High Level Audio API and Policy Proposal

October 19th, 2017
François Thibault, Audiokinetic
Tai Vuong, Audiokinetic
Audio Architecture Proposal Overview

- Application UI Bindings
- Application Bindings
- Application Interface Binding
- Audio Policy Binding
- Hardware Abstraction Binding
- HW-specific Bindings
- Kernel Drivers
- Hardware
AGL Audio Agent Layers Proposal

High Level Audio Binding
- Single entry point for all audio applications needs with simple, stable interface
- Expose all device capabilities in uniform way to applications
- Allow fine grain security permissions control, policy enforcement and provide isolation between different application audio stream controls
- Priority-based and audio role specific endpoint selection / stream routings (automatic or explicit) and aggregation of different audio domains (ALSA, Pulse)
- Audio stream and endpoint controls (volume, mute, state, properties)

Audio Policy Binding
- Customized audio business logic (audio role specific ducking rules, interrupt behaviors, …)
- Implement audio actions influenced by vehicle information (e.g. ALC)
- Dispatch policy actions to different low-level audio frameworks

HW Abstraction Binding
- Provide portability of audio implementation across different audio hardware
- HW control ID mappings to expose standard control set
- Dispatch to HW specific binding for additional functionalities

Hardware Control Bindings
- ALSA core → generic ALSA hardware controls
- Implement/expose additional hardware capabilities (e.g. ADSP or Unicens)
High-level Audio Binding API Concepts

• Audio roles (e.g. entertainment, warning, communications, etc.)
• Audio endpoints (source and sink endpoints)
  – Provide applications display name for device (e.g. UI selection)
  – Provide applications device URI to stream to selected endpoint
  – Automatically retrieve associated volume control for ALSA softvol URI
  – Volume and properties (numeric (e.g. balance, EQ), or string (e.g. preset))
• Audio streams (audio role assignment)
  – Stream state (e.g. idle/running/suspended)
  – Stream mute state
• Sound events (audio role assignment)
  – Integrate sound generation with audio stream management
    • Connect to a custom renderer (e.g. HMI events, startup/ending sound, etc., AVAS, …)
High Level Audio Binding Audio Routings

• Audio role specific audio endpoint enumeration and monitoring
• Device routings (automatic or explicit)
  – Provided with audio role and endpoint type
  – Selected according to config priority (and optionally current state/concurrency information)
  – Return appropriate device URI to application
  – Return target endpoint for volume/property changes
• Dynamic device handling and re-routing currently missing
High-level Audio Binding Configuration

Simple audio role based configuration
• Preferred routings (for automatic endpoint selection)
• Interrupt behaviors
• Role priorities
• Supported events

```
"policy_module": "AudioPolicy_v1",
"audio_roles": [
  {
    "name": "Warning",
    "id": 0,
    "description": "Safety-relevant or critical alerts/alarms",
    "priority": 100,
    "output": [
      "alsa.plug:Warning_Main",
      "alsa.plug:Warning_DriverHR"
    ],
    "events": [
      "emergency_brake",
      "collision_warning",
      "blind_spot_warning"
    ],
    "interrupt_behavior": "pause"
  },
```

AGL AMM Dresden
Permissions, Role Privileges and Access Controls

API verbs permissions

- Stream control → Stream start/pause/resume/mute/unmute,…
- Audio streaming → Stream open/close
- Sound event → Trigger/notify about audio asset playback
- Currently monitoring is allowed for everyone (but can be changed)

Role privileges

- Different levels of privileges based on roles also possible

Access controls

- Application can only control/affect stream and endpoints on which they have ownership
  - Reduce potential side effects, enforce role of policy
High-level Audio Binding Policy Module

- Audio role specific priorities and interrupt behaviors provided by high-level binding config file
- HLB expose relevant state information to policy module
- API verbs that affect state of audio streams or endpoints must go through policy first
  - Policy can accept or reject the change
- Policy implements custom business logic e.g.
  - ducking, state changes, forbidden behaviors etc.
- Policy actions are dispatched to appropriate low-level technology
API Overview

• Endpoint enumeration
  – GetSources / GetSinks → for explicit routing

• Stream and routing management
  – StreamOpen / StreamClose → application streaming (e.g. media player)
  – Stream open with source and sink can be used for routings (e.g. handsfree)

• Stream control
  – Get/SetStreamState → Transitions from idle, running, suspended
  – Get/SetStreamMute

• Endpoints (source or sinks)
  – Set/Get Volume
  – Set/Get Properties
  – GetListProperties → capabilities

• Sound events
  – PostSoundEvent → Sound generation services
  – GetListEvents → Configuration defined available audio events

• Events
  – Endpoints volume/status/property changes (e.g. from policy application)
  – Endpoint availability changes
  – Audio streaming changes (start/stop/pause/resume, etc.)
  – Stream/routing activity changes (endpoint URI changes)
Simple API Usage (Start Playback)

- If explicit device selection:
  - GetSinks('entertainment')
    - List available endpoints for role
  - StreamOpen('entertainment', 'sink', [endpointID])
    - StreamOpen(PolicyCtx)
      - Allowed/denied

- If app initiated stream state change:
  - SetStreamState(streamID, 'running')
    - Allowed/denied
  - SetStreamState(PolicyCtx)
    - Allowed/denied

- If state change allowed:
  - <<StreamEvent(Start)>>
    - Application starts audio streaming

Client automatically registered for stream events concerning successfully opened
Possible to do on stream start if preferred
Simple API Usage (Stop Playback)

Pause/Resume sequences would be similar (without stream close)
Demo Architecture

[Diagram showing the Demo Architecture with components such as AGL Native Audio Apps, HTML5 UI, Application Business Logic, ALSA Audio Backend, High Level Audio Shadow API, AGL security Framework, Renderer, High Level Audio API, Audio Policy, HAL (Hardware Abstraction), USB, Bluetooth, Renesas, Intel, ALSA Core, Libasound, and Sound Cards.]

13
Demo

• Use of audio role specific software volume controls
• Endpoint / zone selection with configurable device priorities
• Audio role priorities (ducking and interrupt behaviors)
• Sample policy
  – Volume management
  – Volume acceleration
  – Active source change
  – Active source locking
  – Source interrupts

Requirements and scenarios from
https://wiki.automotivelinux.org/eg-ui-graphics-req-audiorouting
High-level audio binding and sample policy
https://github.com/Audiokinetic-Automotive/afb-audiohighlevel
Sample configuration and demonstration UI and assets
https://github.com/Audiokinetic-Automotive/ak-demo
Demonstration audio back-end (simple ALSA renderer)
https://github.com/Audiokinetic-Automotive/afb-audiobackend
Some changes and additional HAL implementation for demo
https://github.com/huetaivuong/afb-aaaaa

Please provide feedback!