AGL HMI Framework

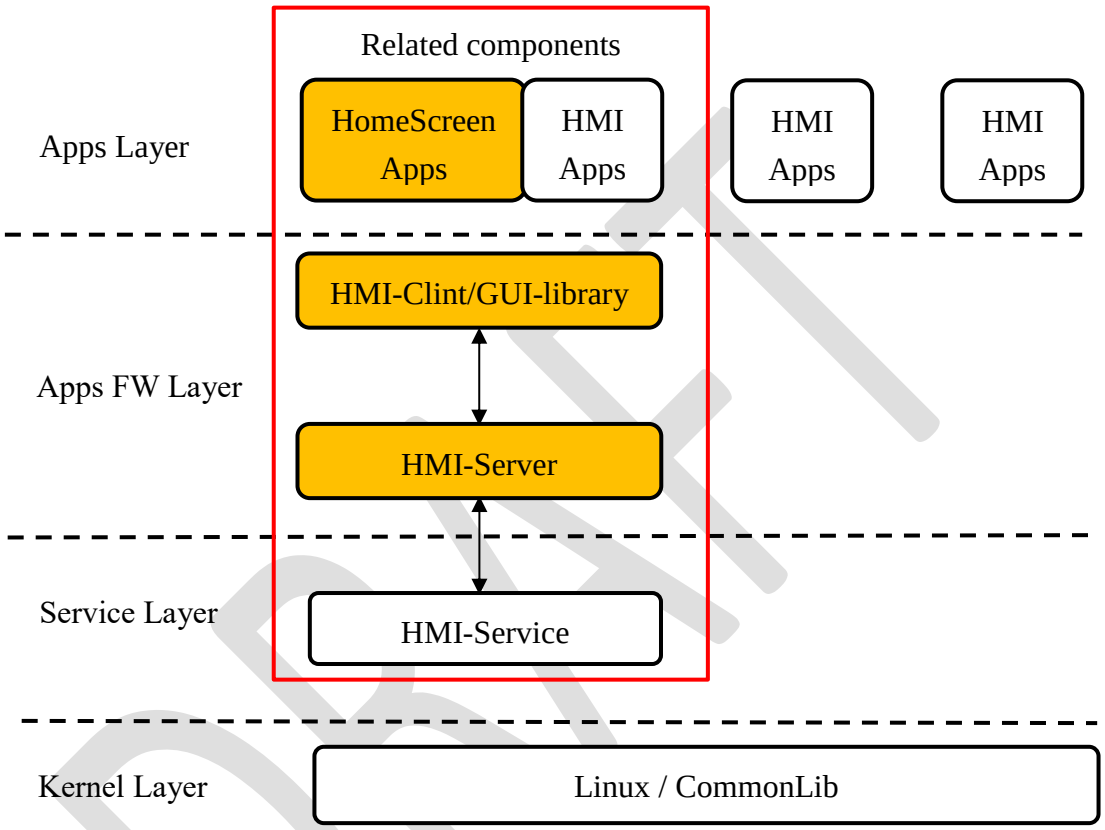
Architecture Document

Version	Date
0.9	2019/x/x

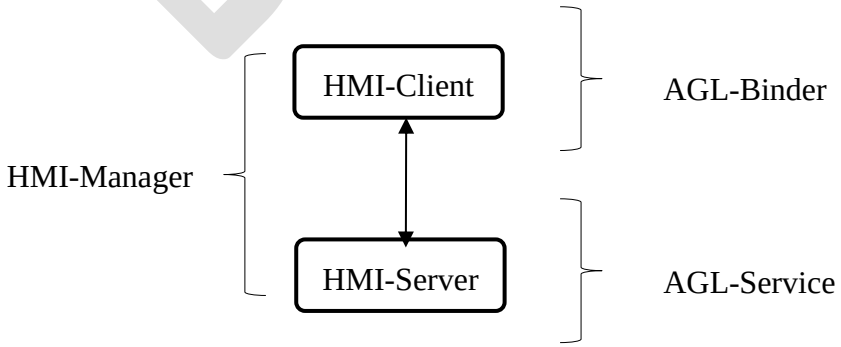
1. HMI Framework overview

1.1. Oerview

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



The relationship between HMI-FW and AGL Apps-FW is shown below.



1.1.1. HMI-FW Components

Components of the HMI-FW are shown below.

GUI-library

OEM 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.

- ✓ WindowManager
- ✓ SoundManager
- ✓ InputManager
- ✓ HomeScreen
- ✓ HMI-ApplicationManager

HomeScreen Apps

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.

- ✓ Menu Bar
- ✓ Onscreen
- ✓ Launcher

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-Service

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

- ✓ Graphics Subsystem：Weston/Graphics Device Driver
- ✓ Sound Subsystem：Audio Manager/ALSA
- ✓ Input Subsystem：Input Device Driver/Vehicle Bus Access

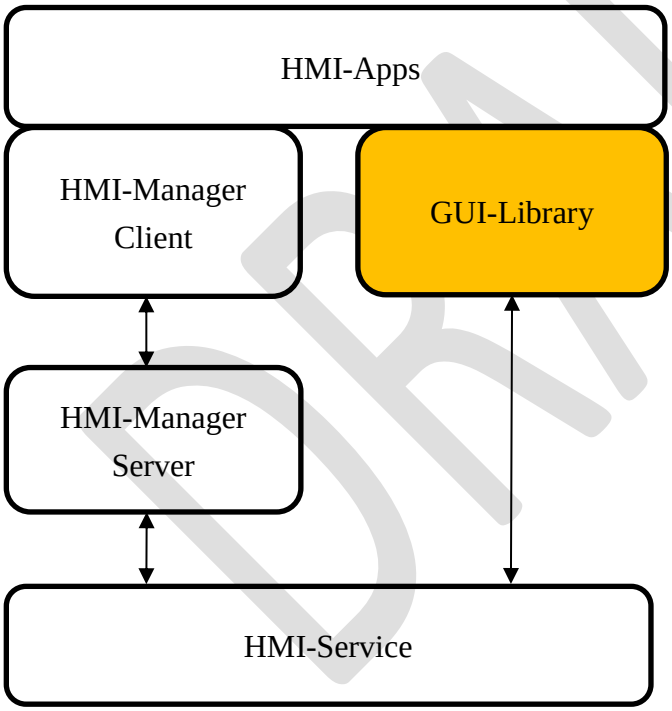
2. GUI-library

2.1. Overview

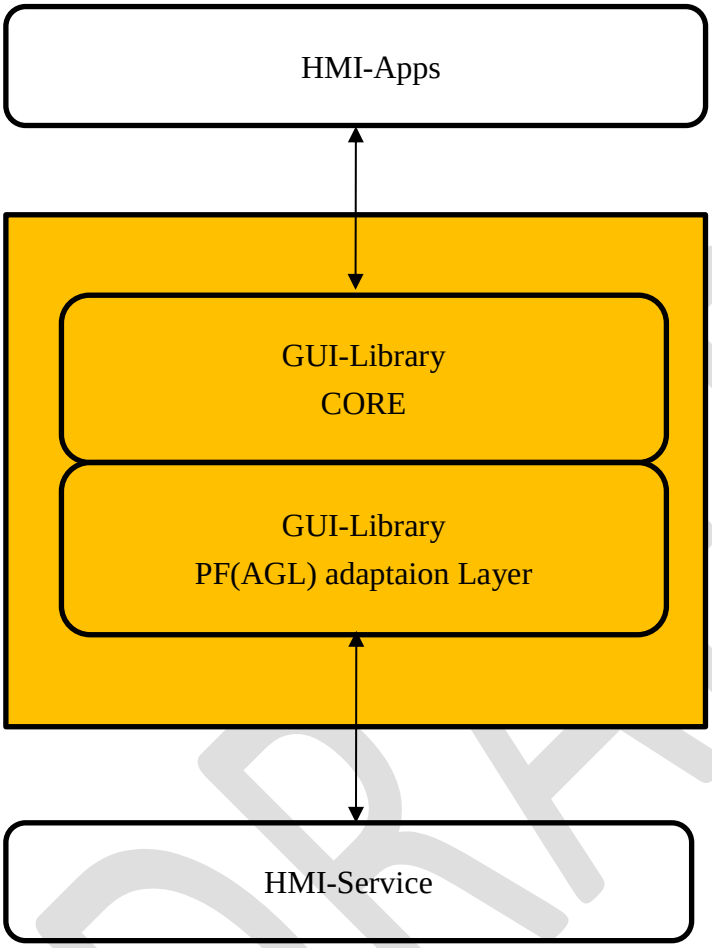
GUI-library is a library that provides HMI functions to applications, and mainly has HMI functions related to graphics, sound, and input.

2.1.1. Related external components

The application developer selects the GUI-library (e.g. Qt, HTML5,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.）



2.1.2. Internal Components



2.2. Graphics functions

Graphics provides rendering functions to the application.

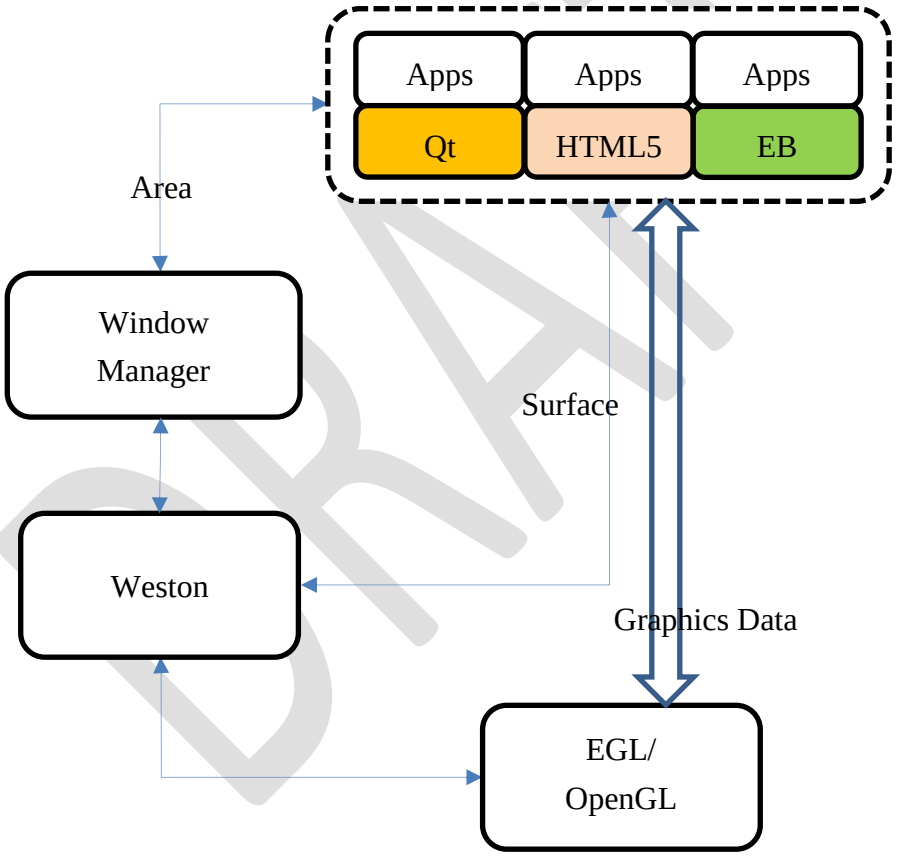
2.2.1. Procedure necessary for HMI-Apps

Graphics draws with the following procedure.

① The application requests Weston to acquire Surface

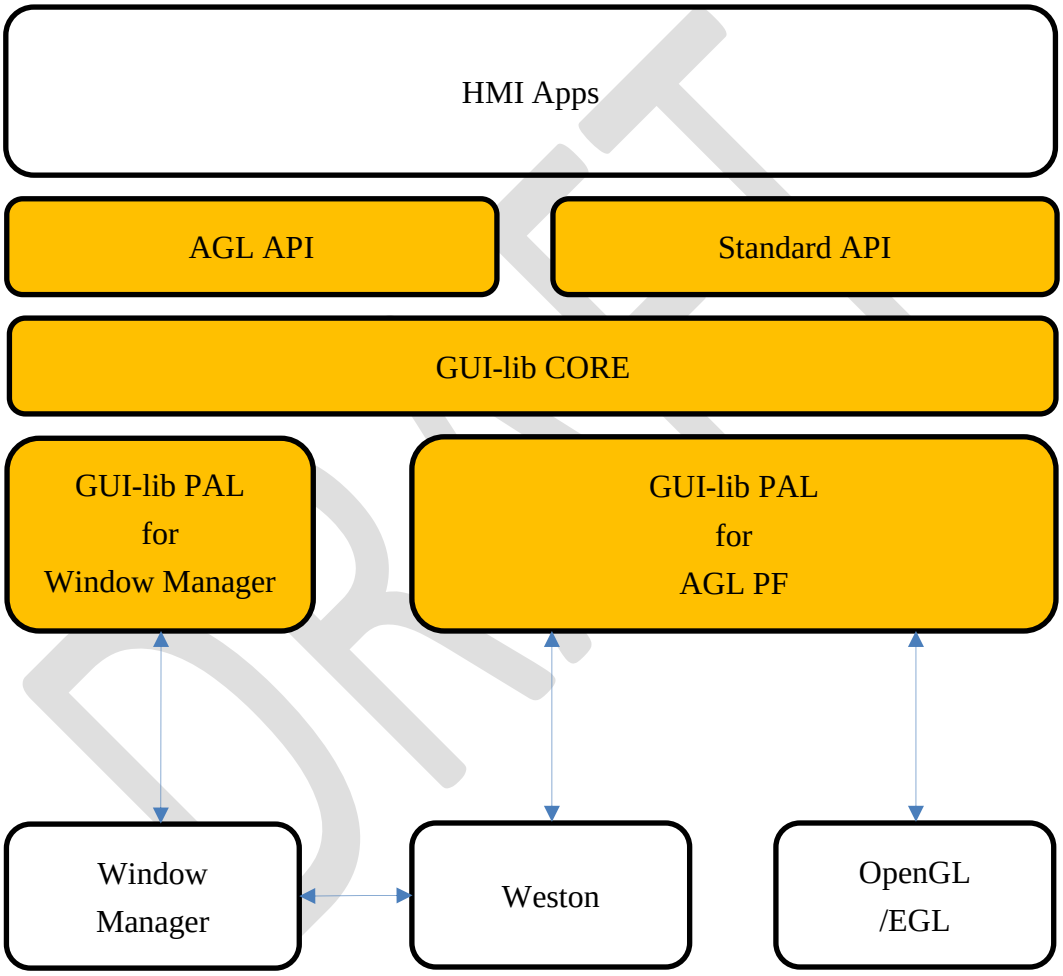
② The application makes resources request to Window Manager（OEM options）

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



2.2.2. 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



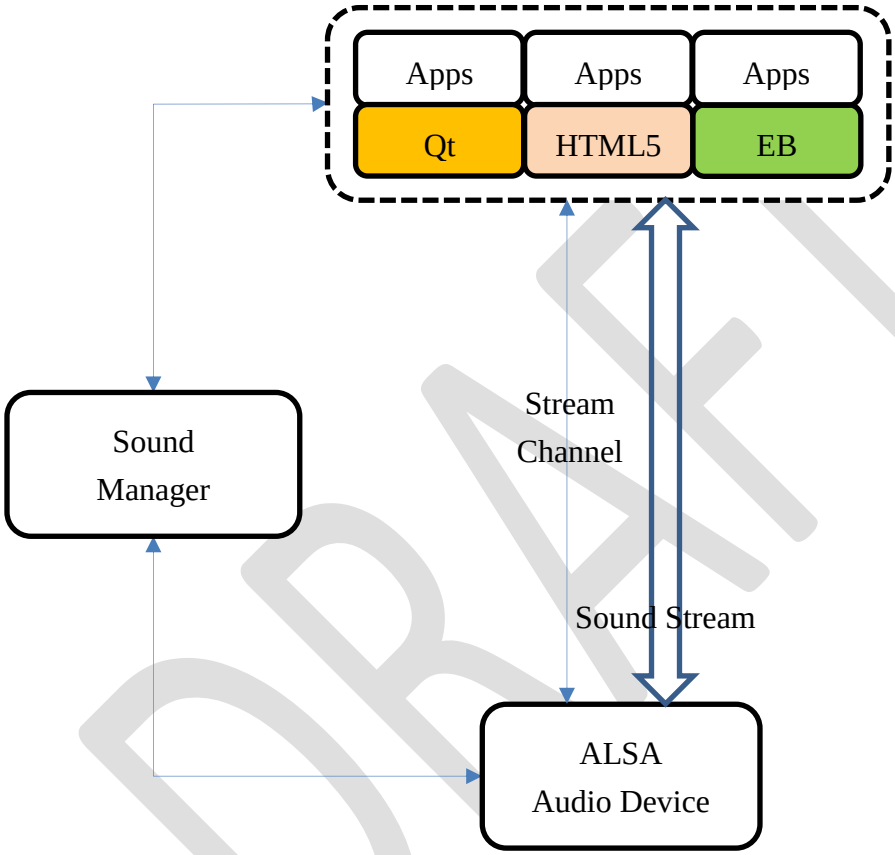
2.3. Sound functions

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

① The application requests ALSA to acquire Stream.

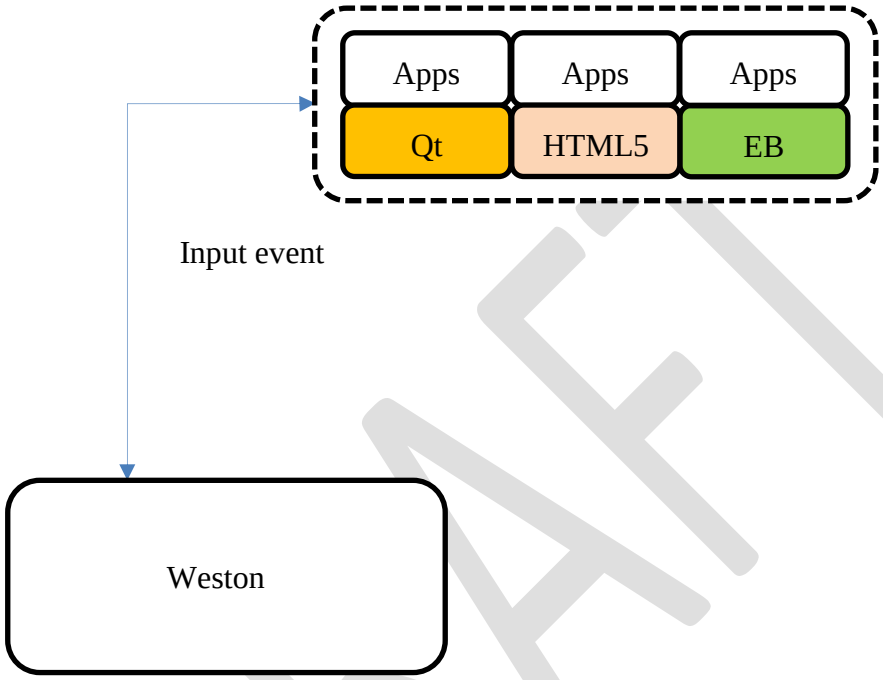
② The application makes resources request to Sound Manager（OEM options）

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



2.4. Input functions

For input under the control of GUI-lib, input function provides the information to Apps according to the method of GUI-lilb.

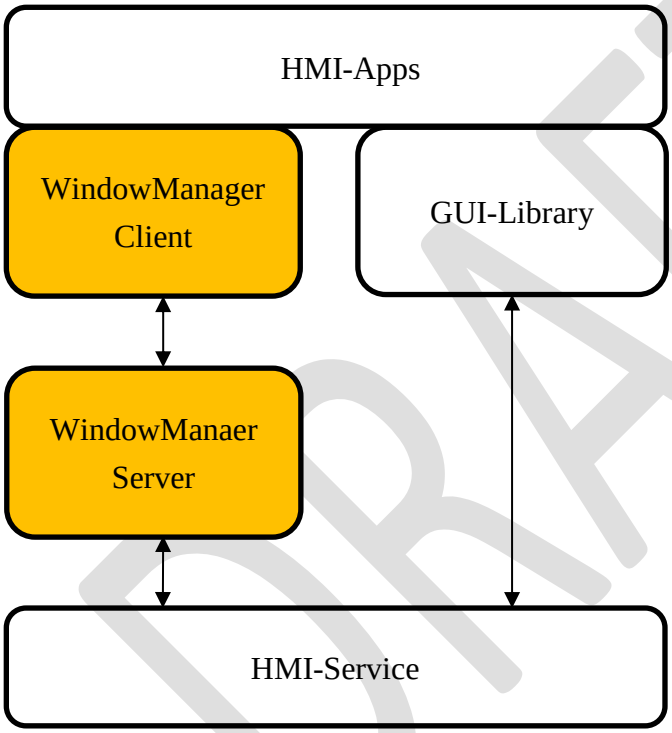


3. Window Manager

3.1. Overview

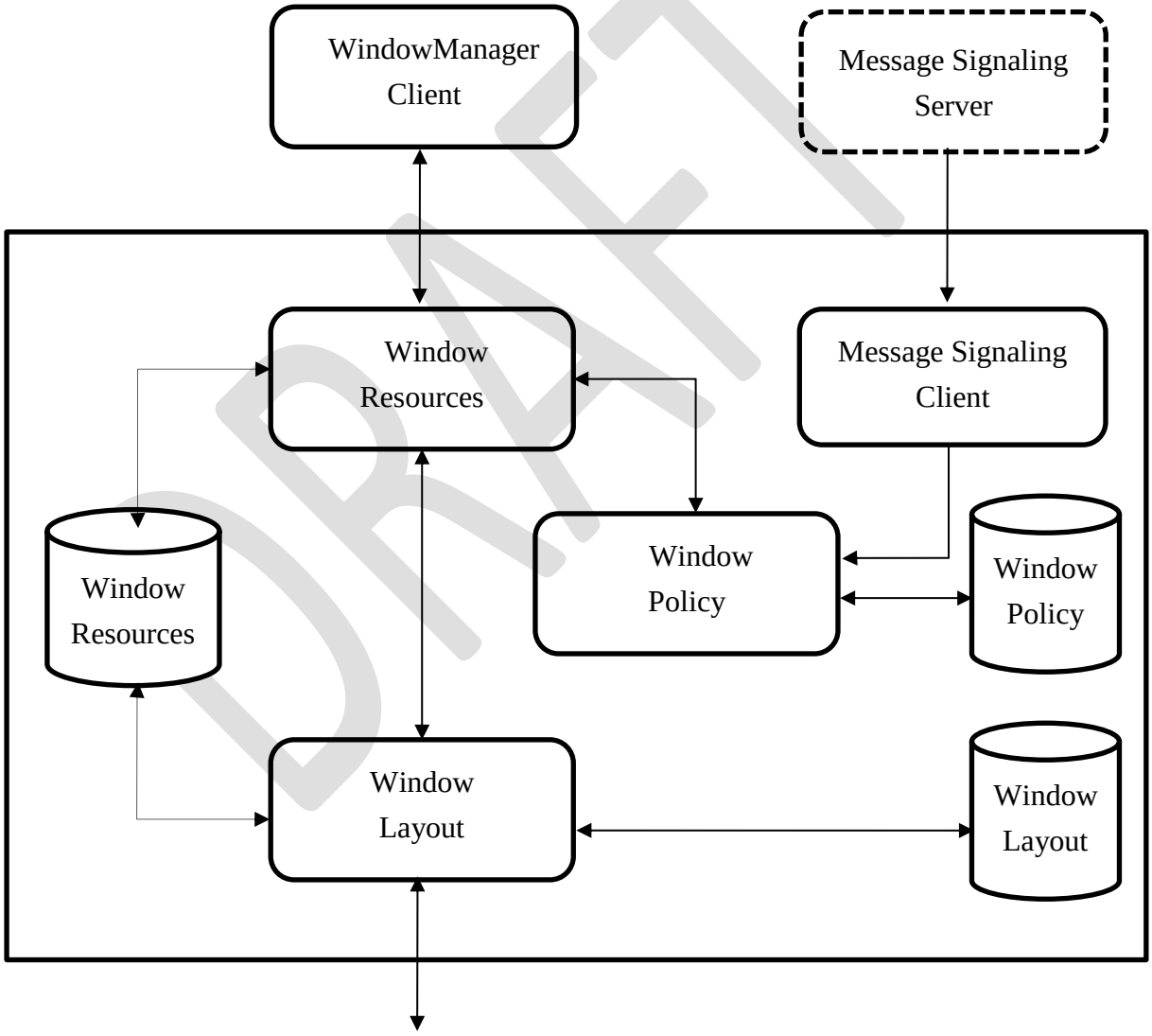
Window Manager determines the optimum screen layout and controls the screen、based on the request from the HMI-Apps.

3.1.1. Related external components



3.1.2. Internal Components

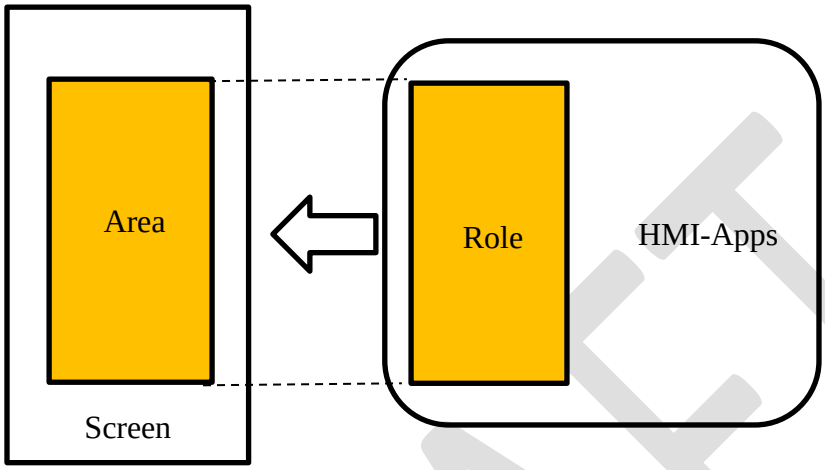
No	Function	Description
1	Window Manager Client	API
2	Window Resource Manager	Window Resource Management
3	Window Policy Manager	Mediation of Window Resources
4	WIndow Layout Manager	Window Layout Management



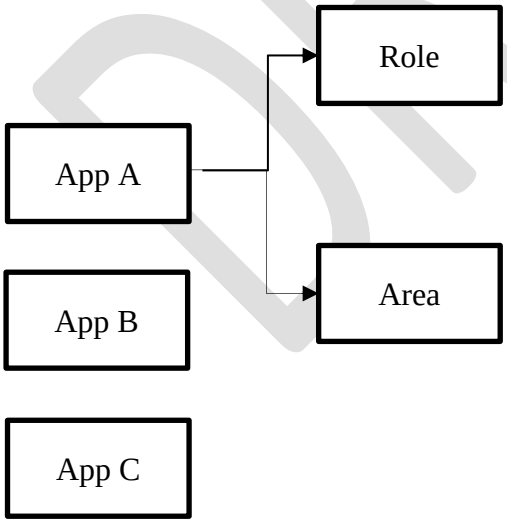
3.2. Window Manager DataBase

3.2.1. Window Resources DB

In WndowManager,
specifies the data to be drawn with Role and specifies the area to be drawn in Area



Window Resources Manager manages information for each application



Role

The content specified by the Role name can be displayed in the acquired Window.
The Role is a system-defined URI, and Window Manager determines the screen layout by Role and vehicle information.
An example of a Role is shown below.

Role Name	Description
Navi	Navigation
Map	Map
MediaPlayer	Media Player
PhoneCall	Phone Call

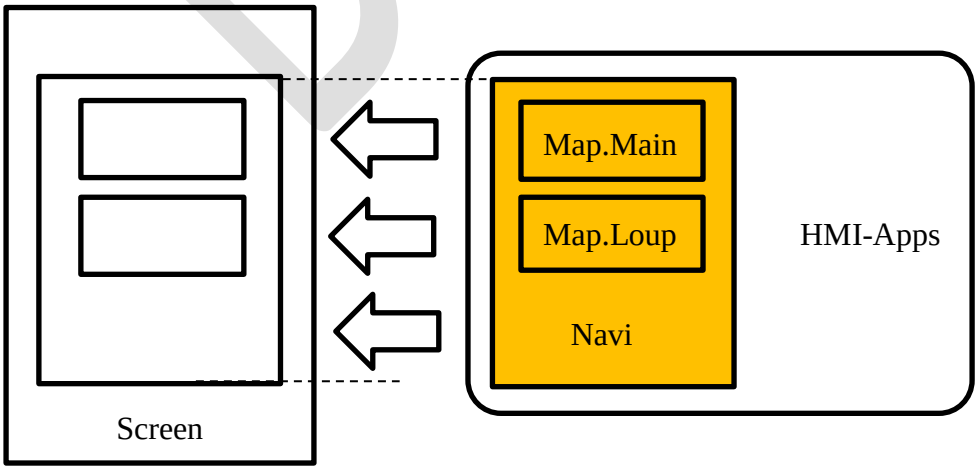
Role is linked with one Area or Layer of the application.
The application controls the area or Layer designated by ROLE through WindowManager.

Sub-Role

When one application has multiple roles, write it as Sub-Role.
Sub-Role is identified by WindowManager by separating it with “.”

e.g. Map.Main Map.Loupe

WindowManaer does not manage management on「SubRole」.
For example, the application must perform display ON / OFF of「SubRole」



Special Role

You can specify Role by specifying the Window state for the WindowManager

Role Name	Description
ROLE or “”	My Role
FOCUS	Currently focused Wndow

Area

The application specifies the area to be drawn with "Area".

Area = [Screen][Layout]Area Name

“Area Name” is defined by each OEM,
Area Nmae samples are shown below.

Area Name	Description
FullScreen	Layout in which one application is displayed full screen.
Normal	One application together with the status bar etc., the layout displayed on the screen
Split.Main	The main area when two applications are displayed on the screen
Split.Sub	The sub area when two applications are displayed on the screen

3.2.2. Window Policy DB

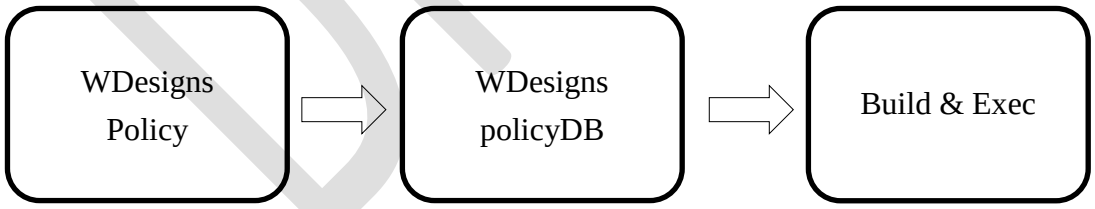
In Window Manager
In order to optimally process drawing requests from the application, incorporate a state machine.

① Design Policy
OEM designs policies from system requirements using optimal tools.

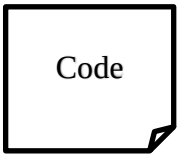
② Design Policy DB
The developer designs policy DB. It is necessary to standardize the policy DB definition (e.g. XML), improve productivity and quality.

The developer can also use the tool（It is desirable that it is common to the tool used by OEM） to automatically create the policyDB from the state transition table and the like.

③ Build & Execution
The developer incorporates the generated code into the system and executes it. However, if you need to change policies dynamically, implement them so that they can be executed directly from PolicyDB.

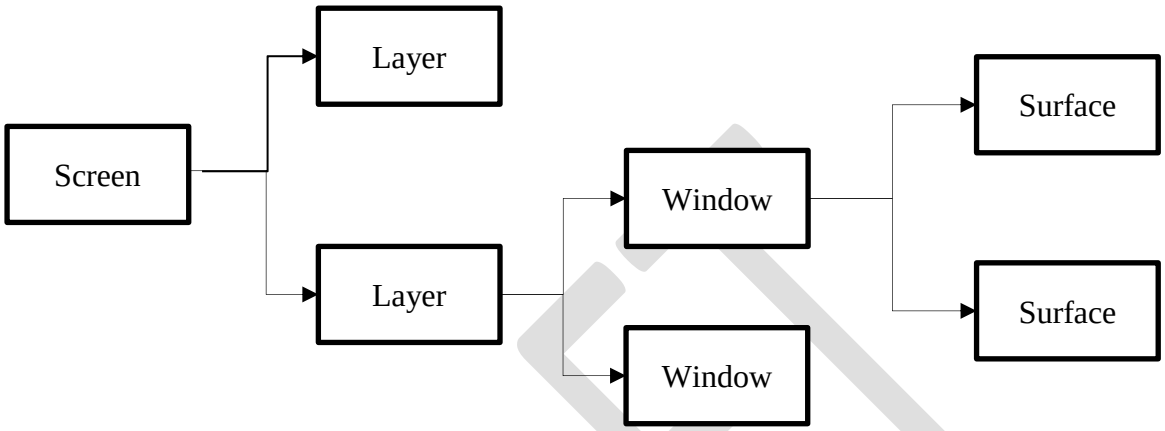


	HS NV BS
S1 S2	
S3	



3.2.3. Window Layout DB

Window Layout are resource information related to the screen managed by the Window Manager and varies depending on the in-vehicle unit configuration



The data items included in Window Layout are shown below.

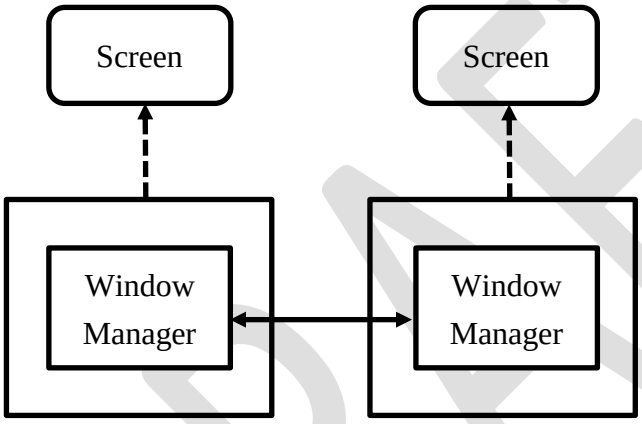
Screen

Display has information on the display device.

A Display can have multiple Layers.

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

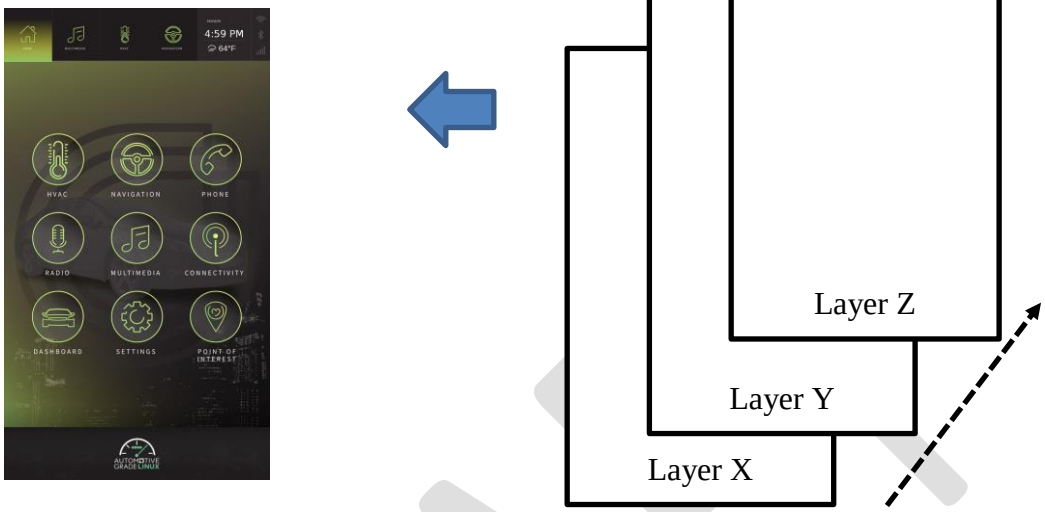
For vehicles with multiple displays it is assumed to have multiple Window Manager.



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

Layer

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



No	Name	Information Source	Description
1	ROLE	Application	Layer Name
2	ID	Graphics Subsystem	Layer ID
3	Position	Graphics Subsystem	Layer Position
4	Size	Graphics Subsystem	Layer Width and Hight
5	Z order	Graphics Subsystem	Layer Zorder
6	Visibility	Graphics Subsystem	Layes Visibility Status
7	α Blend	Graphics Subsystem	Layer Transparent Ratio

Window

Window is the area where the application displays content. When the application displays contents, it is necessary to acquire Window from WindowManager.

No	Name	Information Source	Description
1	Name	Application	Window Name
2	ID	Graphics Subsystem	Layer ID
3	Position	Graphics Subsystem	Layer Position
4	Size	Graphics Subsystem	Layer Width and Hight
5	Z order	Graphics Subsystem	Layer Zorder
6	Visibility	Graphics Subsystem	Layes Visibility Status
7	α Blend	Graphics Subsystem	Layer Transparent Ratio

e.g.) "area": {"rect": { "x": x, "y": y, "width": width, "height": height } }

3.3. Window Manager Client

3.3.1. API

No	Function	W/R	Description
1	init	W	Connect to Window Manager
2	setRole	W	Set Role to My Application
3	attachRoleToApp	W	Attach Role to another Application
4	setRenderOrder	W	Set Render Order our Area
5	activateWindow	W R W R	Request Allocate WR Sequence Sync Draw End Draw Flush Draw
6	dactivateWindow	W	Request Release WR
7	WindowResourcesDB Control	W/R	Get/Set Window Resources DB
8	Window Policy DB Control	W/R	Get/Set Policy DB
9	Window Layout DB Control	W/R	Get/Set Layout DB

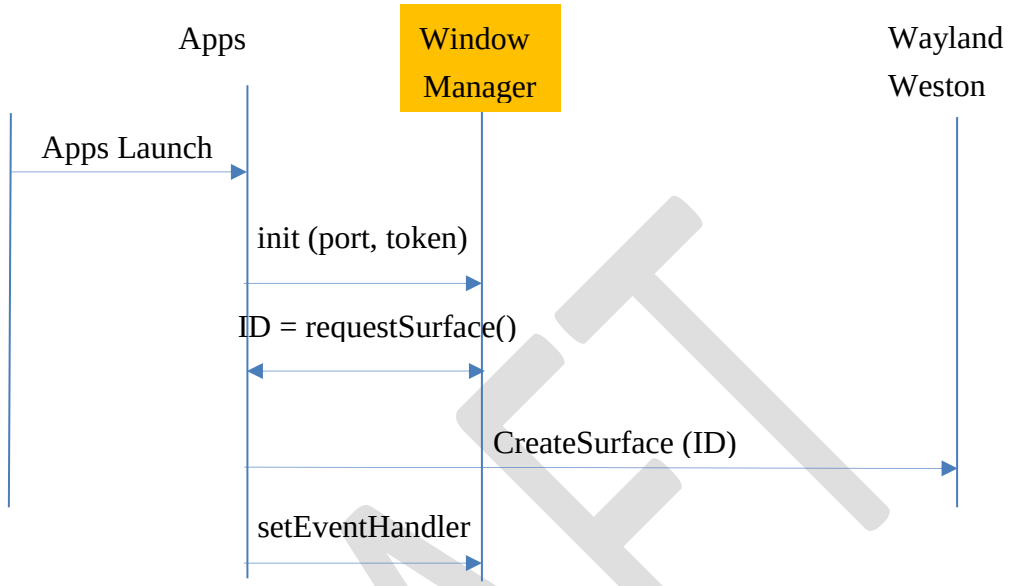
3.3.2. EVENT

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

No	EVENT	Description
1	activate	When own Area becomes Activated
2	deactive	When own Area becomes Deactivated
3	visible	When own Area becomes Visible
4	invisible	When own Area becomes InVisible
5	fous	When own Area becomes Focus
6	outfocus	When own Area becomes outFocus
7	restriction	When own Area becomes restriction

3.4. Window Resources Manager

3.4.1. Initializing Stage



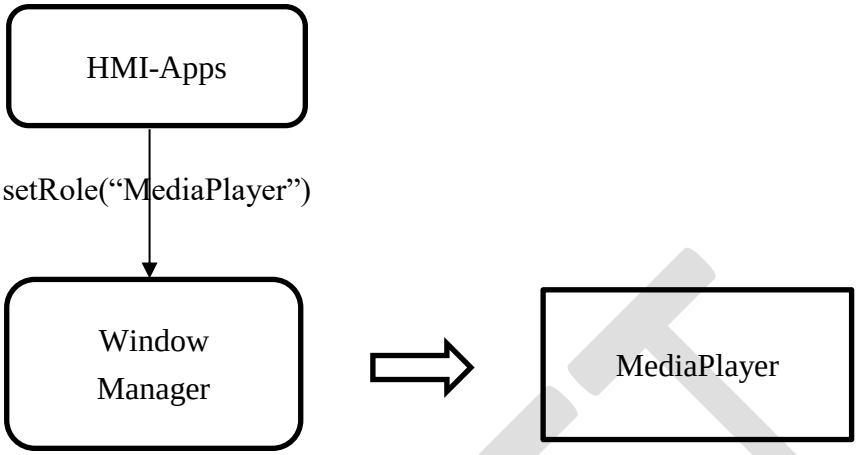
init (port, token) **//depends on AGL**
When an application uses WindowManager, registration of the application is necessary. In AGL, we will implement Connect with WindowManger using AppFW initialization function **init (port, token).**

ID = requestSurface () **//depends on AGL**
Get surface ID of IVI-extention

setEventHandler (“Event”, function)
Register function to receive event from WindowManager.

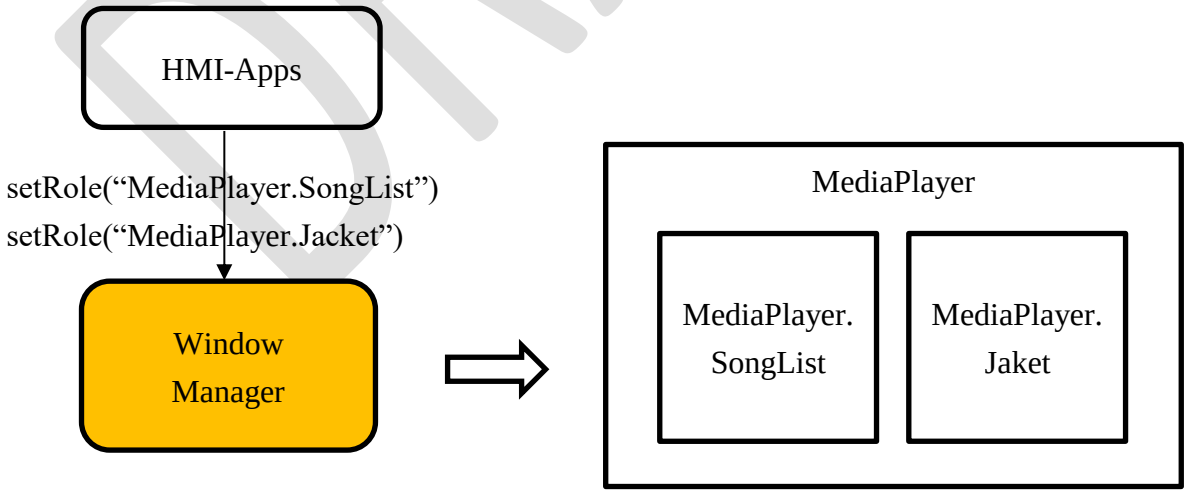
setRole (“Role”)

An application set its own ROLE with WindowManager.



setRole(“SubRole”)

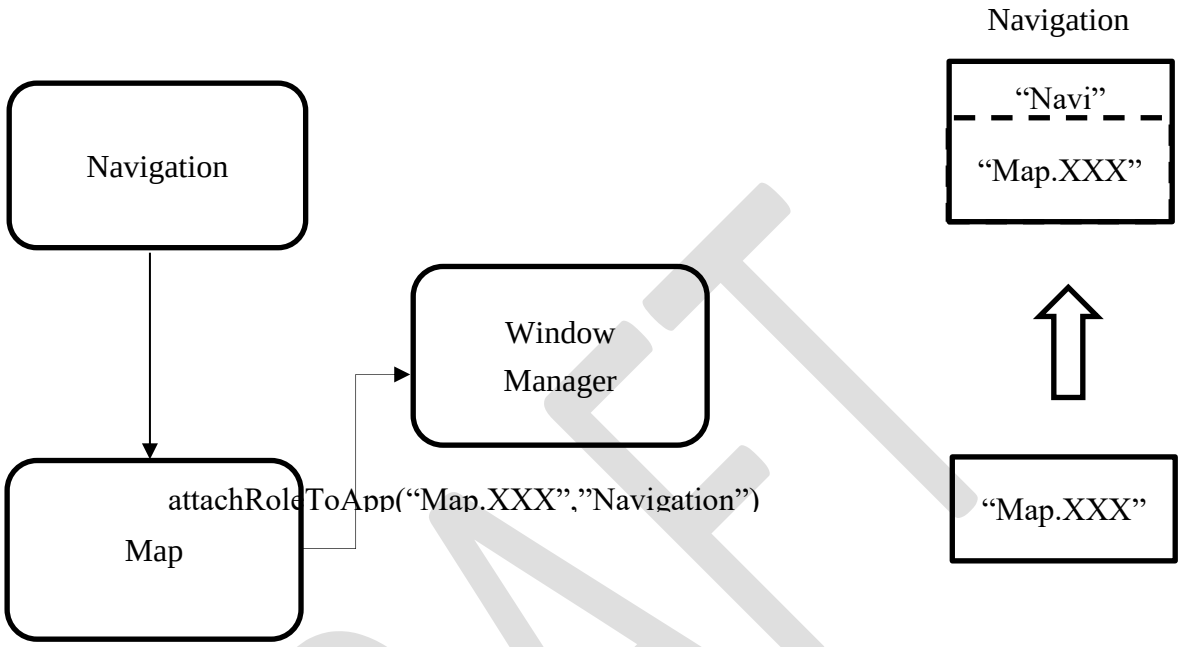
When one application has multiple areas, write it as Sub-Role.
Sub-Role is identified by WindowManager by separating it with “.”



Attach role to Application (Privilege Function)

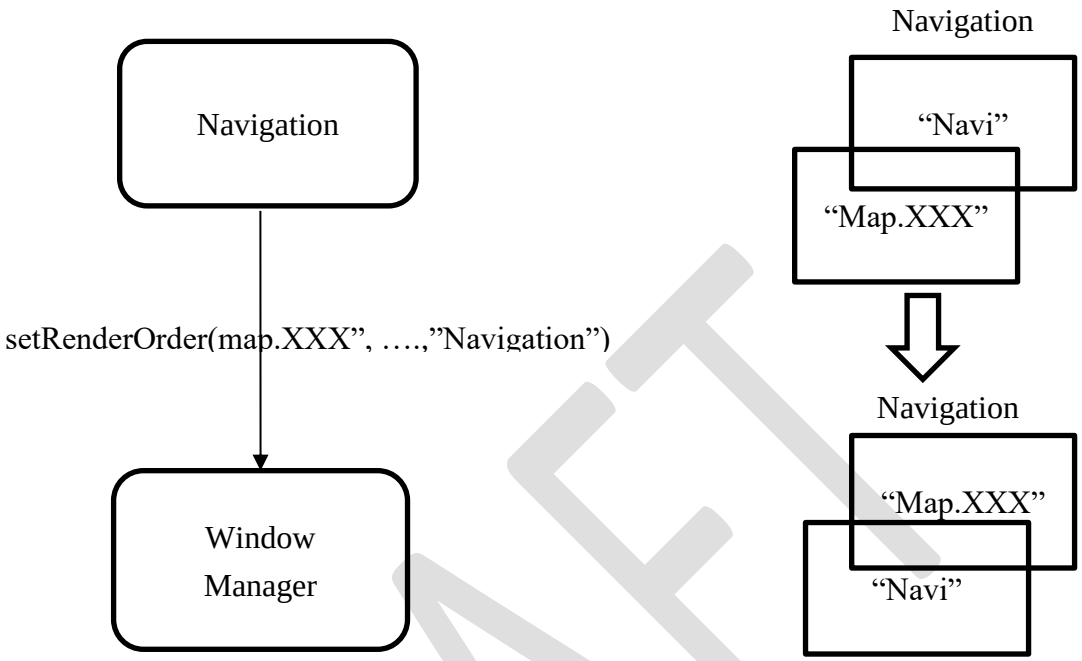
Attach Role to another application.

The way the map application gives Role to Navigation is shown below.



Set RenderOrder(Privilege Function)

We can change display order of multiple roles.



3.4.2. Activate Window

activateWindow (Role, [Area])

When the application wishes to display the drawing data in his drawing area,
it is necessary to activateWindow.

syncDraw (Role, Area)　//EVENT

The area information determined by PolicyManger is notified of the event.
The application draws content based on the designated area (width, height).

WindowManager optimally arranges the Window according to the situation.

e.g.)　activateWindow(“MediaPlayer”)
　　syncDraw(“MediaPlayer”,“Normal”)　//EVENT

If necessary, specify the layout information area as the area for displaying the Window.

e.g.)　activateWindow (“MediaPlayer”, ”FullScreen”)
　　syncDraw(“MediaPlayer”,“FullScreen”)　//EVENT

endDraw (Role)

After the application finishes drawing, it issues EndDraw to WindowManager.

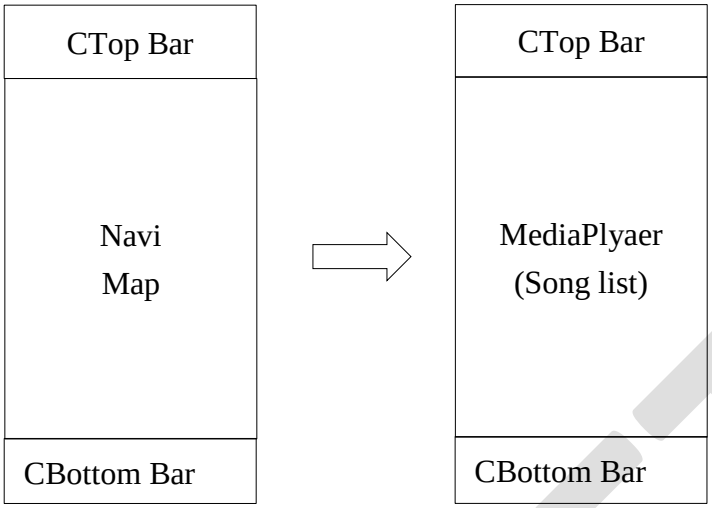
flushDraw()　//EVENT

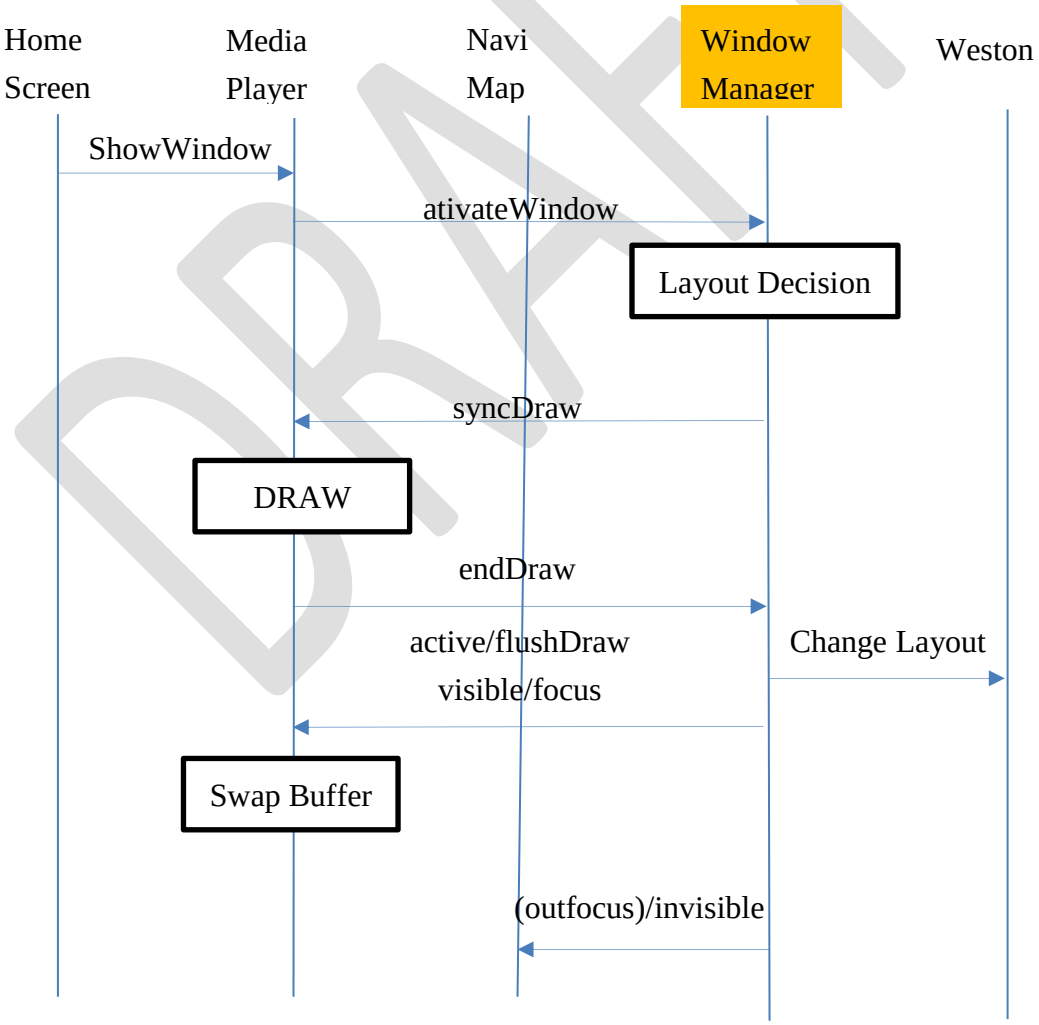
If WindowManager receives all EndDraw, it will issue FlushDraw.
When the application receives this event, issue drawing update command (e.g. Swap Buffer).

active()　//EVENT

It is issued when the application is Activate

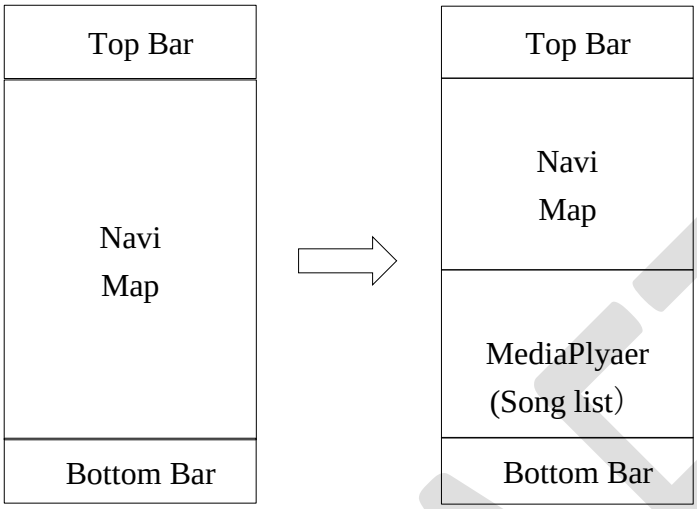
Usecase ①　Normal Window

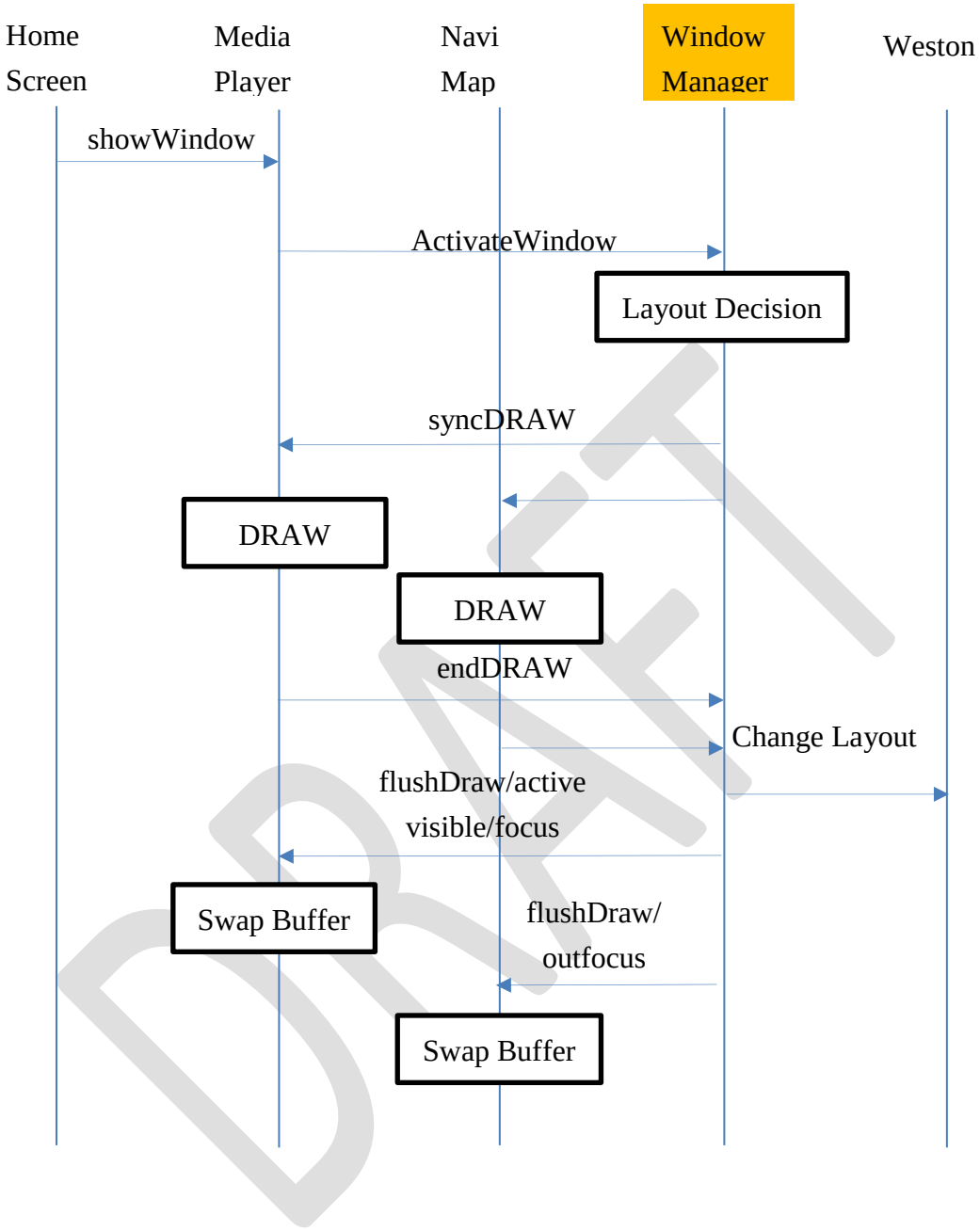




Usecase ②　Split Window

The use cases in which the MediaPlayer displays the song list during Navi map display are shown below. Display of MediaPlyaer is done with shortcut key on HomeScreen Top Bar.





3.4.3. Drawing Stage

The WindowManager notifies the state of the application changed by the layout change

visible()　//EVENT
It is issued when the application is displayed on the screen.
invisible()　//EVENT
When the application screen hides from the screen
focus()　//EVENT
Focus on the application, can receive input events.
outfocus()　//EVENT
when the application gets out of focus

3.4.4. Deactivate Window

Applications issue when resources become unnecessary.
WindowManager hides the application by releasing Window Resources.

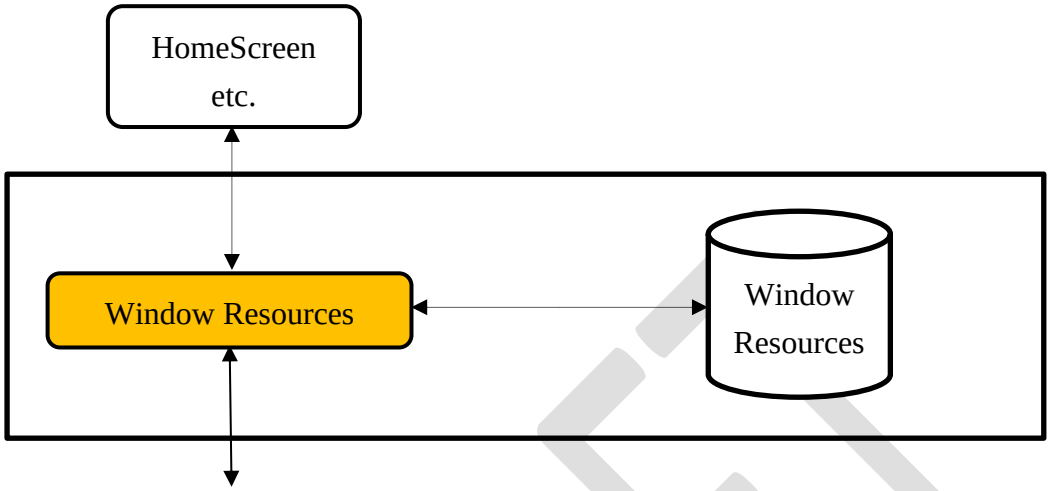
However, depending on the judgment of WindowManager, In this case,
WindowResources is not released but only hidden and the WindowManager sends an EVENT (deactive, invisible) to the application.

deactivateWindow (Role)

deactive()　//EVENT
When the application is DeActivate

3.4.5. Resource DB Control (Privilege Function)

HMI-Manager (HomeScreen etc.) can Get/Set Window Resources.



Set window Rresouces
WindowManager changes the context of the specified Window.

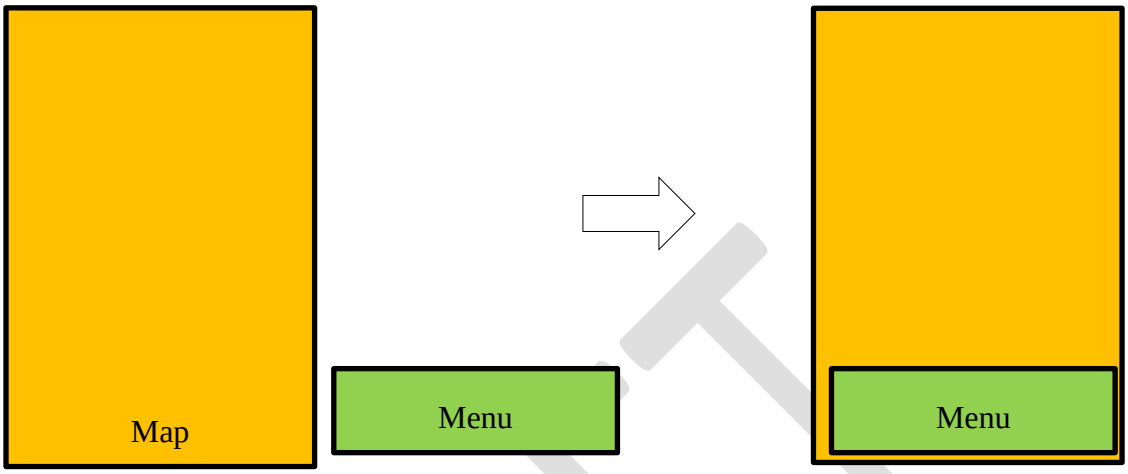
e.g.
・Change the size of the specified application.
・Change the output Screen the specified application

For details,
see HomeScreen's 「Change the Window Size」 「Change the Screen」

Layer Add Area (for Multiple Area Apps)

Add a Area(Surface) to the specified Layer.

e.g. LayerAddArea (“Navi.Menu”)



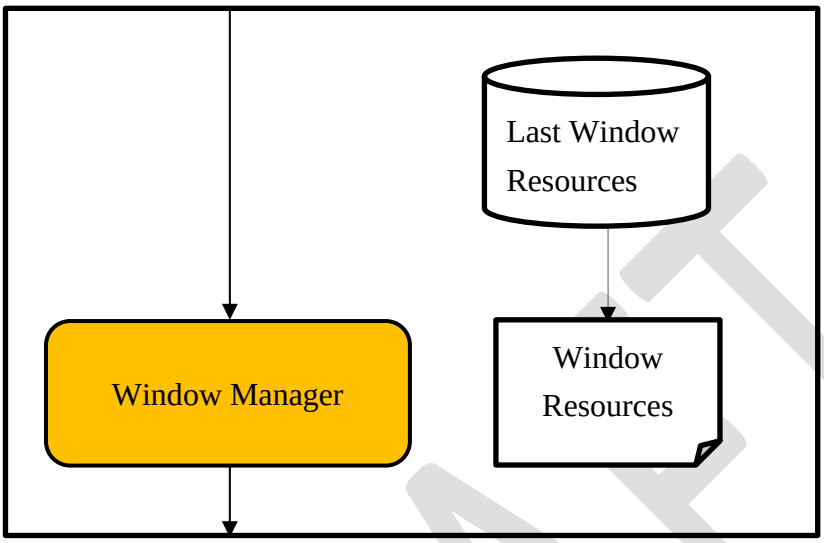
get Area Information

e.g.) "area": {"rect": { "x": x, "y": y, "width": width, "height": height } }

Recover Window Resources (Boot Sequence)

The Window Manager always holds current window resources.

After reboot, Window Manger recoverd the Last Window resources.



3.5. Window Policy Manager

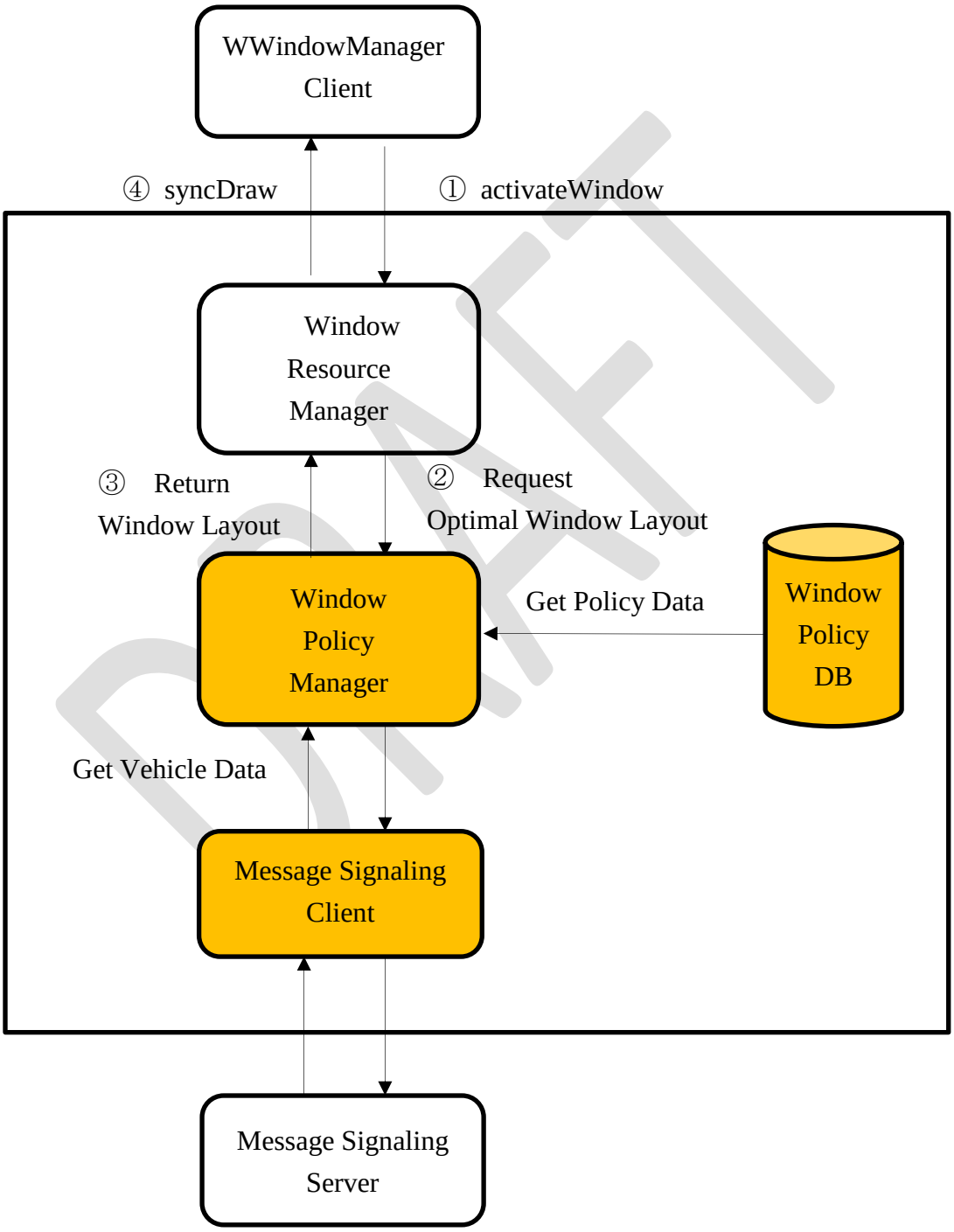
When there is a screen request from the application due to a user operation or a state change of the system, it is common to erase the old screen and display a new screen.
But, setting an optimum screen layout in consideration of the following conditions is an important requirement of an in-vehicle HMI.

- 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.

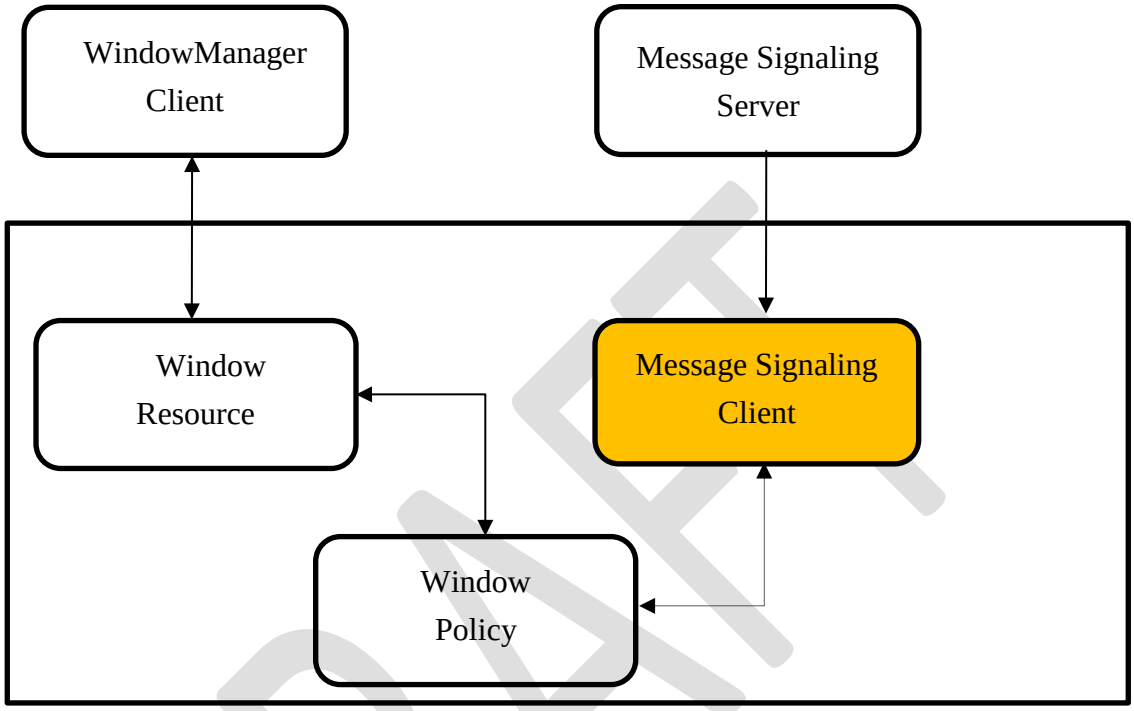
3.5.1. Policy Manager flow chart

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



3.5.2. Message Signaling Client

Policy Manager acquires latest vehicle information from Message Sginaling.

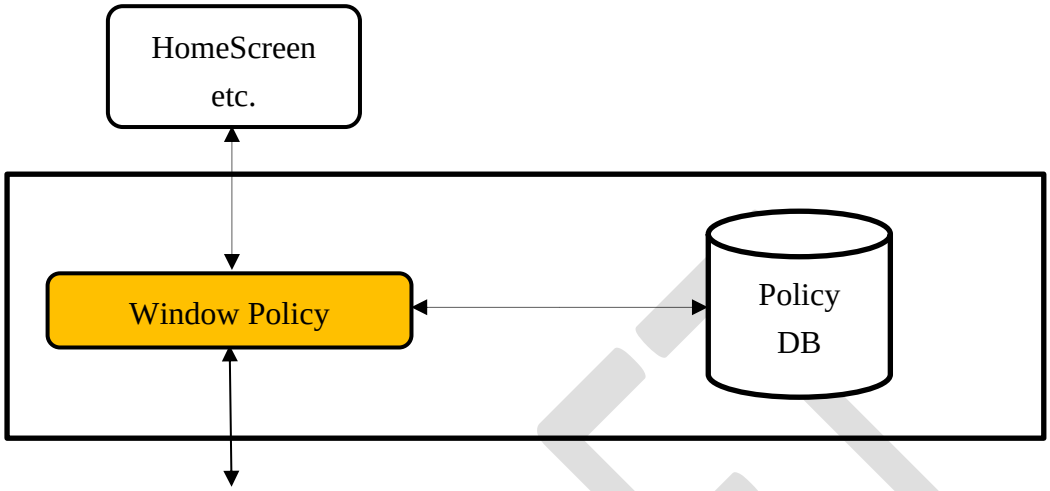


Policy Manager controls the screen according to the state of the car.
For example, when the car is in the Driver Distraction state, the Policy Manager issues an event to the related Client (HMI-Apps).

Restriction()　//EVENT
There is a need to restrict the display while the car is running.

3.5.3. Policy DB Control (Privilege Function)

HMI-Manager (HomeScreen etc.) can Get/Set Window Resources.



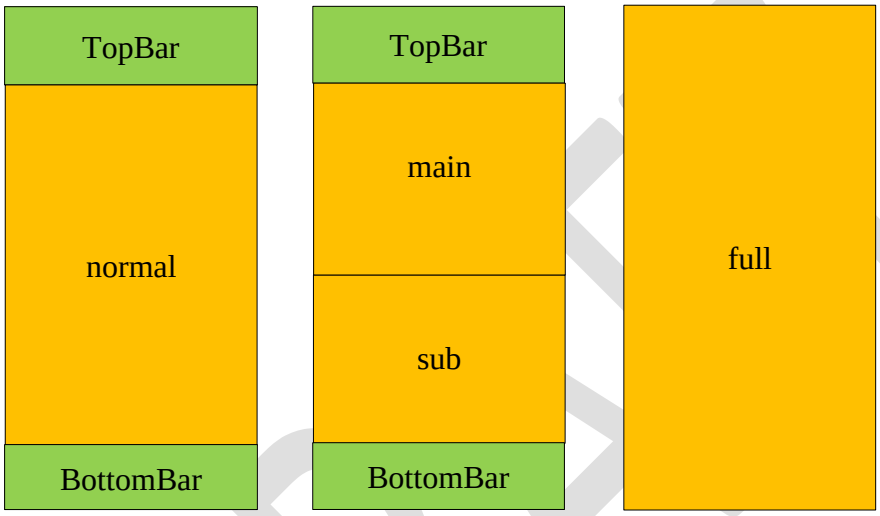
Update the Window Policy DB with the following timing.

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

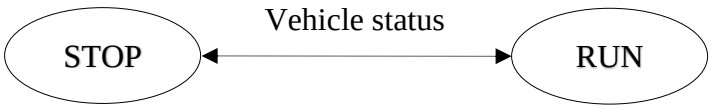
3.5.4. Use Case

An example using Policy Manager is shown below.

PolicyManager changes Navi and the two applications group A and B to the following layout according to the 「Vehicle Information」 「Drawing capability」 「Policy State machine」 etc.



Vehicle status



Drawing Capability

Each Role has an area where display is possible. An example of specifying an area in which a specific Role can be displayed.

```
“DrawingCapability”:
[{
        {
        "role": ”HomeScreen",“layer”: “homescreen”,“area”: “full”
        },{
        "role”: “PhoneCall|SystemError",“layer”: “onscreen”, “area”: “onscreen”
        },{
        "role": ”Map",“layer”: “appsEX”,“area”: “normal|split.main |full”
        },{
        “comment”: “Apps Group A”,
        "role": "MediaPlayer|Radio|Phone|POI", “layer”: “apps”,
        “area”: “normal|split.main|split.sub”
        }
},{
        “comment”: “Apps Group B”,
        "role": "Settings|HVAC|Dashboard|Mixer”,
        “layer”: “apps”,
        “area”: “normal”
}]
```

Policy State Machine

An example of state transition using PoicyManager is shown below.

① STOP

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.

	normal	main	sub	full	NAVI	NAVI:full	APPS-A	APPS-B
n1	NAVI	−	−	−	−	To nf	To n2	To b
n2	−	NAVI	＊	−	To n1	To nf	To n2	To b
nf	−	−	−	NAVI	To n1	−	−	−
a1	APPS-A	−	−	−	To n1	To nf	To a2	To b
a2	−	APPS-A	＊	−	To n1	To nf	To a2	To b
b	APPS-B	−	−	−	To n1	To nf	To a1	To b

First Row：State Name

Second Row and 5th Row：Area Name

The action after the 6th Row for the application request（State Machine Table）

＊：Other applications(Apps-A) different from main

② RUN

The state transition table during RUN is shown below.

In the case of stopping, restore the current state and shift to the STOP state.

	normal	main	sub	full	NAVI	NAVI:full	APPS-A	APPS-B
n1	NAVI	−	−	−	−	To nf	−	−
nf	−	−	−	NAVI	To n1	−	−	−

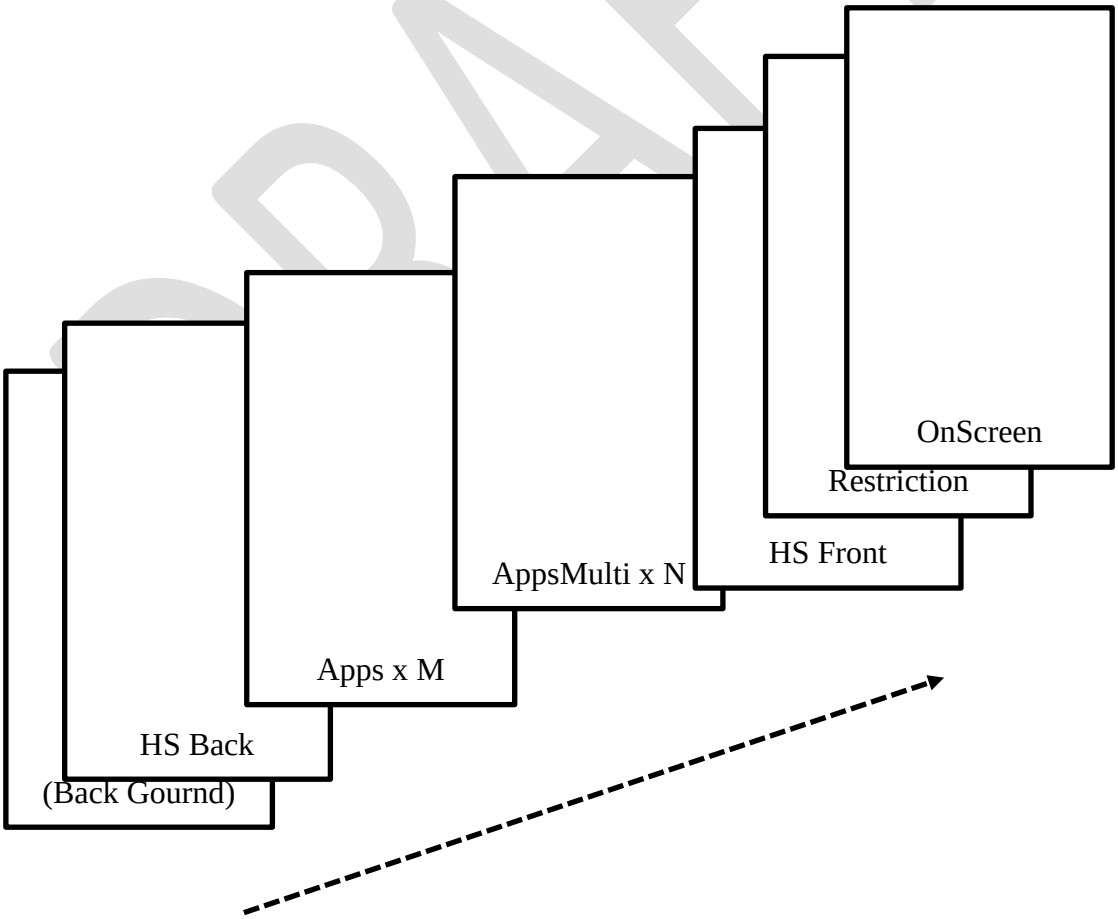
3.6. Window Layout Manager

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

3.6.1. Initial setting of Layer

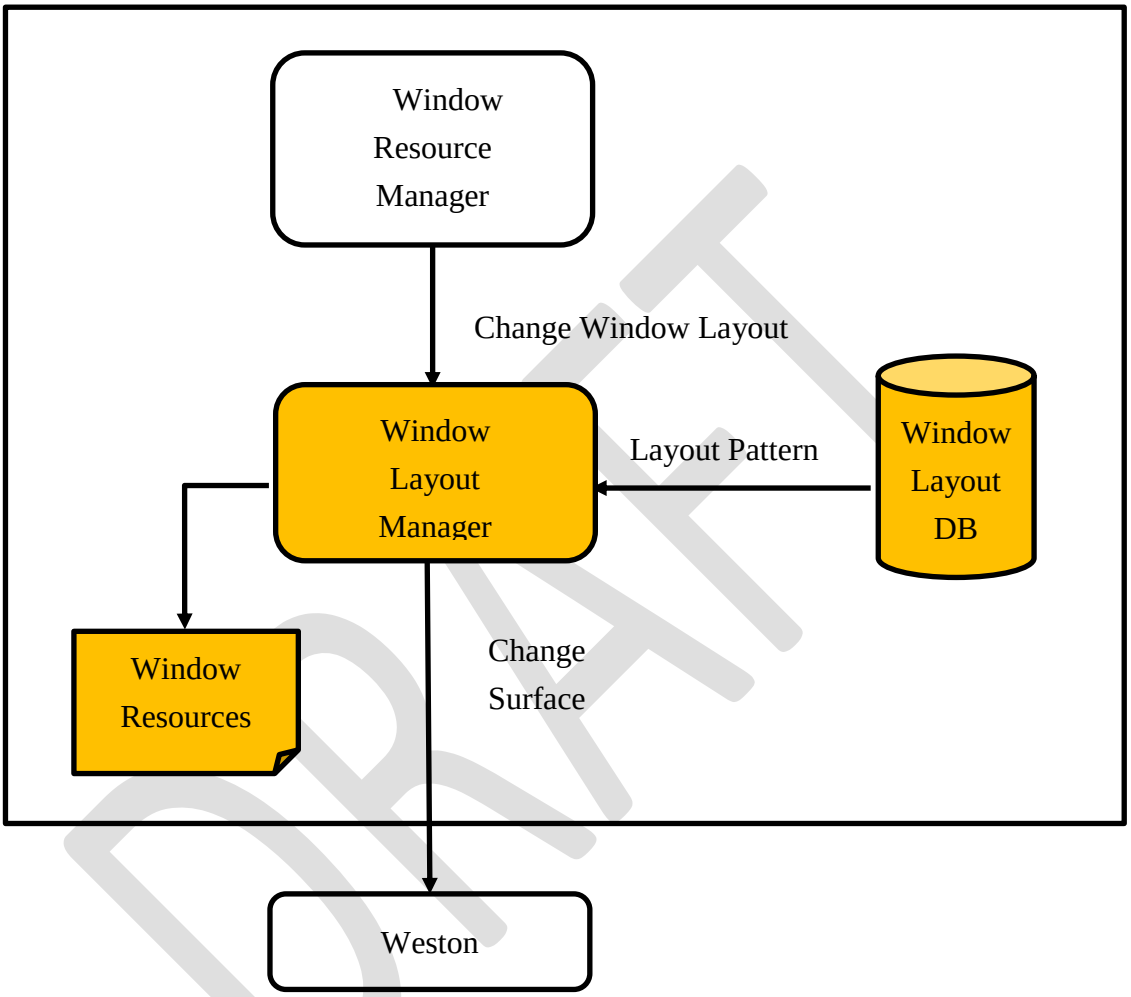
Window Manager initializes the default layer at startup.
An example is shown below.

HS Back: **MenuBar**
Apps*: **Single Area Apps**
AppsMulti*: **Multiple Area Apps (e.g. Navigation)**
HS Front: **Animation, Software KB**
Restriction: **Restriction**
OnScreen: **OnScreen**



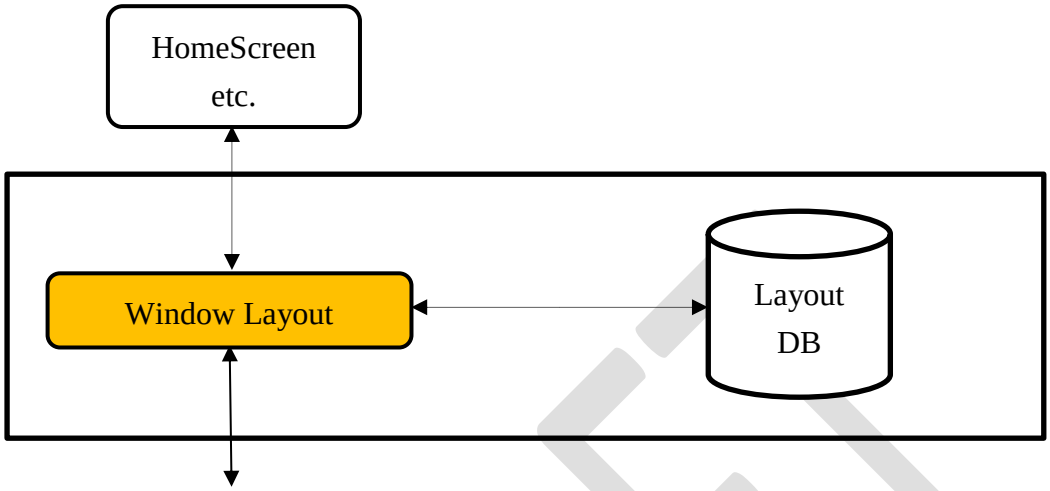
3.6.2. Layout Manager flow chart

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



3.6.3. Layout DB Control (Privilege Function)

HMI-Manager (HomeScreen etc.) can Get/Set Layout DB.



Update the Window Layout DB with the following timing.

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

3.6.4. Use case

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

Layout.DB

① Apps Layout

“normal”: {“comment”: “Normal Screen”,“layer”: “Apps”,
"area": {…}
},"split": {“comment”: “Split Screen”,“layer”: “Apps”,
"main":{ "area": {…}},"sub":{ "area": {…}}
},“full”: {“comment”: “Full Screen”,“layer”: “Apps”,
"area": {…}
}

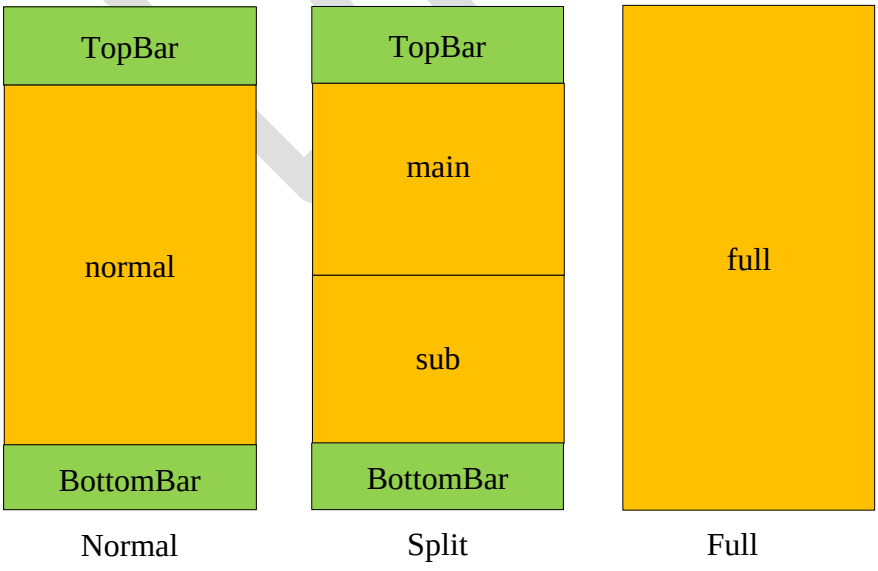
② HomeScreen Layout

"normal": {“comment”: “Home Screen”,“layer”: “HomeScreen”,
"Top":{ "area": {…}},
"Bottom":{ "area": {…}}
}

③ OnScreen Layout

"normal": {“comment”: “OnScreen”,”layer”: “OnScreen”,
"area": {…}
}

(*) "area": {"rect": { "x": x, "y": y, "width": width, "height": height } }



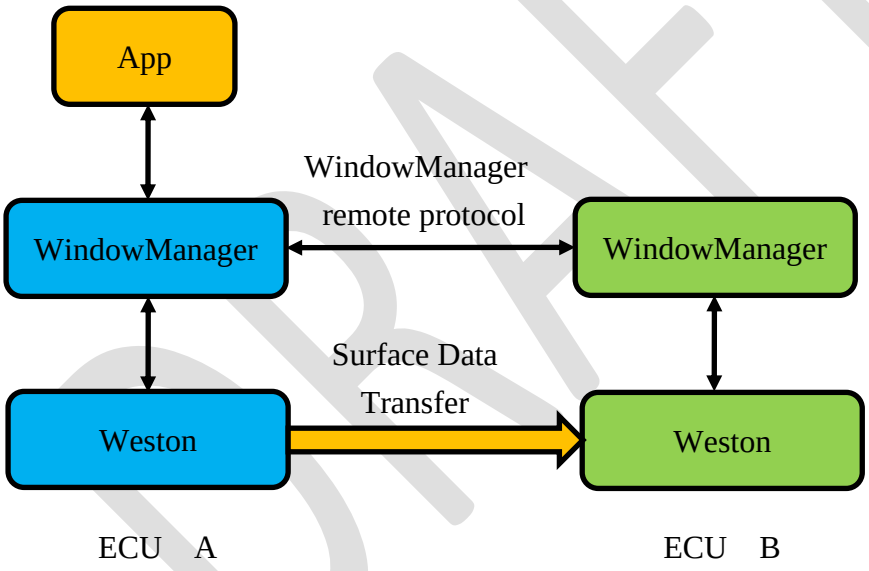
3.7. Multi ECU Extention

3.7.1. Overview

WindowManager can be extended to Multi Display including Multi ECU.

WindowManager remote protocol
Protocol for multiple WindowManagers to cooperate and output to different screen

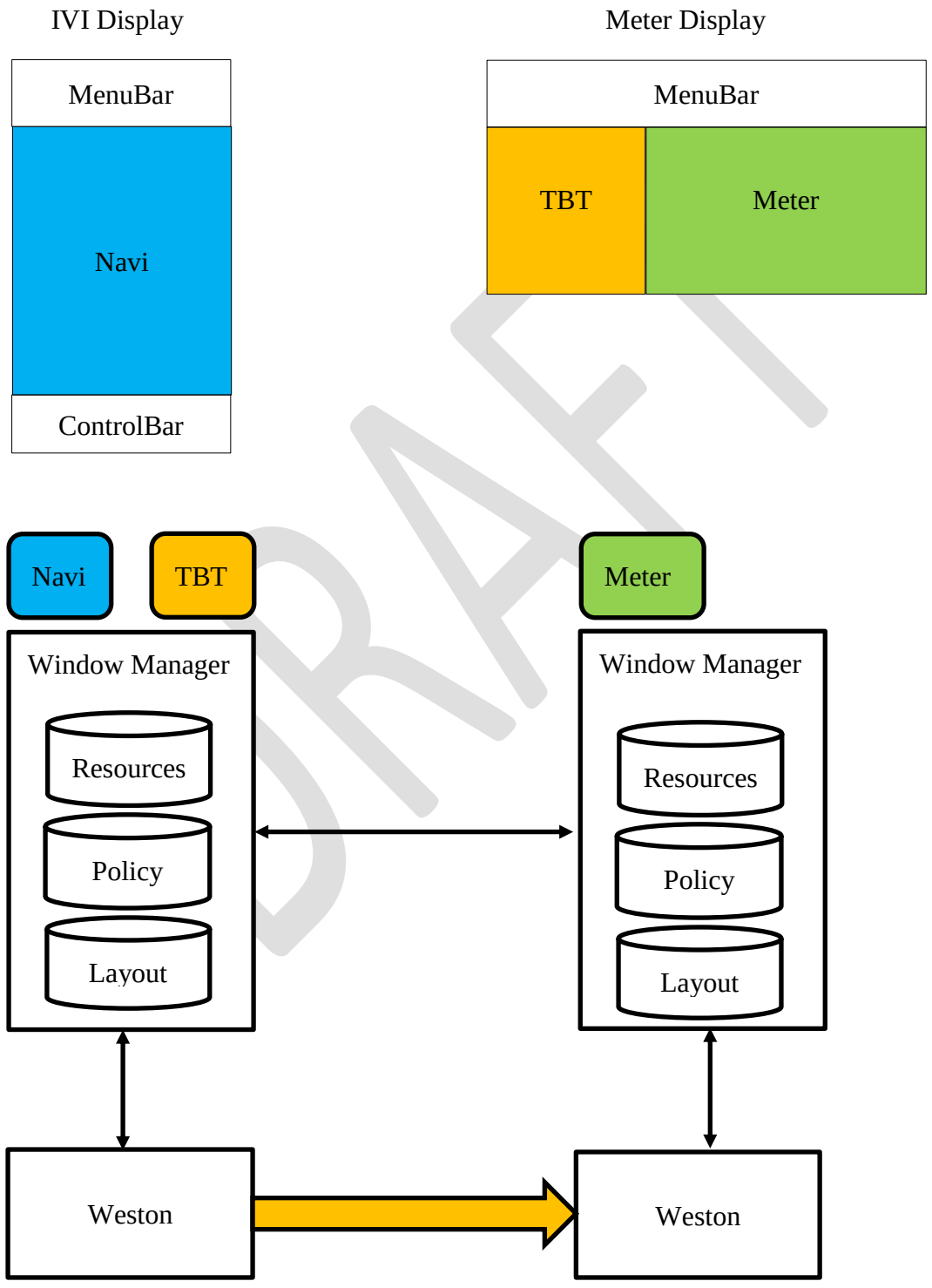
Surface Data Transfer
Transfer screen data (Surface) of application to different ECU according to the instruction of WindowManager



Display Since changing of the output destination is carried out by HomeScreen, modification of the application is unnecessary.
※ For details, see HomeScreen's 「Change the Screen」

3.7.2. Use case

In the following example, the screen of the TBT application on the IVI side is output to the Meter side.

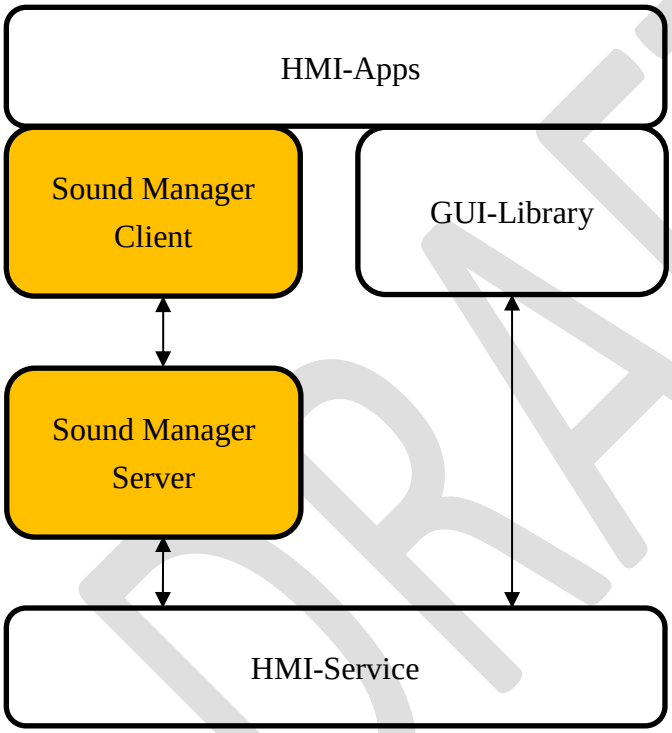


4. Sound Manager

4.1. Overview

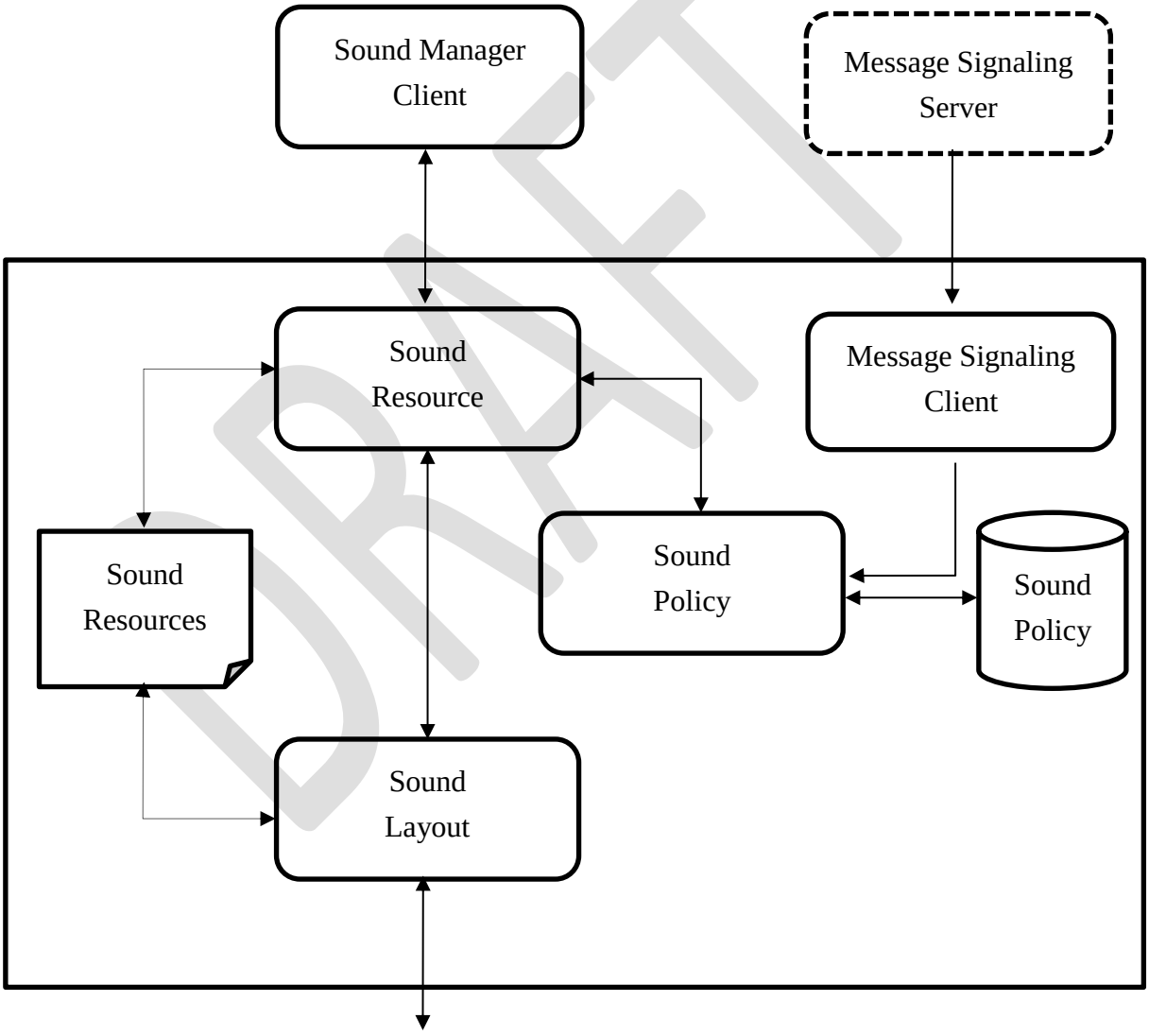
Sound Manager determines the optimum sound layout and controls the sound、based on the request from the HMI-Apps.

4.1.1. Related external components



4.1.2. Internal Components

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



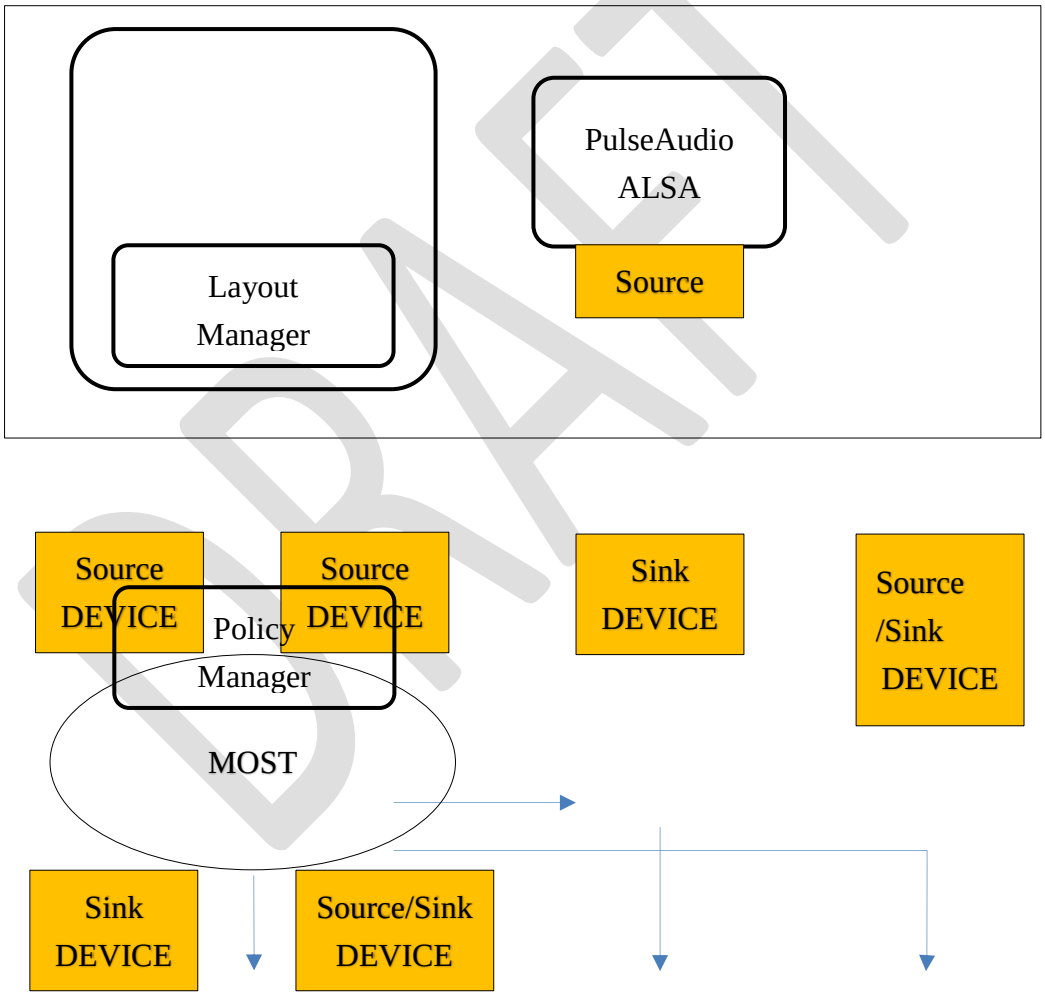
4.1.3. Sound Resources

The sound system has the following two sound nodes.

- ✓ Source: Input Sound Node
- ✓ Sink: Output Sound Node

The connection state of the node is called Sound Layout.

After the 「Sound Policy Manager」 decides to reconfigure the Sound Layout according to the request from the application, 「Sound Layout Manager」 controls connection and disconnection between source and sink



4.2. Sound Manager Client

4.2.1. API

No	Function	R/W	Description
1	Init	W	Connect to Sound Manager
2			
3	Allocate Sound Resources	W	Request Allocate Resources
4	Release Sound Resources	W	Request Release Resources
5	Sound Resources Control	R/W	Get/Set Sound Resources
6	Sound Policy DB Control	R/W	Get/Set Policy DB

4.3. Sound Resources Manager

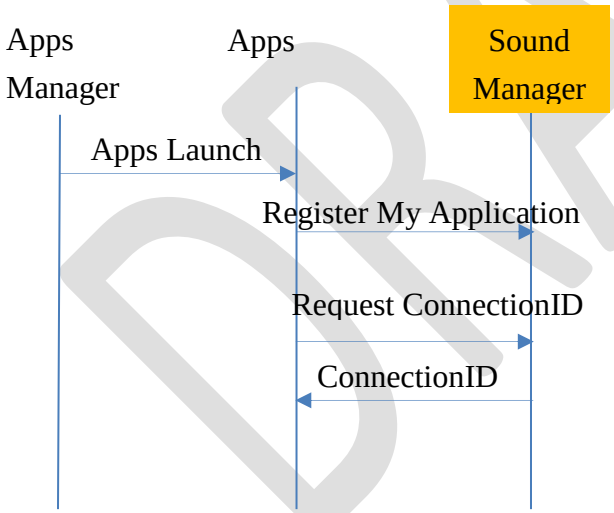
4.3.1. Initializing Stage

Register My Application (API)

When an application uses Sound Manager, registration of the application is necessary.

Request ConnectionID (API)

The application gets ConnectionID.
Sound Manager returns the ConnectionID according to the specified「Sounding Label」
An application can use more than one connection.



4.3.2. Sounding Stage

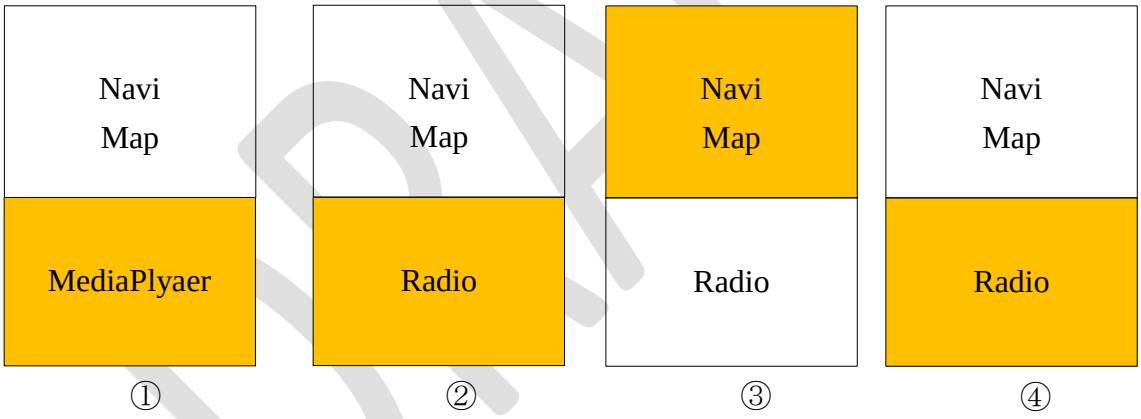
Allocate Sound Resources (API)

When the application starts sounding, it is necessary to acquire Sound Resources.

Use Case of Allocate Sound Resources

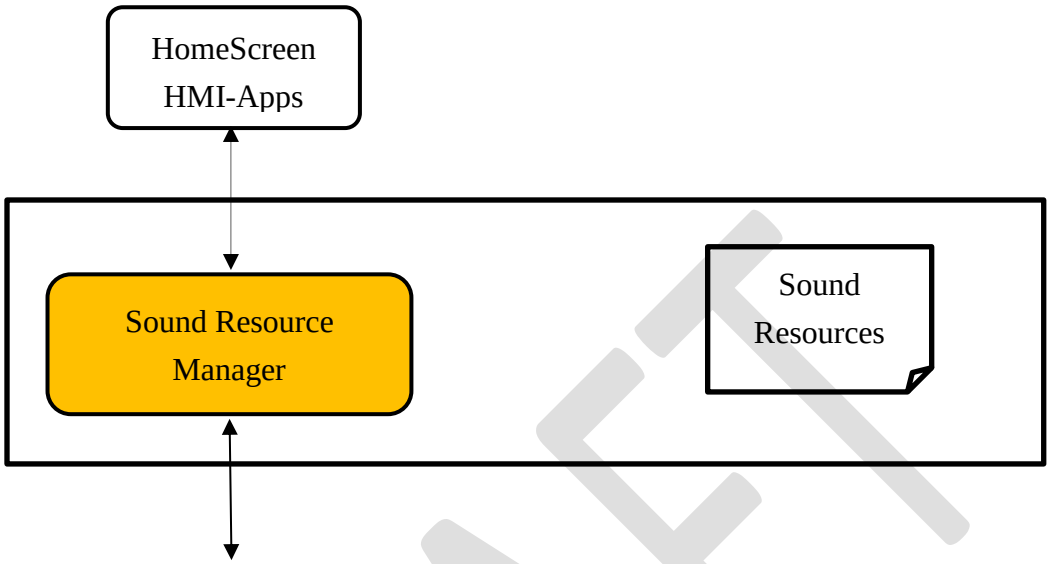
The use cases are shown below.

① Listening to music with Media Player
② A radio is selected (normal)
③ Navi requires a voice guide (Interrupt)
④ Back to the radio

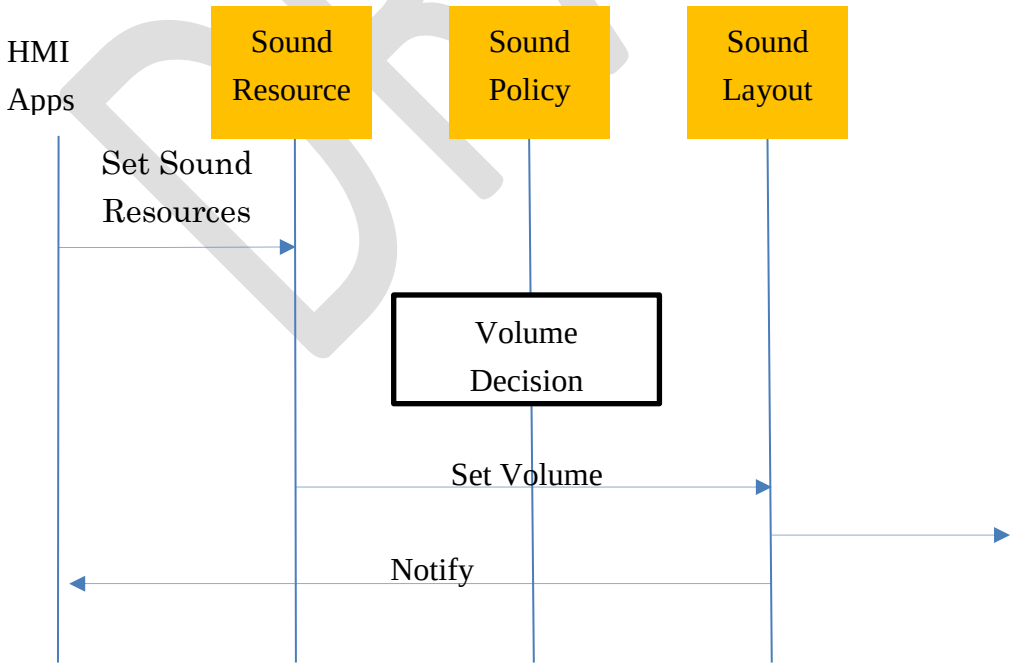


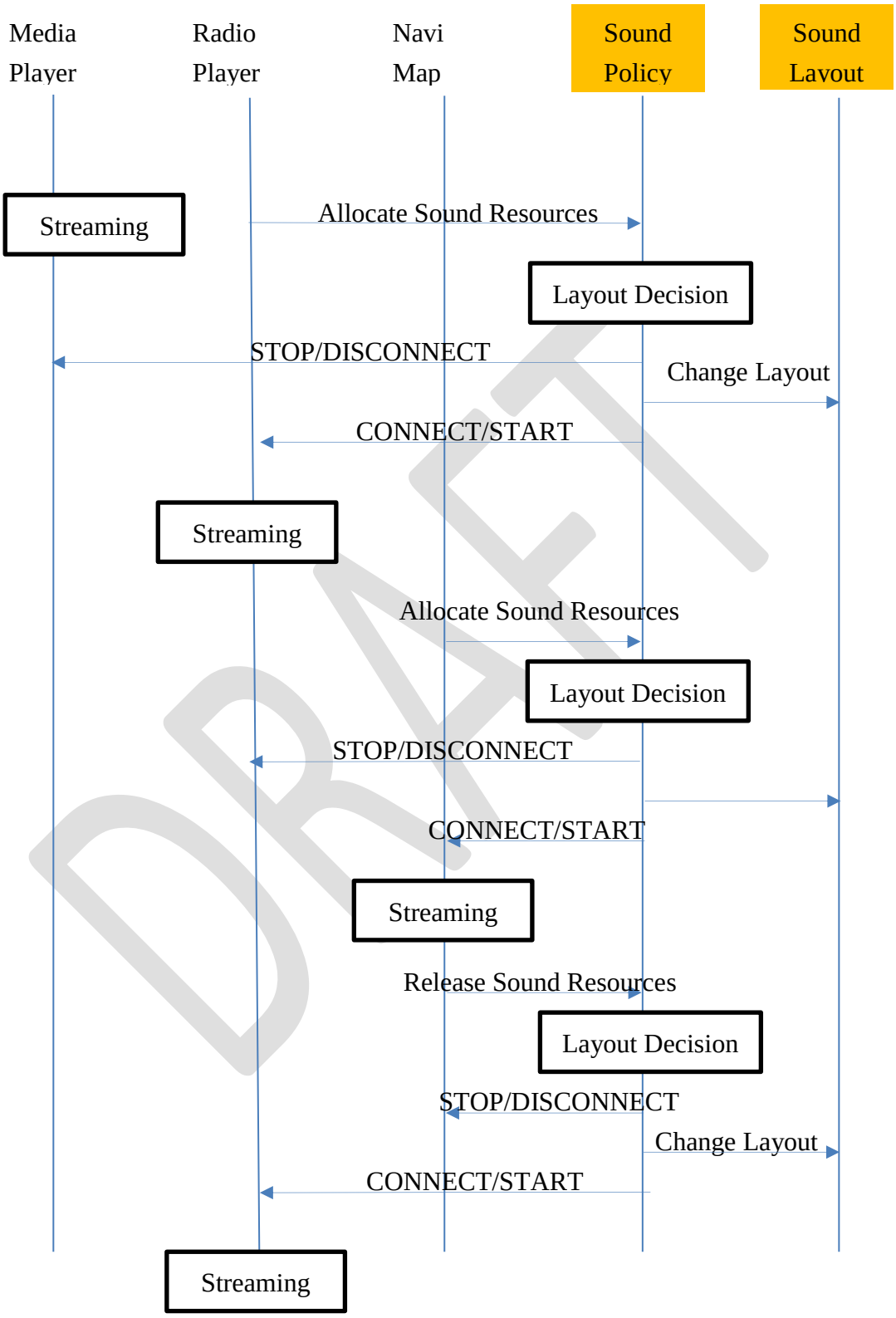
4.3.3. Sound Resource Control (API)

HMI Apps Get/Set Sound Resources.



Use Case of Set Volume





4.4. Sound 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.

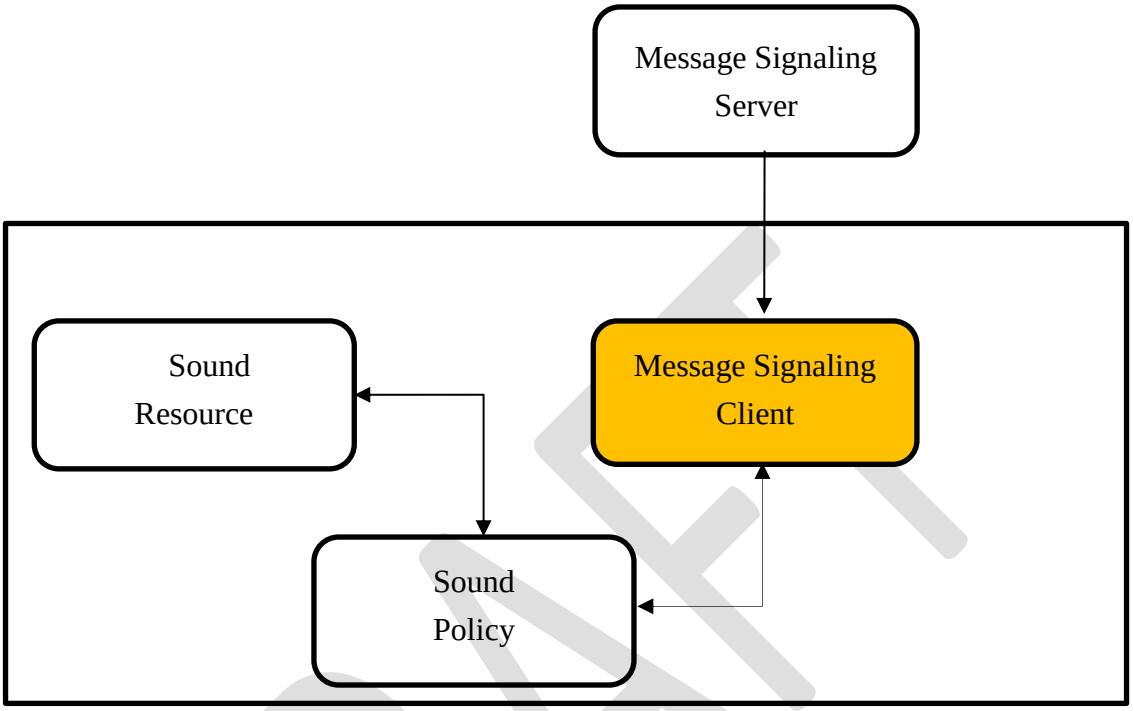
4.4.1. Sound Policy DB Control (Sound Manager API)

Update the Sound Policy DB with the following timing.

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

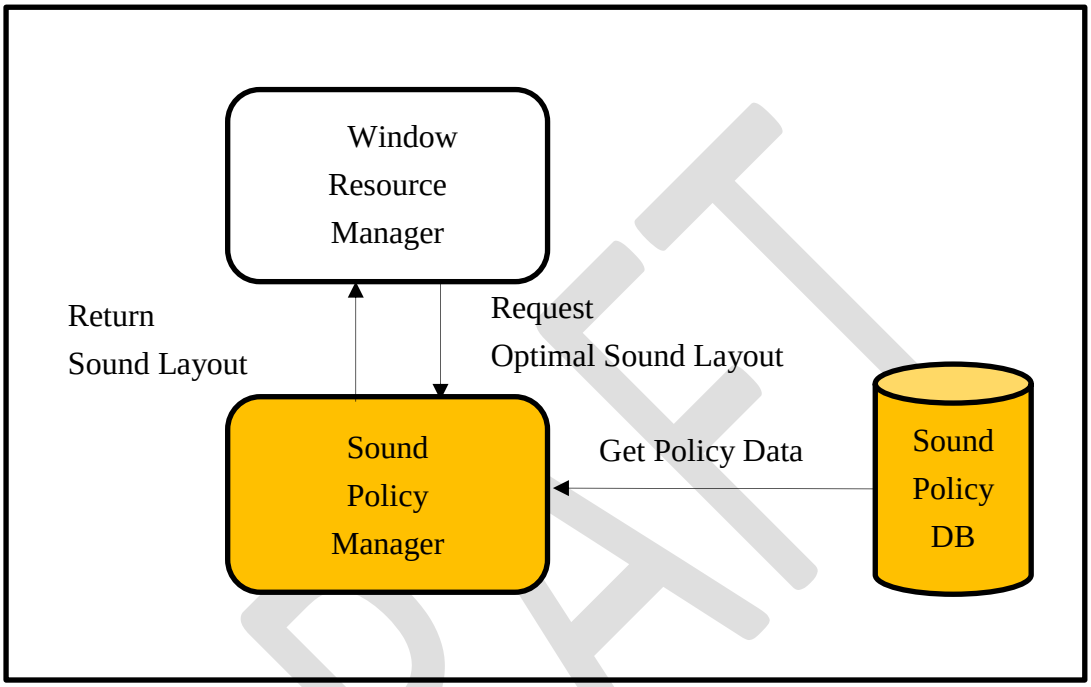
4.4.2. Message Signaling Client

Policy Manager acquires latest vehicle information from Message Sginaling.



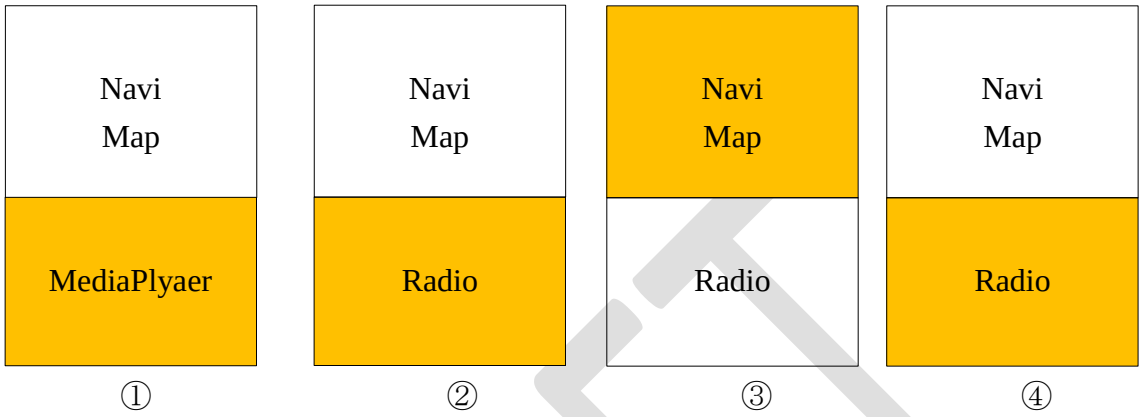
4.4.3. Policy Manager flow chart

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



4.4.4. Policy manager use cases

Policy DB State Machine



	MP	Radio	Navi(Start)	Navi(Stop)
MP	−	To RADIO	PUSH Status(MP) To NAVI	
RADIO	To MP	−	PUSH Status(MP) To NAVI	
NAVI	−	−	−	POP Status To Status

First Row：State Name

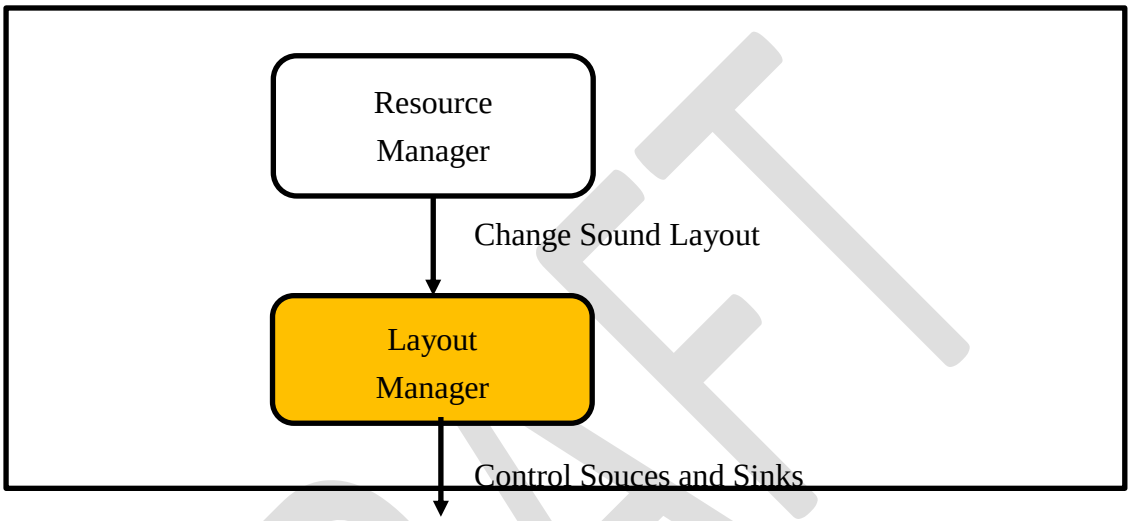
The action after the 2th Row for the application request（State Machine Table）

4.5. Sound Layout Manager

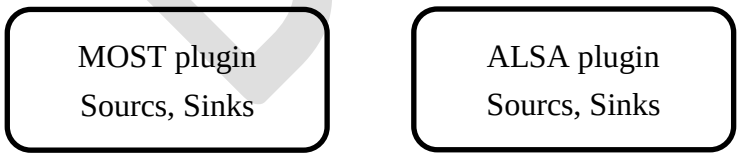
The Sound Layout Manager has the following functions related to Layout.

4.5.1. Change Sound Layout

If Sound Layout Manager receive「Change Sound Layout」, they need control Source and Sinks.



The Layout Manager has a mechanism of plug-in so that it can add a new sound device. Individual plugins have different sources and sinks.

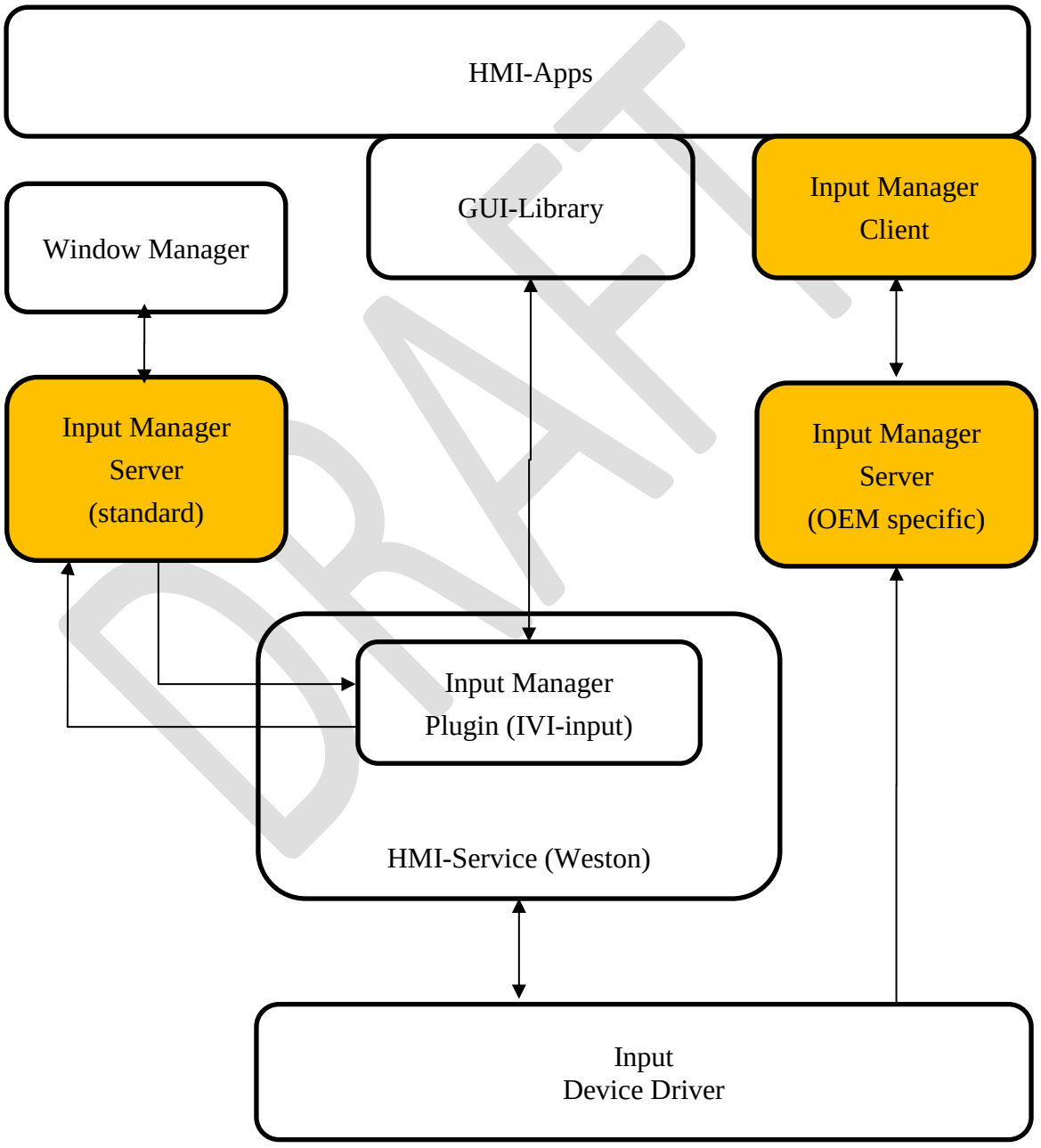


5. Input Manager

5.1. Overview

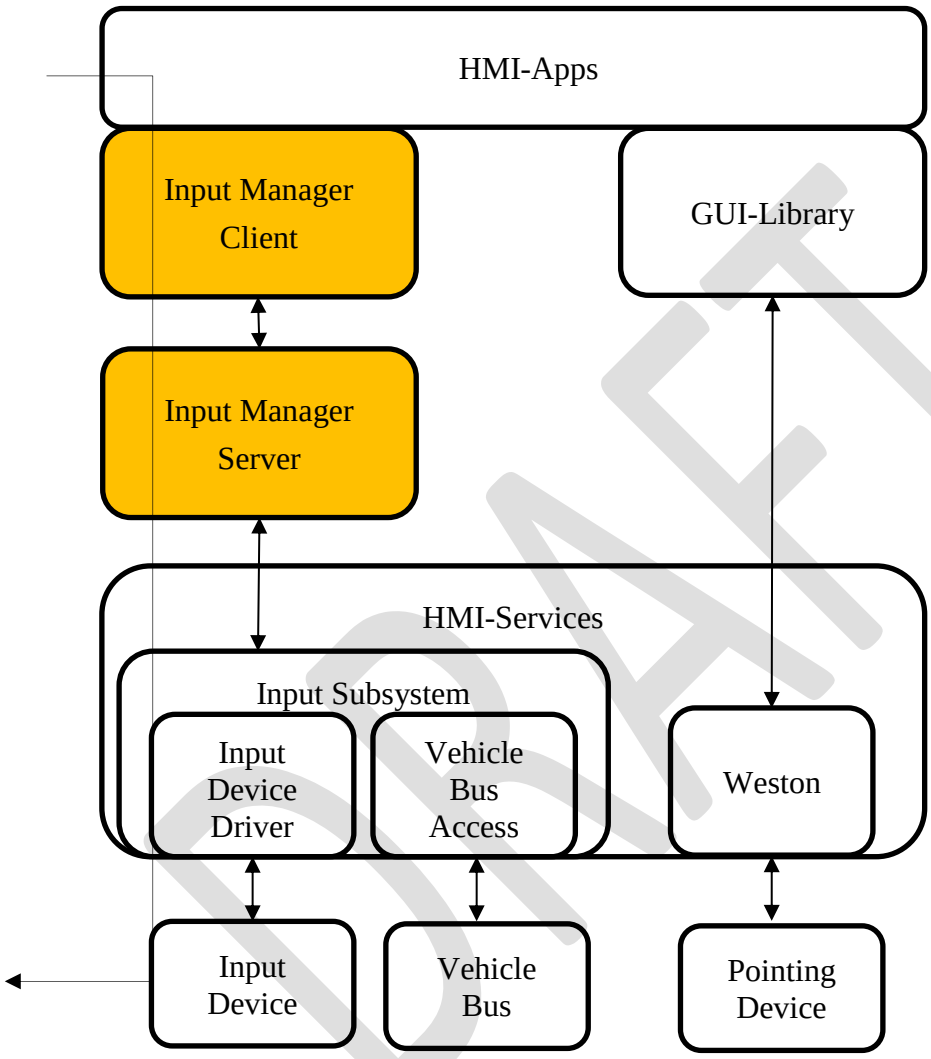
Input Manger provides access to HMI-Apps about input data. Input Manager accepts input data request from HMI-Apps, and deliver the requested input data.

5.1.1. Related external components



Related external components

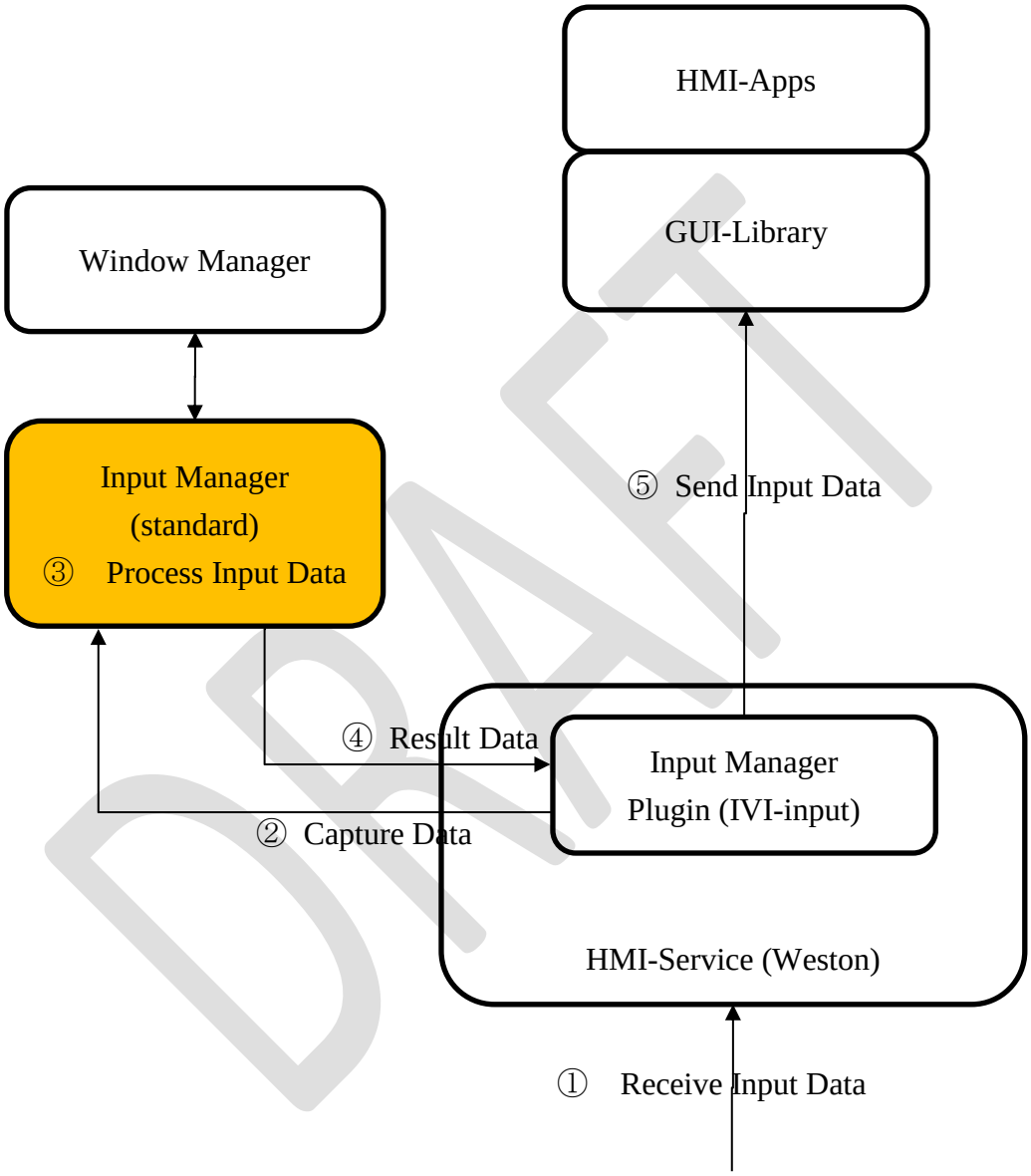
Input Manager doesn't handle the the input data that get from Weston in the HMI-Service and processed by GUI-Library.



Pointing Device is configured so that Weston can handle it. If Pointing Device such as haptic device needs control value for operation feedback, HMI-Apps set the control value to the Pointing Device.

5.2. Input Manager（Standard Device）

InputManager performs special processing of input data on the standard input device (Pointer, Keyboard, Touch) as follows.



An example of InputManager processing is shown below.

Judge focus Window

The InputManager judges the Window where the user operation has been made and notifies the WindowManager.

5.3. Input Manager (OEM Specific Device)

5.3.1. clinet

The API is shown below.

No	Function	R/W	Description
1	Init	W	Connect to InputManager
2	activateInput	W	OEM Specific Device
3	deactivateInput	W	OEM Specific Device

5.3.2. Server

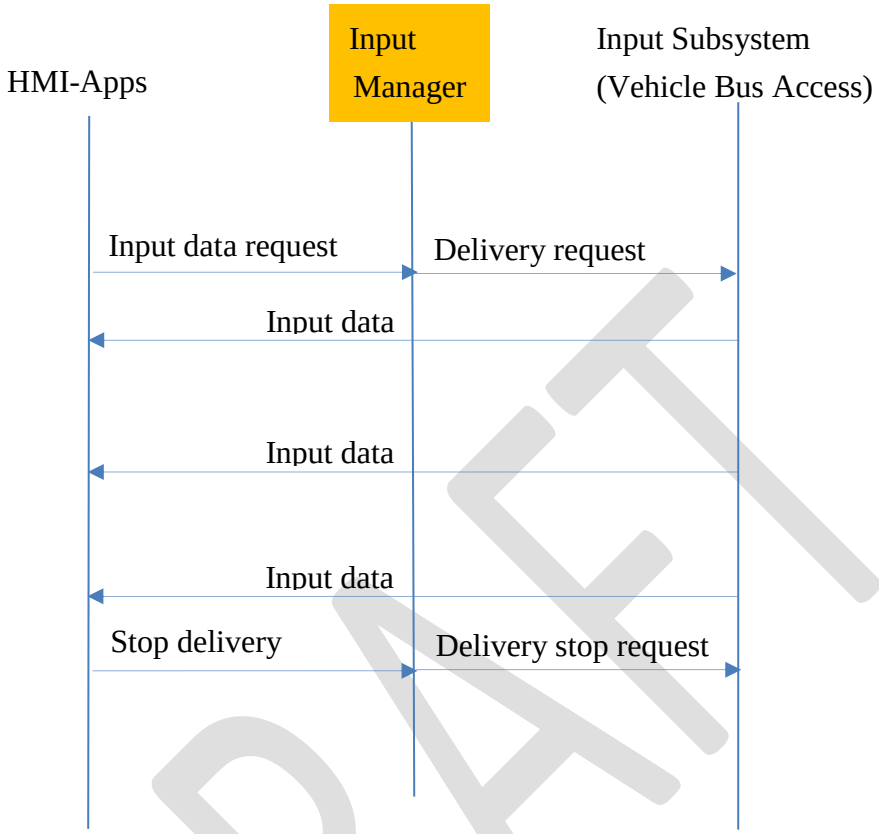
activateInput (Role, Area)

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

deactivateInput (Role, Area)

Applications issue when resources become unnecessary.

Following is Vehicle Bus Access of sequence chart.

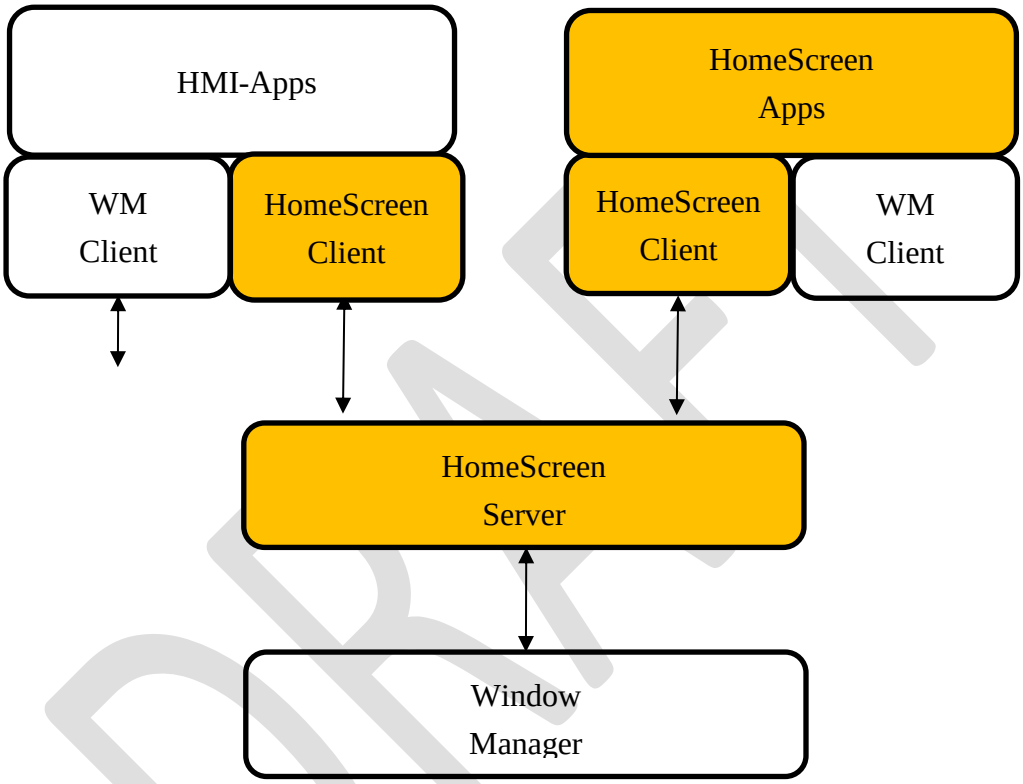


6. Home Screen

6.1. OverView

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

6.1.1. Related external components



6.1.2. Internal Components

No	Component	Description
1	Client	HomeScreen Library
2	Server	HomeScreen Service
3	MenuBar	MenuBar (HomeScreen Apps)
5	Restriction	Display Restriction (HomeScreen Apps)
6	OnScreen	OnScreen (HomeScreen Apps)
7	Launcher	Apps Launcher (HomeScreen Apps)
8	SKB	Software Key Board (HomeScreen Apps)

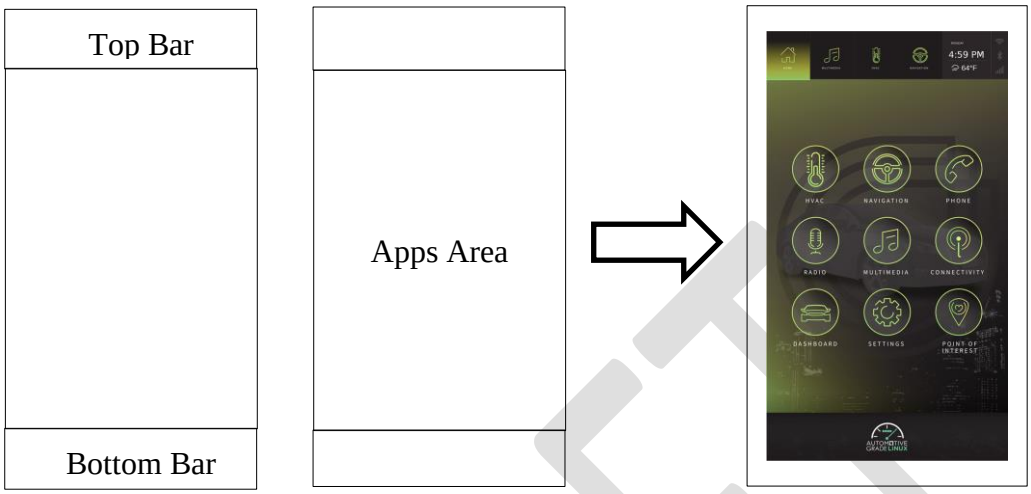
6.2. Home Screen Client

6.2.1. API

No	Function	R/W	Description
1	Home Screen Resources Control	R/W	Get/Set Resources
2	Short Cut key	W	
3	Status Bar	W	
4	Information Bar	W	
5	Home Key	W	
6	OnScreen	W	
7	Apps Lancher	W	
8	Software KeyBoard	W	
9	Notify	R	

6.3. HomeScreen Server

The standard Home Screen sample is shown below. (Sample)



6.3.1. Initial Setting

HomeScrenn acquires the surfaceID from Window Manager at startup.

① HOMESCREEN NormalSurface (HomeScreen Layer)
② ONSCREEN OnScreen Surface (OnScreen Layer)
③ RESTRICTION Restriction Surface (Restriction Layer)

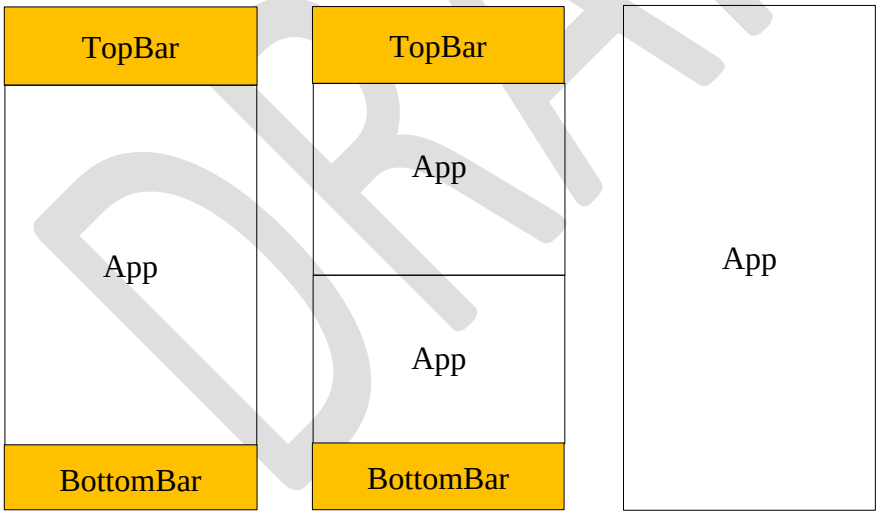
6.4. HomeScreen Apps

6.4.1. Menu Bar (HomeScreen Layer)

General HomeScreen functions are shown below.

Shortcut key
The user selects an application to use with Shortcut menu.
Status Bar
The Status Bar shows status information by notification command from each application.
Information Bar
The Information Bar shows application information by information data from each application.

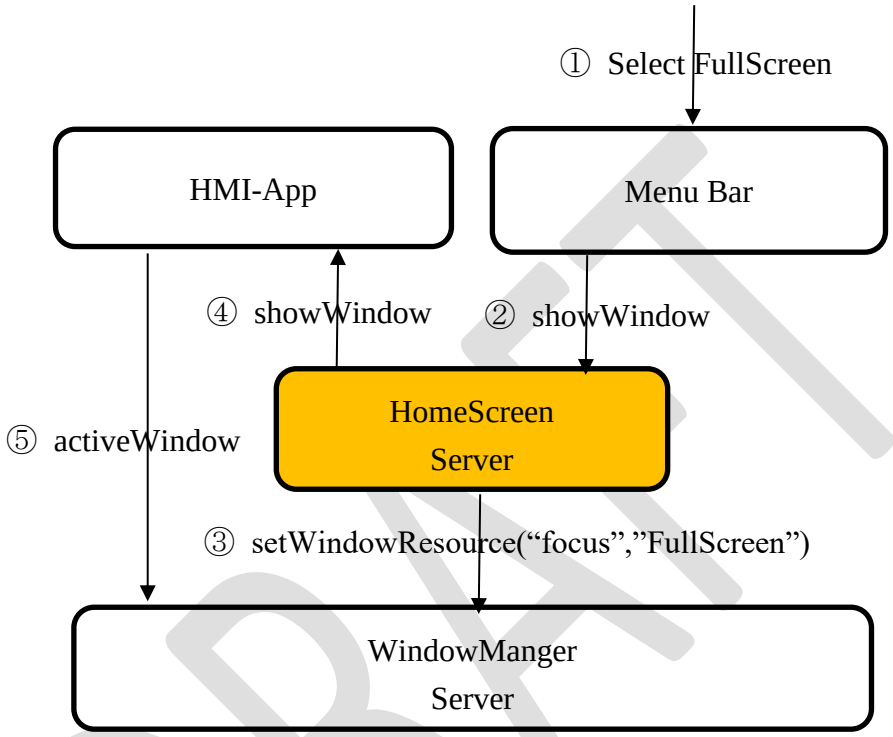
The position of the menu bar is shown below.
If the application is full screen, the menu bar is hidden.



Change the Window Size

MenuBar controls the active window by user operation.

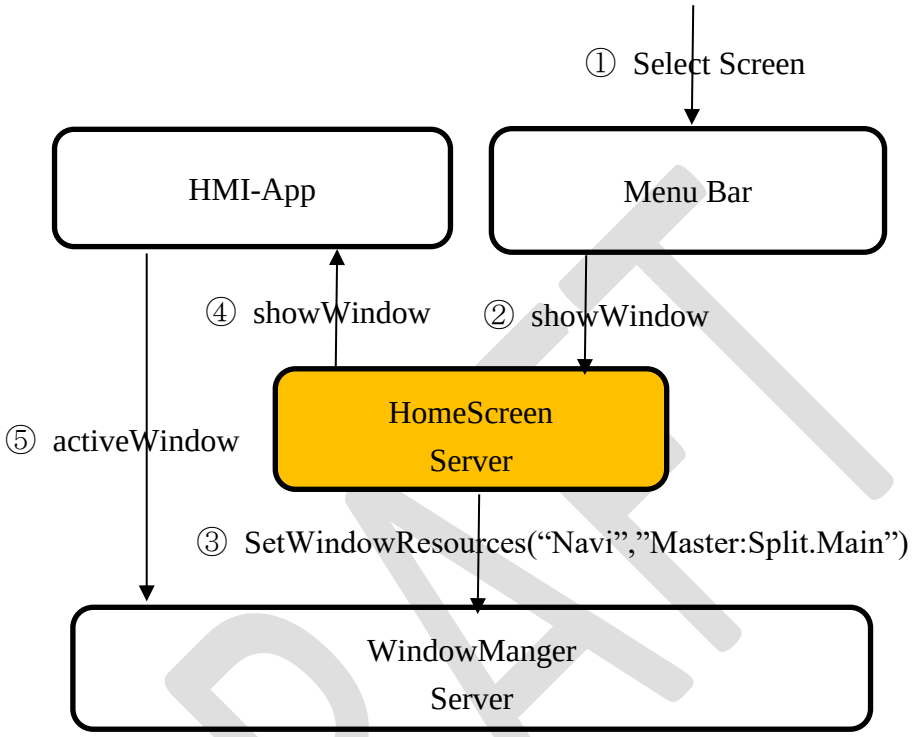
Below is an example of FullScreen operation.



(*) If “focus” is specified for Role, it is Window on focus.

Change the Screen

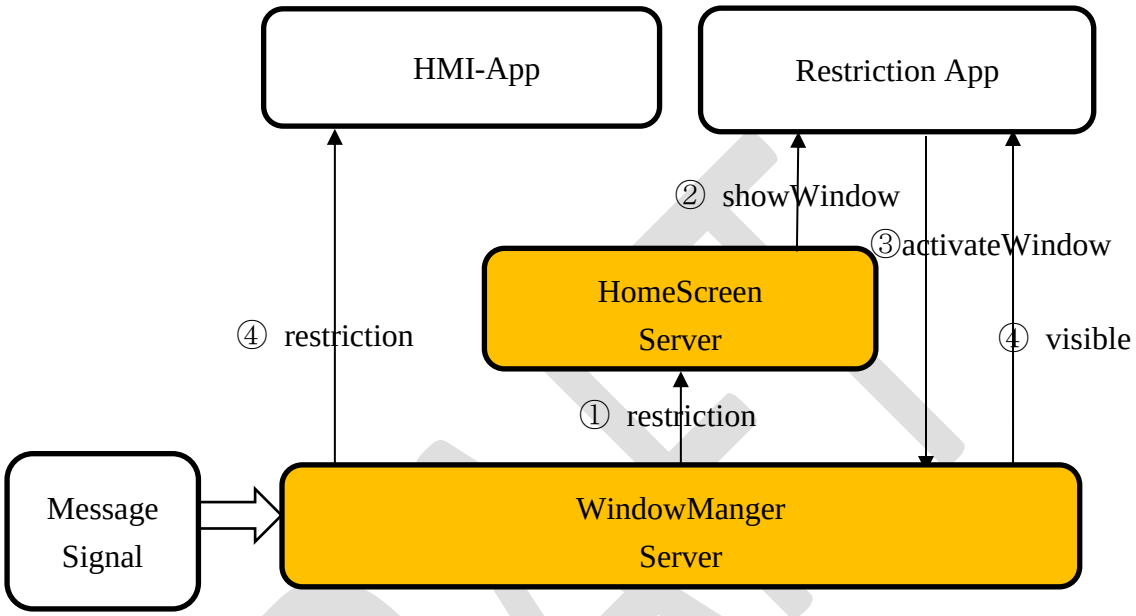
Below is an example of Change the Screen by user operation.



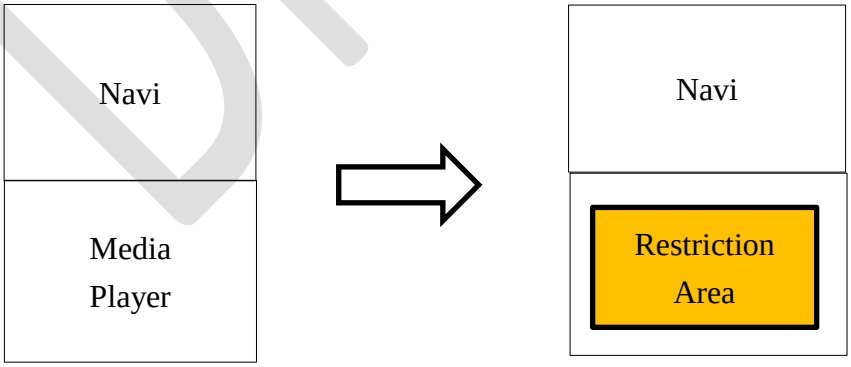
(*) If “focus” is specified for Role, it is Window on focus.

6.4.2. Display Restriction (Restriction Layer)

When the vehicle starts running, the WindwManager sends the event to the HomeScreen. The HomeScreen can request the HomeScreen to restrict running screen.



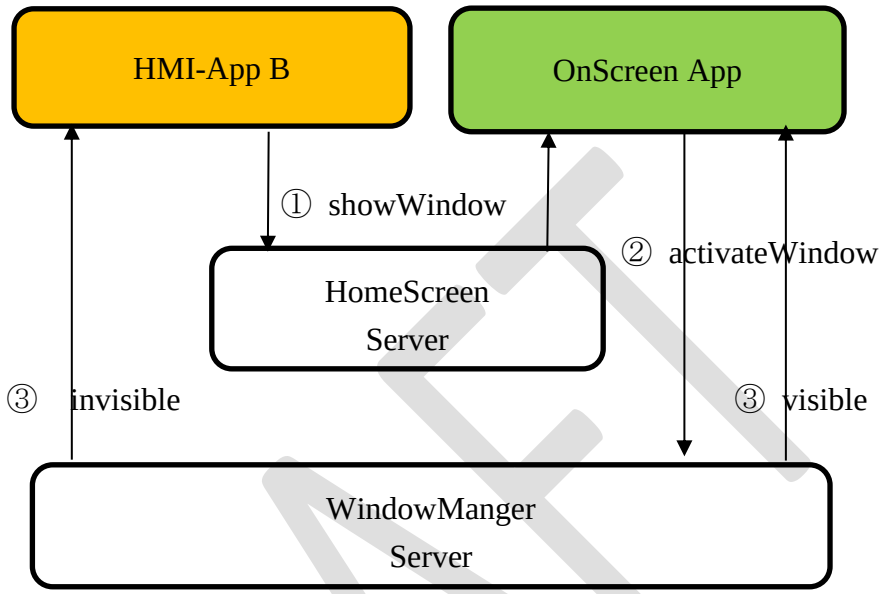
HomeScreen displays the restriction screen on the application screen so as not to accept input.

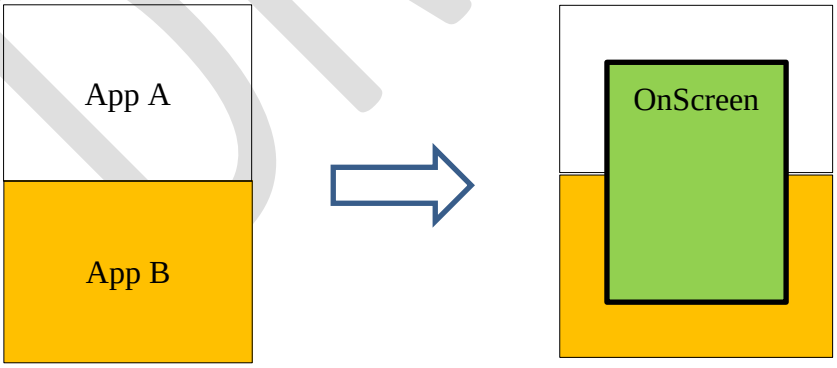


6.4.3. OnScreen (OnScreen Layer)

On Screen displays on the screen when notification from each application is received.

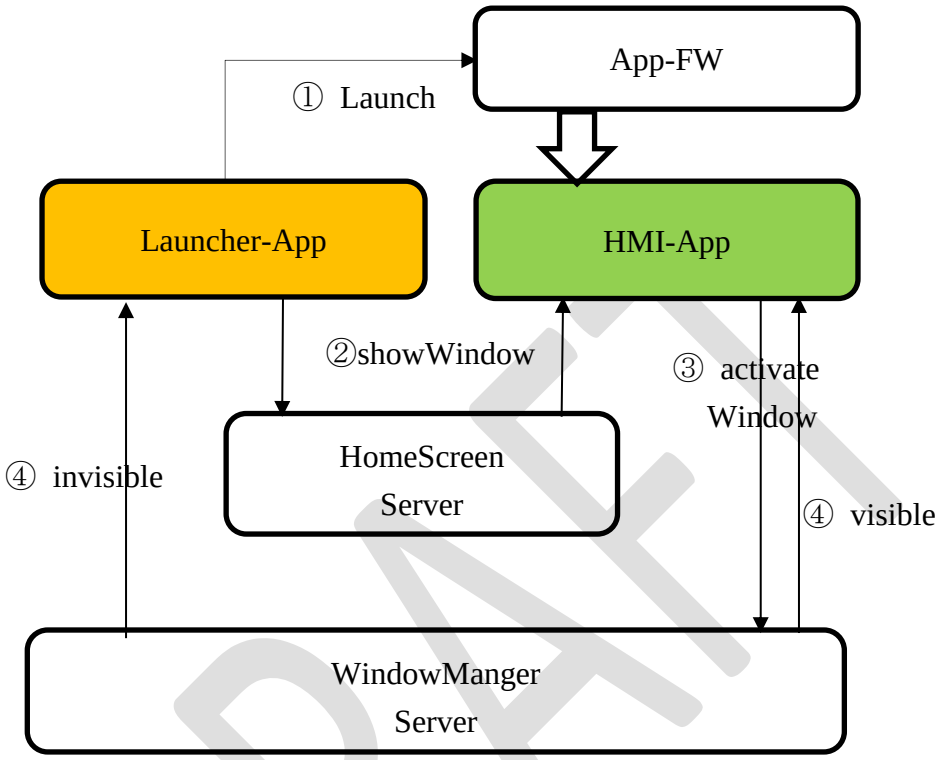
OnScreen Sample

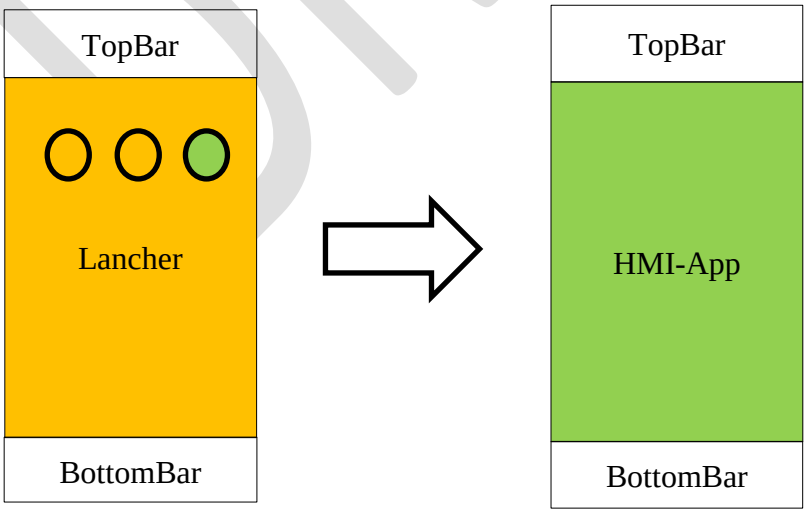




6.4.4. Apps launcher (Apps Layer)

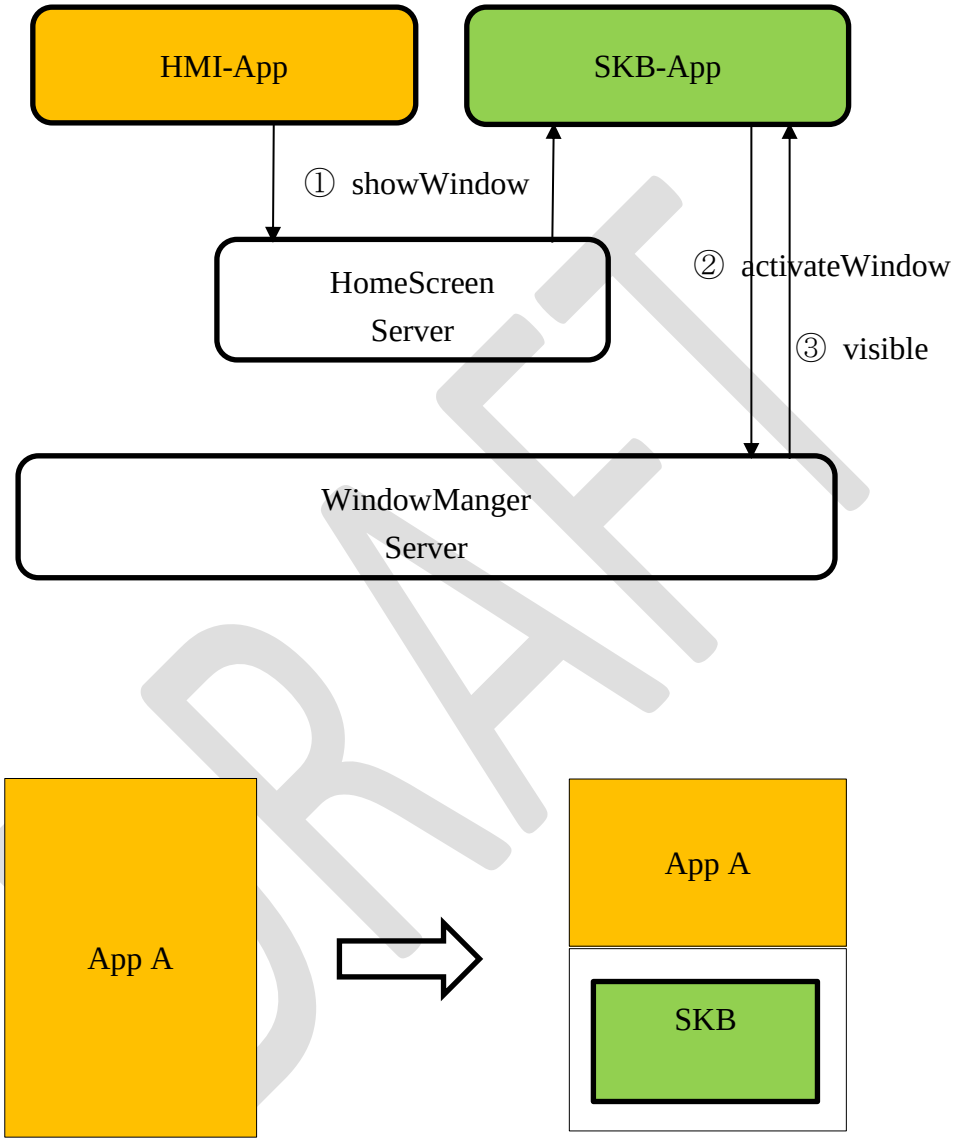
The user can select necessary applications from the application menu.





6.4.5. Software Key Board (Apps Layer or HS Layer)

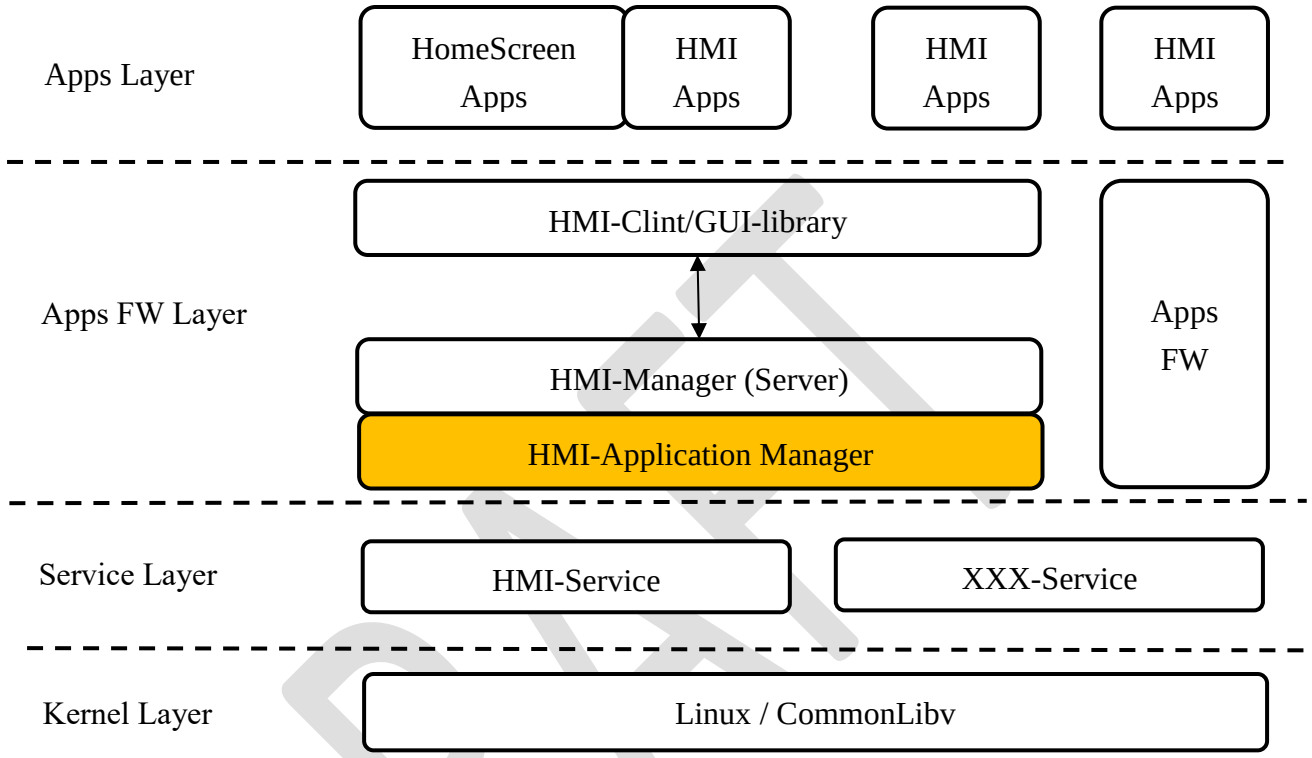
The application can call SKB with user operation.



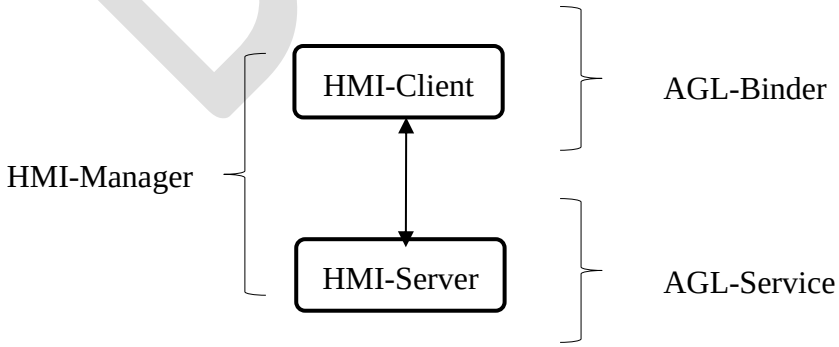
7. HMI-Application Manager

7.1. Overview

7.1.1. HMI Application Manger position in AGL



The relationship between HMI-FW and AGL Apps-FW is shown below.

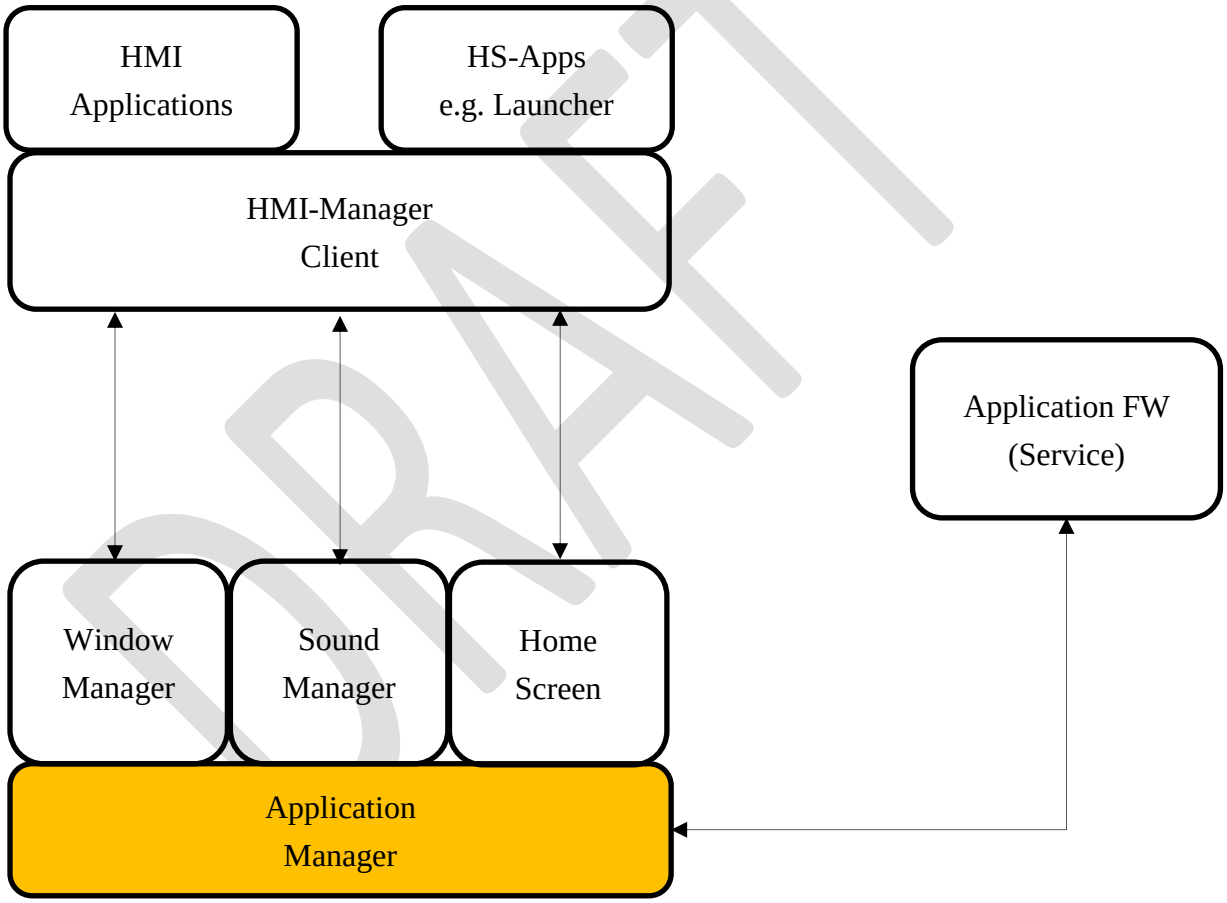


7.1.2. Related external components

HMI-manager group

No	Name	Service Description
1	WM	Window Manager
2	SM	Sound Manager
3	HS	Home Screen

Software structure



7.1.3. Internal Components

No	Function	Description
1	Application Manaager Client	API
2	Application Manager	Application Manager Service

7.2. Application Manager Client

7.2.1. API

No	Function	W/R	Who is ordered?	Server@v0.8
1	Start Application	W	PrivilegeApps	HS
2	Stop Applicaton	W	PrivilegeApps	HS
3	Get Application Information	R	The App itself / PrivilegeApps	HS
4	Get ListSTOPApps	R	PrivilegeApps	HS
5	Get ListRUNApps	R	PrivilegeApps	HS
6	GetListSUSPENDApps	R	PrivilegeApps	T.B.D.

Privilege Apps: Launcher, MenuBar etc.

7.2.2. EVENT

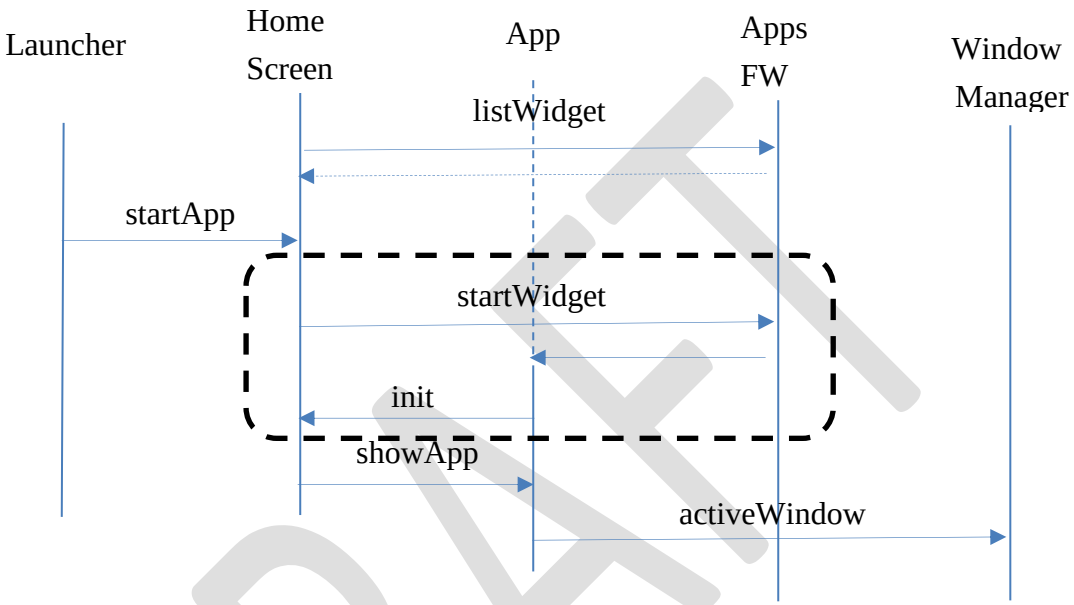
Window Manager notifies the application at the event when the situation of Window Rersources changes.

No	EVENT	Description
1	RUN	When own App becomes RUN
2	SUSPEND	When own App becomes SUSPEND
3	STOP	When own App becomes STOP

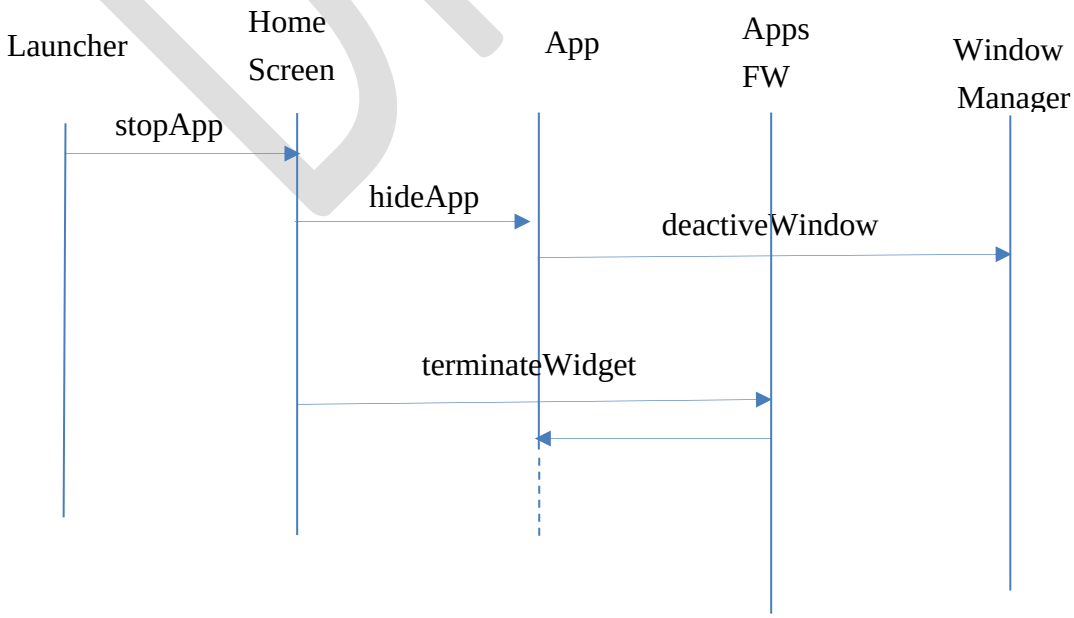
7.3. Application Manager

7.3.1. Start application

If the application is already installed and in the STOP state, instruct AppFW to launch the application.



7.3.2. Stop application

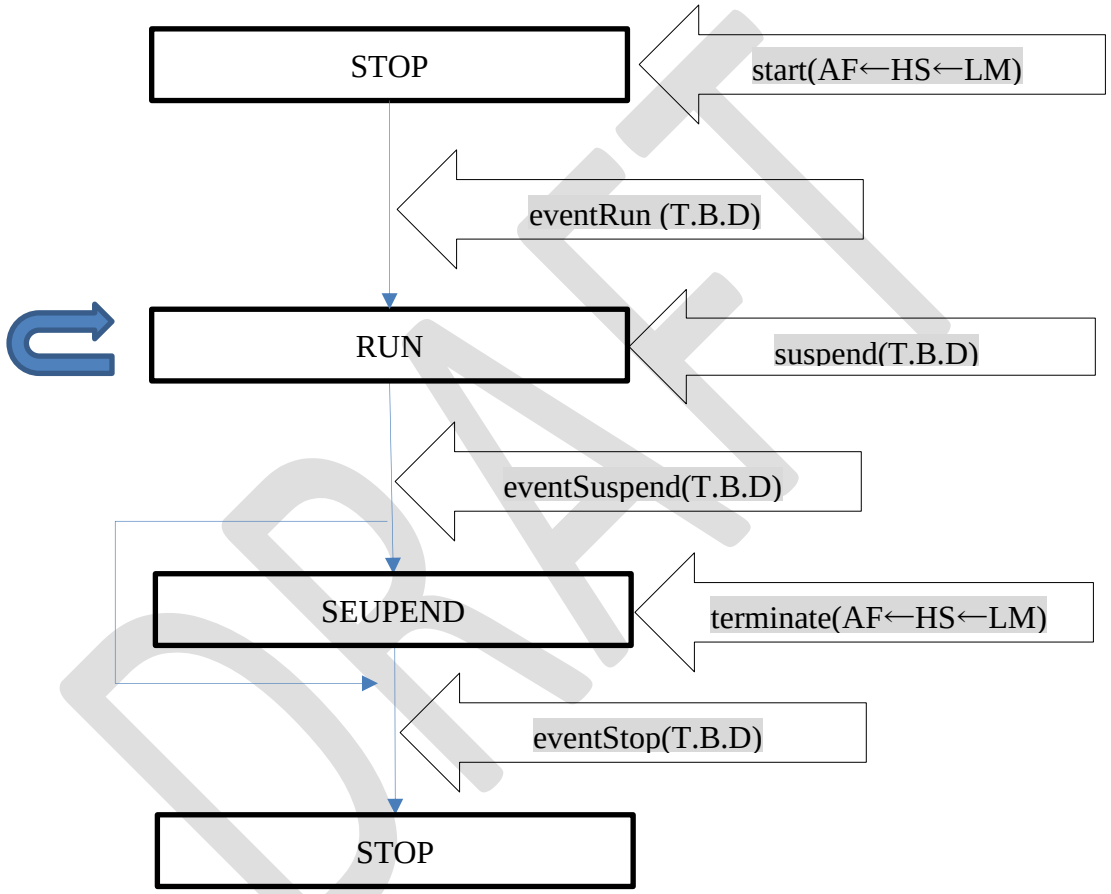


7.4. Application Lifecycle

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

AF: ApplicationFramework
HS: HomeScreen
LM: Lancher or MenuBar

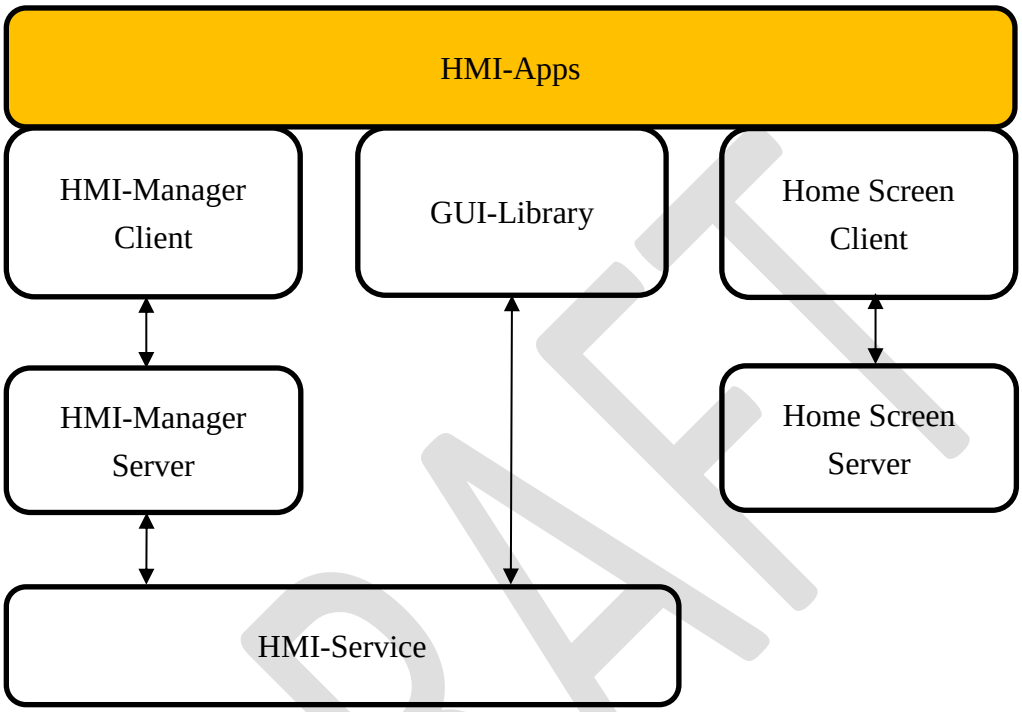
App Life Cycle



8. HMI-Apps （HMI-FW Related components）

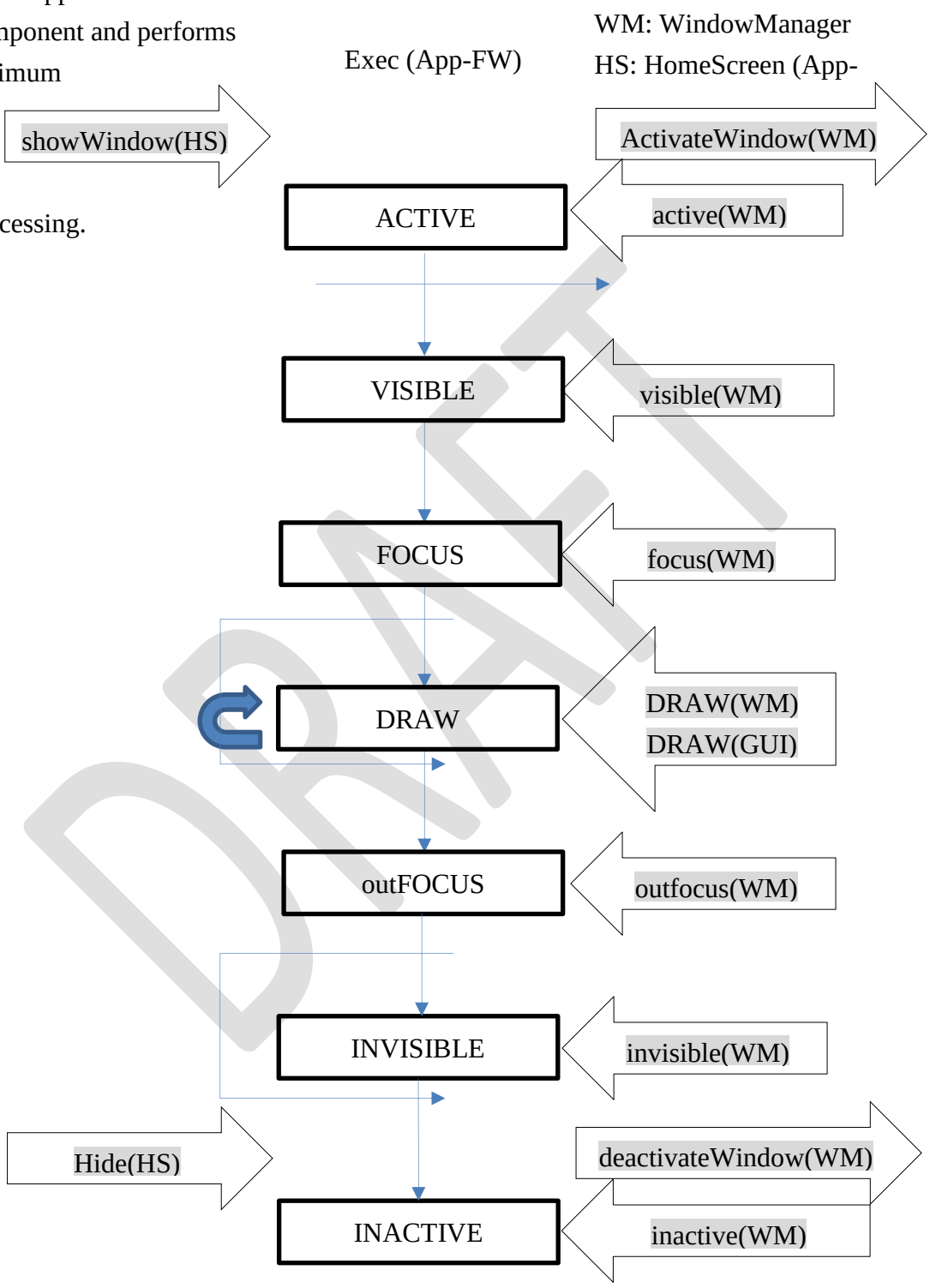
8.1. Overviw

8.1.1. Related external components



8.1.2. HMI-Apps Life Cycle

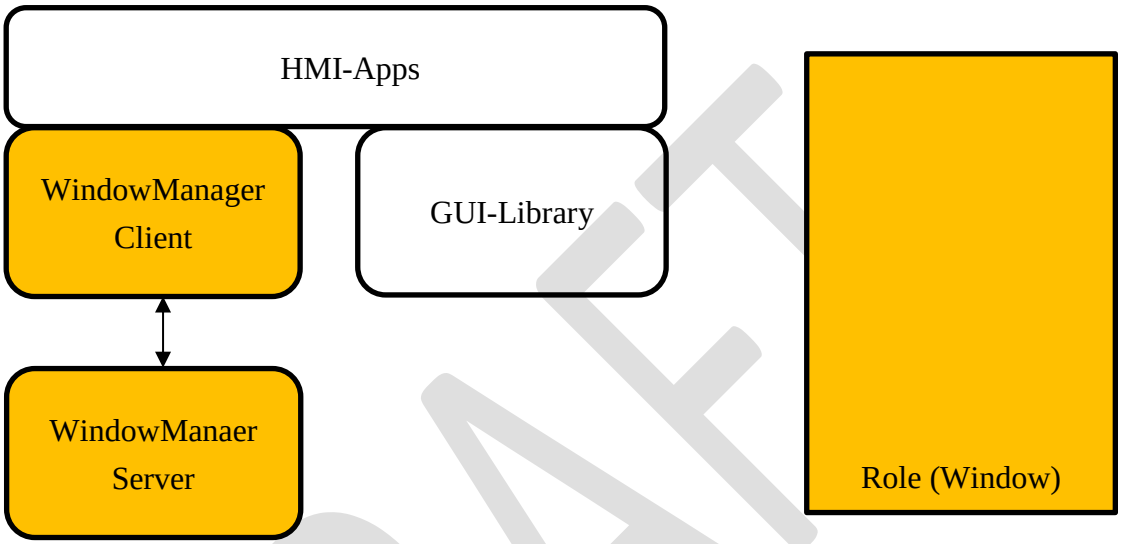
HMI-Apps receives events from each component and performs optimum processing.



8.2. HMI-Application Area Type

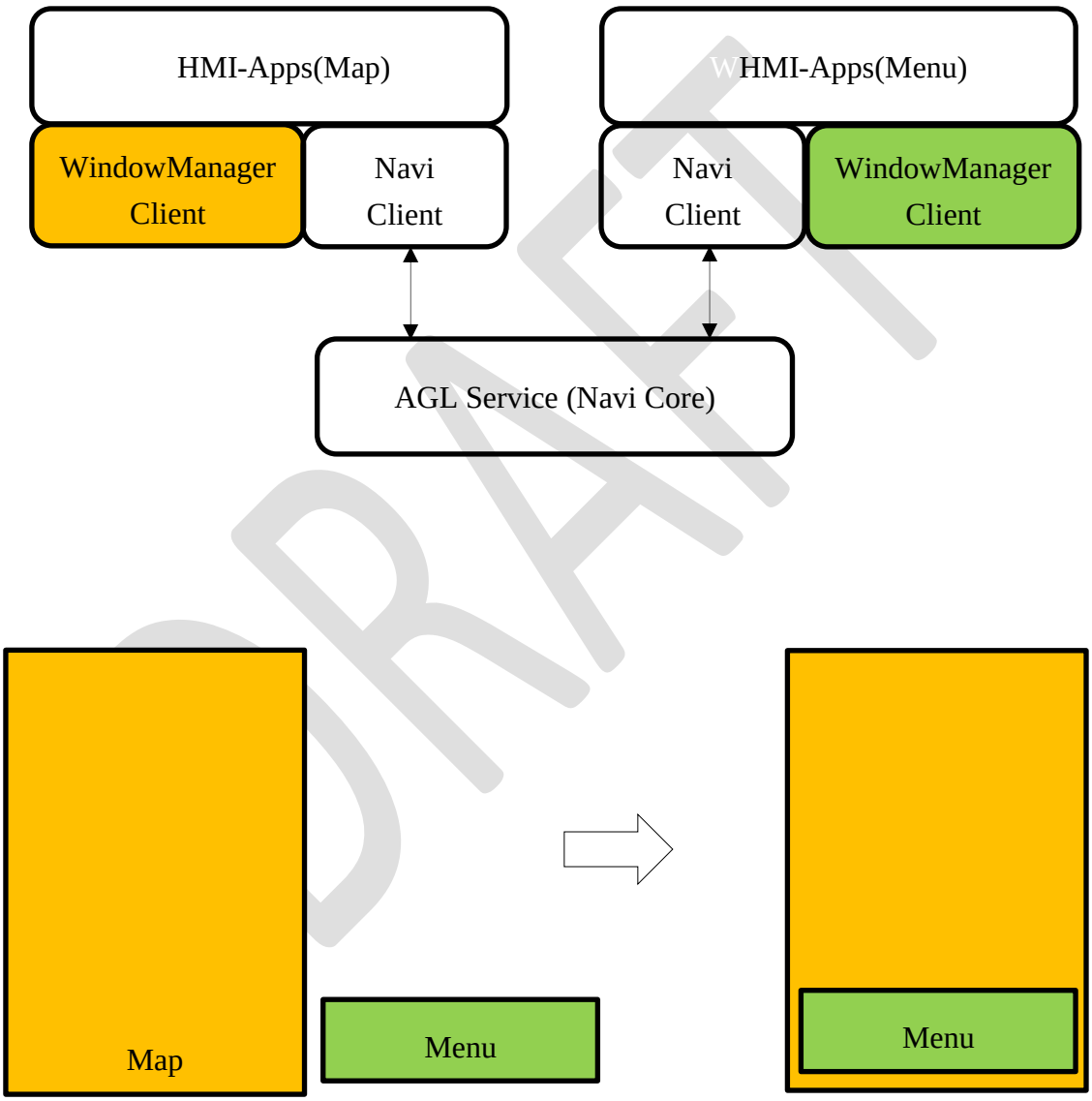
8.2.1. Single Role

A general application has one area and requests the WindowManager to draw area and then starts drawing using GUI-Library.



8.2.2. Multiple Role

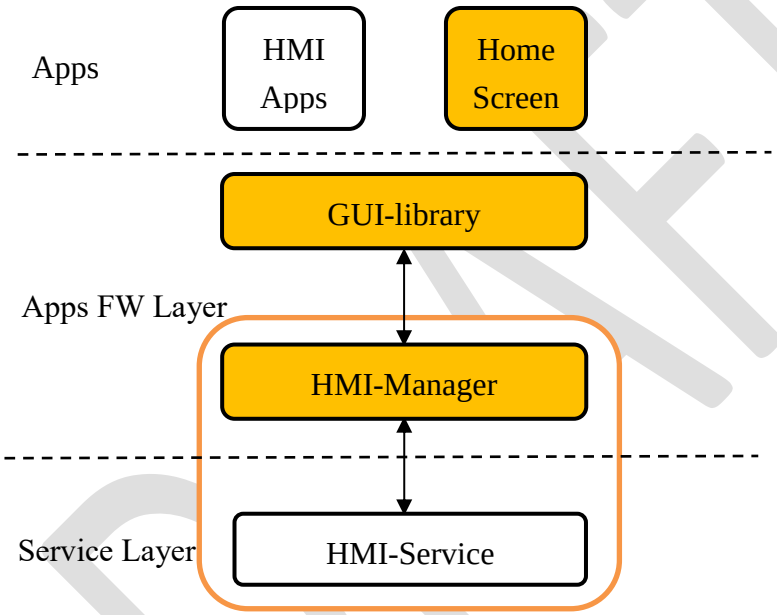
An application such as navigation may have a dedicated Layer and multiple roles on the Layer. and by superimposing each area, it becomes final information.
Generally, in order for each application to display in a coordinated manner, an AGL service is prepared.



9. Glossary

9.1. 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.



9.2. 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 — —	△	
	Input	Mouse Gesture	○	Mouse Gesture	○	