

# App Framework Update

**Automotive Grade Linux / Collabora**

Daniel Stone <[daniels@collabora.com](mailto:daniels@collabora.com)>

Simon McVittie

# Agenda and overview

- Introduce Simon McVittie (Collabora colleague)
- Recap of March VF2F app FW discussion
- Overview of existing AGL app/service management (all EGs)
- Development proposal for app FW evolution
- Overview of existing OSS solutions
- Discussion of usecases and targets for development

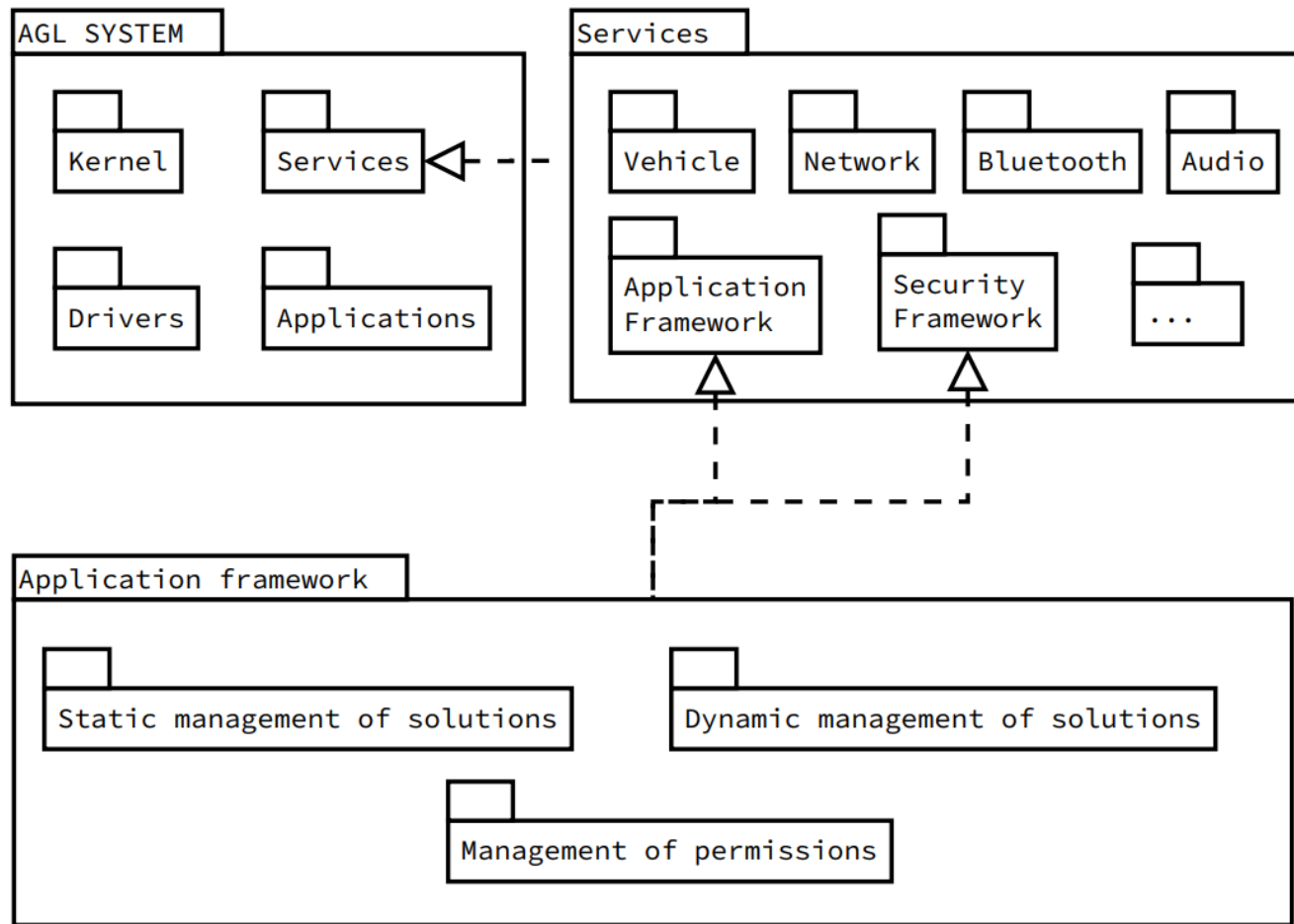




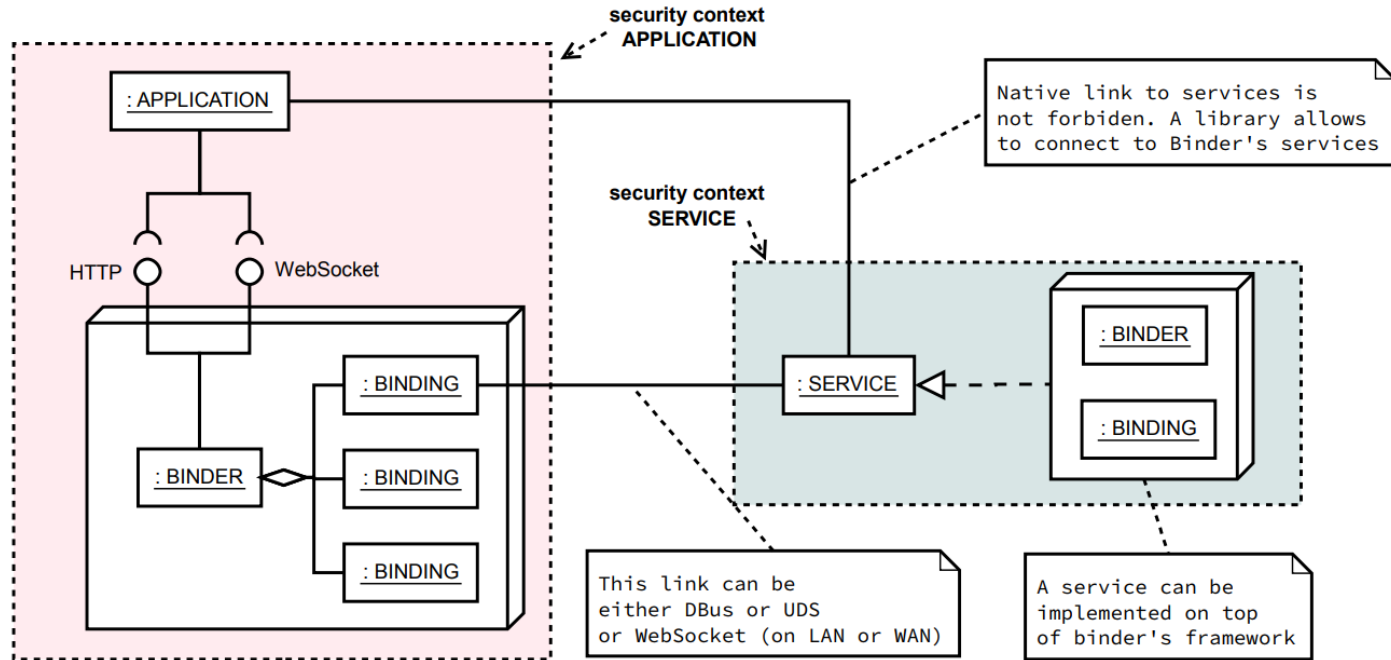
COLLABORA

# Current IVI App FW

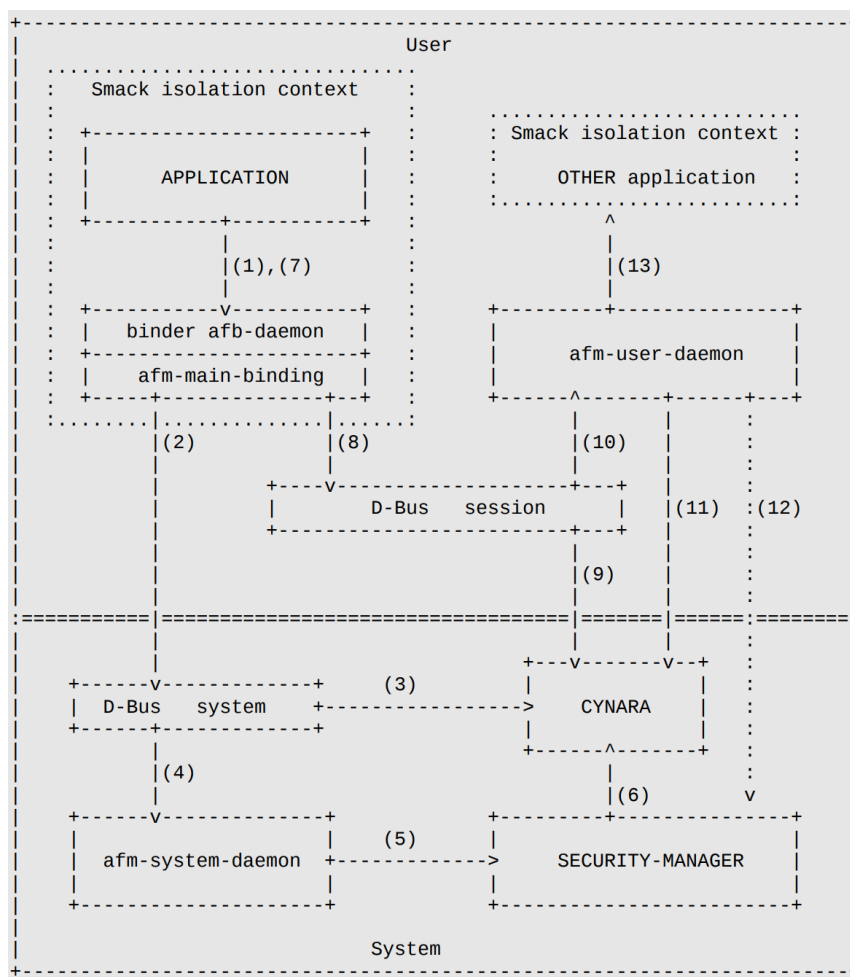
# IVI App FW (current)



# IVI App FW (current)



# IVI App FW (current)



# IVI app framework: takeaways

- Three large functional areas:
- System management (afm-system-daemon)
- Session management (afm-user-daemon)
- Service and IPC mediation (afm-binder)
- Authorization handled via Cynara/SMACK
- App installation and discovery through W3C .wgt format
- Support native apps (Qt, Flutter) as well as WAM
- **Most mature solution in AGL**





COLLABORA

# PR App FW proposal



PR App FW  
(proposal)

App / HMI

HomeScreen, navi, audio, hvac, radio, etc.

Application Framework

HMI Service

Window Manager  
Sound Manager  
Input Manager, etc.

HMI Framework

Application Framework

Platform services

Bluetooth, Wifi,  
telephony,  
location management,

Scope of this document

System anomaly detection,  
Power state management,  
System resource management,  
etc.

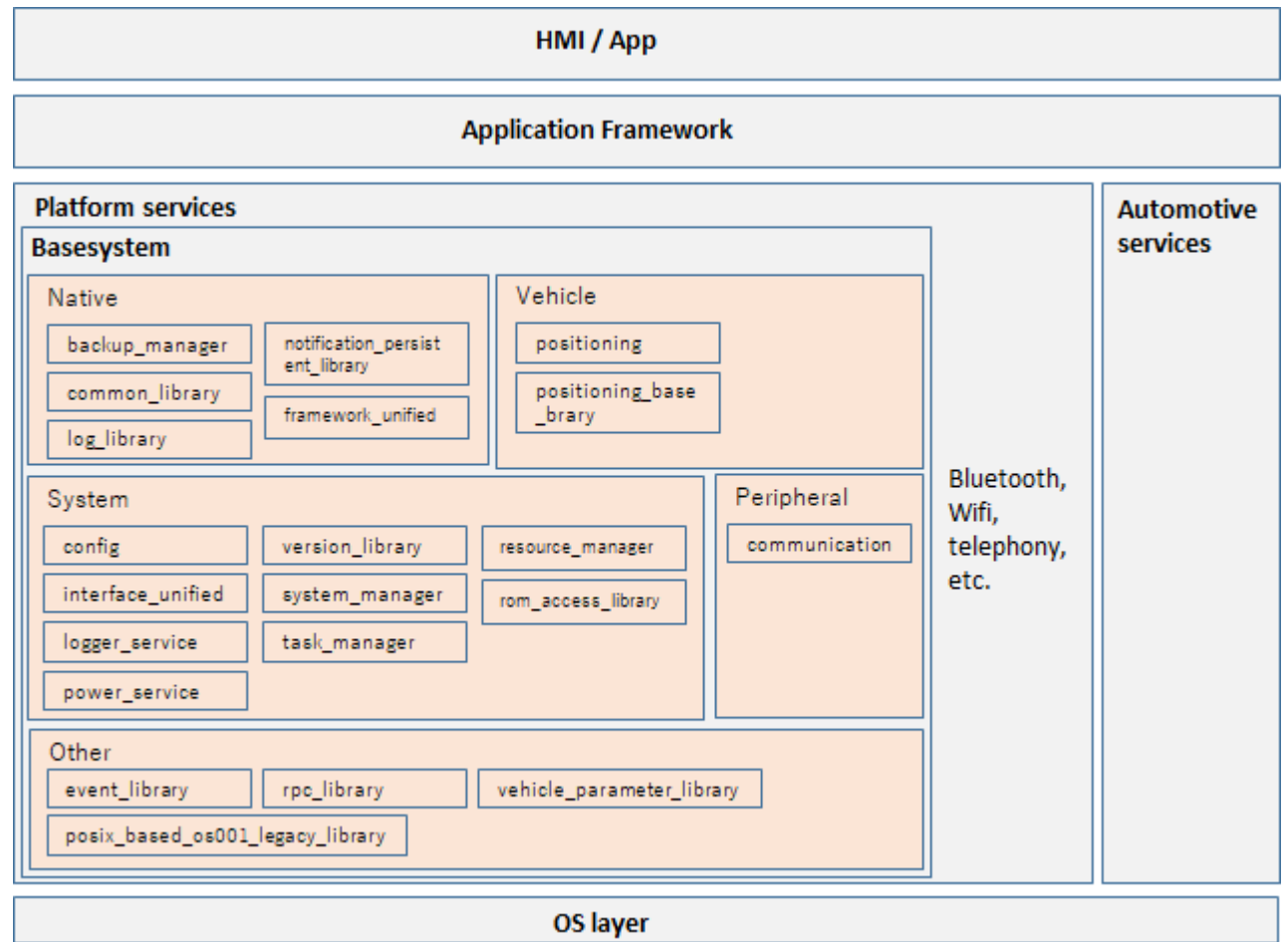
Automotive services

Audio service,  
Speech service,  
Tuner service,  
camera,  
telematics,  
etc.

OS layer



PR Base System (proposal)




# PR app framework: takeaways

- App Framework for PR currently undefined
- Common functional areas:
  - Service lifecycle management (launch/terminate)
  - System and service logging management
- Native apps (Flutter)
- Integration with VirtIO (common 'HAL')
- **Clear overlap with IVI EG, different implementation**





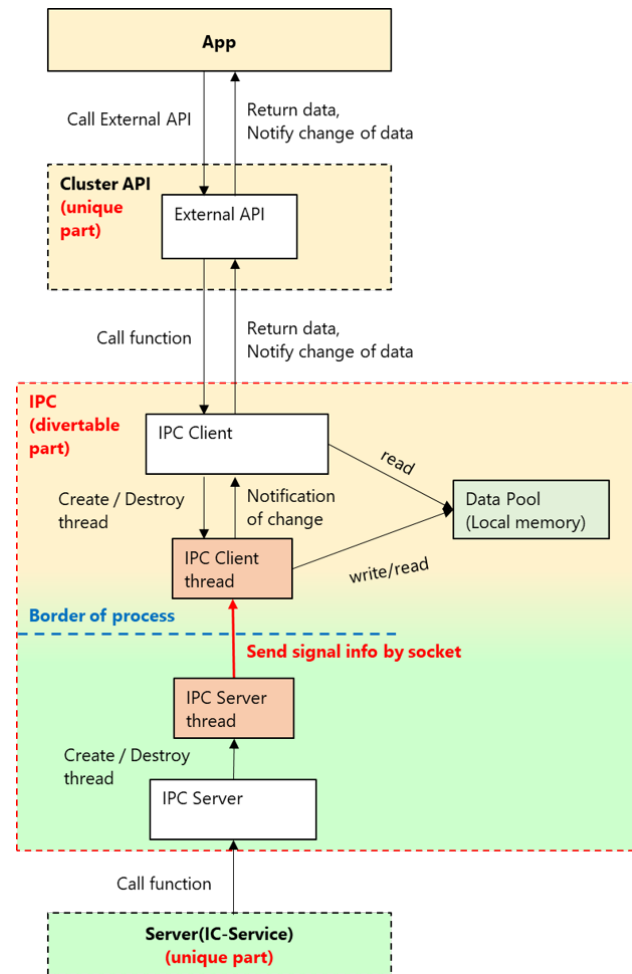
# IC App management proposal?



**IC App FW  
(proposal)**



# IC IPC (proposal)



# IC app framework: takeaways

- App Framework for IC currently undefined
- Static service management?
- Unclear how services will start and be monitored
- Native apps all run in single IC container
- Design of IC-custom IPC API
  - Based on UNIX sockets (local)
  - Based on ICCOM (distant)
- **Most limited usecases**





# Container & mesh proposal



# Container & mesh: takeaways

- Common and cloud-inspired tooling and design (AWS)
- All services built into containers
- Tooling like Envoy/Traefik for inter-service routing
- IPC like gRPC or similar
- Service management with Kubernetes
- Based on explicit declaration of services and interconnects
- Cloud tooling embraces failure: retry, restart, capture
- Cloud tooling based on dynamic workload definition
- **More complex than IC/IVI usecases!**

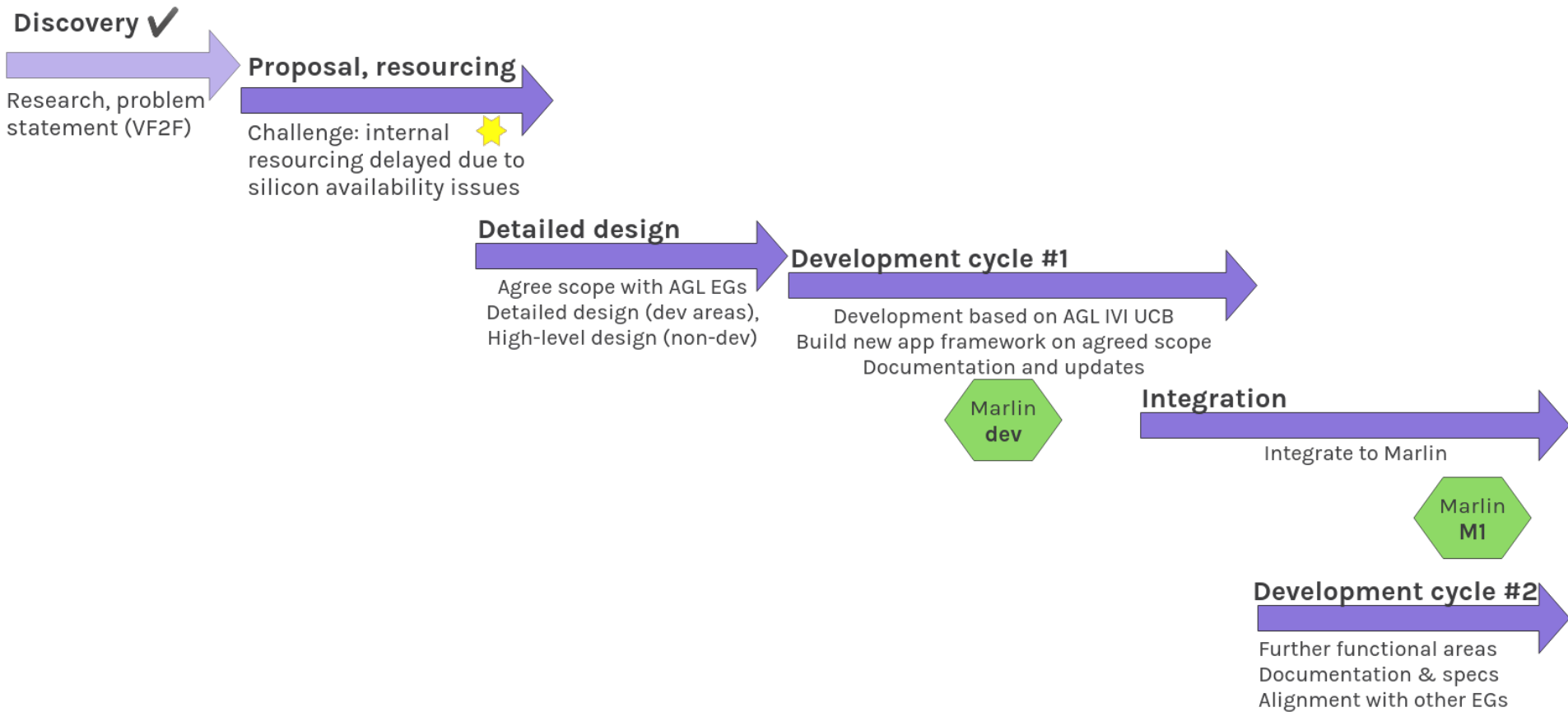




COLLABORA

# Development proposal

March	April	May	June	July	Aug	Sep	Oct	Nov	Dec
-------	-------	-----	------	------	-----	-----	-----	-----	-----



# Development status

- Resourcing for developers delayed due to worldwide supply-chain issues (silicon delay)
- Initial high-level research discussed in March F2F
- Proposal approved by AGL members
- Further research and high-level design discussed here
- Agree areas of priority for design or active development
- Work with and enable other AGL EGs



# IVI EG development principles

- Advance *state of the art* for automotive software
- Close alignment with (& contribution to) *upstream*
- Avoid *duplication* of PR/IC/VirtIO/Mesh development effort
- Support *native & web apps*
- Limited development resources: priority is *value for money*
- Provide *building blocks* not full solutions
- Clear focus on *specific development areas*



# Current development in AGL

	IVI App FW	PR EG	IC EG	Virt EG	Mesh EG
Service provision	✓		✓		✓
App lifecycle	✓				✓
IPC framework	✓		✓		
Service bindings	✓			✓	



# Current development in open source

	systemd	Flatpak	Kubernetes	Envoy	gRPC
Service provision	✓		✓		✓ ✗
App lifecycle	✓	✓	✓		
IPC framework					✓
Service bindings				✓	✓

Note: gRPC provides service IPC but no native discovery/enumeration

# App lifecycle scope

- AGL IVI must be able to launch applications on demand
  - Launch from homescreen (direct interaction)
  - Launch from other-app ('intent' or indirect)
- Applications should be monitored and restarted on crash
- Heartbeat mechanism to ensure app responsiveness
- Capture logs and enable developers to test applications
- Support native & web applications
- Align closely with containerised usecases
- Allow for security policy and usage limits





# App lifecycle out-of-scope

- Not intended for direct deployment to production
- Not intended to provide ‘app store’ distribution
- Not intended to provide *complete* system security model
- Not intended to duplicate existing PR/IC/mesh service models
- Not intended to create bespoke IPC mechanism
- Not intended to comply with functional safety framework
- Focus on development usecases and alignment with upstream + other EGs



# Service provision scope

- Allow AGL system services to be discovered and enumerated
- Services activated on demand by application or system
- Services running within system sandbox
- Services given information about security context of requesting application (including WAM context)
- Allow services to be written in any language
- Allow services to use most appropriate IPC mechanism (UNIX socket, TCP/gRPC, D-Bus, etc ...)



# Service provision out-of-scope

- No custom IPC mechanism: there are already many mature examples: gRPC, ICCOM, D-Bus, etc
- Avoid reimplementation of specific services: reuse open base frameworks unless necessary
- Do not dictate runtime/container mechanisms: allow reuse of whatever makes the most sense
- Delegate system-wide security policy to individual EGs: implementations are incompatible, no point adding more



# Possible upstream bases

- systemd provides most of what we need today
  - Scoped per-session management, logging
  - Isolation and security via cgroups, seccomp, AppArmor
  - Launch native apps from root filesystem
- Flatpak provides further isolation through containers
  - Containerised applications built on common runtime
  - Base runtimes built with Yocto
  - OS services exposed via device nodes, D-Bus, TCP
  - No notion of lifecycle or activation



# Integration of app framework

- Identify most appropriate system services to provide
- Examples of running system services under systemd with activation and lifecycle management
- Use systemd system scope for services (OEM/Tier-1)
- Package app-relevant part of AGL UCB into Flatpak runtime
- Use systemd session scope for apps (ISV)
- Examples of native Flatpak apps, activated by systemd
- Sensible security policies and use limits for example apps
- Document both to show clear best principles



# Development outcomes

- A stripped-back UCB, divided into tier-1/OEM and ISV worlds
- Examples of how to develop services/applications which can be useful for integration into all AGL profiles
- Provide a 'halfway house' between native applications running directly on system (IC, PR) and containerised applications (state-of-the-art IVI, mesh)
- Continue to support WAM and HTML-based apps
- Clear documentation and design principles
- Reuse upstream design decisions and principles



# Integration challenges

- Writing AGL binder definitions for every service makes services available to WAM, but means that every service must be wrapped and multiple definitions maintained
- Need to provide WAM bridges for RPC mechanisms (gRPC, D-Bus?)
- Going from current UCB to new world with sensible transitions
- Unified CES demonstrator between IVI/IC already based on outdated branches



# Open questions

- What do we demonstrate at CES, and how do we show it?
- How do we balance the demands and conflicting designs of other AGL EGs?
- What is the most valuable contribution to AGL?
- What is the most valuable contribution to the community?
- How are we resourcing the demo apps, and who is doing non-system work (e.g. UI and design)?
- Is the proposed timeline viable for Marlin?
- Are the suggested usecases defined and agreed?







**Thank you!**



COLLABORA

**Open First**