AGL HMI Framework

Architecture Document

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9</td>
<td>2019/x/x</td>
</tr>
</tbody>
</table>
1. HMI Framework overview ................................................................. 4
   1.1. Overview ................................................................................. 4
       1.1.1. HMI-FW Components ......................................................... 5
       1.1.2. Related components ............................................................ 6
2. GUI-library .................................................................................. 7
   2.1. Overview ................................................................................. 7
       2.1.1. Related external components ............................................... 7
       2.1.2. Internal Components ............................................................ 8
   2.2. Graphics functions ................................................................. 9
       2.2.1. Procedure necessary for HMI-Apps ...................................... 9
       2.2.2. Software configuration of GUI-lib ....................................... 10
   2.3. Sound functions ..................................................................... 12
   2.4. Input functions ...................................................................... 13
3. Window Manager ........................................................................ 14
   3.1. Overview ................................................................................. 14
       3.1.1. Related external components ............................................... 14
       3.1.2. Internal Components ............................................................ 15
   3.2. Window Manager DataBase ................................................... 16
       3.2.1. Window Resources DB ......................................................... 16
       3.2.2. Window Policy DB ............................................................... 20
       3.2.3. Window Layout DB ............................................................... 21
   3.3. Window Manager Client ......................................................... 25
       3.3.1. API .................................................................................... 25
       3.3.2. EVENT ............................................................................ 25
   3.4. Window Resources Manager .................................................. 26
       3.4.1. Initializing Stage ................................................................ 26
       3.4.2. Activate Window ................................................................ 30
       3.4.3. Drawing Stage .................................................................. 34
       3.4.4. Deactivate Window ............................................................ 35
       3.4.5. Resource DB Control (Privilege Function) ......................... 36
   3.5. Window Policy Manager ........................................................ 39
       3.5.1. Policy Manager flow chart ................................................ 42
       3.5.2. Message Signaling Client .................................................. 43
       3.5.3. Policy DB Control (Privilege Function) .............................. 44
3.5.4. Use Case .......................................................... 45
3.6. Window Layout Manager ............................................. 48
  3.6.1. Initial setting of Layer .......................................... 48
  3.6.2. Layout Manager flow chart .................................... 50
  3.6.3. Layout DB Control (Privilege Function) ................. 51
  3.6.4. Use case ....................................................... 52
3.7. Multi ECU Extention ................................................. 53
  3.7.1. Overview ...................................................... 53
  3.7.2. Use case ....................................................... 54
4. Sound Manager .......................................................... 55
  4.1. Overview .......................................................... 55
    4.1.1. Related external components ................................. 55
    4.1.2. Internal Components ......................................... 56
    4.1.3. Sound Resources ............................................... 57
  4.2. Sound Manager Client .............................................. 59
    4.2.1. API .......................................................... 59
4.3. Sound Resources Manager .......................................... 60
  4.3.1. Initializing Stage ............................................... 60
  4.3.2. Sounding Stage ................................................ 62
  4.3.3. Sound Resource Control (API) ............................... 63
4.4. Sound Policy Manager ............................................... 65
  4.4.1. Sound Policy DB Control (Sound Manager API) ........ 65
  4.4.2. Message Signaling Client ..................................... 66
  4.4.3. Policy Manager flow chart .................................... 67
  4.4.4. Policy manager use cases ..................................... 69
4.5. Sound Layout Manager ................................................. 70
  4.5.1. Change Sound Layout .......................................... 70
5. Input Manager .......................................................... 71
  5.1. Overview .......................................................... 71
    5.1.1. Related external components ................................. 71
  5.2. Input Manager (Standard Device) ............................... 74
  5.3. Input Manager (OEM Specific Device) ......................... 75
    5.3.1. clinet ........................................................ 75
    5.3.2. Server ........................................................ 75
6. Home Screen ............................................................ 78
  6.1. OverView .......................................................... 78
6.1.1. Related external components................................................................. 78
6.1.2. Internal Components ........................................................................ 78
6.2. Home Screen Client ............................................................................. 80
  6.2.1. API.................................................................................................. 80
6.3. HomeScreen Server ............................................................................. 81
  6.3.1. Initial Setting .................................................................................. 81
6.4. HomeScreen Apps ................................................................................ 82
  6.4.1. Menu Bar (HomeScreen Layer) ..................................................... 82
  6.4.2. Display Restriction (Restriction Layer) ......................................... 85
  6.4.3. OnScreen (OnScreen Layer) ............................................................ 86
  6.4.4. Apps launcher (Apps Layer) ......................................................... 87
  6.4.5. Software Key Board (Apps Layer or HS Layer) ......................... 88
7. HMI-Application Manager ...................................................................... 89
  7.1. Overview ......................................................................................... 89
    7.1.1. HMI Application Manager position in AGL ......................... 89
    7.1.2. Related external components ............................................... 90
    7.1.3. Internal Components ............................................................... 91
  7.2. Application Manager Client ............................................................... 92
    7.2.1. API .......................................................................................... 92
    7.2.2. EVENT .................................................................................... 92
  7.3. Application Manager .......................................................................... 93
    7.3.1. Start application .......................................................................... 93
    7.3.2. Stop application .......................................................................... 93
  7.4. Application Lifecycle ......................................................................... 94
8. HMI-Apps (HMI-FW Related components) ............................................ 96
  8.1. Overview .......................................................................................... 96
    8.1.1. Related external components ............................................... 96
    8.1.2. HMI-Apps Life Cycle ............................................................... 97
  8.2. HMI-Application Area Type ............................................................... 99
    8.2.1. Single Role ................................................................................ 99
    8.2.2. Multiple Role ........................................................................... 100
9. Glossary ................................................................................................ 102
  9.1. Considerations on implementation .................................................. 102
  9.2. GUI-lib Standard Functions List (Reference material) .................. 103
1. HMI Framework overview

1.1. Overview

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 depend on each GUI-library, refer to each specification.)
2.1.2. Internal Components

![Diagram]

HMI-Apps

GUI-Library
CORE

GUI-Library
PF(AGL) adaptation Layer

HMI-Service
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.

![Diagram of sound functions]

- Apps
- ALSA Audio Device
- Sound Stream
- Sound Channel
- HTML5
- Qt
- EB
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-lib.
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

<table>
<thead>
<tr>
<th>No</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Window Manager Client API</td>
<td>API</td>
</tr>
<tr>
<td>2</td>
<td>Window Resource Manager</td>
<td>Window Resource Management</td>
</tr>
<tr>
<td>3</td>
<td>Window Policy Manager DB</td>
<td>Mediation of Window Resources</td>
</tr>
<tr>
<td>4</td>
<td>Window Layout Manager DB</td>
<td>Window Layout Management</td>
</tr>
</tbody>
</table>

```
AGL HMI Framework Architecture Document

Copyright (c) 2017, 2018, 2019 TOYOTA MOTOR CORPORATION
```
3.2. Window Manager DataBase

3.2.1. Window Resources DB

In WindowManager, 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.

<table>
<thead>
<tr>
<th>Role Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Navi</td>
<td>Navigation</td>
</tr>
<tr>
<td>Map</td>
<td>Map</td>
</tr>
<tr>
<td>MediaPlayer</td>
<td>Media Player</td>
</tr>
<tr>
<td>PhoneCall</td>
<td>Phone Call</td>
</tr>
</tbody>
</table>

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

WindowManager 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

<table>
<thead>
<tr>
<th>Role Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROLE</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>FOCUS</td>
<td>Currently focused Window</td>
</tr>
</tbody>
</table>
Area

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

Area = [Screen][Layout/Area Name]

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

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

<table>
<thead>
<tr>
<th>No</th>
<th>Name</th>
<th>Information Source</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ID</td>
<td>Graphics Subsystem</td>
<td>Screen ID</td>
</tr>
<tr>
<td>2</td>
<td>Name</td>
<td>—</td>
<td>Screen Name</td>
</tr>
<tr>
<td>3</td>
<td>Size</td>
<td>Graphics Subsystem</td>
<td>Screen Width and Height</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>No</th>
<th>Name</th>
<th>Information Source</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ROLE</td>
<td>Application</td>
<td>Layer Name</td>
</tr>
<tr>
<td>2</td>
<td>ID</td>
<td>Graphics Subsystem</td>
<td>Layer ID</td>
</tr>
<tr>
<td>3</td>
<td>Position</td>
<td>Graphics Subsystem</td>
<td>Layer Position</td>
</tr>
<tr>
<td>4</td>
<td>Size</td>
<td>Graphics Subsystem</td>
<td>Layer Width and Height</td>
</tr>
<tr>
<td>5</td>
<td>Z order</td>
<td>Graphics Subsystem</td>
<td>Layer Zorder</td>
</tr>
<tr>
<td>6</td>
<td>Visibility</td>
<td>Graphics Subsystem</td>
<td>Layer Visibility Status</td>
</tr>
<tr>
<td>7</td>
<td>α Blend</td>
<td>Graphics Subsystem</td>
<td>Layer Transparent Ratio</td>
</tr>
</tbody>
</table>
Window

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

<table>
<thead>
<tr>
<th>No</th>
<th>Name</th>
<th>Information Source</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Name</td>
<td>Application</td>
<td>Window Name</td>
</tr>
<tr>
<td>2</td>
<td>ID</td>
<td>Graphics Subsystem</td>
<td>Layer ID</td>
</tr>
<tr>
<td>3</td>
<td>Position</td>
<td>Graphics Subsystem</td>
<td>Layer Position</td>
</tr>
<tr>
<td>4</td>
<td>Size</td>
<td>Graphics Subsystem</td>
<td>Layer Width and Height</td>
</tr>
<tr>
<td>5</td>
<td>Z order</td>
<td>Graphics Subsystem</td>
<td>Layer Zorder</td>
</tr>
<tr>
<td>6</td>
<td>Visibility</td>
<td>Graphics Subsystem</td>
<td>Layer Visibility Status</td>
</tr>
<tr>
<td>7</td>
<td>α Blend</td>
<td>Graphics Subsystem</td>
<td>Layer Transparent Ratio</td>
</tr>
</tbody>
</table>

e.g.) "area": {"rect": { "x": x, "y": y, "width": width, "height": height } }
3.3. Window Manager Client

3.3.1. API

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

3.3.2. EVENT

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

<table>
<thead>
<tr>
<th>No</th>
<th>EVENT</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>activate</td>
<td>When own Area becomes Activated</td>
</tr>
<tr>
<td>2</td>
<td>deactive</td>
<td>When own Area becomes Deactivated</td>
</tr>
<tr>
<td>3</td>
<td>visible</td>
<td>When own Area becomes Visible</td>
</tr>
<tr>
<td>4</td>
<td>invisible</td>
<td>When own Area becomes InVisible</td>
</tr>
<tr>
<td>5</td>
<td>focus</td>
<td>When own Area becomes Focus</td>
</tr>
<tr>
<td>6</td>
<td>outfocus</td>
<td>When own Area becomes outFocus</td>
</tr>
<tr>
<td>7</td>
<td>restriction</td>
<td>When own Area becomes restriction</td>
</tr>
</tbody>
</table>
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 WindowManager using AppFW initialization function **init (port, token)**.

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

- **setEventHandler (“Event”, function)**
  Register function to receive event from WindowManager.
**setRole ("Role")**

An application sets its own ROLE with WindowManager.

![Diagram showing setRole("MediaPlayer")](image)

**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.

```
setRenderOrder(map.XXX”,…..”Navigation”)
```

```
<table>
<thead>
<tr>
<th>Navigation</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Navi”</td>
</tr>
<tr>
<td>“Map.XXX”</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Window Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>Navigation</td>
</tr>
<tr>
<td>“Map.XXX”</td>
</tr>
<tr>
<td>“Navi”</td>
</tr>
</tbody>
</table>
3.4.2. activateWindow

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
Use case ① Normal Window

CTop Bar

Navi Map

CBottom Bar

Media Player

CTop Bar

Media Player (Song list)

CBottom Bar

Home Screen

Media Player

Navi Map

Weston

ShowWindow

activateWindow

Layout Decision

syncDraw

draw

visible/focus

Navi Map Manager

Switch Buffer

(outofocus)/invisible

Copyright (c) 2017, 2018, 2019 TOYOTA MOTOR CORPORATION
Usecase ② Split Window
The use cases in which the MediaPlayer displays the song list during Navi map display are shown below. Display of MediaPlayer is done with shortcut key on HomeScreen Top Bar.

```
<table>
<thead>
<tr>
<th>Top Bar</th>
<th>Top Bar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Navi Map</td>
<td>Navi Map</td>
</tr>
<tr>
<td>Bottom Bar</td>
<td>Bottom Bar</td>
</tr>
<tr>
<td>MediaPlyaer</td>
<td>MediaPlyaer</td>
</tr>
<tr>
<td>(Song list)</td>
<td>(Song list)</td>
</tr>
</tbody>
</table>
```

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)

dehactive() //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 Resources
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")

![Diagram showing the addition of an area to a layer.](image)

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 Manager 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 Manager 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.

![Flow chart diagram]

Copyright (c) 2017, 2018, 2019 TOYOTA MOTOR CORPORATION
3.5.2. Message Signaling Client

Policy Manager acquires latest vehicle information from Message Signaling.

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

![Vehicle status diagram]

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.

<table>
<thead>
<tr>
<th></th>
<th>normal</th>
<th>main</th>
<th>sub</th>
<th>full</th>
<th>NAVI</th>
<th>NAVI:full</th>
<th>APPS-A</th>
<th>APPS-B</th>
</tr>
</thead>
<tbody>
<tr>
<td>n1</td>
<td>NAVI</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>To nf</td>
<td>To n2</td>
<td>To b</td>
</tr>
<tr>
<td>n2</td>
<td>-</td>
<td>NAVI</td>
<td>*</td>
<td>-</td>
<td>To n1</td>
<td>To nf</td>
<td>To n2</td>
<td>To b</td>
</tr>
<tr>
<td>nf</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>NAVI</td>
<td>To n1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>a1</td>
<td>APPS-A</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>To n1</td>
<td>To nf</td>
<td>To a2</td>
<td>To b</td>
</tr>
<tr>
<td>a2</td>
<td>-</td>
<td>APPS-A</td>
<td>*</td>
<td>-</td>
<td>To n1</td>
<td>To nf</td>
<td>To a2</td>
<td>To b</td>
</tr>
<tr>
<td>b</td>
<td>APPS-B</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>To n1</td>
<td>To nf</td>
<td>To a1</td>
<td>To b</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th></th>
<th>normal</th>
<th>main</th>
<th>sub</th>
<th>full</th>
<th>NAVI</th>
<th>NAVI:full</th>
<th>APPS-A</th>
<th>APPS-B</th>
</tr>
</thead>
<tbody>
<tr>
<td>n1</td>
<td>NAVI</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>To nf</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>nf</td>
<td>-</td>
<td>-</td>
<td>NAVI</td>
<td>-</td>
<td>To n1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
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 Surface」 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

<table>
<thead>
<tr>
<th>No</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sound Manager Client API</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Sound Resource Manager</td>
<td>Sound Resource Management</td>
</tr>
<tr>
<td>3</td>
<td>Sound Policy Manager</td>
<td>Mediation of Sound Resources</td>
</tr>
<tr>
<td>4</td>
<td>Sound Layout Manager</td>
<td>Sound Layout Management</td>
</tr>
</tbody>
</table>

![Diagram of internal components]
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

<table>
<thead>
<tr>
<th>No</th>
<th>Function</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Init</td>
<td>W</td>
<td>Connect to Sound Manager</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Allocate Sound Resources</td>
<td>W</td>
<td>Request Allocate Resources</td>
</tr>
<tr>
<td>4</td>
<td>Release Sound Resources</td>
<td>W</td>
<td>Request Release Resources</td>
</tr>
<tr>
<td>5</td>
<td>Sound Resources Control</td>
<td>R/W</td>
<td>Get/Set Sound Resources</td>
</tr>
<tr>
<td>6</td>
<td>Sound Policy DB Control</td>
<td>R/W</td>
<td>Get/Set Policy DB</td>
</tr>
</tbody>
</table>
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 Manager 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 Signaling.

![Diagram showing the Message Signaling Client and its interactions with Sound Resource, Sound Policy, and Message Signaling Server.]
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

<table>
<thead>
<tr>
<th>MP</th>
<th>Radio</th>
<th>Navi(Start)</th>
<th>Navi(Stop)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP</td>
<td>-</td>
<td>To RADIO</td>
<td>PUSH Status(MP)</td>
</tr>
<tr>
<td>RADIO</td>
<td>To MP</td>
<td>-</td>
<td>To NAVI</td>
</tr>
<tr>
<td>NAVI</td>
<td>-</td>
<td>-</td>
<td>PUSH Status(MP)</td>
</tr>
</tbody>
</table>

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 Manager 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

![Diagram showing the relationship between components]

- HMI-Apps
- GUI-Library
- Input Manager Client
- Input Manager Server (standard)
- Input Manager Server (OEM specific)
- Input Manager Plugin (IVI-input)
- HMI-Service (Weston)
- Input Device Driver
Related external components
Input Manager doesn’t handle 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.

<table>
<thead>
<tr>
<th>No</th>
<th>Function</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Init</td>
<td>W</td>
<td>Connect to InputManager</td>
</tr>
<tr>
<td>2</td>
<td>activateInput</td>
<td>W</td>
<td>OEM Specific Device</td>
</tr>
<tr>
<td>3</td>
<td>deactivateInput</td>
<td>W</td>
<td>OEM Specific Device</td>
</tr>
</tbody>
</table>

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.

![Sequence Chart Diagram]
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

<table>
<thead>
<tr>
<th>No</th>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Client</td>
<td>HomeScreen Library</td>
</tr>
<tr>
<td>2</td>
<td>Server</td>
<td>HomeScreen Service</td>
</tr>
<tr>
<td>3</td>
<td>MenuBar</td>
<td>MenuBar (HomeScreen Apps)</td>
</tr>
<tr>
<td>5</td>
<td>Restriction</td>
<td>Display Restriction (HomeScreen Apps)</td>
</tr>
<tr>
<td>6</td>
<td>OnScreen</td>
<td>OnScreen (HomeScreen Apps)</td>
</tr>
<tr>
<td>7</td>
<td>Launcher</td>
<td>Apps Launcher (HomeScreen Apps)</td>
</tr>
<tr>
<td>8</td>
<td>SKB</td>
<td>Software Key Board (HomeScreen Apps)</td>
</tr>
</tbody>
</table>
## 6.2. Home Screen Client

### 6.2.1. API

<table>
<thead>
<tr>
<th>No</th>
<th>Function</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Home Screen Resources Control</td>
<td>R/W</td>
<td>Get/Set Resources</td>
</tr>
<tr>
<td>2</td>
<td>Short Cut key</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Status Bar</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Information Bar</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Home Key</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>OnScreen</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Apps Lancher</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Software Keyboard</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Notify</td>
<td>R</td>
<td></td>
</tr>
</tbody>
</table>
6.3. **HomeScreen Server**

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

![Diagram of HomeScreen Server]

6.3.1. **Initial Setting**

HomeScreen acquires the surface ID 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**

![Diagram](image-url)

- ① `showWindow`
- ② `activateWindow`
- ③ `invisible`
- ④ `visible`
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.

HMI-Client

AGL-Binder

HMI-Server

AGL-Service

Copyright (c) 2017, 2018, 2019 TOYOTA MOTOR CORPORATION
7.1.2. Related external components

HMI-manager group

<table>
<thead>
<tr>
<th>No</th>
<th>Name</th>
<th>Service Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>WM</td>
<td>Window Manager</td>
</tr>
<tr>
<td>2</td>
<td>SM</td>
<td>Sound Manager</td>
</tr>
<tr>
<td>3</td>
<td>HS</td>
<td>Home Screen</td>
</tr>
</tbody>
</table>

Software structure
7.1.3. Internal Components

<table>
<thead>
<tr>
<th>No</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Application Manager Client</td>
<td>API</td>
</tr>
<tr>
<td>2</td>
<td>Application Manager</td>
<td>Application Manager Service</td>
</tr>
</tbody>
</table>
7.2. Application Manager Client

7.2.1. API

<table>
<thead>
<tr>
<th>No</th>
<th>Function</th>
<th>W/R</th>
<th>Who is ordered?</th>
<th>Server@v0.8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Start Application</td>
<td>W</td>
<td>PrivilegeApps</td>
<td>HS</td>
</tr>
<tr>
<td>2</td>
<td>Stop Application</td>
<td>W</td>
<td>PrivilegeApps</td>
<td>HS</td>
</tr>
<tr>
<td>3</td>
<td>Get Application Information</td>
<td>R</td>
<td>The App itself / PrivilegeApps</td>
<td>HS</td>
</tr>
<tr>
<td>4</td>
<td>Get List STOP Apps</td>
<td>R</td>
<td>PrivilegeApps</td>
<td>HS</td>
</tr>
<tr>
<td>5</td>
<td>Get List RUN Apps</td>
<td>R</td>
<td>PrivilegeApps</td>
<td>HS</td>
</tr>
<tr>
<td>6</td>
<td>Get List SUSPEND Apps</td>
<td>R</td>
<td>PrivilegeApps</td>
<td>T.B.D.</td>
</tr>
</tbody>
</table>

Privilege Apps: Launcher, MenuBar etc.

7.2.2. EVENT

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

<table>
<thead>
<tr>
<th>No</th>
<th>EVENT</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RUN</td>
<td>When own App becomes RUN</td>
</tr>
<tr>
<td>2</td>
<td>SUSPEND</td>
<td>When own App becomes SUSPEND</td>
</tr>
<tr>
<td>3</td>
<td>STOP</td>
<td>When own App becomes STOP</td>
</tr>
</tbody>
</table>
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

1. STOP
   - start(AF←HS←LM)
   - eventRun (T.B.D)
   - eventSuspend(T.B.D)
   - terminate(AF←HS←LM)
   - eventStop(T.B.D)

2. RUN
   - suspend(T.B.D)
   - eventSuspend(T.B.D)

3. SEUPEND
   - eventStop(T.B.D)

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

8.1. Overview

8.1.1. Related external components

![Diagram of HMI-Apps]

- HMI-Apps
- HMI-Manager Client
- HMI-Manager Server
- GUI-Library
- HMI-Service
- Home Screen Client
- Home Screen Server
8.1.2. HMI-Apps Life Cycle

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

- `showWindow(HS)`
- `Exec (App-FW)`
- `WM: WindowManager`
- `HS: HomeScreen (App-FW) Cycle`
- `activateWindow(WM)`
- `WM: WindowManager`
- `HS: HomeScreen (App-FW) Cycle`
- `active(WM)`
- `visible(WM)`
- `focus(WM)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)`
- `DRAW(WM)`
- `DRAW(GUI)
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. 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 Functions List (Reference material)

<table>
<thead>
<tr>
<th>Functions</th>
<th>Qt</th>
<th>JavaFX</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D Window</td>
<td>Qt GUI</td>
<td>○ Stage</td>
<td>○ Pop-up Window</td>
</tr>
<tr>
<td>3D Canvas</td>
<td>Painter2D</td>
<td>○ Canvas2D</td>
<td>○ WebView</td>
</tr>
<tr>
<td>3D SceneGraph</td>
<td>Material</td>
<td>○ Camera/Light</td>
<td>○ WebView</td>
</tr>
<tr>
<td>3D SceneGraph</td>
<td>Transform</td>
<td>○ Transform</td>
<td>○ Visual Effect</td>
</tr>
<tr>
<td>3D SceneGraph</td>
<td>Animation</td>
<td>○ Pick</td>
<td>○ Sub-Scene</td>
</tr>
<tr>
<td>3D SceneGraph</td>
<td>Clip-Node</td>
<td>○ Sub-Scene</td>
<td>○ Sub-Scene</td>
</tr>
<tr>
<td>3D Graphics</td>
<td>OpenGL/ES</td>
<td>○ 2D Share</td>
<td>○ 3D Share</td>
</tr>
<tr>
<td>3D Graphics</td>
<td>Canvas 3D</td>
<td>△</td>
<td></td>
</tr>
<tr>
<td>3D Graphics</td>
<td>(WebGL)</td>
<td>△</td>
<td></td>
</tr>
<tr>
<td>ML ETC</td>
<td>QML</td>
<td>○ FXML</td>
<td>○</td>
</tr>
<tr>
<td>MultiMedia</td>
<td>Qt package</td>
<td>△</td>
<td>Java OSGI</td>
</tr>
<tr>
<td>MultiMedia</td>
<td>Audio</td>
<td>○ Audio</td>
<td>○</td>
</tr>
<tr>
<td>MultiMedia</td>
<td>Video</td>
<td>○ Video</td>
<td></td>
</tr>
<tr>
<td>MultiMedia</td>
<td>Camera</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>MultiMedia</td>
<td>Radio</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>Mouse</td>
<td>○ Mouse</td>
<td>○</td>
</tr>
<tr>
<td>Input</td>
<td>Gesture</td>
<td>△</td>
<td></td>
</tr>
</tbody>
</table>