

**AGL HMI Framework**

Design Document

|  |  |
| --- | --- |
| Version | Date |
| 0.20 | 2017/XX/XX |

Index

[1. HMI Framework overview 4](¥l)

[1.1. HMI-FW Related components 4](¥l)

[1.1.1. Related components 5](¥l)

[1.1.2. HMI-FW Components 6](¥l)

[1.2. Considerations on implementation 7](¥l)

[2. HMI-Apps （HMI-FW Related components） 8](¥l)

[2.1. Overviw 8](¥l)

[2.1.1. Related external components 8](¥l)

[2.1.2. HMI-Apps Life Cycle 9](¥l)

[3. GUI-library 11](¥l)

[3.1. Overview 11](¥l)

[3.1.1. Related external components 11](¥l)

[3.1.2. Internal Components 12](¥l)

[3.2. Graphics functions 13](¥l)

[3.2.1. Procedure necessary for HMI-Apps 13](¥l)

[3.2.2. Software configuration of GUI-lib 14](¥l)

[3.3. Sound functions 16](¥l)

[3.4. Input functions 17](¥l)

[3.5. GUI-lib Standard Funcitions List (Reference material) 18](¥l)

[4. Window Manager 19](¥l)

[4.1. Overview 19](¥l)

[4.1.1. Related external components 19](¥l)

[4.1.2. Internal Components 20](¥l)

[4.1.3. Window Resources 21](¥l)

[4.2. Window Manager Client (API) 27](¥l)

[4.3. Window Resources Manager 28](¥l)

[4.3.1. Recover Window Resources (Boot Sequence) 28](¥l)

[4.3.2. Window Resource Control (Window Manager API) 29](¥l)

[4.3.3. Register My Application　(Window Manager API) 30](¥l)

[4.3.4. Allocate/Release Window Resources (Window Manager API) 31](¥l)

[4.3.5. Notify Window Resources Status (Window Manager API:EVENT) 35](¥l)

[4.4. Window Policy Manager 36](¥l)

[4.4.1. Window Layout Decision 37](¥l)

[4.4.2. Message Signaling Client 37](¥l)

[4.4.3. Window Policy DB Control(Window Manager API) 38](¥l)

[4.4.4. Window Policy DB use cases 38](¥l)

[4.5. Window Layout Manager 41](¥l)

[4.5.1. Change Window Layout 41](¥l)

[4.5.2. Window Layout DB Control(Window Manager API) 42](¥l)

[4.5.3. Window Layout Pattern Data (DB) sample 42](¥l)

[5. Home Screen 44](¥l)

[5.1. OverView 44](¥l)

[5.1.1. Related external components 44](¥l)

[5.1.2. Internal Components 45](¥l)

[5.2. Home Screen CORE API 46](¥l)

[5.3. HomeScreen initial processing 47](¥l)

[5.3.1. Initial setting of 「Window Manager」 47](¥l)

[5.3.2. Initial setting of 「Sound Manager」 48](¥l)

[5.3.3. Initial setting of 「Input Manager」 48](¥l)

[5.4. HomeScreen CORE functions 49](¥l)

[5.4.1. Shortcut key 49](¥l)

[5.4.2. Status Bar 49](¥l)

[5.4.3. Apps Area 50](¥l)

[5.4.4. Home Key 52](¥l)

[5.4.5. On Screen Area (Floating Area) 53](¥l)

[5.5. Apps shipped with HomeScreen 54](¥l)

[5.5.1. Apps launcher 54](¥l)

[5.5.2. Input Method Editor 55](¥l)

[6. Sound Manager（T.B.D） 56](¥l)

[7. Input Manager（T.B.D） 57](¥l)

# HMI Framework overview

## HMI-FW Related components

The related components are shown below.

（Orange box components included in HMI-FW）

Apps FW Layer

GUI-library

Service Layer

Apps

Home

Screen

HMI

Apps

HMI-Manager

HMI-Service

### 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

### HMI-FW Components

Components of the HMI-FW are shown below.

#### 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

#### 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

## 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.

Apps FW Layer

GUI-library

Service Layer

Apps

Home

Screen

HMI

Apps

HMI-Manager

HMI-Service

# HMI-Apps （HMI-FW Related components）

## Overviw

### Related external components

HMI-Apps

GUI-Library

HMI-Manager  
Client

HMI-Service

HMI-Manager

Server

Home Screen

Client

Home Screen

CORE

### HMI-Apps Life Cycle

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

CREATE

VISIBLE

DRAW

ACTIVE

InACTIVE

InVISIBLE

DESTROY

Apps Manager

Apps Manager

Window Manager

Window Manager

Window Manager

(syncDRAW)

GUI-library

Window Manager

Window Manager

Life Cycle

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

| No | Life  Cycle | Description | HMI-Apps Action  **To Window Manager** |
| --- | --- | --- | --- |
| １ | CREATE | Apps Launch | Register My Application  Allocate Window Resource |
| ２ | VISIBLE | Area VISIBLE |  |
| ３ | ACTIVE | Area ACTIVE |  |
| ４ | DRAW | Change Layout Start  ・RESIZE  ・RESTORE  Change Layout Stop | Draw END |
| REDRAW(GUI-Lib) | ➖ |
| ５ | inACTIVE | Area inACTIVE |  |
| ６ | inVISIBLE | Area inVISIBLE | Allocate Window Resource |
| ７ | DESTROY |  |  |

# GUI-library

## Overview

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

### 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.）

WHMI-Apps

GUI-Library

HMI-Manager  
Client

HMI-Service

HMI-Manager

Server

### Internal Components

WHMI-Apps

GUI-Library

PF(AGL) adaptaion Layer

GUI-Library

CORE

HMI-Service

## Graphics functions

Graphics provides rendering functions to the application.

### 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.

GUI-lib

Window

Manager

Weston

EGL/  
OpenGL

EGL

Apps

Apps

Apps

GUI-lib

GUI-lib

Graphics Data

Surface

Area

### 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

GUI-lib CORE

Standard API

AGL API

GUI-lib PAL

for

Window Manager

GUI-lib PAL

for

AGL PF

Window

Manager

Weston

OpenGL

/EGL

HMI Apps

## 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.

GUI-lib

Sound

Manager

Audio

manager

ALSA

Audio Device

Apps

Apps

Apps

GUI-lib

GUI-lib

Sound Stream

Stream

Channel

Zone

## 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

## 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 | ○ |  |

# Window Manager

## Overview

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

### Related external components

WHMI-Apps

GUI-Library

WindowManager  
Client

HMI-Service

WindowManaer

Server

### 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 |

WWindow

Policy

Manager

WindowPolicy

DB

Window

Layout

DB

Window

Layout

Manager

WWindow

Resource

Manager

WWindowManager

Client

Manager

Window

Resources

Message Signaling

Server

Message Signaling

Client

Manager

### 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.

Window

Manager

Window

Manager

Display

Display

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

Display

Layer

Layer

Area

Area

Surface

#### Layer

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

HomeScreen

(Back Gournd)

HomeScreen

(Fore Ground)

Apps Layer X

Apps Layer Y

OnScreen



A layer can maps multiple areas.

Apps Layer X

Area B

Area A

| 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.

Parent Surface

Child

Surface B

Area

Child Surface

| 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 | ー | Area Position |
| 8 | Size | ー | Area Width and Hight |
| 9 | Z order | ー | Area Z order |
| 10 | Visibility | ー | 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 | ー | Surface position |
| 4 | Size | ー | Width and Hight |
| 5 | Z order | ー | Surface Z order |
| 6 | Visibility | ー | Visibility Status |
| 7 | Alpha Blend | ー | 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] |

## Window Manager Client (API)

The API is shown below.

| No | Function | R/W | Description |
| --- | --- | --- | --- |
| １ | Window Resources Control | R/W | Get/Set 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 | Get/Set Policy DB |
| ７ | Window Layout DB Control | R/W | Get/Set Layout DB |

## Window Resources Manager

### Recover Window Resources (Boot Sequence)

The Window Manager always holds current window resources.

After reboot, Window Manger recoverd the Last Window resources.

Window Manager

Last Window  
Resources

Window

Resources

### Window Resource Control (Window Manager API)

HMI Apps can Get/Set Window Resources.

* HomeScreen Get/Set Display and Layer Info.
* HMI-Apps Set/Set Area Info.

HomeScreen

HMI-Apps

Window Manager

Weston

Window

Resources

#### Sample: Set Change Layer by Home Screen

Window Resources Control

(Set Change Layer)

Home  
Screen

Window  
Manager

Weston

Resource

Control

Notify (Resource Controle end)

)

Change Layer

### Register My Application　(Window Manager API)

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

Request Surface

Apps

Launch

Apps  
Manager

Apps

WaylandWeston

Register My Application

Window

Manager

OK

Apps

Window

Manager  
Manager

Weston

1. Register My Application （PID, Surface ID）

②　Get Surface Data

Window

Resources

### 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.

CShortCutIcon

Navi

Map

CHomeButton

Before

After

CShortCutIcon

MediaPlyaer

Song list

Apps

CHomeButton

Navi

Map

Apps

Allocate Window Resources

SyncDRAW

t)

Change Surface

Request Apps

Home  
Screen

Media

Player

Window  
Manager

Weston

endDRAW

Navi

Map

flushDRAW

Draw

Draw

Swap Buffer

Swap Buffer

Layout Decision Buffer

The internal sequence of WindowManage during 「AllocateWindowResources」 execution is shown below.

1. The app issues「AllocateWindowReasources」to Window Policy Manager.
2. Window Reasources Manager respons to App.The App draw own area.
3. Window Reasources Manager rsposns to another App.The App draw own Area.
4. Window Reasources Manager receive「endDRAW」from Apps
5. Window Reasources Manager issues 「ChangeLayout」 to Window Layout Manager

WWindow

Policy

Manager

Window

Layout

Manager

WWindow

Resource

Manager

⑤

1. ④

②　③

#### Use Case of inVISIBLE Window Resources

CShortCutIcon

Navi

Map

CHomeButton

Before

After

CShortCutIcon

MediaPlyaer

Song list

CHomeButton

AllocateWindowResources

SyncDRAW

t)

Change Surface

Request Apps

Home  
Screen

Media

Player

Window  
Manager

Weston

endDRAW

Navi

Map

flushDRAW

inVISIBLE

Swap Buffer

Layout Decision Buffer

DRAW

inVISIBLE

RAW

### 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 | When own Area becomes Visible |
| ２ | inVISIBLE | R | When own Area becomes InVisible |
| ３ | ACTIVE | R | When own Area becomes Active |
| ４ | inACTIVE | R | When own Area becomes inActive |
| ５ | syncDRAW | R | The app must draw own Area.After the end, response「endDRAW」. |
| ６ | flushDRAW | R | The app must flush own Area. |

## 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.

### 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.

Window

Policy

DB

DB

Window

Policy

Manager

Request

Optimal Window Layout

Get

Optimal Layout

Return

Window Layout

Set

Vehicle Info.

WWindow

Resource

Manager

Manager

### Message Signaling Client

Policy Manager acquires latest vehicle information from Message Sginaling.

### 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

### Window Policy DB use cases

#### Precondition

* Window Layer Pattern（have 2 Layers）

1. Base Layer (HomeScreen)
2. Apps Layer

CControl Bar

CHomeButton

Base

Layer

C

Apps Area

Apps

Layer



* Apps Layout Pattern（have 2 types）

1. FULL Full Apps Area
2. HALF MAIN: Upper Apps Area

SUB： Lower Apps Area

CControlBar

FULL

CHomeButton

1. FULL

CControlBar

SUB

Apps

CHomeButton

MAIN

Apps

1. HALF

* Displayable area of application（３Apps）

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 | — | — | — | — |

## Window Layout Manager

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

### Change Window Layout

If Window Layout Manager receive「Change Window Layout」

They need update Window Resources and send「Change Surfase」to Weston.

Window

Layout

DB

Window

Layout

Manager

Change Window Layout

Layout Pattern

Window

Resources

Change

Surface

WWindow

Resource

Manager

Manager

Weston

### 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

### 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

CControl Bar

CHomeButton

HomeScreen

Basic

C

Apps A

Apps B

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

}]

}

1. 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

}]

}

# Home Screen

## OverView

Home Screen is a component for performing user operation.

It is possible to have different Home Screen for each in-vehicle device.

### Related external components

GUI-Library

HMI-Manager

Client

Home Screen

Core

HMI-Service

HMI-Manager

WHMI-Apps

Home Screen

Client

Manager

Apps shipped with HomeScreen

### Internal Components

| No | Function | Description |
| --- | --- | --- |
| １ | Home Screen Client | API |
| ２ | Home Screen Core | Home Screen CORE functions |
| ３ | Apps Launcher | HMI-Apps |
| ４ | Input Method Editor | HMI-Apps |

## Home Screen CORE 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 | Home Key | W |  |
| ５ | On Screen | W |  |
| ６ | Notify | R |  |

## HomeScreen initial processing

The HomeScreen performs the following processing at startup.

For details, see the chapter of each manager.

### 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.

HomeScreen

(Back Gournd)

HomeScreen

(Fore Ground)

Apps Layer X

Apps Layer Y

OnScreen

An application may monopolize one layer or share it.

#### Setting Area

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

### Initial setting of 「Sound Manager」（T.B.D）

### Initial setting of 「Input Manager」（T.B.D）

## HomeScreen CORE functions

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



### Shortcut key

The user selects an application to use with apps menu.

### Status Bar

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

### Apps Area

Apps Area is the area for the application to display.

#### 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.

App C

(Invisible)

App A

App B

(Visible)

App B

App A

(Visible)

Apps Layer X

#### Stack Display

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

App A

App B

Called from A

App C

Called from B

App C

Apps Layer X

### 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.

HomeScreen

App A

App A

HomeScreen

### On Screen Area (Floating Area)

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

#### On Screen Sample

Apps A

App B

App A

Apps B

Caution!

Allocate Window Resources

syncDRAW

Change Surface

Request Onscreen

HMI-  
App A

Home

Screen

Window  
Manager

Weston

endDRAW

flushDRAW

HMI-  
App B

Swap Buffer

Layout Decision Buffer

Draw

## Apps shipped with HomeScreen

### 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.

Apps

Launch

Request Surface

Apps

Launch

Apps  
Manager

Apps

WaylandWeston

Register My Application

Window

Manager

OK

Apps

Launcher

#### Multi Page Menu

Menu 1

Menu 3

Menu 2

### Input Method Editor

The application can call IME with user operation.

App A

App B

B

App A

IME

IME

App B

Input

Manager

Window

Manager

App A

Request IME

Request OnScreen

and Draw Editor

Draw

IME

Execute

IME

Response

# Sound Manager（T.B.D）

# Input Manager（T.B.D）