
IOT
BZH

Table of Contents

Introduction

Meta	*Data*
Title	AGL_4A QuickStart
Author	IoT.Bzh Team
Description	This is a quick guide on 4a
Keywords	AGL, Development, Iotbzh
Language	English
Published	Published August 2018 as an electronic book
Updated	Tue Aug 07 2018 10:28:33 GMT+0200 (CEST)
Collection	Open-source
Website	http://iot.bzh

Play Music

Aknowledge roles

To play music you must know which roles you want to use, for music it's going to be the multimedia role.

To display the available roles enter:

```
4a-api roles
```

Play command

4a-play must be used like this.

```
4a-play <file> [role]
```

So in our example it will look like this:

```
4a-play Happy_MBB_75.ogg multimedia
```

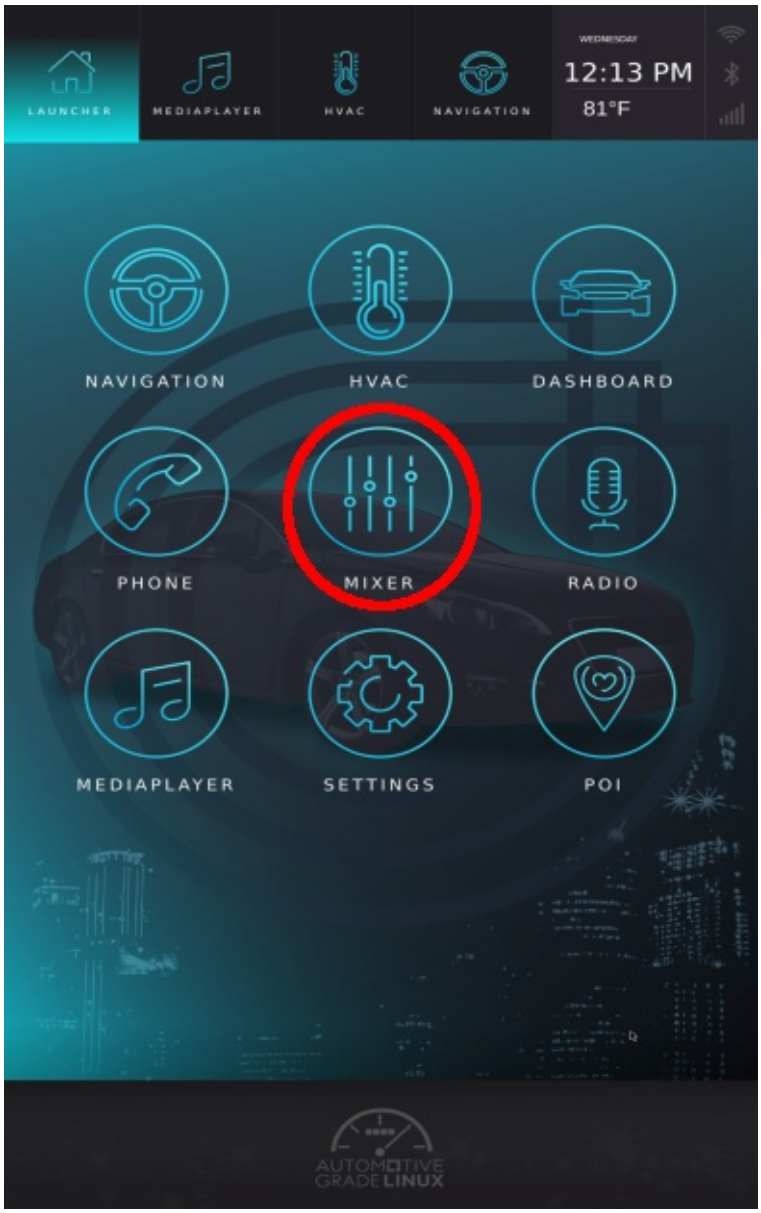
Where *file* is the path to the file that you want to play, *role* is the role of the 4a-play (multimedia for example).

> ***NOTE**: For AGL versions before FF.RC3 you had to specify the device to use. The device that matches with "multimedia" is "hw:0,0,2" (for more details please read the "Get HALs status" section). The command would have been :*
>
> ```
> 4a-play Happy_MBB_75.ogg hw:0,0,2 multimedia
> ```

Adjust the sound

Music is too loud ? What about adjusting the sound ?

On the main menu, hit the "Mixer" icon.



Then drag the cursor to change the sound volume of the wanted role.

Play multiple Sounds on different roles

While your music is playing you might want to simulate an emergency sound, to do so, in a new terminal, enter:

```
4a-play emergencySound.ogg emergency
```

Your music should automatically be lowered so you can hear the emergency sound. As your emergency sound turns off, your music returns automatically to the previous volume.

> ***NOTE**: For AGL versions before FF.RC3 you had to specify the device to use. The device that matches with "emergency" is "hw:0,0,4" (for more details please read the "Get HALs status" section). The command would have been :*

```
4a-play emergencySound.ogg hw:0,0,4 emergency
```

Tips & Tricks

In this section you'll find usefull tips and tricks to help you in case of trouble.

- Get Roles list
- Get Service Status
- Check API
- Get HALs Status
- Get Available Devices
- Change Hal

Get roles list

`4a-api roles` gives you the list of different roles in a json format.

```
$ 4a-api roles
Detected systemd unit file!
Port detected: 1025
ON-REPLY 1:ahl-4a/get_roles: OK
{
  "response":[
    "radio",
    "multimedia",
    "emergency",
    "navigation"
  ],
  "jtype":"afb-reply",
  "request":{
    "status":"success"
  }
}
```

Get service status

4a-status gives you the 4a status.

Here is an example when the service is running fine:

```
$ 4a-status
---- Audio cards detected ----
card 0: Loopback
card 1: Intel

---- snd-aloop driver availability ----
SUCCESS: Built into the kernel

---- 4a service status ----
SUCCESS: Service is currently running!
```

And now an example when an error happened during service startup:

```
---- Audio cards detected ----
card 0: Loopback
card 1: Intel

---- snd-aloop driver availability ----
SUCCESS: Built into the kernel

---- 4a service status ----
WARNING: Service is not currently running!
It can be started using the following command:
systemctl restart *agl-service-audio-4a*.service
```

Check that an api is responding

You might want to test if an api has not crashed or is running, to do so use the command:

4a-api api ping

For exemple :

```
$ a4-api api smixer ping
Detected systemd unit file!
Port detected: 1025
ON-REPLY 1:smixer/ping: OK
{
  "response":3,
  "jtype":"afb-reply",
  "request":{
    "status":"success"
  }
}
```

Softmixer is responding !

Get HALs status

`4a-api hals` gives you the ready-to-use hal(s) on your system.

Following, an example of this command on my system:

```
$ 4a-api hals
Detected systemd unit file!
Port detected: 1025
ON-REPLY 1:4a-hal-manager/loaded: OK
{
  "response":[
    "4a-hal-intel-qemu"
  ],
  "jtype":"afb-reply",
  "request":{
    "status":"success",
    "info":"Requested data"
  }
}
```

And now an example when an error happened during service startup:

```
Detected systemd unit file!
Port detected: 1025
ON-REPLY 1:4a-hal-manager/loaded: OK
{
  "response":[
  ],
  "jtype":"afb-reply",
  "request":{
    "status":"success",
    "info":"Requested data",
    "uuid":"magic"
  }
}
```

In my case, I have only one hal that can be used.

To get more information about this HAL you can use this command:

`4a-api api <api_name> info`

In this case my api name is "4a-hal-intel-qemu", so the command will be

```
$ 4a-api api 4a-hal-intel-qemu info
Detected systemd unit file!
Port detected: 1025
```

```
ON-REPLY 1:4a-hal-intel-qemu/info: OK
{
  "response":{
    "streams":[
      {
        "name":"multimedia",
        "cardId":"hw:0,0,2"
      },
      {
        "name":"navigation",
        "cardId":"hw:0,0,3"
      },
      {
        "name":"emergency",
        "cardId":"hw:0,0,4"
      }
    ],
    "playbacks":[
      {
        "name":"playback",
        "mixer-name":"INTEL-QEMU:playback"
      }
    ],
    "captures":[
      {
        "name":"capture",
        "mixer-name":"INTEL-QEMU:capture"
      }
    ]
  },
  "jtype":"afb-reply",
  "request":{
    "status":"success",
    "info":"Requested data"
  }
}
```


> ***NOTE*** *This is usefull for versions before FF.RC3, as you can see here you can get the cardId that matches with the corresponding role.*

Get available devices

Here you have two methodes, one using aplay and another using the infoget verb of the alsacore api.

Using aplay

```
$ aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: Loopback [Loopback], device 0: Loopback PCM [Loopback PCM]
  Subdevices: 8/8
  Subdevice #0: subdevice #0
  Subdevice #1: subdevice #1
  Subdevice #2: subdevice #2
  Subdevice #3: subdevice #3
  Subdevice #4: subdevice #4
  Subdevice #5: subdevice #5
  Subdevice #6: subdevice #6
  Subdevice #7: subdevice #7
card 0: Loopback [Loopback], device 1: Loopback PCM [Loopback PCM]
  Subdevices: 8/8
  Subdevice #0: subdevice #0
  Subdevice #1: subdevice #1
  Subdevice #2: subdevice #2
  Subdevice #3: subdevice #3
  Subdevice #4: subdevice #4
  Subdevice #5: subdevice #5
  Subdevice #6: subdevice #6
  Subdevice #7: subdevice #7
card 1: Intel [HDA Intel], device 0: Generic Analog [Generic Analog]
  Subdevices: 0/1
  Subdevice #0: subdevice #0
```

In this list you get the available devices with their subdivices.

Using alsacore api

```
$ 4a-api api alsacore infoget
Detected systemd unit file!
Port detected: 1025
ON-REPLY 1:alsacore/infoget: OK
{
  "response":[
    {
      "devid":"hw:0",
      "name":"Loopback",
```

```
      "driver":"Loopback",
      "info":"Loopback 1"
    },
    {
      "devid":"hw:1",
      "name":"HDA Intel",
      "driver":"HDA-Intel",
      "info":"HDA Intel at 0xfebf0000 irq 28"
    }
  ],
  "jtype":"afb-reply",
  "request":{
    "status":"success"
  }
}
```

Change HAL

Your HAL are known as json files in 4a, they are stored in */usr/libexec/agl/4a-hal*. Unused HAL are stored in *etc.available/* and used HAL in *etc/*.

To change you HAL, move your current one from *etc/* to *etc.available/* and your wanted one from *etc.available/* to *etc/*.

Example:

```
$ ls etc etc.available/
etc:
hal-4a-csl-cm106-8ch-usb.json

etc.available/:
hal-4a-2ch-generic-usb.json  hal-4a-intel.json           hal-4a-rcar-m3.json
hal-4a-ensoniq.json         hal-4a-jabra.json           hal-4a-rcar-m3kf.json
hal-4a-intel-minnow.json     hal-4a-m3ulcbkf-radio-to-2ch.json
hal-4a-intel-qemu.json         hal-4a-raspberry-pi-3.json
$ mv etc/hal-4a-csl-cm106-8ch-usb.json  etc.available/.
$ mv etc.available/hal-4a-2ch-generic-usb.json etc/.
$ ls etc etc.available/
etc:
hal-4a-2ch-generic-usb.json

etc.available/:
hal-4a-csl-cm106-8ch-usb.json  hal-4a-intel.json          hal-4a-rcar-m3.json
hal-4a-ensoniq.json           hal-4a-jabra.json          hal-4a-rcar-m3kf.json
hal-4a-intel-minnow.json       hal-4a-m3ulcbkf-radio-to-2ch.json
hal-4a-intel-qemu.json           hal-4a-raspberry-pi-3.json
$ sync
$ reboot
```

As you can see in the exemple, I wanted to have *hal-4a-2ch-generic-usb.json* enabled, so I swapped the hals. After it dont forget to sync and reboot.