed with the platform key
	• Sachin: in a car we will have multiple displays with different permissions because displays show different things
	  ◦ Piotr: I will make a proposition for this because AFAIK this is not supported by Google	

10:30-10:45	Break	
10:45-11:30	External service approach - how to use Adaptive AUTOSAR Identity & Access Management (IAM) (Alexander)	

	• Alexander had a sync with Giovanni Vergine & Stefan Wysocki on Adaptive Autosar IAM prior to the F2F but we realized there are many open issues we need to look at
	• First we need a solution for accessing or exposing the data in the vehicle.
	  ◦ For example: Ambient light, you can change the color, brightness etc. During development, the application developer is directly communicating with the ECU developer to talk about the colors etc. But this does not scale.
	  ◦ We need a system that can handle data like what are the colors enums etc
	• Question is: how to authorize this and how can we ensure this communication.
	  ◦ Idea: Authentication service, Data service.
	  ◦ Data service would request the data
	  ◦ Authentication would check if this App can get this data and give a token to the data service
	  ◦ The data service is then using the token to get data
	• Alex: yet another example
	  ◦ parking assistance data cannot be abstracted using VSS because there are too many states,
	• Alex: explains why we need some kind of abstractions of vehicle data to share knowledge with developers who are spread geographically (they cannot share their knowledge of the car like if they were sitting in the same room)
	• Alex: would like to have a specific service offered by the platform to get access to the vehicle data
	• Alex: Autosar does not seem to be so extensible
	• Alex; we will need to generate tokens to enable the access to applications (token valid for an hour or one day or...)
	• Alex: in VSS we have leaves which are the data, we will have also groups, we do not know which granularity should be shown at the application manifest level
	• Android is activity based. If we make the data service in the framework does it have an advantage ?
	• Advantages
	  ◦ We can control better, for example not to forward the request to the data service if the authentication service stopped it
	  ◦ Also we would have one way to access the data
	• Disadvantages
	  ◦ Bottleneck for the applications always passing through the framework
	• Data server app authentication process have been added to the architectural concepts diagrams, look at AuthenticationMethods
	• We need to think about the idea, what is better - service layer or application layer ?	

11:30-12:30	Users vs. permissions - presentation of the zone concept in Android 10 (Piotr)	

	• Piotr shows this Users and permissions.pptx
	• overview of Android account types (taken from Google documentation)
	• Piotr explains how a user can be identified, how to recognize user
	  ◦ Primary, secondary, guest currently available on android
	• discussion on how to check app/user permission, connection to what was discussed earlier
	• In Android Automotive now, we have the following situation
	  ◦ Headless system user: main services, owner of services, etc.
	  ◦ Secondary user: driver etc.
	• First example - audio
	  ◦ as a rear user I'd like to play my audio on rear headphones
	  ◦ the display in the rear or audio in the rear are now solved by using a different user id
	  ◦ HAL is not aware of the user: that's the problem
	• Same solution as above, application will ask the authorization service for a token and then the application can use this token in order to access graphql (i.e. the vehicle database)
	• Other possibility is a content provider, this is the most common way in Android for the application layer to access data
	• Main point is that there is a single point of access for example to the HAL
	• One example to understand more: let us assume we have one tyre monitoring application developed by BMW
	  ◦ if I log as a BMW user, I got all data (tyre temperature for instance in addition)
	  ◦ if I log to the same application as a Mobis user, I do not get all data
	  ◦ There might be that the dealer would like to know how many times the wheel over-heated, this is regardless of the user
	• Guru: regarding users, we have a lot cases - valet, chauffeur, kids mode, etc.
	• Guru: can we solve this with the VSS model or with an API created for this purpose ?
	• Do we need an application that runs differently for different users ?
	• Kevin: it's up to the application to show the UI according to the user
	• This is also a point that the framework should not be deciding on behalf of the application what data is accessible
	• Is this needed for the boardnet data?
	• Do we have specific examples to show that we need a different user for the same application
	• Do we need to give different permissions to different users?
	• Gunnar: can we come up with such a use case ?
	• Gunnar: one option is to create a specific API when you need to access very finely controlled data and not embed the access control in the VSS database
	• Another example: the taxi driver's situation
	  ◦ for Mercedes Benz a taxi belongs to a specific car line (i.e. a taxi has different features than a normal car)
	  ◦ These users are not really the Linux users but they have their own places to store the data
	• Sachin: what we want to achieve is to expose to the Android world the vehicle properties which we have been handling in the Linux-based infotainment systems so far
	• Alex: can you and your wife be logged in the system at the same time?
	• Sachin: the answer is no
	• Sachin: the control of the display is only based on its position not on whom is using it
	• Sachin: the restriction at application level does make sense
	• End of long discussion on how to control access to data at the application or the service (framework) level
	
	wrap-up (before and after lunch)
	
	• data access verification component: application permissions and user permissions need to be combined
	• Letting the application decide what to do is good but not for a closed system (TBC)
	• Framework needs also to have restriction
	• Permission management can solve this by creating permission of groups
	• Gunnar: we need UID (user identity) to be shared down to the low layers
	• Discussion about what we need to solve, what is the problem to solve ?
	• There are different solutions of course even with Android	

12:30-13:45	Lunch Break	
13:45-14 :50	VSS layers	

	• Please refer to this VSS composable layers for an introduction to VSS layers concept
	• Gunnar presents a short implementation of Android permissions in VSS (TODO Gunnar to provide link to example code)
	  ◦ New extension Apspec that links the main signal database to the permission of each
	  ◦ In the standard there are signals, attribute, sensors
	  ◦ The permissions are different for different systems so the permissions files need to be independent of systems
	  ◦ The idea is that we can have metadata such as Android permissions in a separate file
	• Alex: coins the idea of a deployment file Apspec and have the permission "attribute" as a generic thing in the VSS layer and the "Apspecs" in the deployment data for Android Automotive
	  ◦ for instance in Autosar you would have a different folder with the permission
	• Gunnar: shows the implementation of VSS layers he has in mind
	  ◦ The ideas are written in the file fuelsystem.appspec
	  ▪ Think about generic solution
	  ▪ Think about grouping
	  ▪ Think about permission level
	  ▪ Does write access also give read access ?
	  ▪ Closer to VSS and easily parse-able
	  ◦ Gunnar: we need to write some tooling to process the VSS layers
	  ◦ Apspec could have no include just the entire name
	  ◦ Is it better to keep the same structure?
	  ◦ Opposite mapping - entity
	  ▪ Nearest to the Android
	  ▪ Protection level can be included
	  ▪ More user friendly
	  ▪ This is preferred
	  ◦ There is no point of thinking of very complex scenario, because at some point we can create custom API
	
	Wrap-up
	
	• We will investigate the following tracks
	  ◦ Track #1: do not include users, just the authentication service and application talking to directly to data server, the users will be dealt with a different custom solution
	  ◦ Track #2: include the users and let the application only communicate with data server from one point of entry
	  ◦ For Android applications, we will have a list of permissions listed in a file, and then we have the (VSS ?) leaves listed inside each permission
	
	Information
	
	• Gunnar: provides the link to the W3C public mailing list where the VSS layers concept is discussed
	• W3C list – vss layers: https://lists.w3.org/Archives/Public/public-automotive/2020Jan/thread.html	

	Vehicle HAL Security design ends	
14:50-15:05	Break	
15:05-15:20	Agenda re-shuffling	
	Vehicle HAL Technical Proposals - further refinement	
15:20-16:00	Android internal service (Some signal-connecting library using VSS standard) (Stefan)	

	• Stefan shows this VSS to standard Vehicle HAL.pdf on the translation from VSS to VHAL properties
	• Long discussion on variant III slide and content of vehicle HAL (custom HAL vs. vendor-extension)
	• Discussion on various alternatives to implement things in Treble or the framework
	  ◦ We don't write in HAL but in HAL extension
	  ◦ We have some work to do to agree on the model before having people work on implementation
	  ◦ Mapping from vehicle data to VSS and properties is needed
	  ◦ Kevin: as a third party app developer, do I choose if I want a permission of Android or a VSS permission? Are there two ways of getting data?
	  ◦ Yes but the Google way is key/value, with a limited API
	  ◦ For instance VSS has battery status, Google does not, what happens in the future ?
	  ◦ So yes there is two ways of working
	  ▪ SOA or Data driven approach
	  ▪ We need to define which data are cached or which data has to be directly taken
	  ◦ Property value change propagation will be covered by the protocol implementation we use to communicate between the data server and the app
	  ◦ If we only have get/set/subscribe we don't need an API	

16:00-16:10	Break	
16:10-17:05	External services - SOME/IP (Gunnar)	

	• Gunnar shows the slide with Some/IP usage between the vehicle nodes (look at slide 14 Architectural proposal III (via Global SomeIP Service) of this Vehicle HAL Architectural Design Concepts
	• The idea is to mirror in the framework layer services that are outside the system,.
	• This is related to connecting Android and non-Android systems
	• The general connectivity concept in Autosar is the following
	  ◦ there is a request, followed by a send to authentication: is this request allowed? Then the request is denied or the data are sent back
	  ◦ The concept here is saying: any app will check in the database if the request is allowed or not
	• But in case of external apps/db to the system, the app will only check if the system can request data but not per application
	• So the system needs to implement a way to check if the application can access the data from the external app/db
	• Note: The full SomeIP specs tree is available online for more reading at https://www.autosar.org/standards/foundation/ , select Release R19-11
	• When we have a VSS, we can better describe or know where the data is coming from exactly
	• The main problem with app directly or via socket communicating with external services is that we won't have control over them, there is no check if this app is allowed to access it
	• Piotr: what about a service that does some service discovery and tries to do the request of the API ?
	• Johan: cannot we put the managers in the HAL ?
	  ◦ Actually we discussed it before, it's not a hardware and this is why we shouldn't officially put it in the HAL, this is not where it belongs
	  ◦ But we can put it in the HAL because we can make it just to say that this is another abstraction layer
	• interesting discussion on why to use SomeIP, how to aggregate data through an ad-hoc data server talking via someip to get the elementary data	

17:05-17:15	Agenda review for Day 2 • We need to change the architecture diagrams • We need to decide what do we want to implement • We need to name the ideas or concepts appropriately	
17:15	End of Day 1	

Time	Day 2	TODO
09:15	Start	
09:15-10:00	Summary of expectations	

	• Sachin: Which design to use (develop), or at least prioritize ?
	• Alex: which directions should we go ?
	  ◦ external data server as a service,
	  ◦ SomeIP connection to framework layer and apps
	
	Sachin: it looks promising in particular the integration of VSS, we need to select one approach and projectize it
	
	Recap of yesterday's discussion
	
	• approach 1: (data server app) is sitting somewhere, on a Linux partition or a different ECU, we have an App installed on Android, this App needs a web token to get the data
	• approach 2: (architecture proposal via custom HAL), the app does not have any direct communication to the data service
	• approach 3: SomeIP approach there is a SomeIP service somewhere and there is a generic SomeIP service in the framework
	• Nadym: where is the bottleneck in approach 2 ?
	• Alex: this has to do with the number of requests for data
	• Sachin: we need to be compliant with the Android Auto architecture (not from a CTS standpoint, only from the code structuring standpoint), do we follow the principles of code organization set up by Google ?
	• Alex: GAS App do not have any service inside the framework
	• Alex: let us skip the SomeIP approach because we know how to do it with model transformation and code generation
	• Alex: let us focus on the vehicle data server
	• Gunnar: we all agree
	
	AASIG Statement Of Work - **DECISION**
	Vehicle Data Access Architectural Design & Implementation
	
	• AA SIG will develop a PoC implementation of the External Data Server (priority one).
	• AA SIG will develop a PoC implementation of the Data Server inside the Framework (so-called Internal Data Server) (priority two)
	• AA SIG will develop a PoC implementation of the SomeIP stack inside the Framework (priority three)
	• AASIG will develop a PoC implementation of the Google VHAL + OEM Extensions inside (priority four)	

10:00-10:10	Break - meeting split - Vehicle HAL and Audio HAL tracks	
10:10-12:15	**Vehicle HAL / Vehicle Data Access Track**	
	Work Breakdown Structure for the External Data Server Proof-Of-Concept • The objective of this session is to develop an initial WBS for implementing the External Data Server. • The TODOs listed below were added to the EA block-diagram, look at ExternalDataServerPoC-WBS.pptx • After review and addition of a *description of work and definition of done* to each TODO, these TODOs will be entered into Jira. Vehicle HAL split meeting participants elaborated the WBS below • TODO Develop an initial WBS for implementing an InternalDataServer (Priority 2) Proof-Of-Concept)	

	VSS feeder component
	
	• todo finalize permissions layer concept (independent work item)
	• todo create a layer concept for the Franca to VSS leaf mapping (model transformation)
	• todo design and implement franca service (SomeIP)
	• todo implement feeder as PoC
	• todo check signal to service translation in Adaptive Autosar
	• todo agree on PoC use cases for the implementation
	• todo create PoC Someip simulation component to playback agreed use cases
	  ◦ note: we could consider using an Adaptive Autosar node like what we did for the FARACON demo
	  ◦ note: VSS data base could be merged into the VSS feeder
	• todo select and implement VSS data storage (e.g. VISS, Geotab W3C PoC implementation)
	  ◦ look at W3C wiki page on VISS
	  ◦ https://at.projects.genivi.org/wiki/pages/viewpage.action?pageId=40403466
	  ◦ look at melco: https://github.com/MEAE-GOT
	
	VSS data server component
	
	• todo APP implement authentication to access the data server (eg JWT), e.g. json Web token
	  ◦ relates to Vehicle Signal Authentication
	• todo APP implement access for in-vehicle data, (e.g. App manifest permissions layer concept)
	• todo APP implement request/response serialization
	  ◦ Gunnar: this relates to a communication protocol (so-called binary protocol)
	• todo resolve requested data for the APP from the VSS data structure
	• todo change/write data values for requested data leaves
	  ◦ look at https://github.com/w3c/automotive/issues/322
	• todo handle the subscription for APPs
	• todo write a generator which will handle permissions in the data server
	  ◦ look at on-going discussion in W3C on the dynamic registry https://www.w3.org/2019/11/W3C_Gen2_dynamic_registry.pdf (Geotab proposal)
	
	Framework Layer / Authentication Service
	
	• todo request APP permissions from package manager
	• todo generate access token for the APP including the APP permissions
	
	Application Layer
	
	• todo Implement the APP permissions based on permissions defined/proposed in VSS layers
	• todo implement access token request
	• todo APP implement request/response serialization for the client
	• todo implement the selected use cases	

13:00-13:45	Lunch break	
13:45-16:00	**Vehicle HAL / Vehicle Data Access Track**	

Roadmap for the PoC(s)
Philippe : suggests the following roadmap for the PoC(s) implementation

- stage 1 Spring AMM (12-14 May)
- stage 2 Go to Google readiness check (early Q3 - July ?)
- stage 3 Fall tech summit (Q4 – October-November ?)
- stage 4 CES 2021 (January 2021)

Target platform for the PoCs

- VSS data server on laptop/linux with vsomeip component
- AOSP+application layer on Renesas R-Car H3 or NXP boards with AOSP P

External Data Server Concept – BOM (technology selection)

- Platform: Renesas R-Car H3 or NXP boards with AOSP P
  - 1- Android App in Application Layer
    - Android Apollo Plugin
  - 2- Authentication Service in Framework layer
    - Java Implementation
- Platform: Notebook with Linux / Docker / SomeIP (vsomeip)
  - 1- NodeJS
  - 2- Apollo GraphQL Server (Data Server)
    - Schema generated out of VSS
    - Permissions generated out of VSS
    - JWT authentification.
    - implement Resolvers
    - implement Mutations
    - implement Subscriptions
  - 3- Feeder

Use Cases for the PoCs

1. Battery status (high voltage)
2. Tire pressure
3. Air Conditioning

Work Breakdown Structure for the Internal Data Server Proof-Of-Concept

- rework of WBS for the External Data Server in order to adapt it to the internal data server architecture
- look at EA diagram in InternalDataServerPoC-WBS.pptx

Identify the deliverables / outcome

Deliverable #1 - **Vehicle Data Access Architecture**
type: technical brief
content:

- External Data Server concept, pros and cons, why this is the baseline concept
- Internal Data Server concept, pros and cons
- SomeIP inside the framework, pros & cons

Deliverable #2 - **Vehicle Data Access Permission Management**
type: technical brief
content:

- External Data Server concept, permission management baseline
- Internal Data Server concept: permission management as a variant of previous one
- SomeIP inside the framework concept: permission management (TBC)

Deliverable #3 - **Vehicle Data Access Proof-Of-Concept Bills of Materials**
type: technical brief
content:

- External Data Server concept - component & technology selection
- Internal Data Server concept - component & technology selection
- SomeIP inside the framework concept - component & technology selection

- Question: is there a component for which it would be worth writing formalized specifications ?

16:00-16:05	Vehicle HAL – recap for the Audio HAL participants • Alex introduces the work breakdown structure for priority 1 PoC (external data server)	
	Vehicle HAL Track ends	
10:10-12:15	**Audio HAL Track**	
	Agenda for this track follows the List of prioritized topics for the Audio HAL	
	Introduction to Android Automotive Audio • Bartosz (Tieto) introduces current audio system, including some various issues using this AA Audio-Source Management.pptx • There must be a primary audio HAL this is typically used for car speakers other HALs exist (USB audio etc) • !!! The default is that every output is mixed and heard. • For automotive, added: Audio contexts. Provide additional info to HAL about the source, • so that HAL impl can modify mixing rules etc. • In automotive, it is not really expected to handle volume in the source side but rather in the HAL, or possibly external amplifier etc. • Audio HAL can mix internal and external sources easily... • Google interfaces exist, are defined. Vendors often provide some extensions, because there were not. • Sachin: Native Android AOSP provides some implementation of the HAL though, right? • Some parts of the implementation thus exist... • Piotr: Yes, a kind of reference implementation, e.g. for emulator • Android provides a HAL abstraction. • Android can be updated without changing the HAL. • Proper ALSA configuration should make it work. • Reference implementation works with emulator, but does not appear to be a lot of work into making it production quality and flexible enough for production, etc. • TinyALSA is optimized, unnecessary things removed... • System Player - the visible player is a kind of skin/UI on the system player. ◦ Including the codec support.	
	(Point 11) **Source Management** • External source (like external Tuner chip) needs to be input to AHAL • HwAudioSource configures some "triggers" .e.g to set up the correct mixing inside AudioFlinger • The audio stream enters the HAL however. • There are plans to improve volume control for external sources. These are TODO comments in code.	

	Audio Capture
	• Discussing different ways (before and now) for listening for HotWord ("Hey Google", "Siri", "Alexa, ")
	• Pre-Android 10
	  ◦ One active capture client at one time created limits.
	  ◦ HW SoundTrigger used.
	• Android 10
	  ◦ AHAL now required to allow simultaneous activation of input streams
	  ◦ but still two audio apps cannot record at the same time (background apps are still connected but receive silence)
	• Some privileged applications can always get the audio stream (e.g. for hotword recognition)
	• Vendors can create privileged applications if they want.
	• We think it is intended behavior to not allow applications to record (i.e. to make sure they receive silence)
	• Possibly an app records the phone call in background.
	• Piotr: Documentation usually mentions the new feature/change, but to understand the logic you have to read the code.	

Android 10: Multi-Zone audio

- Application can be played in a zone
- A zone contains audio devices
- A zone has separate volume
- A zone can be requested by app
- Not yet any automatic mapping of app to zone (based on which display was used) (maybe it was fixed in updates of 10, or in later release)
- HW volume keys controls primary zone only
- Primary Zone is typically used for driver screen
- Secondary Zone typically rear-seat
- Volume to gain mapping curves
  - This exists in Android (phone) but not used in Automotive
  - The mapping is a problem because it only has min, max, and step but realistically you want non-linear gain.
  - Is it OK for Audio HAL to apply the non-linear system ?

Pre-android 10 AudioFocus

- not enforced (applications must respect the setup)
- only phone calls prioritized but not several levels of phone call types (for example)

Android 10: CarAudioFocus

- Internal interaction matrix (currently fixed ⚠️)
- Support multi-zone audio (maintains focus per zone)
- Interaction Matrix – Reject, Exclusive, Concurrent
- Can the Caraudiofocus interaction matrix be modified ?
  - (Yes) Only by modifying the AOSP (system partition) – part of car service
  - System update would overwrite the change.
- Now: Will the matrix be customized for OEM production projects?
- Later: Will it become possible to configure for the vendor/OEM?
- Discussion: This changes behavior of the system, therefore it might be in Google's interest to make it as fixed as possible (according to the non-fragmentation / identical behavior direction Google prefers for Android).
- Configurability of the matrix is desired! Also, potentially more categories than currently.
- Radio support is relatively poor currently.
- There is a concept of global effects. No real way to control them.
- Control panel can be used to modify the effects (but this is a user feature).
- It is lacking system calibration features.
- TODO What is the shortlist of issues that must be solved? (Gap analysis)

(Point 7) **Global Effect HAL**

- Assuming that the other APIs are stable, and sources are connected to ALSA and working then we could define global effects.
- We should have (create) a HAL API to modify the global effects.
- See Architecture picture in "Global Effect HAL" AudioEffects.pptx and look at the following components

1. Global Effect HAL
2. Global Effect Service

- Both of these are proposed to be "GENIVI implemented" blue components (the meaning of this is only that there is a potential for a common implementation across many systems, and thus could be done once, as open source).
- Gunnar: if these can be implemented once for all systems, why are they part of HAL from Google perspective? (In other words, if it could be common, why would it not be part of System Partition / AOSP, a.k.a. Green code?)
  - Piotr: Yes, there are slight differences (e.g. DSP choice) but presumably some of the implementation could be common, and an abstraction to the system/DSP specific parts could be provided as part of this component.

More questions

- Question: Let's say you install Spotify, how can you integrate the desired behavior?
- Piotr: Simply, Spotify uses the primary audio device today.
- Driver can route output to a display or audio zone (e.g. to a particular headphone jack)
- TV use cases (video/audio mix). i.e. lipsync feature.
- Audio setup directly influences the user experience.

12:45-13:00

Audio HAL recap

- Sachin: very complex topic, too many open points
- Sachin: from the HAL perpective, we will get this from the vendor
- Sachin: Tieto made a proposal for standardization between audio HAL and audio services
- TV - Gunnar: we open up the topic but need to further discuss it
- audio is strongly related to user experience and therefore an important topic for OEMs

13:00-13:45	Lunch break	
	Multi-source management: Multi-source multi-sink	

	(Point 1) Networked Audio
	
	• What's the concrete question here?
	• Idea: Answer this question: "Is AVB supported well in Android?"
	• investigate AVB for the purpose of learning how a network audio system would be designed
	  ◦ (because MOST is not so popular).
	• A2B based networks are an option.
	• What other technologies are similar and would drive us towards better understanding ?
	• What is meant by device in the question asked to the F2F ? Device = speaker, or device inside Linux/Android system, a sound card or another ECU
	• Hypothesis: ALSA interface is appropriate (enough*) abstraction for both the control question and the stream. If this is not true, someone provide counter-examples. Concrete example needed of either the architecture we want to solve, or the use case.
	  ◦ *However, things like the added latency of the ALSA driver in question might be required to know...	

	(Point 4) Audio data transfer • Bandwidth issues? • Piotr: You normally would push this over a socket of some sort. There is generally no bandwidth problem even pushing multiple PCM streams across a socket. • (Summarized discussion) • Shared memory? There should not be an issue to just use shared memory mechanism the Linux kernel provides to transfer data between processes • However, if we are speaking of transfer between different hardware, or a virtualized system, it is not obvious but at the same time, solutions exist. • To what level is this intended to be answered? Some aspects are clearly hardware / platform specific. Others ought to be "known" (i.e. the Linux kernel API and what is theoretically possible to do). To go deeper would require a concrete shared open-source implementation. Is this what the group expects?	

	(Point 5) Equalization
	
	• Can be part of the bigger topic global effects (Point 7). Concrete proposal from Piotr to define the API of a HAL for global effects.
	• "There is a need for a generic interface for controlling audio effects at HAL level, global effects are designed for input streams but control over them is limited by the available interface."
	
	1. The user controlling the effects (setting surround sound etc.)
	2. Globally applied system calibration is different (done by OEM/supplier at integration phase). This too needs more support. It must be very configurable.
	  • (A system calibration file could be inserted in HAL level without crossing the HAL boundary (not needed in the system)).
	  • For development (maybe even in production ?), the calibration data might need to be tweaked in real-time, access limited and not available to user.
	  • It would be easier if it is part of HAL API for that reason.
	
	• Both of these aspects should be considered when designing the system.
	• All in all, the system needs a much more generic interface with more capabilities.
	• Next step: Just do it....	

	(Point 9) Multiple audio channels • "The challenge is to adapt the AA framework to HW I/O, e.g. there are 4 audio channels that need to be presented as 2 audio channels to AA" • We don't know what exact issue this is referring to? • Henric: It might have been an isolated thing we had to do in a previous project. • There are 8 channels in Android.	

	(Point 15)
	
	• (Virtualization, and splitting audio function between Android and "safety OS") may lead to scheduling challenges.
	• Yes, this is known, but it is hard to say how to solve without a concrete case.
	• Are there any features that might help, in general ?
	  ◦ Process Priority.
	  ◦ Kernel tweaks.
	  ◦ Fixing/rewriting kernel drivers.
	  ◦ Hypervisor specific issues...	

	(Point 14) Noted phone call priority above in the discussion we already had. (Point 16) This is just a description of how things are. We can't really change it. (In some other points we mentioned places where limitations require changes in the framework anyway, even if you are not supposed to)	
	(Point 13) Combination of BT stream with other streams? • We are not sure what the problem is exactly. Probably it is the general question about audio routing. • Connect microphone and speaker to the BT input and output. Handsfree function. • General problem with not good documentation for the Audio HAL (and lacking features). Only after the bus devices were introduced, then this was possible to solve. We set up device types using call_rx/tx but those come from the cellphone call audio, which is really a different use case. • With dynamic routing enabled this is still a problem. • Wassim: In addition there is audio processing involved including echo cancel, noise reduction. • Bartosz: There are interfaces prepared for this. • Some improvements in Android 10, e.g. a reference echo stream is handled. • We suspect really advanced setups might not fit into this. (More than one microphone, more than one speaker stream that needs to be affect cancellation in different ways).	
	(Point 12) "Audio focus can be lost at any time." • A bit improved in Android 10 with the interaction matrix • Already covered before. (Point 8) Latencies • Agreed. Android has quite an inefficient audio stack and latencies are an issue. • AAudio - https://developer.android.com/ndk/guides/audio/aaudio/aaudio ◦ this is made to rewrite and improve performance. A kind of direct API towards ALSA (for system programming, but could be used by Applications through NDK) "Designed specially for applications that require low latency"	

	Concrete development plan
	
	Global effects
	
	1. Implement a simple proof of concept
	2. Make a list of all possible effects
	3. Design the Global Effects API using experience from 1/2. (code first)
	
	• A first simple effect is filter/equalizer
	• Proves ability to manipulate the sound in real time.
	• It exists but no simple way to apply it globally. Effects are provided but each application sets their own effect settings.
	
	Multi-zone connected to displays
	
	• Not yet connected displays
	• Implement the possibility to change the audio zone system-wide
	  ◦ (e.g. force an application's sound to a particular zone)
	  1. Implement a simple proof of concept
	  2. Need to track carefully the changes Android will make here because there is a plan to connect zones and displays...	

16:00-16:15	recap (Gunnar)
	
	• 2 ideas of concrete projects came up
	  ◦ General Audio Effects service interface
	  ◦ rerouting audio streams	

	Audio HAL Track ends	
	End of Day 2	