AGL HMI Framework

Design Document

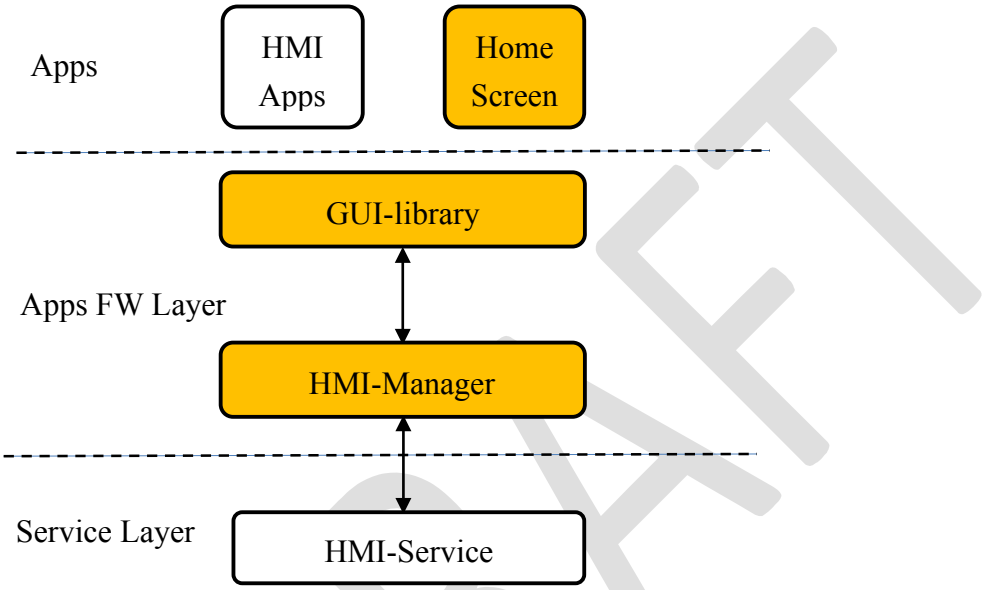
Version	Date
0.20	2017/XX/XX

Index

1. HMI Framework overview

1.1. HMI-FW Related components

The related components are shown below.
（Orange box components included in HMI-FW）



1.1.1. HMI-FW Components

Components of the HMI-FW are shown below.

Home Screen

Home Screen have an auxiliary screen other than the application screen and interact with the user.

There are various Home screens, but the following representative auxiliary screens are shown below.

- ✓ Short Cut Bar
- ✓ Status Bar
- ✓ Onscreen Bar

GUI-library

You can select the GUI-library (e.g. Qt, HTML5, JavaFX, EB) suitable for HMI with the software necessary for representing HMI.

- ✓ 2D/3D Graphics、Image Output
- ✓ Sound Output
- ✓ Input Event

HMI-Manager

HMI-Manager located between upper GUI-library and lower HMI-Service and has the following components for each HMI.

- ✓ Window Manager
- ✓ Sound Manager
- ✓ Input Manager

1.1.2. Related components

It is not included in HMI-FW, but related components are shown below.

HMI-Apps

An application including HMI (drawing, voice, input) processing is called HMI-Apps. HMI-Apps expresses HMI by calling components of HMI-FW.

HMI-Apps has the following responsibilities

- ✓ HMI-Apps is used after requesting the HMI resource required for HMI-Manager
- ✓ HMI-Apps will do the appropriate processing when the HMI rights are deprived from Manager

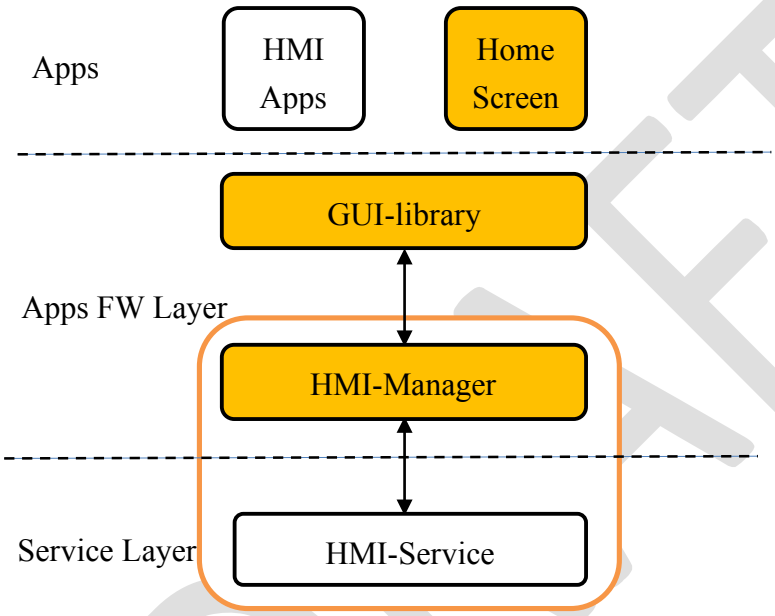
HMI-Services

It belongs to AGL Service Layer by HMI (drawing, voice, input) control software.

- ✓ Graphics Subsystem：Weston/Graphics Device Driver
- ✓ Sound Subsystem：Audio Manager/ALS
- ✓ Input Subsystem：T.B.D

1.2. Considerations on implementation

Since HMI-Manager often has different functions depending on OEM and system, it should be separated from HMI-Service.
However, if implemented according to this specification, the application calls HMI - Service twice, and performance and sequence issues remain.
Therefore, it is also possible to implement the integration of HMI - Manager and HMI - Service modules.

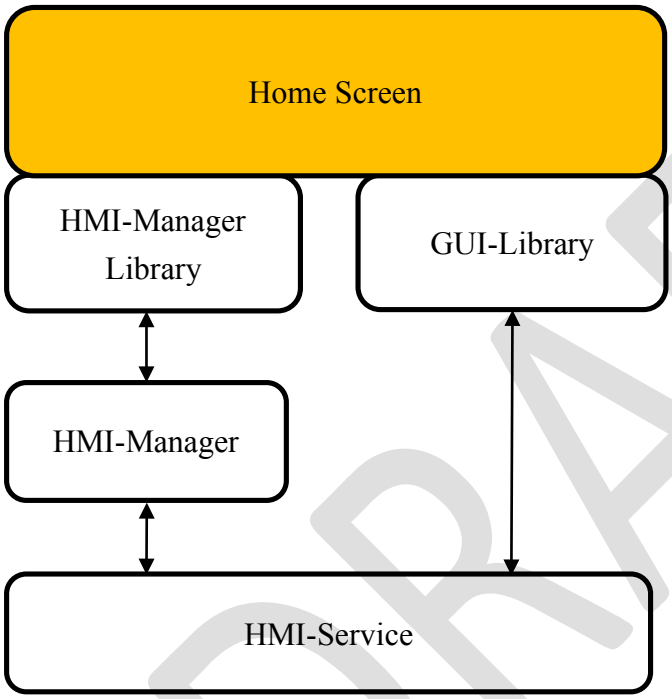


2. Home Screen

2.1. OverView

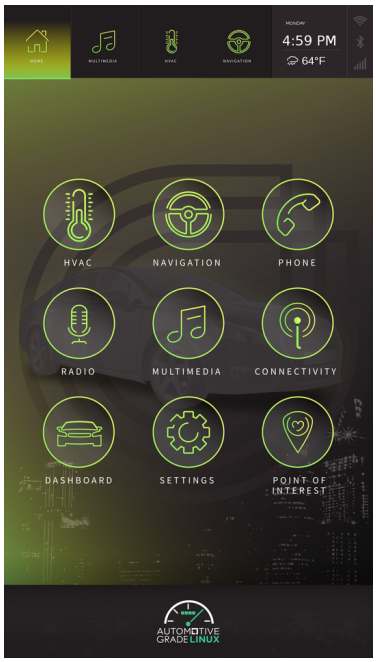
Home Screen is a component for performing user operation.
It is possible to have different Home Screen for each in-vehicle device.

2.1.1. Related external components



2.2. HomeScreen Basic Components（Sample）

The standard Home Screen sample is shown below.



2.2.1. Shortcut key

The user selects an application to use with apps menu.

2.2.2. Status Bar

The Status Bar shows status information by notification command from each application.

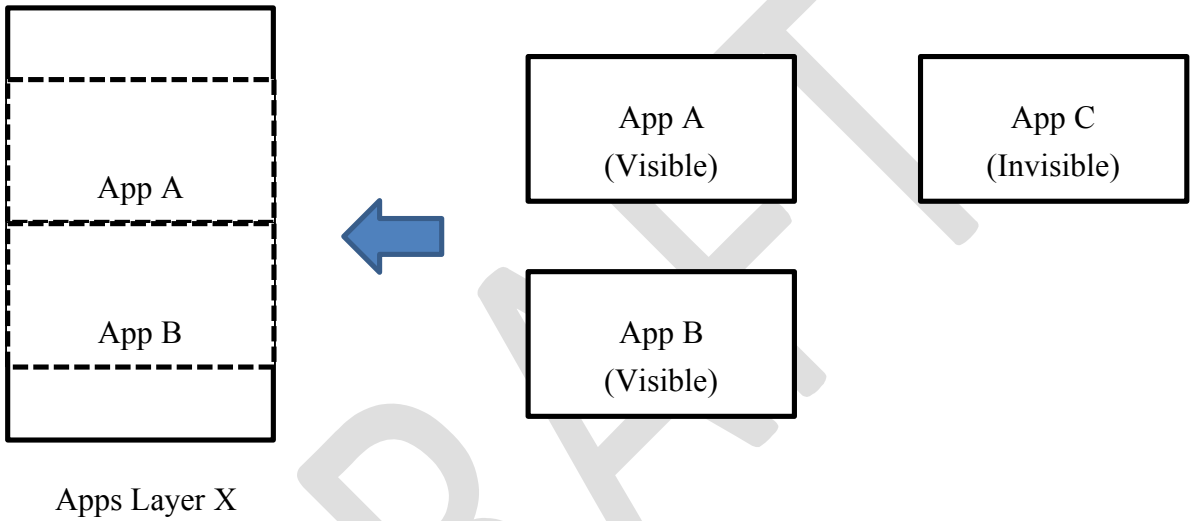
2.2.3. Apps Area

Apps Area is the area for the application to display. E

Share Apps Layer

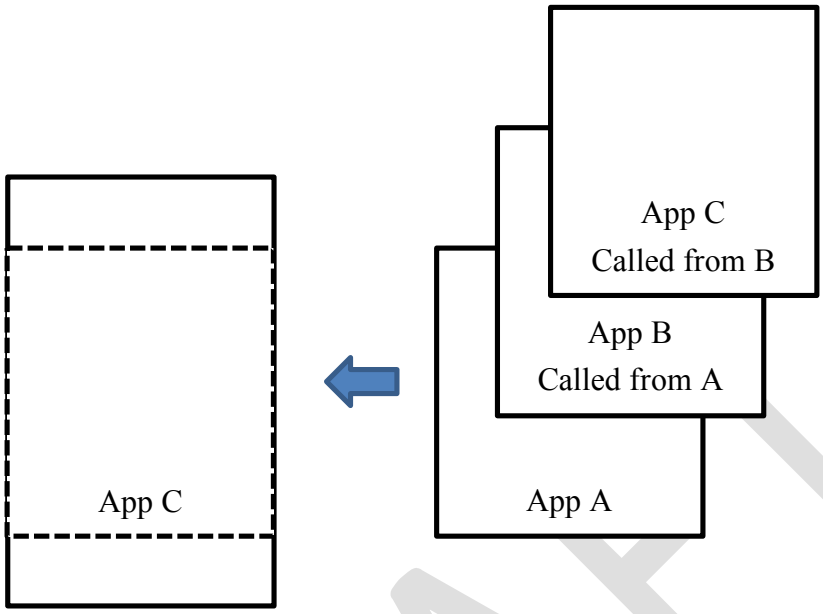
Each application needs to acquire screen rights to Window Manager.
Applications that lose screen rights are set to hide the area.

Depending on the screen size, multiple applications can share the Apps Layer.



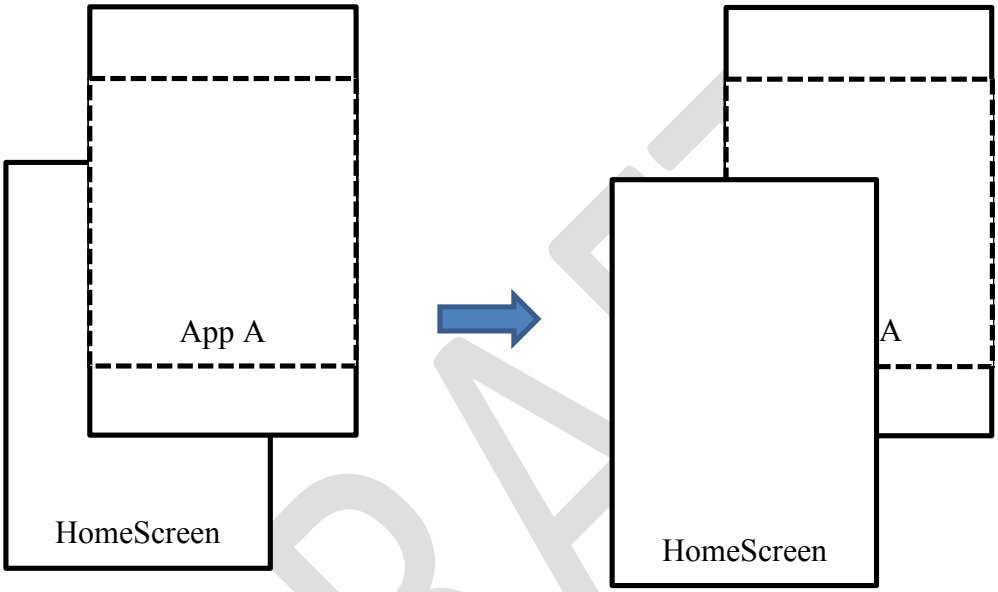
Stack Display (T.B.D)

When an executing application calls another application, another application occupies the screen.



2.2.4. Home Key

HomeScreen hides the current application and returns to the home screen window.
In the following example, the displayed application（app A） is not displayed and HomeScreen is displayed.



2.2.5. On Screen (Floating Area)

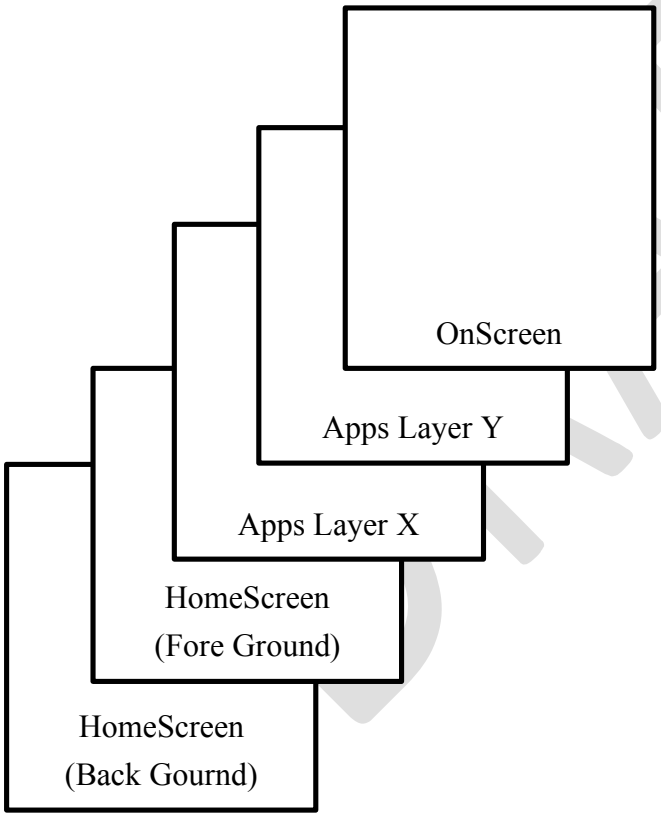
On Screen displays on the screen when notification from each application is received.
Each application needs to acquire screen rights to Window Manager.

2.3. HomeScreen initial processing

The HomeScreen performs the following processing at startup.
For details, see the chapter of each manager.

2.3.1. Initial setting of 「Window Manager」

Setting Layer
HomeScreen needs to set up multiple layers including applications at startup.
An example of setting of multiple layers is shown below.



An application may monopolize one layer or share it.

Setting Area

HomeScreen needs to set multiple areas for「OnScreen」and「HomeScreen」at startup.

2.3.2. Initial setting of 「Sound Manager」

2.3.3. Initial setting of 「Input Manager」

2.4. Apps shipped with HomeScreen

2.4.1. Apps launcher

The user can select necessary applications from the application menu.
The HomeScreen informs the application that it is selected.
If the selected application is not activated, the HomeScreen requests the application management to start the application.

2.4.2. Input Method Editor

The application can call IME with user operation.

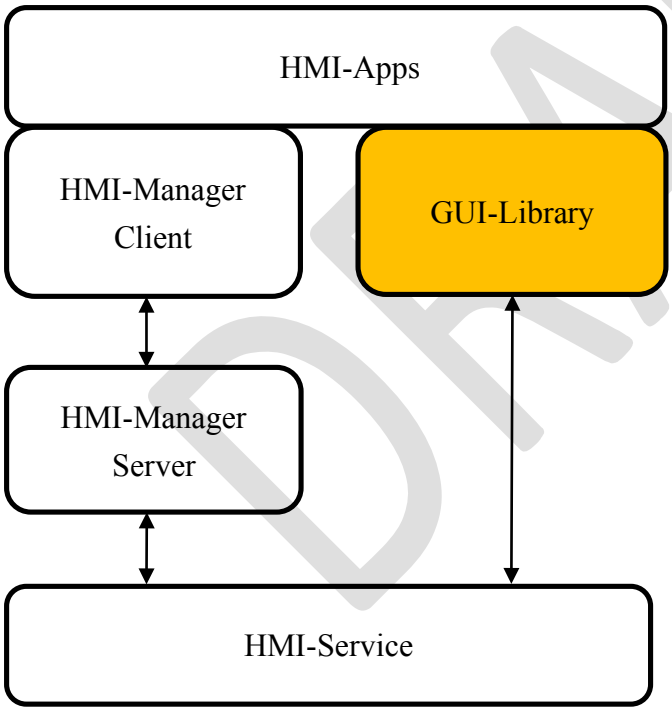
3. GUI-library

3.1. Overview

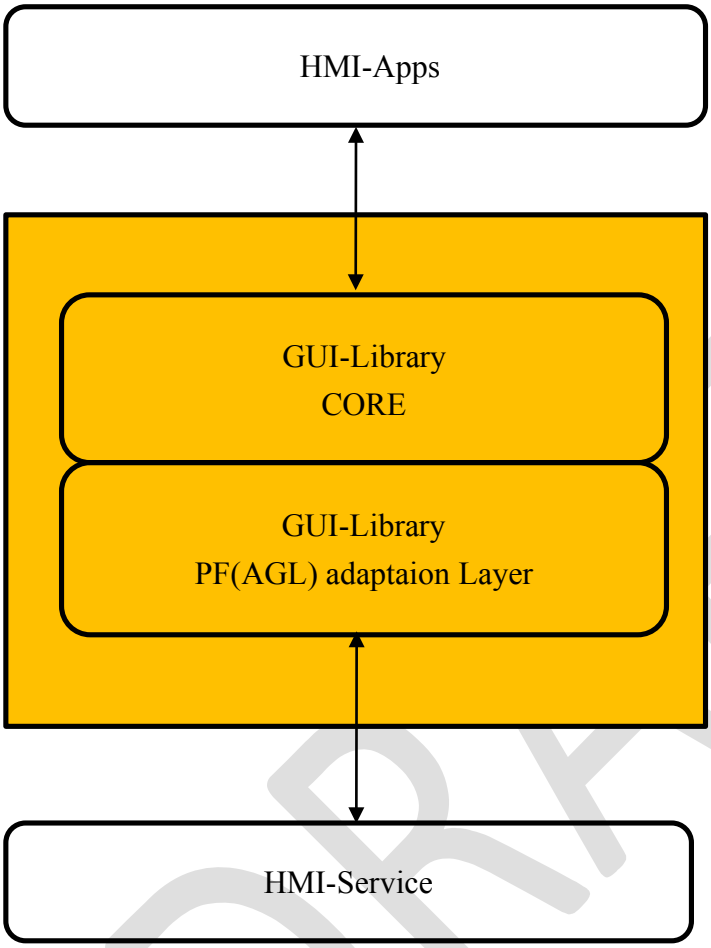
GUI-library is a library that provides HMI functions to applications, and mainly has HMI functions related to graphics, sound, and input.

3.1.1. Related external components

The application developer selects the GUI-library (e.g. Qt, HTML5, JavaFX, EB) according to the required HMI expression, and issues Upper API depending on each GUI-Library.
（As API functions depends on each GUI-library, refer to each specification.）



3.1.2. Internal Components



3.2. Graphics functions

Graphics provides rendering functions to the application.

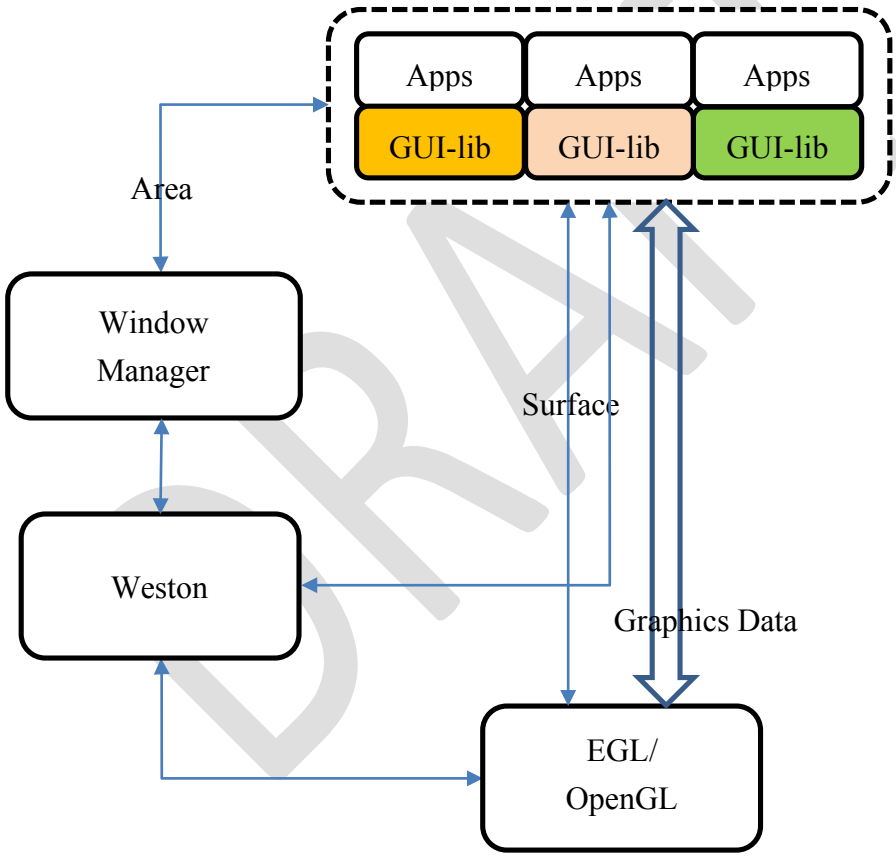
3.2.1. Procedure necessary for HMI-Apps

Graphics draws with the following procedure.

① The application requests Weston to acquire Surface

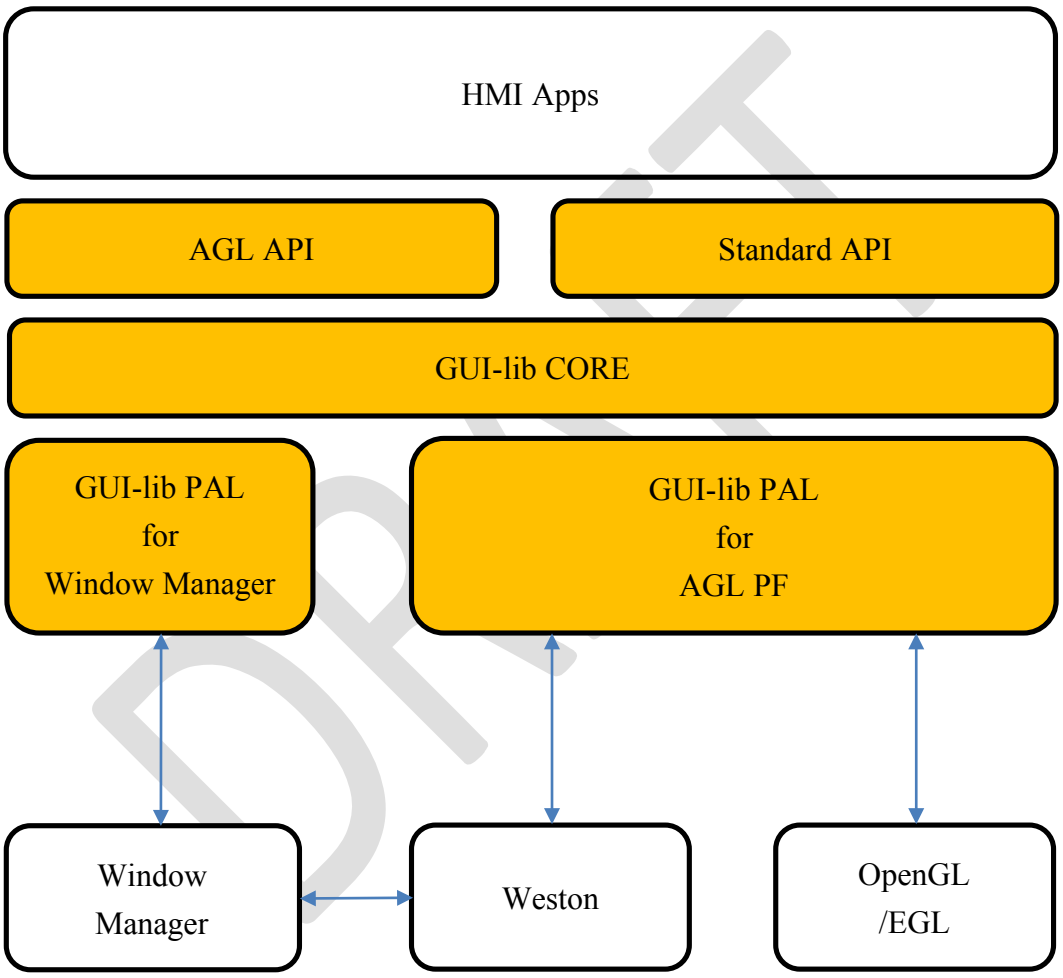
② The application makes Area request to Window Manager（OEM options）

③ The application inputs and outputs Graphics data with the Graphics Device Driver.



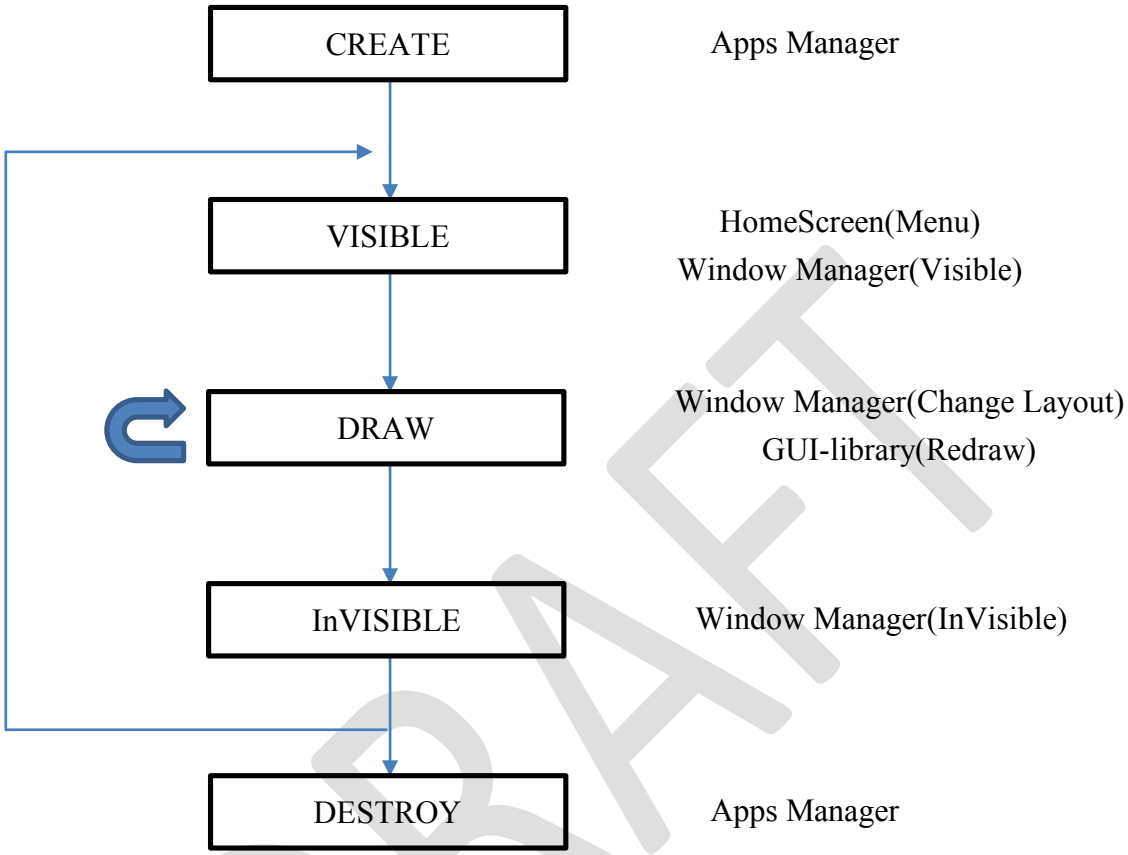
3.2.2. Software configuration of GUI-lib

GUI-lib has an API specific to AGL besides the standard drawing API.
Software vendors providing GUI-lib do not modify GUI-lib CORE、but need to delete functions other than GUI prescribed in AGL.
Software vendors need to remodel PAL(*) according to AGL.
(*) PAL = PF Adaptation Layer



3.2.3. HMI-Apps Drawing Life Cycle

HMI-Apps receives events from each component and performs optimum processing.



When receiving an event, HMI-Apps makes the following request to WindowManager.

① CREATE: Register My Application
② VISIBLE: Allocate Window Resources
③ DRAW: DrawEnd（When called from Window Manager）
④ InVISIBLE: Release Window Resources

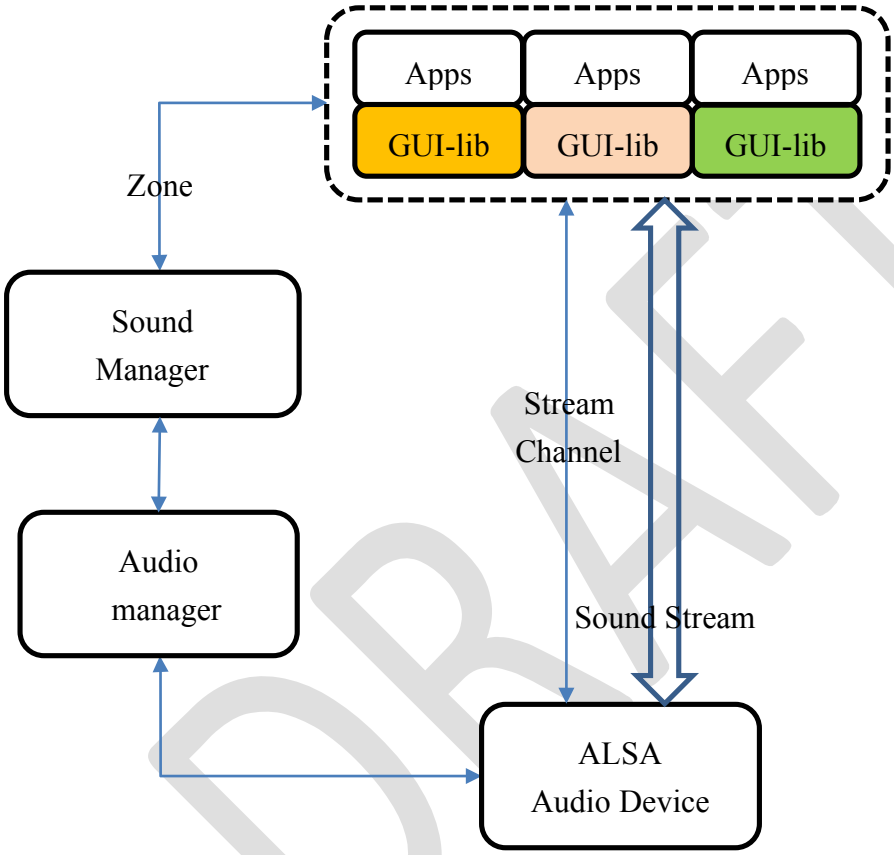
3.3. Sound functions

Sound provides sounding functions to the application with the following procedure.

① The application requests ALSA to acquire Stream.

② The application makes zeon request to Sound Manager（OEM options）

③ The application inputs and outputs Sound data with the Sound Device Driver.

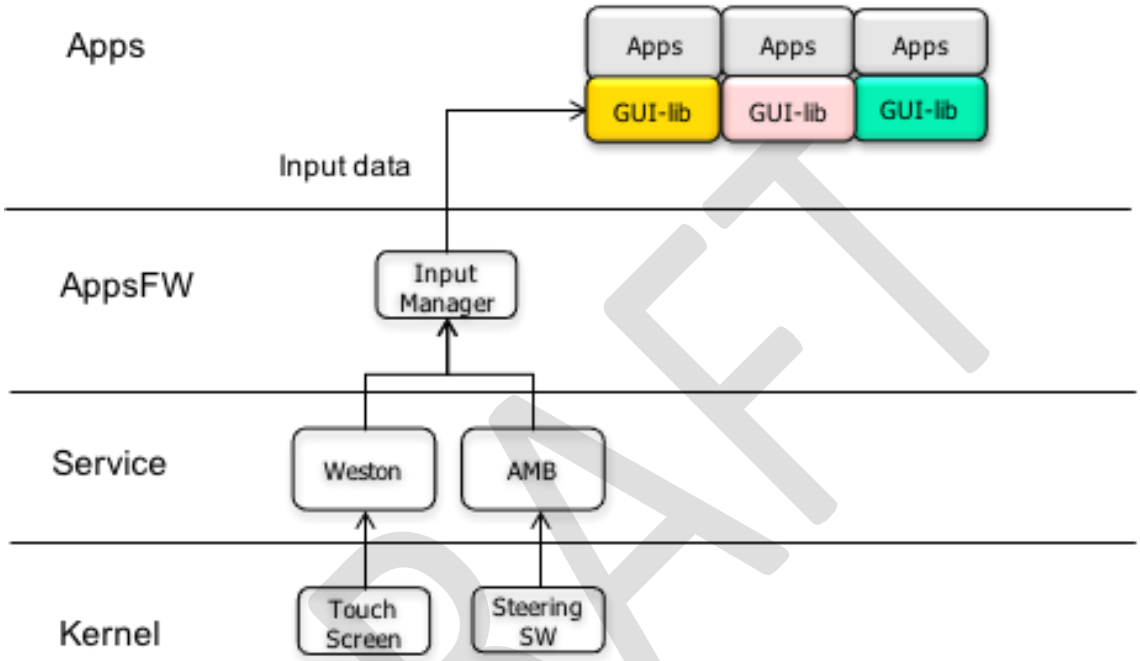


3.4. Input functions

Input provides Input data functions to the application with the following procedure.

① Input-Manager collects input data from each service.

② I nput-Manager determines applications to distribute data based on policy （OEM options）



Input Manager

3.5. GUI-lib Standard Funcitions List (Reference material)

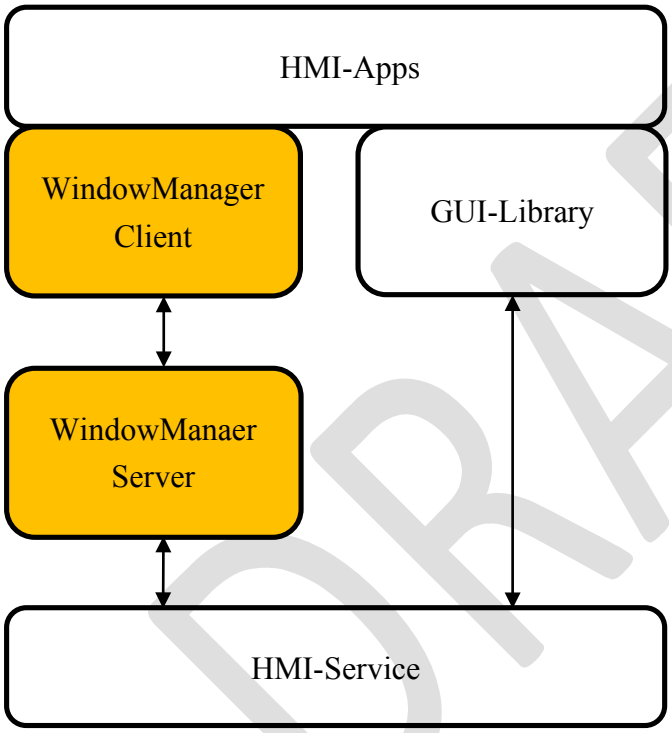
	Fuctions	Qt		JavaFX		Description
2D	Window	Qt GUI	○	Stage Popup-Widnow	○	
	Canvas	Painter2D WebView	○	Canvas2D WebView	△	
3D	SceneGraph	Material Transfrorm Animation Clip-Node Opacity	○	Camera/Light Transform Visual Effect Pick Sub-Scene	○	SceneGraph (Data Structure) neither Qt nor JavaFX is not Open.
	Graphics	OpenGL/ES Canvas 3D (WebGL)	○	2D Share 3D Share	△	
	ML	QML	○	FXML	○	
ETC	Package	Qt package	△	Java OSGI	○	
	MultiMedia	Audio Video Camera Radio	○	Audio Video ー ー	△	
	Input	Mouse Gesture	○	Mouse Gesture	○	

4. Window Manager

4.1. Overview

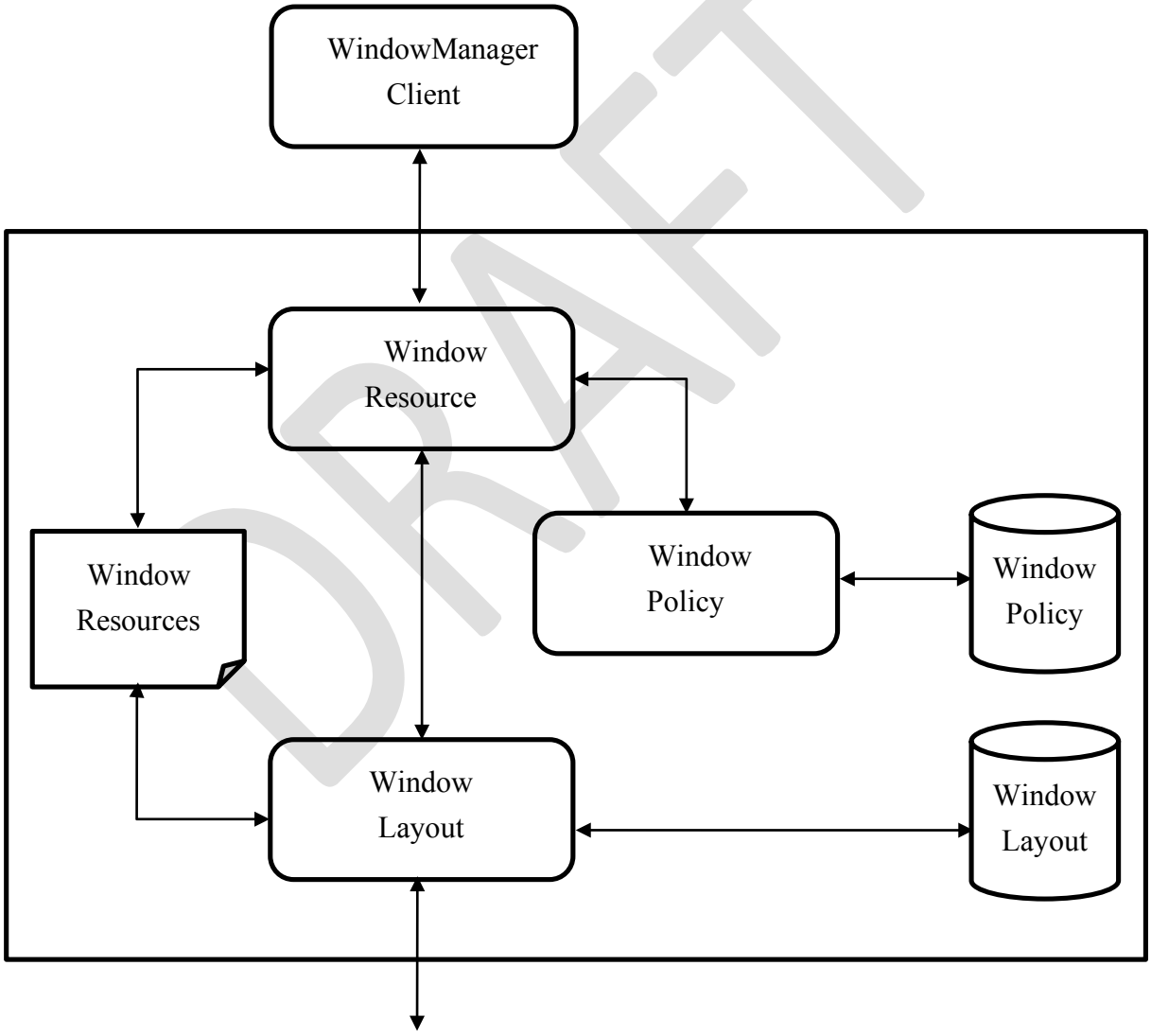
Window Manager determines the optimum screen layout and controls the screen、based on the request from the HMI-Apps.

4.1.1. Related external components



4.1.2. Internal Components

No	Function	Description
1	Window Manager Client	API
2	Window Resource Manager	Window Resource Management
3	Window Policy Manager	Mediation of Window Resources
4	WIndow Layout Manager	Window Layout Management



4.1.3. Window Resources

Window Resources are resource information related to the screen managed by the Window Manager and varies depending on the in-vehicle unit configuration (e.g. display).
The data items included in Window Resources are shown below.

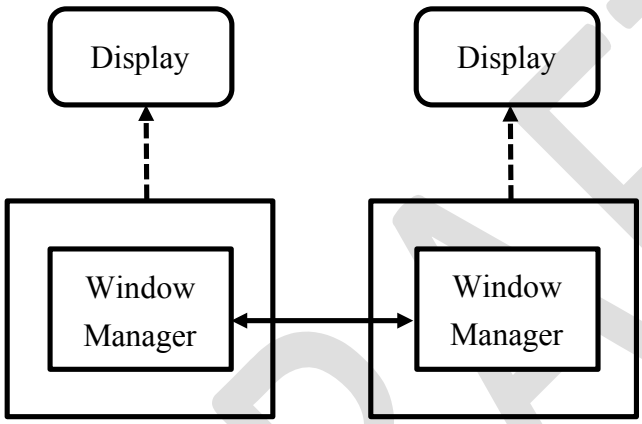
Display

Display has information on the display device.

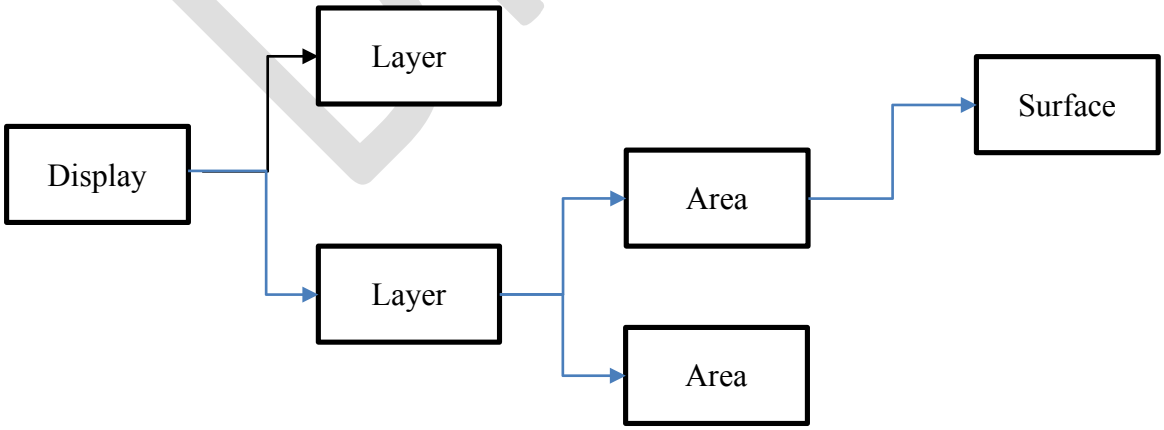
A Display can have multiple Layers.

No	Name	Information Source	Description
1	ID	Graphics Subsystem	Display ID
2	Name	–	Display Name
3	Size	Graphics Subsystem	Display Width and Hight

For vehicles with multiple displays it is assumed to have multiple Window Manager.

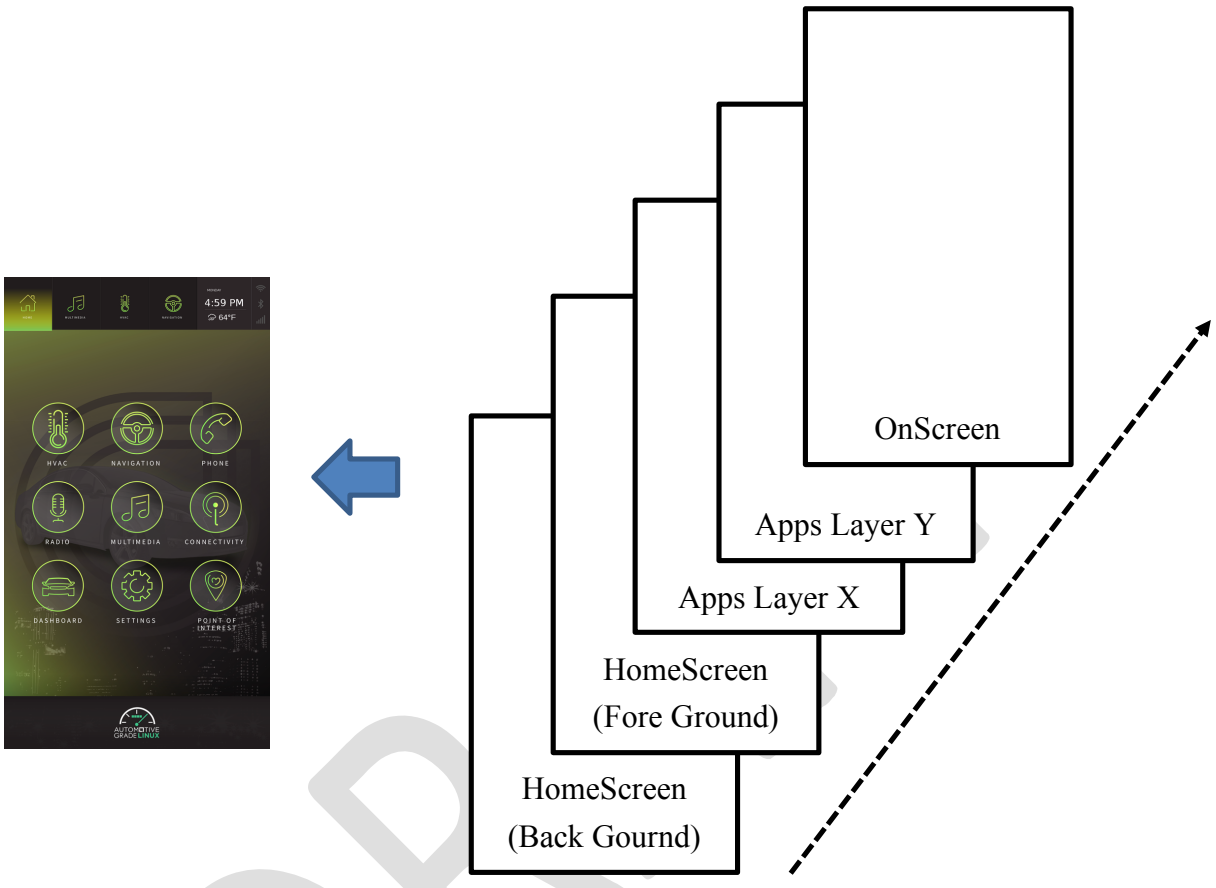


The Window Manager maintain the link state of the window resources.

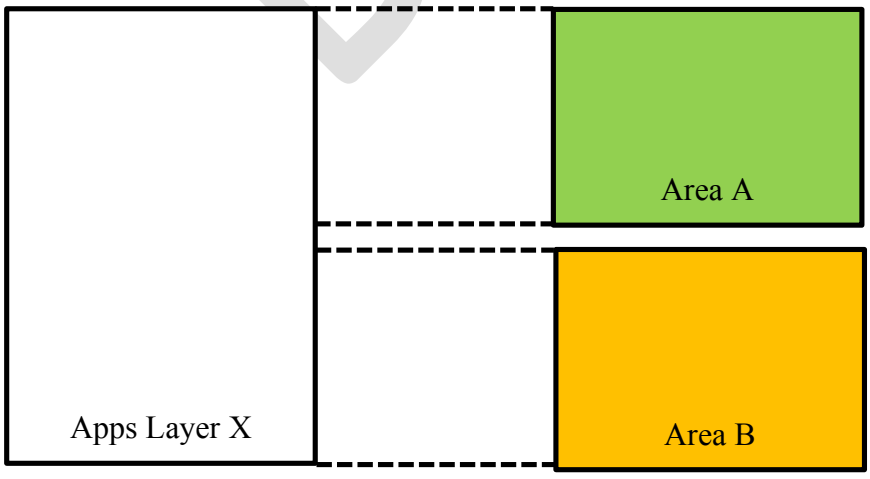


Layer

「Layer」 is the information representing the depth of display.



A layer can maps multiple areas.



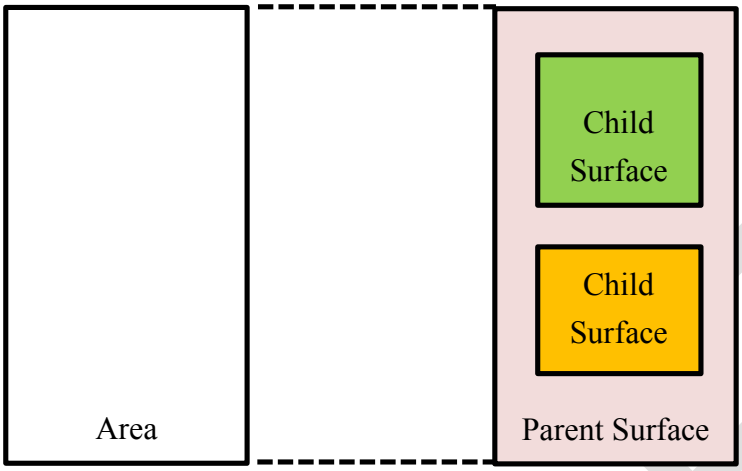
No	Name	Information Source	Description
1	ID	Home Screen	Layer ID
2	Name	Home Screen	Layer Name
3	Z order	Home Screen	Layer Zorder
4	Visibility	Home Screen	Layes Visibility Status
5	Alpha Blend	Home Screen	Layer Transparent Ratio (α Blend)

Area

The Area is the information of the area where the application draws.

A Area maps one Parent Surface.

Parent Surface can have multiple child Surfaces.



No	Name	Information Source	Description
1	ID	Application	Area ID
2	Name	Application	Area Name
3	AppID	Application	Application uniqu ID
4	Pid	Application	Application Process ID
5	Parent ID	Application	Parent Surface ID
6	Child ID	Application	Next Child Surface ID
7	Position	—	Area Position
8	Size	—	Area Width and Hight
9	Z order	—	Area Z order
10	Visibility	—	Area Visibility Status

Surface

Surface is information of display material frame buffer managed by Graphics Subsystem (Weston).

No	Name	Information Source	Description
1	ID	Application	Surface ID
2	SourceSize	Application	FrameBuffer Size
3	Position	—	Surface position
4	Size	—	Width and Hight
5	Z order	—	Surface Z order
6	Visibility	—	Visibility Status
7	Alpha Blend	—	Transparent Ratio (α Blend)

How to create Surface ID

Layer ID [31:24]	Area ID [23:16] (Parent Surface ID)	Surface ID [15:0]

How to create Zorder

Layer Zorder [31:24]	Area Zorder [23:16]	Surface Zorder [15:0]

4.2. Window Manager Client (API)

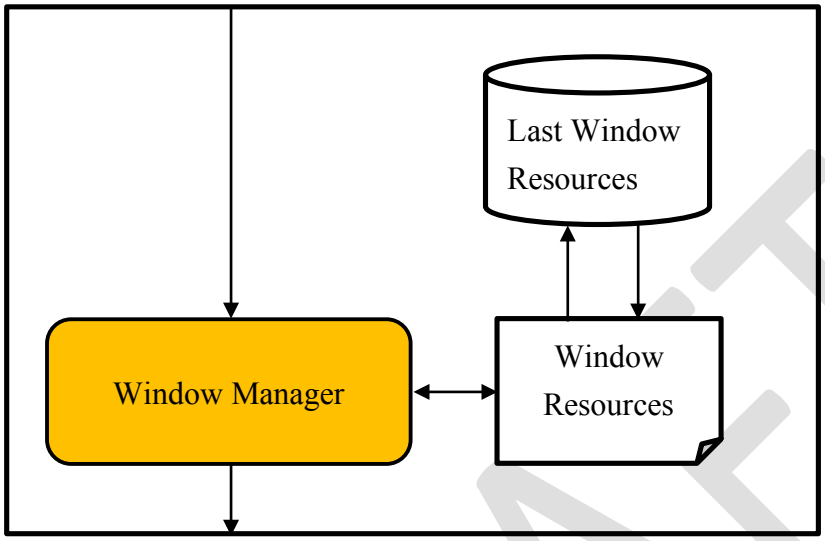
The API is shown below.

No	Function	R/W	Description
1	Window Resources Control	R/W	Read/Write Window Resources
2	Register My Application	W	Registration Own process(PID, SurfaceID)
3	Allocate Window Resources	W	Request Allocate Area(AreaName)
4	Release Window Resources	W	Request Release Area(AreaName)
5	Notify Window Resources Status	R	Post Window Resources Status to Apps
6	Window Policy DB Control	R/W	Read/Write Policy DB
7	Window Layout DB Control	R/W	Read/Write Layout DB

4.3. Window Resources Manager

4.3.1. Recover Window Resources (Boot Sequence)

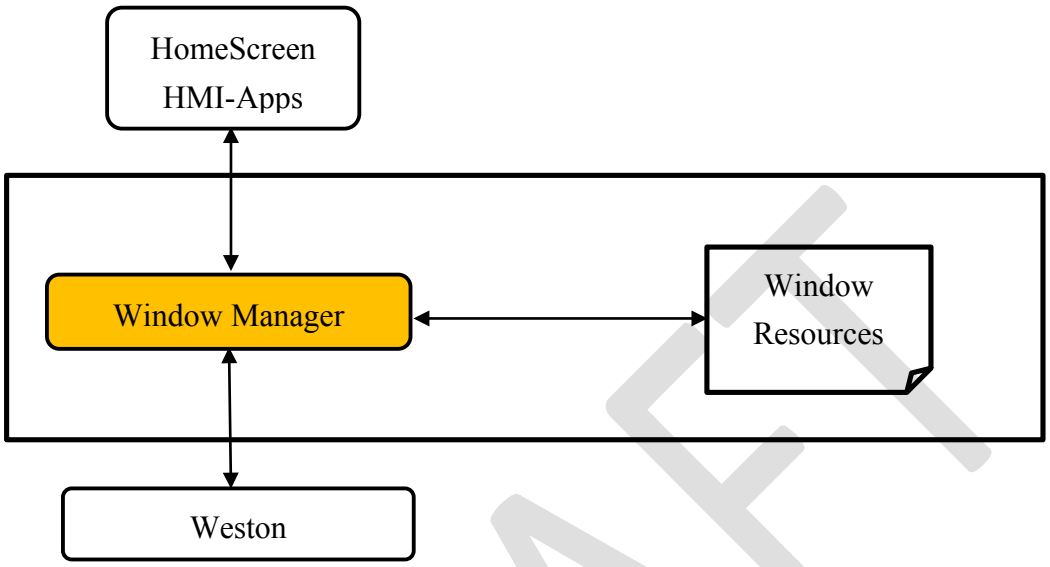
The Window Manager always holds current window resources.
After reboot, Window Manger recoverd the Last Window resources.



4.3.2. Window Resource Control (Window Manager API)

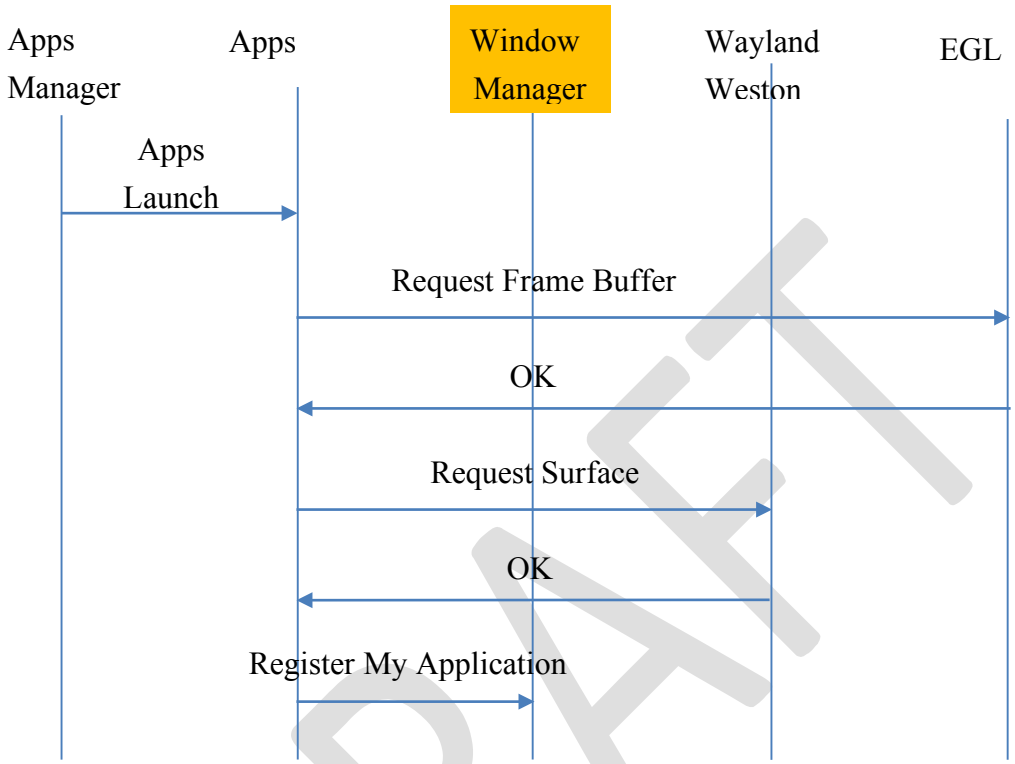
HMI Apps can Read/Wirte Window Resources.

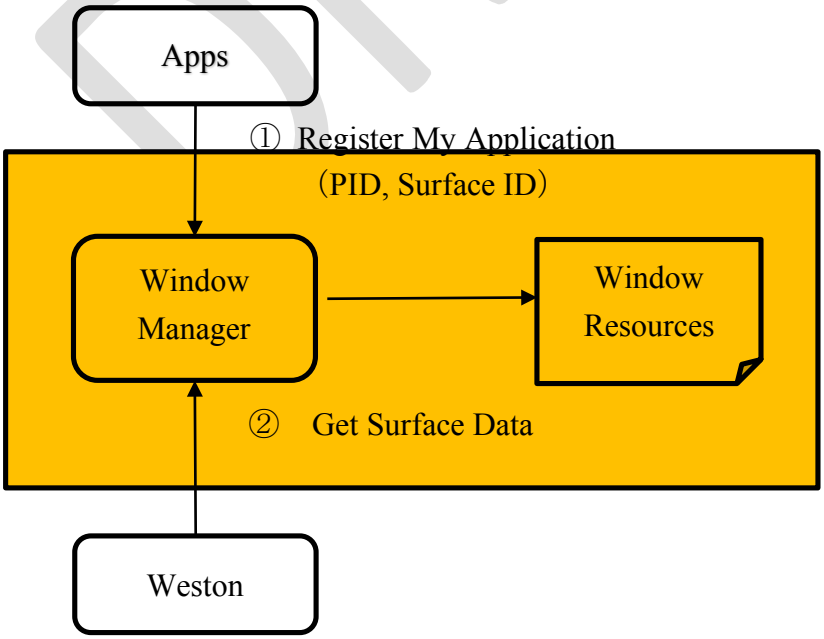
- ✓ HomeScreen Read/Write Display and Layer Info.
- ✓ HMI-Apps Read/Write Area Info.



4.3.3. Register My Application(Window Manager API)

When an application uses WindowManager, registration of the application is necessary.



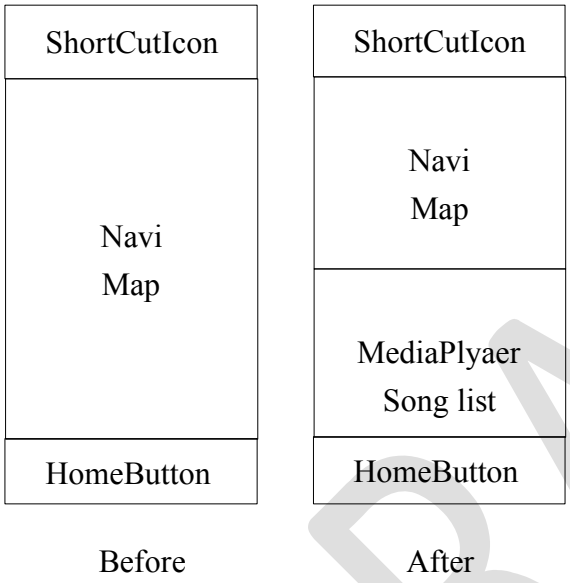


4.3.4. Allocate/Release Window Resources(Window Manager API)

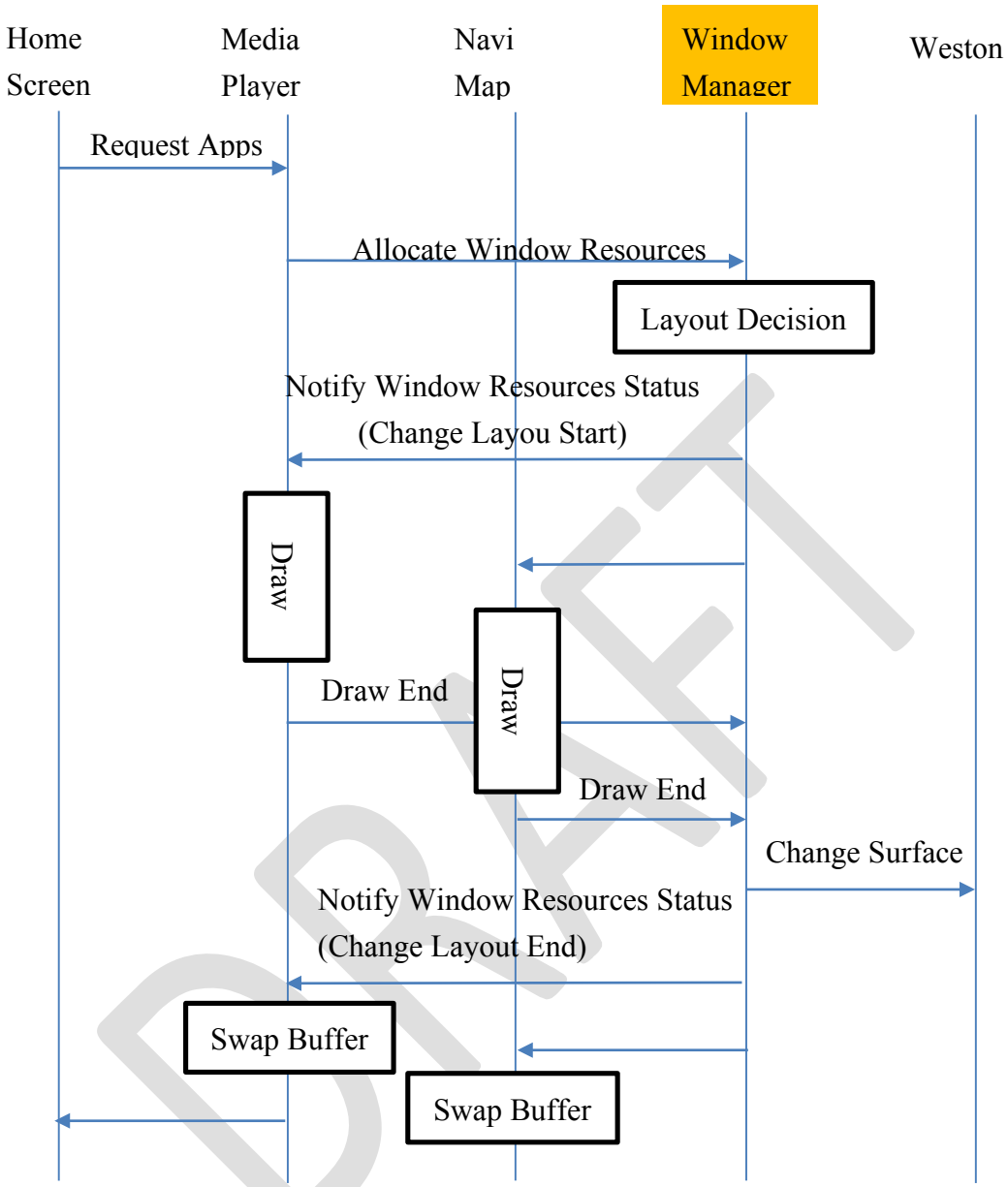
When the application starts drawing, it is necessary to acquire Window Resources.

Use Case of Allocate Window Resources

The use cases in which the MediaPlayer displays the song list during Navi map display are shown below.



① The user presses the shortcutIcon (HomeScreen).
② Home Screen notifies Media Player.
③ The Media Player issues「Allocate Window Resources」to Window Manager.
④ Window Manager determines the optimal layout using Policy DB.
⑤ The media player receives a response and draws it.
⑥ The navi receives「Notify Window Resources Status」, Redraw Map.
⑦ Window Manager waits for drawing ⑤、⑥ and issues Change Layout to Weston.
⑧ And finally Apps issue Swap Buffer.



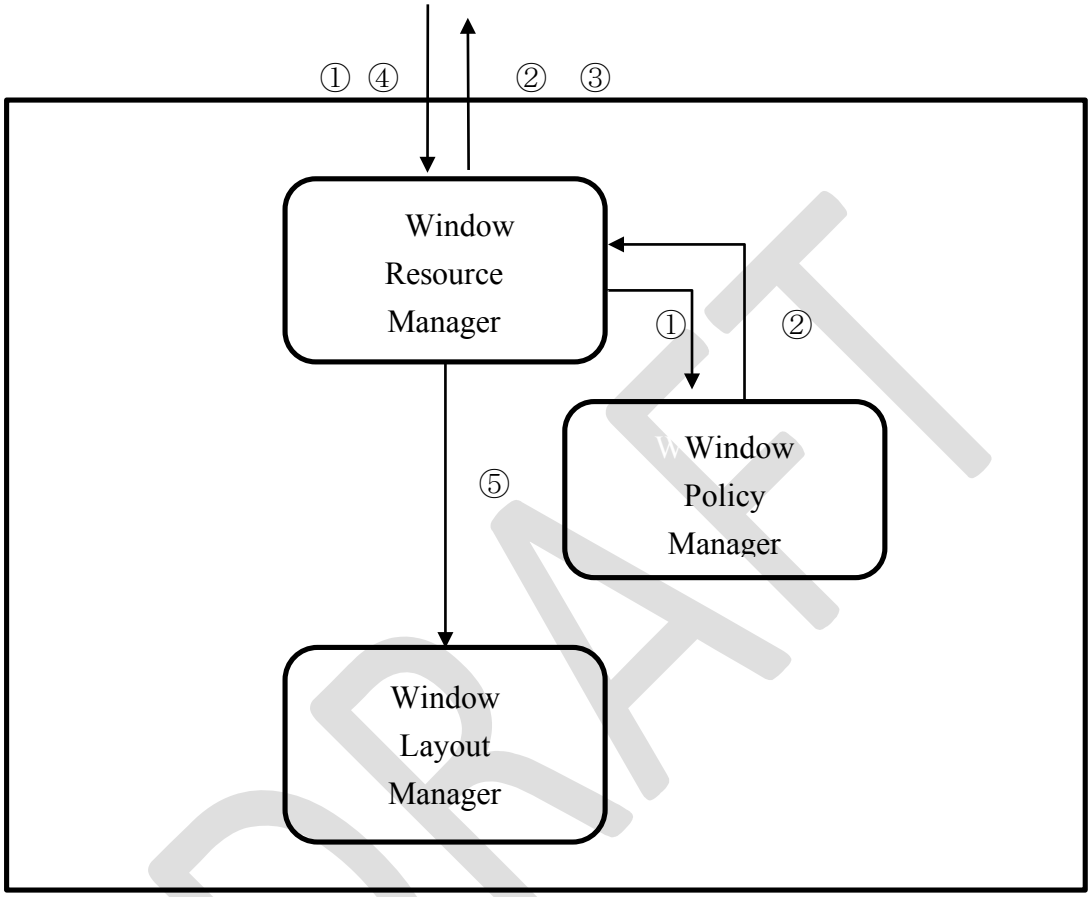
The internal sequence of WindowManage during 「Allocate Window Resources」 execution is shown below.

① The app issues「Allocate Window Reasources」 to Window Policy Manager.
② Window Reasources Manager respons「OK」to App.
The App draw own area.
③ Window Reasources Manager rsposns「Notify Window Status」(e.g.Area Size, Visible) to another App

The App redraw own Area.

④ Window Reasources Manager receive「Draw End」from Apps

⑤ Window Reasources Manager issues 「Change Layout」 to Window Layout Manager



4.3.5. Notify Window Resources Status(Window Manager API:EVENT)

Window Manager notifies the application at the event when the situation of Winodw Rersources changes.

No	EVENT	R/W	Description
1	Visible	R	Area Visible
2	InVisible	R	Area InVisible
3	Change Layout Start	R	Apps must redraw according to the layout After the end, response「Draw END」.
4	Change Layout End	R	

4.4. Window Policy Manager

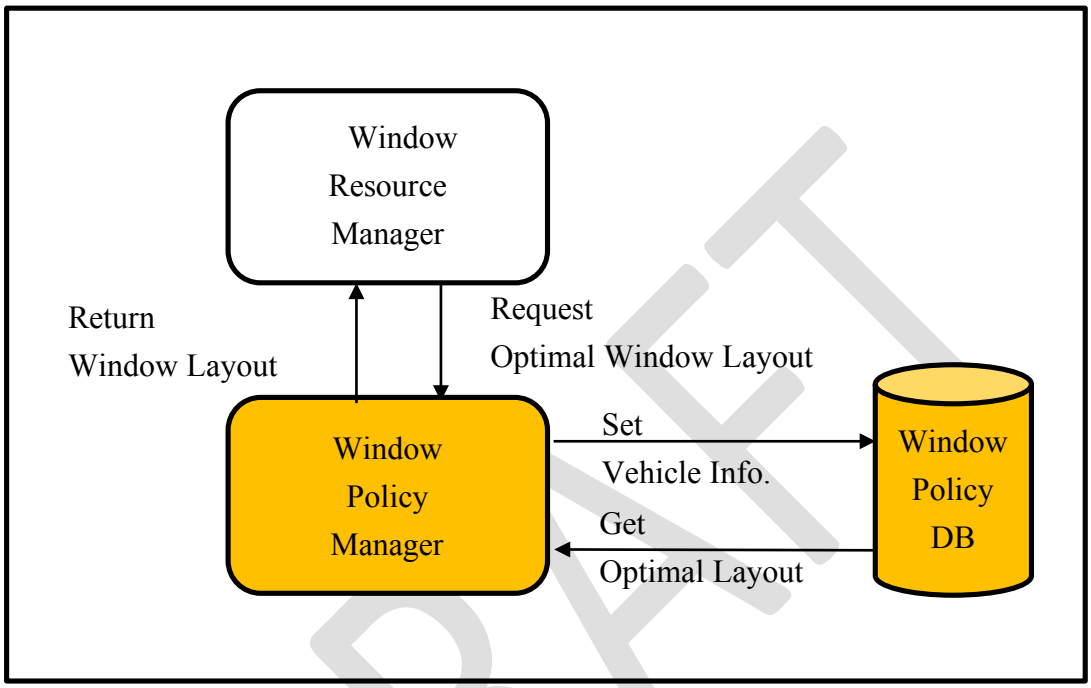
When there is a screen request from the application due to a user operation or a state change of the system, It is common to erase the old screen and display a new screen.
But, Setting an optimum screen layout in consideration of the following conditions is an important requirement of an in-vehicle HMI.

- Application Priority
- Driving restrictions

This requirement is called "HMI Policy".
However, HMI Policy is often different for each OEM and each in-vehicle device.
So, Window Policy Manger have policy logic based on PolicyDB prepared in advance.

4.4.1. Window Layout Decision

According to a request from "Window Resource Manager",
Window Policy Manager decides Layout based on Window Policy DB
and responds to Window Resource Manager.



4.4.2. Window Policy DB Control(Window Manager API)

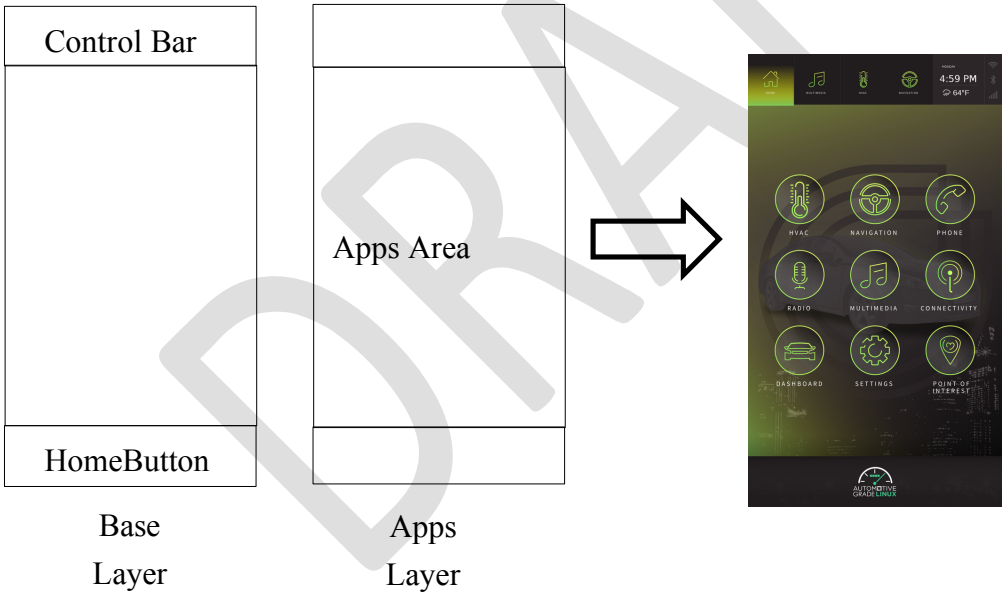
Update the Window Policy DB with the following timing.

- ✓ Hardware
  in-vehicle unit setting
- ✓ Software
  Software update、Application delivery

4.4.3. Window Policy DB use cases

Precondition

- ✓ Window Layer Pattern（have 2 Layers）
  ① Base Layer (HomeScreen)
  ② Apps Layer

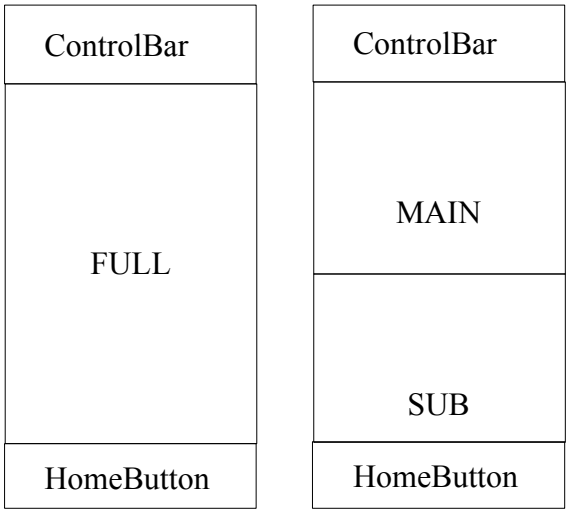


✓ Apps Layout Pattern（have 2 types）

① FULL Full Apps Area

② HALF MAIN: Upper Apps Area

SUB： Lower Apps Area



① FULL ② HALF

✓ Displayable area of application（3 Apps）

① HOMESCREEN： FULL

② NAVI： FULL or MAIN

③ BASE(General Apps)： FULL or MAIN or SUB

Policy DB (State Machine at stopping)

The state transition table during STOP is shown below.
In the case of driving start, save the current state and shift to the RUN state.

STOP

	MAIN	SUB	HOME	NAVI	BASE
Hs	HOME	HOME	−	To n1	To b1
n1	NAVI	NAVI	To Hs	−	To n2
n2	NAVI	＊	To Hs	To n1	MAIN:NAVI SUB:BASE
b1	BASE	BASE	To Hs	To n1	−
b2	BASE	＊	To Hs	To n1	To b1

First Low：State Name
Second Low、Third low：Area Name
（When MAIN and SUB are the same, it is regarded as FULL screen）
The action after the 4th column is for the application request（State Machine Table）
＊：Other applications different from MAIN

Policy DB (State Machine at running)

The state transition table during RUN is shown below.
In the case of stopping, restore the current state and shift to the STOP state.

RUN

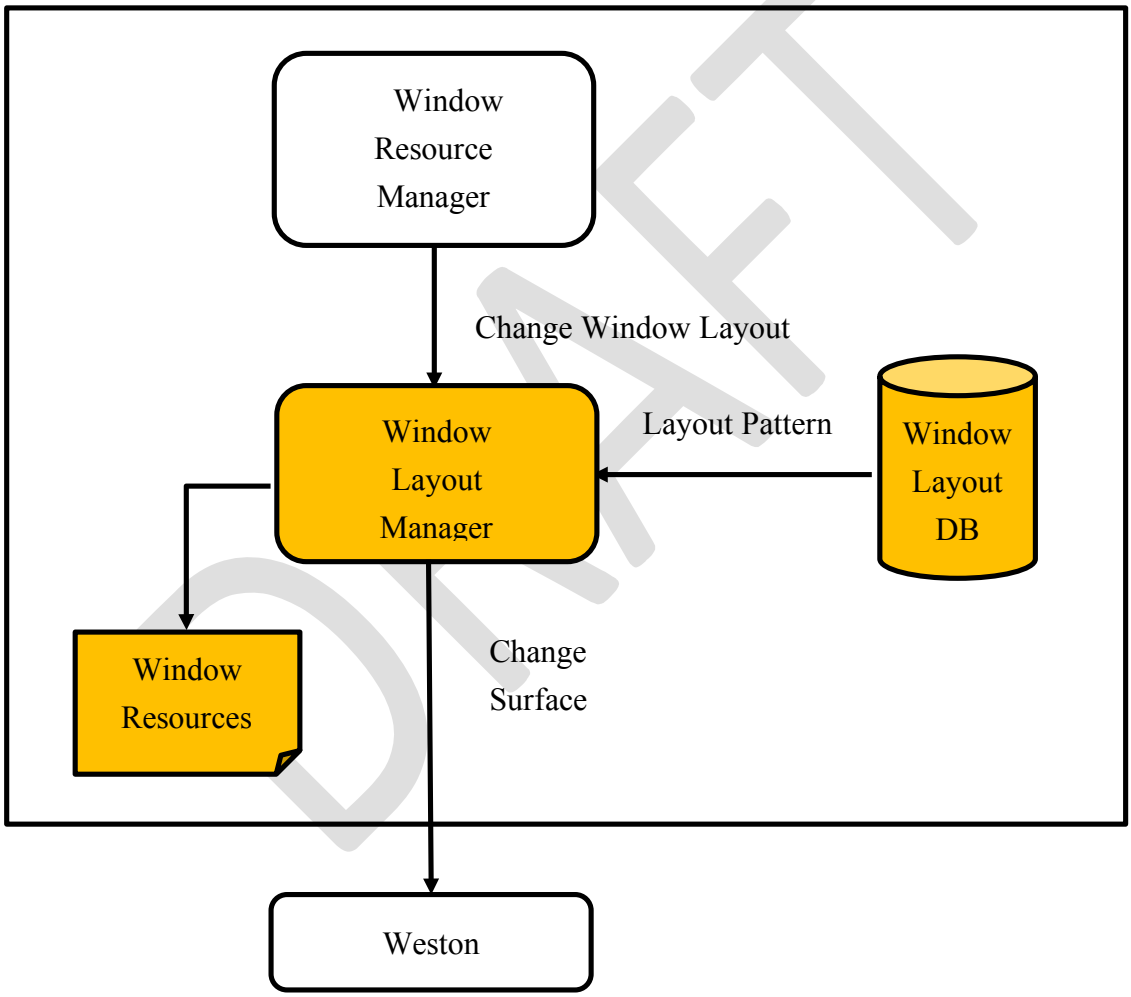
	MAIN	SUB	HOME	NAVI	MM	BASE
n1	NAVI	NAVI	−	−	−	−

4.5. Window Layout Manager

The Window Layout Manager has the following functions related to Layout.

4.5.1. Change Window Layout

If Window Layout Manager receive「Change Window Layout」
They need update Window Resources and send「Change Surfase」to Weston.



4.5.2. Window Layout DB Control(Window Manager API)

Update the Window Layout DB with the following timing.

- ✓ Hardware
  in-vehicle unit setting
- ✓ Software
  Software update、Application deliver

4.5.3. Window Layout Pattern Data (DB) sample

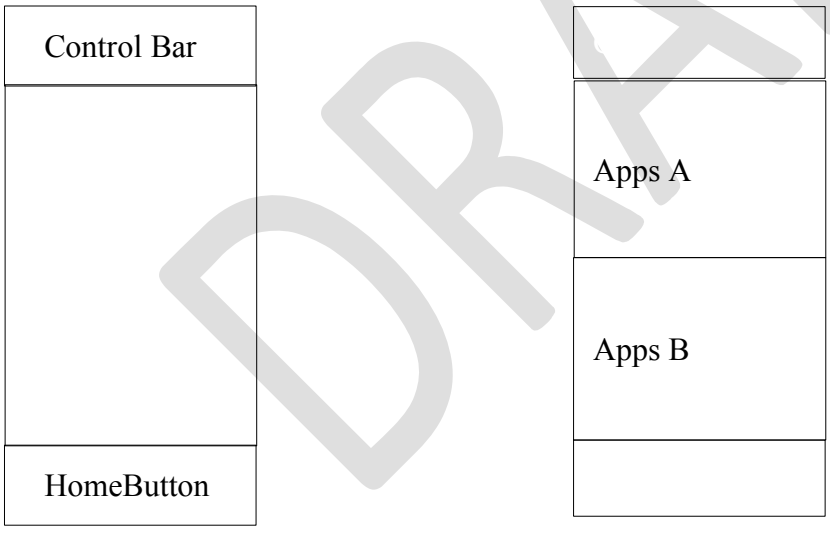
The layout Pattern Data is shown below together with data description (JSON).

Precondition

Layout Pattern　(2 patterns)

① HomeScreen Basic

② Apps Half Basic



HomeScreen Basic

Apps Half Basic

Pattern Description (JSON)

Display_height=1920
Display_width =1024

ControleBar_ height =200
HomeButton_ height=200

Apps_height= (Display_height- ControleBar_ height- HomeButton_ height)/2
Main_y = ControleBar_ height
Sub_y= Main_y+Apps_height
HomeButton_y= Sub_y+ HomeButton_ height

① Home Screen Basic (Home Screen Layer)

```
“Layout”
{
        “name”:”HomeScreenBasic”
        “areas”:
        [{
                "name":"ControleBar", "x”:0, “y”:0, “width”:”display_width”,
                    “height”: “display_height”, "zorder":0
        },
        {
                "name":"HomeButton", "x”:0,“y”:HomeButton_y,“width”:”HomeButton_width”,
                    “height”: “Display_height”, "zorder":0
        }]
}
```

② Apps Half Basic (Apps Layer)

```
“Layout”
{
        “name”:”ApssHalfBasic”
        “areas”:
        [{
                "name":"Main", "x”:0, “y”: Main_y,
                    “width”:”Display_width”, “height”: “Apps_height”, zorder":0
        },
        {
                "name":"Sub", "x”:”0”, “y”:Sub_y,
                    “width”:”Display_width”, “height”: “Apps_height”, "zorder":0
        }]
}
```

5. Sound Manager（T.B.D）

6. Input Manager（T.B.D）