Case study: Apply Weston IVI shell to product

Nobuhiko Tanibata

1st June 2015

Project Manger of Advanced Driver Information Technology

ADIT is joint venture company of DENSO Corporation and Bosch GmbH

Contents

Advanced Driver Information Technology

- **Future of Graphic stacks in automotive segment**
- **Trend of Graphic stacks in automotive segment**
- **Comparison of UI requirements**
- **What is IVI shell + Wayland IVI Extension?**
- **Case study**
  - Synchronization of creating surface and visibility
  - Speed restriction
  - GPU usage reduction
  - Fastboot
  - Physical plane
  - Application framework co-existence
  - Multi screens and inputs devices
  - Multi Nodes

Future of Graphics in automotive segment

- **Center Display, Meter Cluster, Head-up Display,,,,**
- **Touch panel, Steering switch, Special devices,,,,**
  - Many screens, Many devices to be managed by Central HMI controller for integrated HMI of Face of **‘Cockpit system’**

Not so much synchronized each other

Integrated HMI: Face of Cockpit system

Trend of Graphic stacks in automotive segment

- **Proprietary to Common**
- **Complexity to Light weight**
- **Wayland/Weston is one of candidates**
  - Distill out functions from X server

- **Trends**
  - GENIVI compliant 8.0 supports wayland-ivi-extension protocol
    - GENIVI Demo platform provides good reference including Wayland/Weston
    - http://wiki.projects.genivi.org/index.php/GENIVI_Demo_Platform
  - Many companies shift proprietary stacks to using Wayland/Weston

Comparison of UI style; Desktop PC, Tablet/smartphone, and IVI system

IVI system UI is not completely designed by using Desktop PC or Tablet/Smartphone UI system without customization.

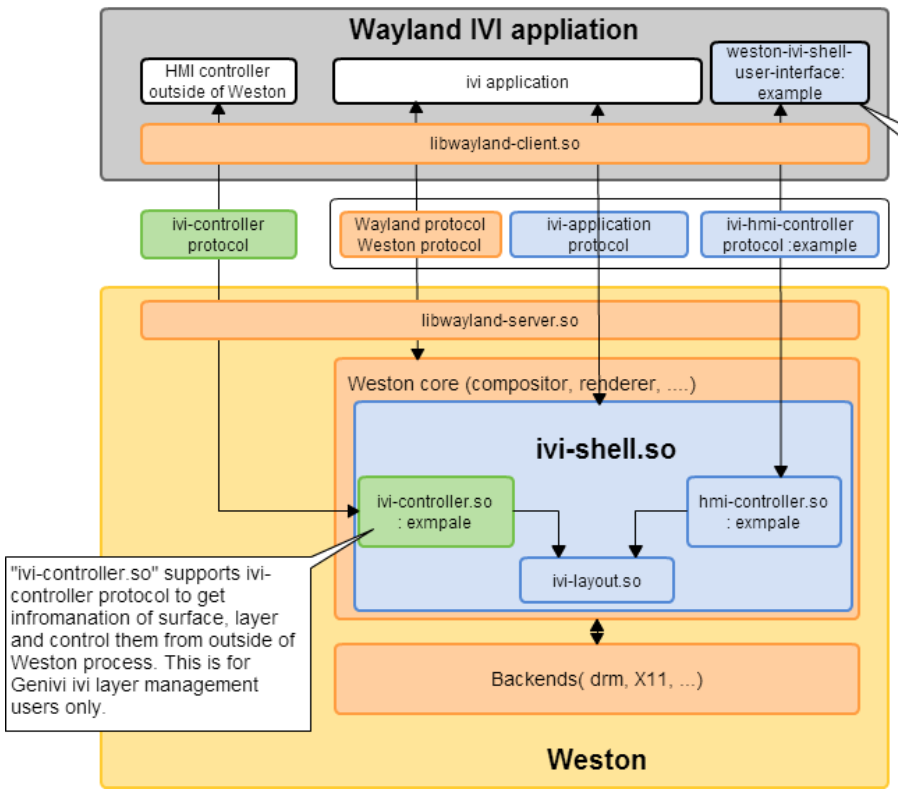
	Desktop PC	Tablet/ Smartphone	IVI System
Window size/position	Free	Fixed per mode	Fixed per mode
Window management	User can request	Central control	Central control
Layer concept	not required so much	not required so much	Strongly required
Physical plain control	not required so used for app	not required so used for app	Strongly required
Fastboot	not required so much	not required so much	Required
Multi screens	**Clone or Extended**	**N/A**	**Separately managed**
Multi input devices	**Pointer/ Keyboard**	**Pointer/ Keyboard**	**Could be many**

What is IVI shell + Wayland IVI Extension?

- **Add missing requirement on Desktop/Tablet window management**
  - Window central management, Layer concept
  - Physical plane control, and Fastboot
  - **Multi screens, Multi input devices** <- Important features for Future Cockpit system
- **IVI-shell**
  - ivi-application protocol: mapping surface ID to wl_surface
    - HMI “central” controller can identify specific surface and set properties.
  - Allow us to add modules to realize “Physical plane controller” and “Fastboot”
  - Interfaces for a “controller module” loaded on IVI-shell
    - Control properties of “surfaces/layers/**screens**”
    - **Handle multi input devices**
- **Wayland IVI Extension**
  - A “controller module”: ivi-controller.so
    - ivi-controller.so allows HMI controller outside of Weston to control.
    - Allow us to use legacy central controller

Overview of IVI shell and Wayland-ivi-extension

Advanced Driver Information Technology

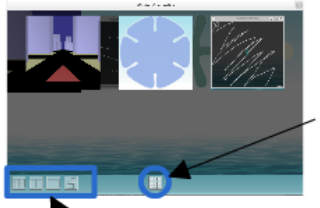


Already available in Wayland community

Contribution to Weston as ivi-shell part.

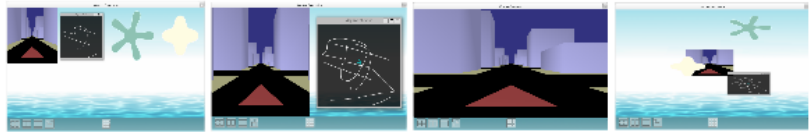
Available in Genivi Project: Wayland-ivi-extension

"hmi-controller.so" is is a reference implementation of setting up User Interface by using ivi-layout.so which supports management of properties of surfaces and layer. The User interface consists of background and panel; three bottom to trigger layout change and home screen to launch IVI applications.



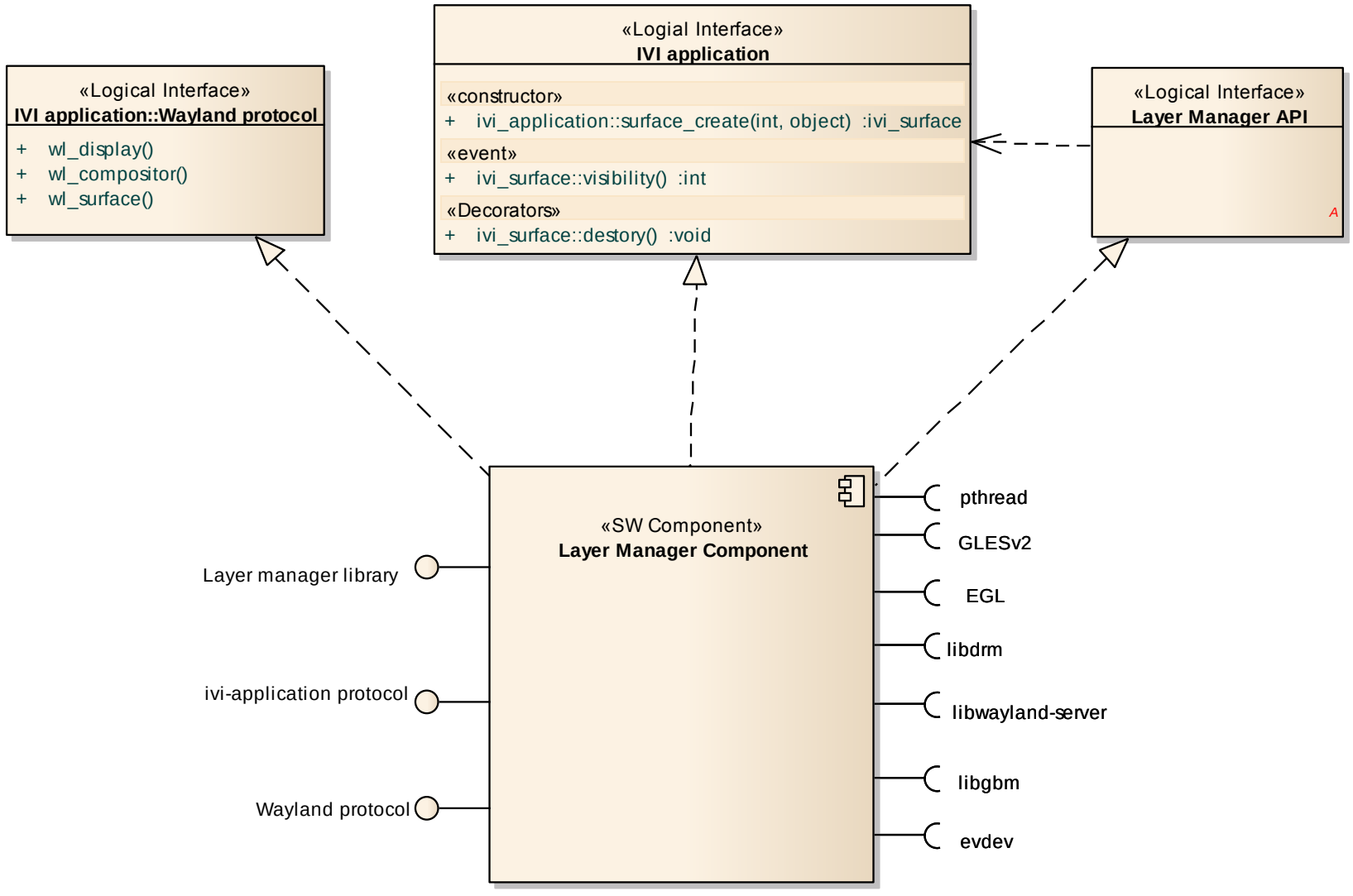
Home button in the center of bottom shows icons to launch IVI applications. The home screen has several page which can be selected by motion of input.

Four buttons trigger layout change as reference how to use libweston-layout.so. It support 4 reference layouts; tiling, side by side, fullscreen, and random.



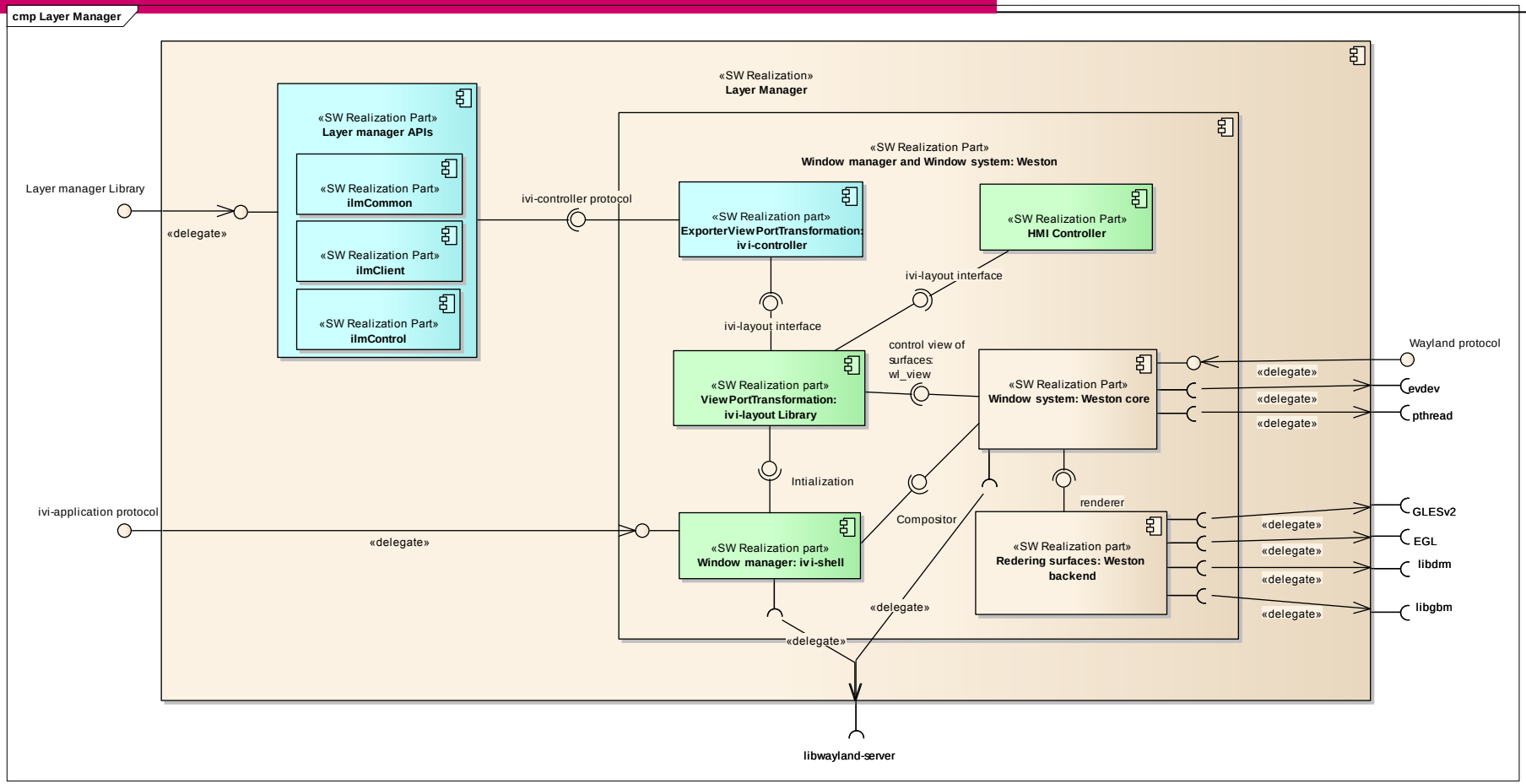
Interfaces

cmp Layer Manager



Component diagram

Advanced Driver Information Technology



Case study

Feedbacks/Questions from Users

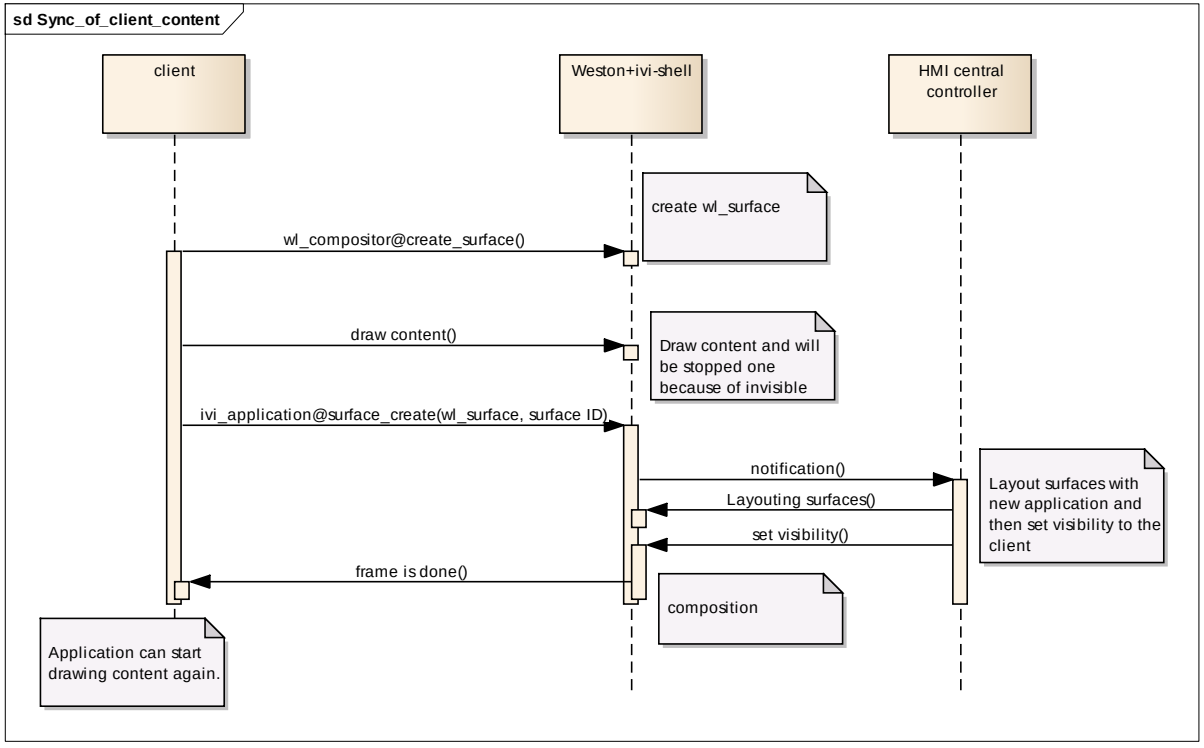
- Synchronization of creating surface and visibility
- Speed restriction
- GPU usage reduction
- Fastboot
- Physical plane
- Application framework co-existence
- **Multi screens and inputs devices**
- **Multi Nodes**

Case study: Synchronization of creating surface and setting visibility

Case: Central HMI controller shows client surface after completely drawing of client content.

Possible solution:

- **Avoid additional communication sequence between them.**
  - Use only Wayland protocol to keep general versatility



Case study: GPU usage reduction

Case: Reduce AC supply consumption

Possible solution:

- **Avoid calling eglSwapbuffers more than vsync e.g. 60fps**
  - EGL supports eglSwapInterval to allow application to call eglSwapbuffers more than Vsync. However, the option shall be avoided.
- **Separately implementing in multi-thread**
  - E.g. calculation of current position of Map and load map data
  - E.g. draw map with Wayland/Weston

Case study: Speed restriction

Case: Invisible a surface e.g. television screen. One of very important feature in automotive

Possible solution:

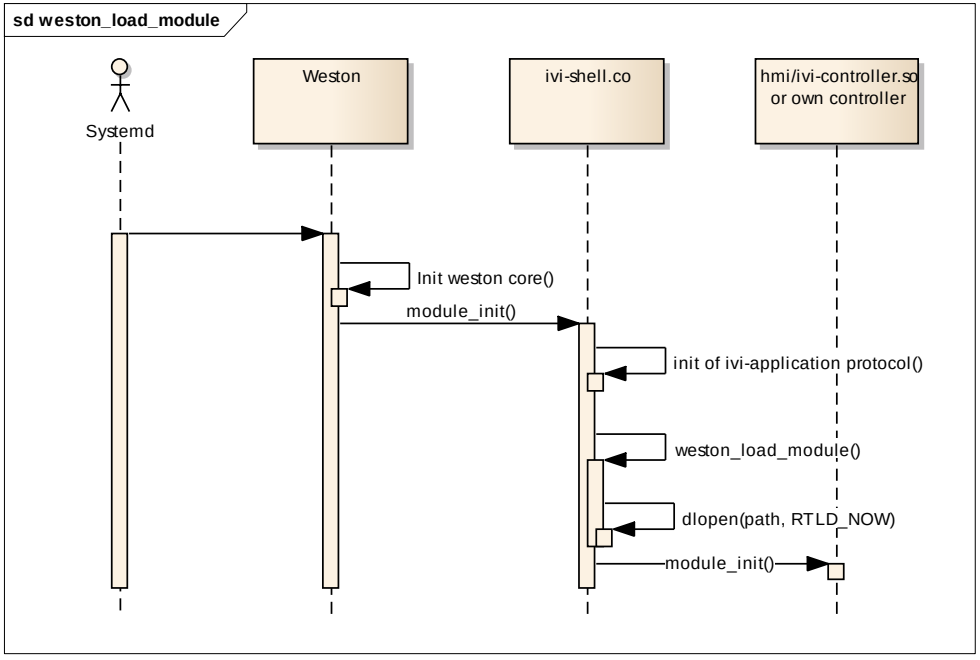
- **Avoid outside of Weston**
  - To reduce case which can not set invisible to a surface. This feature shall be implemented near H/W; H/W > Weston > Application
- **Recommendation**
  - Add logic in your own controller to directly observe speed and set invisible to a surface. Simple solution is better for critical feature!

Case study: Fastboot

Case: Rearview camera, Early video within 2s

Possible solution:

- **Avoid huge number of symbols in your own controller**
- **Avoid huge processing in your module_init before critical functionality, RVC or Early Video.**

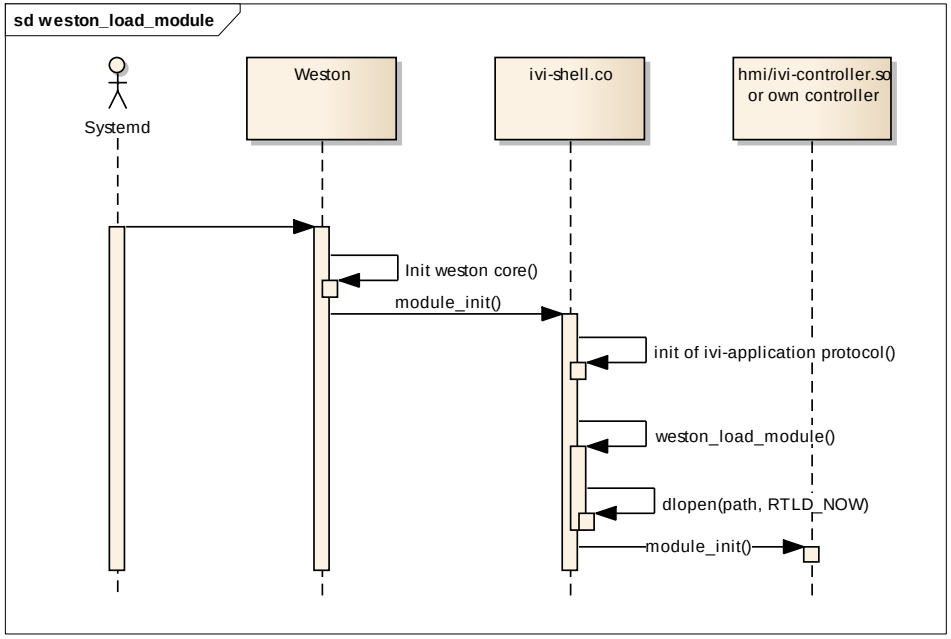


Case study: Physical plane

Case: RVC in Physical plane

Possible solution:

- **Avoid outside of Weston Surface Management**
  - Remove complex handover of surface to Weston
- **Physical plane controller is implemented at the begging of own module_init.**



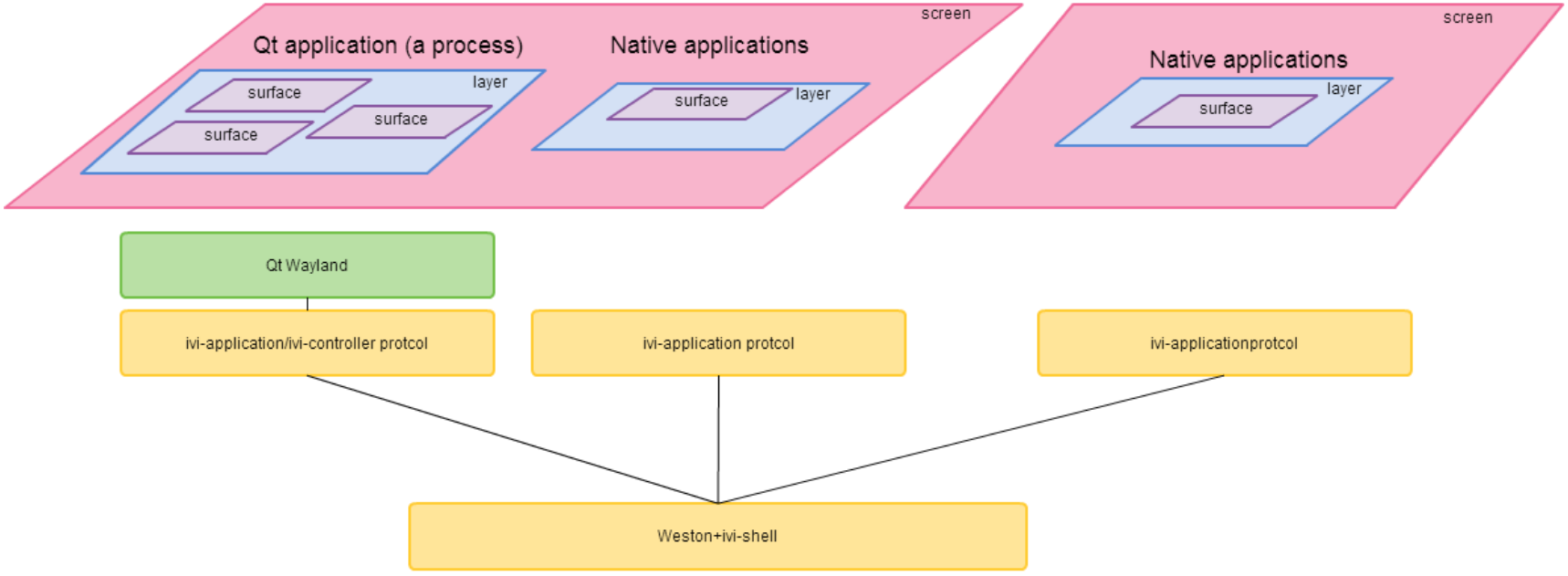
Case study: Application framework co-existence

Case:

- **Management layout of e.g. Qt application + native applications.**
  - Critical native application needs to be managed outside of Qt, means simple layer.

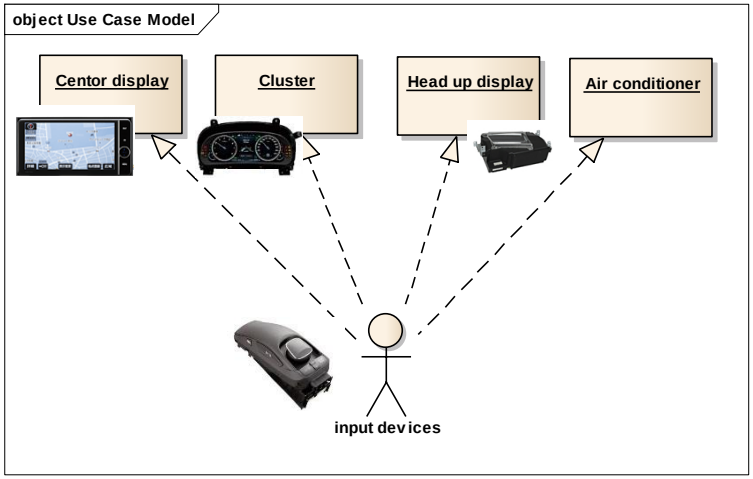
Possible solution:

- **Add layer and surface management in Qt Wayland. (under planning to contribute)**



Case study: Multi screens and input devices

Case: allow user to control a screen with one input device. Reduce input device to control multi screens.



Possible solution:

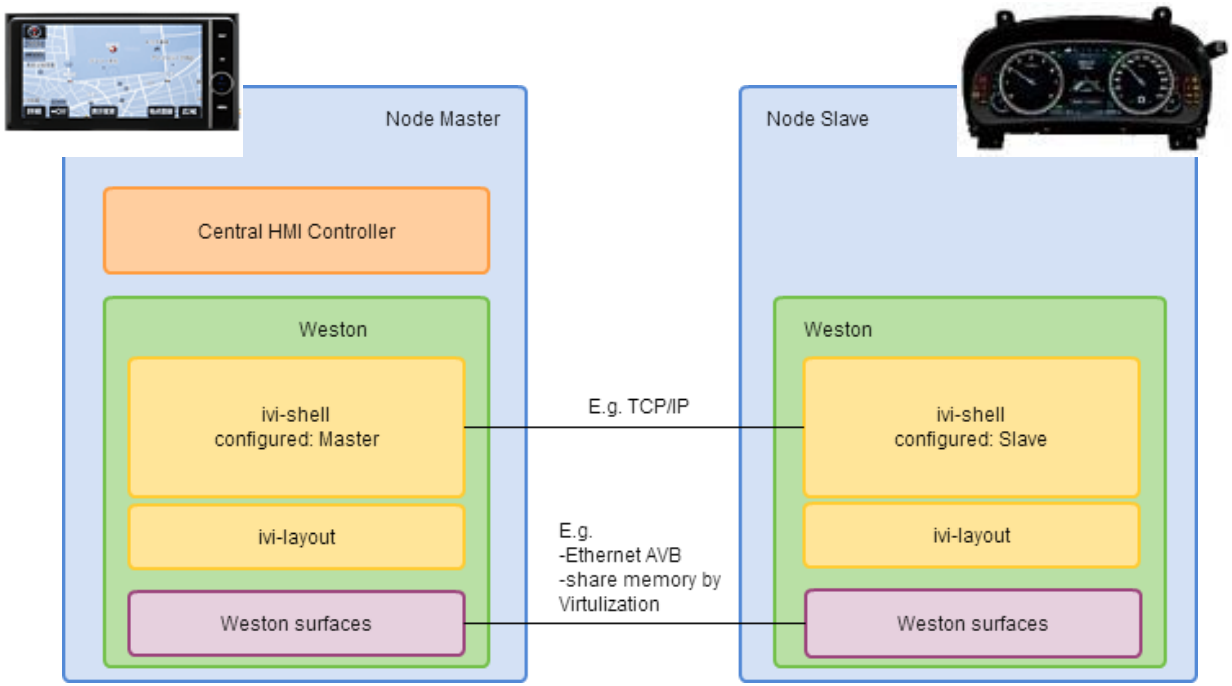
- **IVI-shell and Wayland-ivi-extension supports Input focus APIs**
- **Central HMI controller can switch focus of input device per surface.**
  - http://git.projects.genivi.org/?p=wayland-ivi-extension.git;a=blob;f=protocol/ivi-input.xml

Case study: Multi Nodes

Case: Control Multi Nodes Layout by one Central HMI controller

Possible solution:

- **Master and Slave IVI-shell to communicate with e.g. TCPIP**
- **(option) sharing surfaces by Ethernet AVB or shared memory**



Summary

- **Wayland-ivi-extension + IVI-shell covers automotive UI use cases.**
  - The set of interfaces allows us to layout surface and keep compatibility of applications and Central HMI controller across different systems.

- **Next steps: Future of Graphic for Integrated Cockpit system**
  - Extend IVI-shell to cover requirement of UI for integrated Cockpit system, which integrate several ECUs e.g. Center Display, Cluster, and so on.
    - Target date for POC: the end of 2015
    - Evaluation points
      - Multi input devices handling
      - Performance to be controlled by one central controller
      - How to share surfaces across Nodes (multi OS)

References

- **Wayland**
  - http://wayland.freedesktop.org
  - http://cgit.freedesktop.org/wayland

- **GENIVI Wayland-ivi-extension**
  - http://projects.genivi.org/wayland-ivi-extension