

# 2021 IVI-EG and TOYOTA Activity Plan

- Objectives of this Discussion
- Activities in 2020
- Plan for 2021
- 2022 and Beyond
- Finalize Feature list

- Agree on the scope of TOYOTA's contribution
  - list up what's missing
- Agree on the next steps of IVI-EG
  - update feature list

- Disclosed production-ready functions as planned

- basesystem
- RBA

- Kooky Koi release(Feb/2021)

- Succeeded in Kicking off Production Readiness
- [AGL/meta-agl-devel.git] / meta-oem-production-readiness /

- Areas of Improvement

- We should be more compliant to rules of yocto recipes
- We should avoid huge commits

	IVI	IVI-ProductReady Trial (in March 2021)	IVI-ProductReady 1 <sup>st</sup> Release (in December 2022)
HMI-Apps	Apps For demo	+Test Apps for IVI-PR [Toyota]	+ [Tier1,OEM]
HMI	Qt/Chromium	Qt/Chromium	+T.B.D
AppsFW	AGL-Binder	AGL-Binder	+New Binder[Tier1,OEM]
HMI-Service	WindowManager AudioManager	+ RBA[DN]	+ HMI service[Tier1, OEM]
Other-Service	[AGL]	[AGL]	+ BT, Radio[AGL]
Base-System	Not be merged	+ BaseSystem[Toyota]	+Diag,Logging,Security +Multifunction Detector +HAL[Tier1,OEM]
Middleware Linux Kernel	[Yocto & AGL]	[Yocto & AGL]	[Yocto & AGL]
Evaluation	For demo	Profile Maintenance & Evaluation[Toyota] BSP,HW[Renesas]	+including the part of quality test[Tier1,OEM]

Fig.) Plan of Production Readiness Profile (Jul, 2020)

**Kooky Koi**

New Features available in Kooky Koi:

- General
  - Continue with Yocto LTS version - upgraded 3.1.4 for 11.0.0
- IVI Expert Group
  - Rule Based Arbitrator (RBA) integrated into AGL Compositor as a compile time option.
  - Toyota Base System available as technology demonstrator
- Web Apps
  - Update to Chromium 79
- Speech Recognition
  - Update to Alexa Auto SDK 2.3
- AGL Compositor update
  - Waltham integration
  - Screenshot capability added
- Connectivity
  - Cancelloni support added to enable remote CAN testing.
- Continuous Integration and Test
  - Code coverage for application builds and 'pyagl' tests for agl-service-\*

Fig.) Kooky Koi release list

- Successfully launched bi-weekly IVI-EG
  - Where we can focus on Production Readiness
  - Where OEMs can disclose Production Requirements
  
- Areas of improvement
  - Discussion Topic was too focused on basesystem and review comment for committing basesystem
    - Less attractive
  - Activities related information are not properly maintained in JIRA or Confluence
  - Meeting minutes are not sufficiently described.
    - Less appealing

- Production Readiness Requirement Specification v0.10Draft
  - Draft version for BaseSystem was published
- Areas of improvement
  - Too implementation specific and lack of sufficient Requirements
  - Couldn't held meetings among OEMs as planned

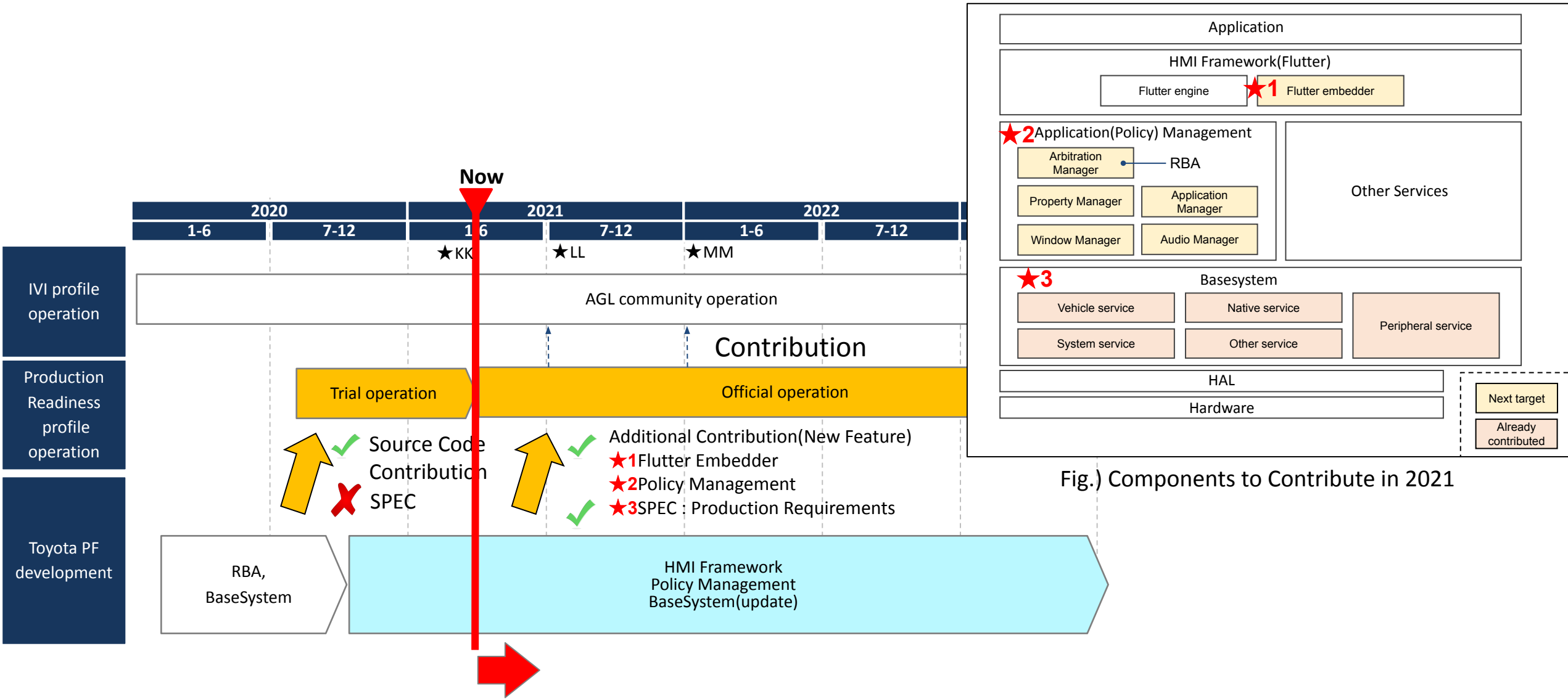


Fig.) Production Readiness 2021 plan

Fig.) Components to Contribute in 2021

- What is Flutter?
  - Flutter is Google's portable UI toolkit for crafting beautiful, natively compiled applications for mobile, web, and desktop from a single codebase.
- Why Flutter for IVI?
  - High performance
  - Smartphone-tier touch mechanics
  - Developer ergonomics
  - Faster iteration from customer feedback
  - BSD 3-Clause "New" or "Revised" License
  - more thoughts in our presentation
    - <https://www.youtube.com/watch?v=zSbsliluixw&t=1963s>

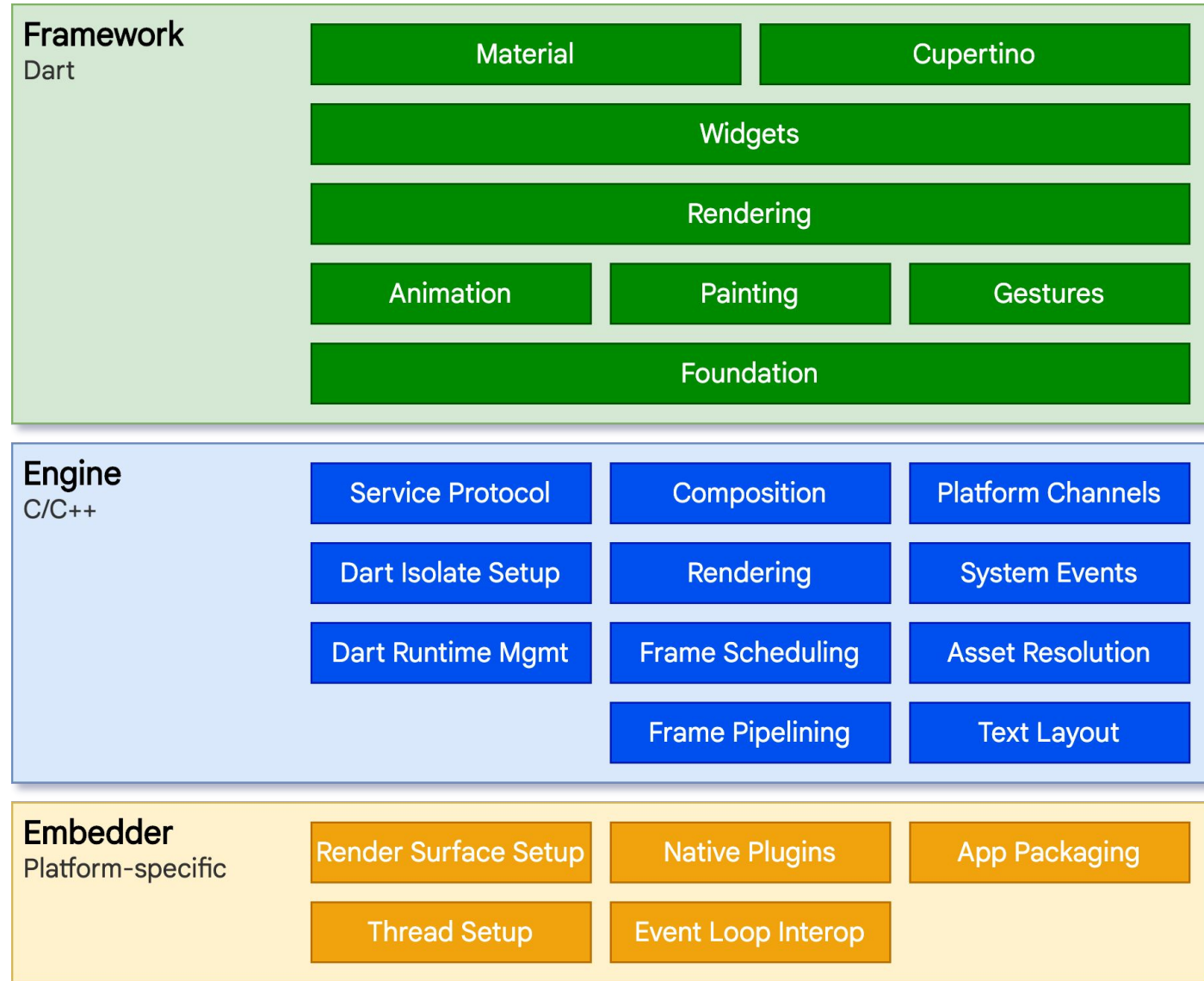


## Flutter Components

- <https://flutter.dev/docs/resources/architectural-overview>

## What can be contributed from TOYOTA?

- Flutter Embedder for AGL(agl-shell)
- Prototype of yocto recipes
  - Flutter build is based on GN + Ninja
  - Engine should not be modified from mainline



- What TOYOTA have tested
  - Run against AGL Icefish(agl-shell)
    - Targeting newer agl-compositor but not updated yet
  - Sample Flutter Apps is running, some Open Source Flutter Apps can work
  - Recipes for minimum yocto image
- Goal of 2021 and (April - June)
  - April - June : Embedder is upstreamed to staging repository
    - Risk : Internal Legal Checks
  - April - June : Sample Flutter Apps can be demonstrated on AGL
    - Not targeting the integration with other AGL services
  - 2021 : Flutter can be an option of HMI FW in AGL
    - Architecture defined, integrated with AGL services
  - 2021 : With help from AGL community, other sample apps will be working

- What's missing?
  - (would like to Ask Community members)
  - (We need someone who can kindly lead this activity from community side)
  - Fund needed?
  
- Rough schedule
  - 21' April. Define the (initial) Architecture and the scope of contribution
    - other options
      - <https://github.com/sony/flutter-embedded-linux>
      - [https://github.com/jwinarske/flutter\\_wayland](https://github.com/jwinarske/flutter_wayland)
      - canonical flutter embedder gtk backend (xdg-backend) <- this could be the option
  - 21' May. Yocto recipes defined. Internal refactoring, developments.
  - 21' June. Complete internal legal check and push to staging

# Policy Management (1/2)

- What is Policy Management?
  - Manage which apps to show on display and starting/stopping sound, judges priority and switchability based on arbitration rule.
  - We internally called this “Application Management” due to some historical reason, but actually this is Policy Management. Not directly related to agl AppFW.
  
- Why Policy Management again? What’s the relationship with RBA?
  - Arbitration Controller that manage both of Window and Sound
  - Purpose : showcase production use cases and the implementation
  - option1 of [https://wiki.automotivelinux.org/\\_media/agl-distro/rulebasedarbitrator\\_a02.pdf](https://wiki.automotivelinux.org/_media/agl-distro/rulebasedarbitrator_a02.pdf)
    - option2 for the integration with agl-compositor

- What TOYOTA plan to contribute
  - Arbitration Manager
  - Detail is under planning
  
- Goal of 2021 and 2021 (April to June)
  - April to June : Define what to disclose and complete internal refactoring
  - April to June : Complete internal legal process
  - April to September : Push source to staging repository
  - April to September : Design Yocto recipes
  - 2021 : Merged to meta-agl-devel/meta-oem-production-readiness

# Product Requirement (1/2)

- We are Code First Community! Still, Production Requirements are important for AGL, especially for OEMs.
- Why?
  - Without the definition of product requirements, we cannot understand the importance of each functions
  - We cannot talk about what's missing in AGL
- Product specific requirements vs (common) Product requirements
  - Many requirements are product specific
    - ex. show OEM logo on display,
  - There are common requirements among OEMs
    - ex. keep log data when failure occurs,
- AGL should focus on common requirements

# Product Requirement (2/2)

---

- Goal of this activity
  - Mid Term(~2022) : Product Requirements are defined as part of AGL Specifications
  - Short Term(2021) : Product Requirements for basesystem and related functionalities are documented
- Looking back on the Trial Period
  - We wanted to start this activity during Trial Period, but we couldn't.
- Approach toward the goal
  - 1st step : Toyota exemplifies what kind of Requirements we need. Toyota will summarize use cases for basesystem and why these functions are required for products.

- Closely work with AppFW-EG
- Show our requirements



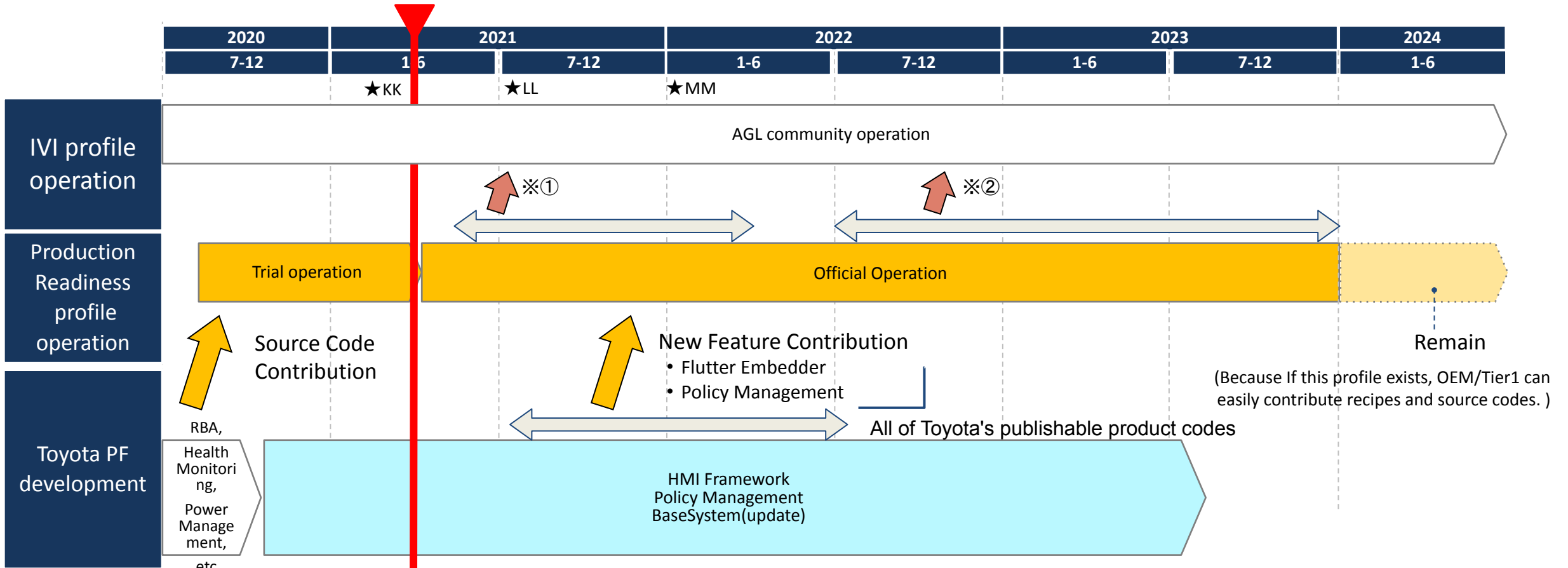


Fig.) Production Readiness future plan

① Toyota will try the following topics.

- Basesystem source codes from staging to src from one selected function.
- Basesystem recipes from meta-agl-devel to the suitable layer (meta-xxx) in accordance with source codes.

② Toyota will try the following topics.

- Policy management and flutter functions source codes from staging to src from one selected function.
- Policy management and flutter recipes from meta-agl-devel to the suitable layer (meta-xxx) in accordance with source codes.