API Standardization in JASPAR for SDV

Japan Automotive Software Platform and Architecture

2026 June 30

Kazuo Tsubouchi
Chairperson
JASPAR API WG

Agenda

1. JASPAR Overview
2. API Standardization in JASPAR
3. Collaboration with COVESA
4. Conclusion

What is JASPAR?

- Japan Automotive Software Platform and Architecture
- Established in 2004
- Focuses on the standardization of automotive software and E/E architecture

Mission

Through standardization activities in Japanese automobile industry,
we will realize the following.

- Development of common foundation for automotive industry
- Increased development productivity
- Promoting technological development

JASPAR promotes the standardization of vehicle systems in Japanese automotive industry

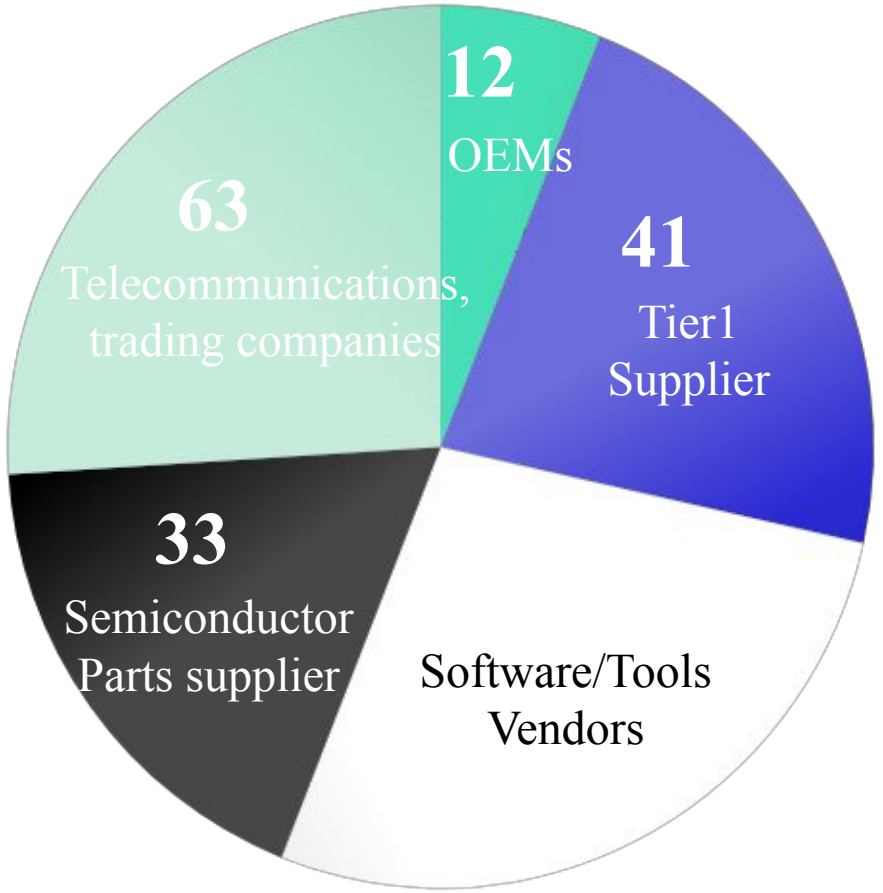
JASPAR Member

JASPAR Member

- Total corporate members: 206 (as of Jun 2026)
- In addition to the above, group companies (44) and universities and research institutes (14) participated

Board Members

- 5 companies lead JASPAR's activities as board companies.
  - Toyota Motor Corporation
  - Nissan Motor Co., Ltd.
  - Honda Motor Co., Ltd.
  - DENSO CORPORATION
  - Toyota Tsusho Corporation



Industry breakdown of corporate members

Consist of OEMs, suppliers, SW development companies, and various other companies/organizations related to automotive industry

JASPAR 20 Years of History

2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	2025～

FlexRay / AUTOSAR

Functional Safety

Bluetooth, Connectivity

Ethernet

Security, OTA

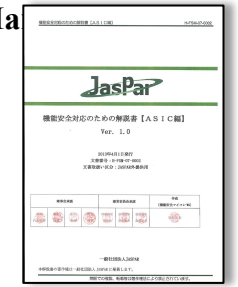
Vehicle I/F

September 2004
Establishment of JASPAR

In 2007
FlexRay/AUTOSAR and Declaration of Cooperation

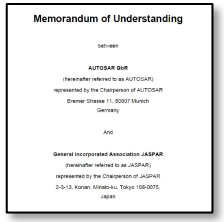
In 2010
National Professional Achievement Presentation

In 2013
Functional Safety Ma

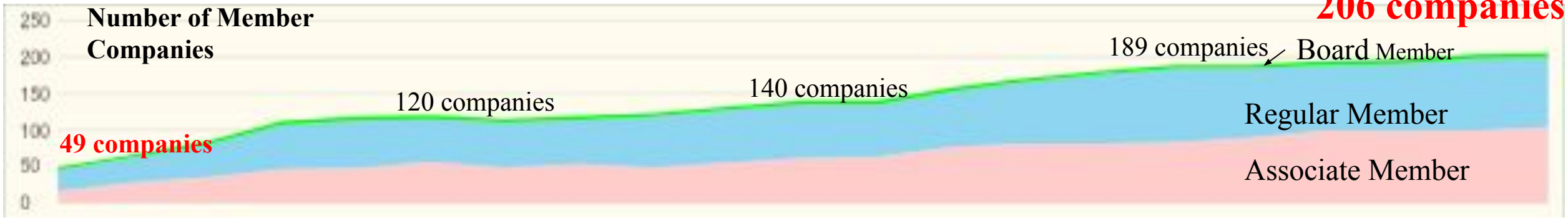


In 2022
Hosted IEEE International Conference

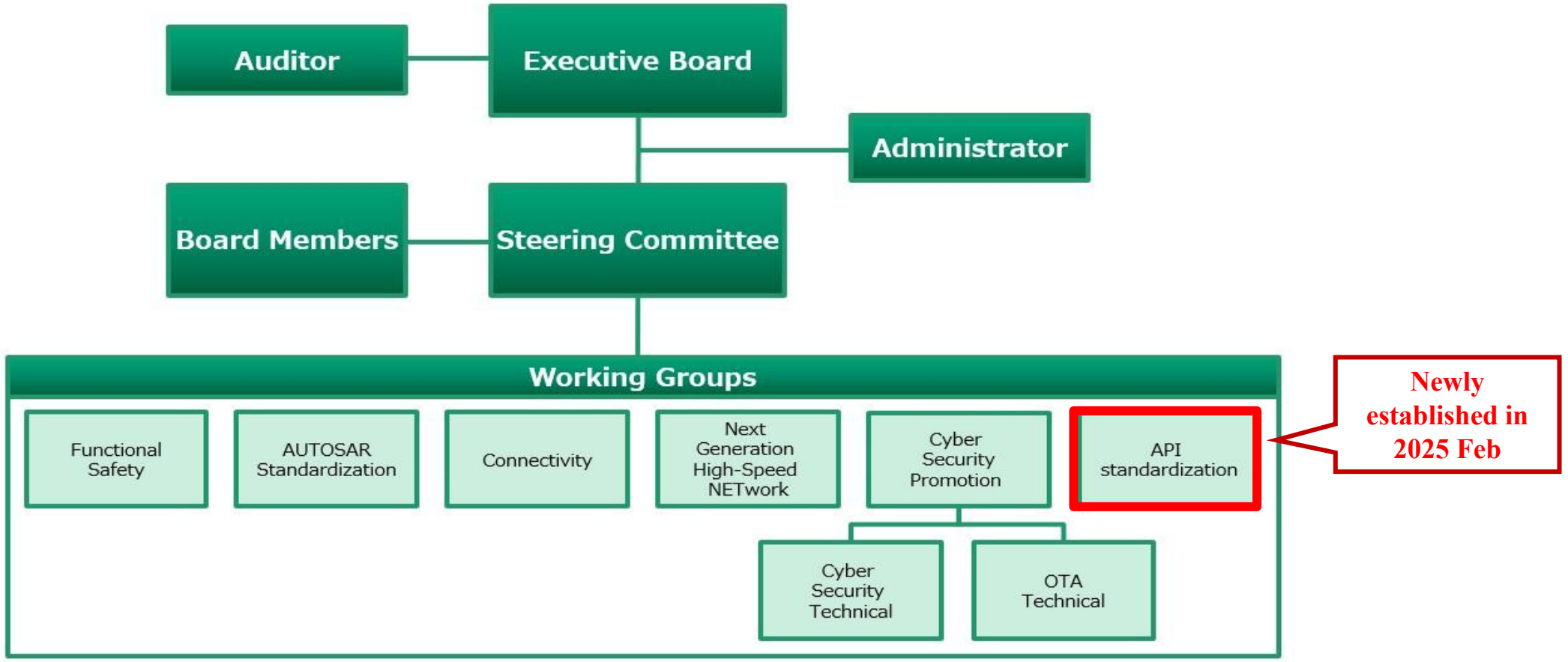
In 2023
Signed AUTOSAR MoU



September 2024
JASPAR 20th Anniversary
First year of SDV



JASPAR Working Groups



7 WGs have been formed to focus on specific technical fields and issues.

Agenda

1. JASPAR Overview
2. API Standardization in JASPAR
3. Collaboration with COVESA
4. Conclusion

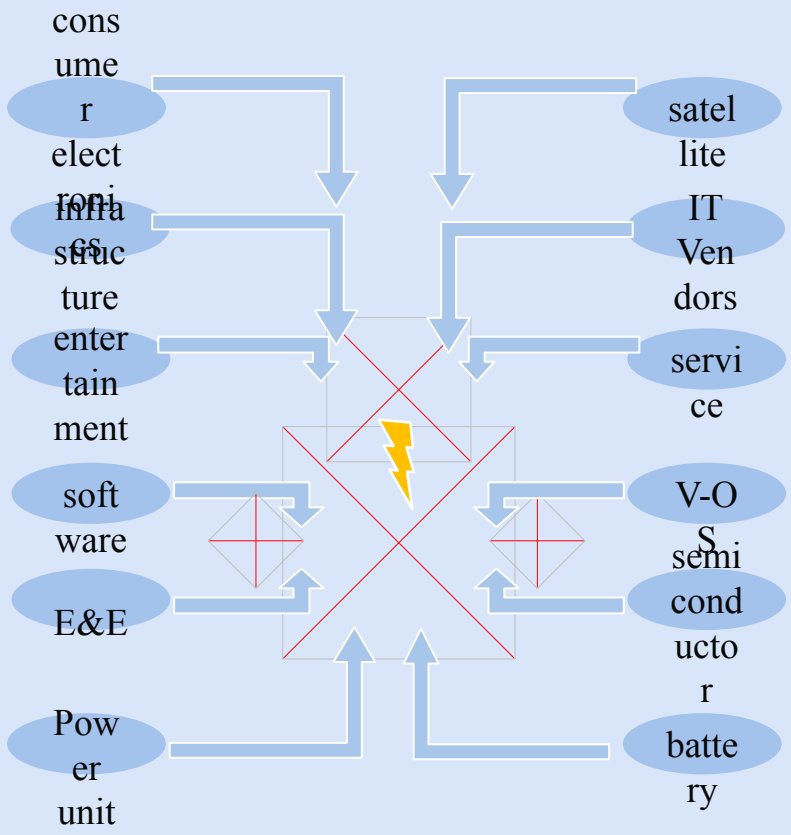
JASPAR for the next 20 years

The mobility industry is challenging a once-in-a-century transformation

Enabling SDV is the most important theme for the industry to address for the next 20 years

Consequence of SDV

SDV will formulate a new mobility ecosystem



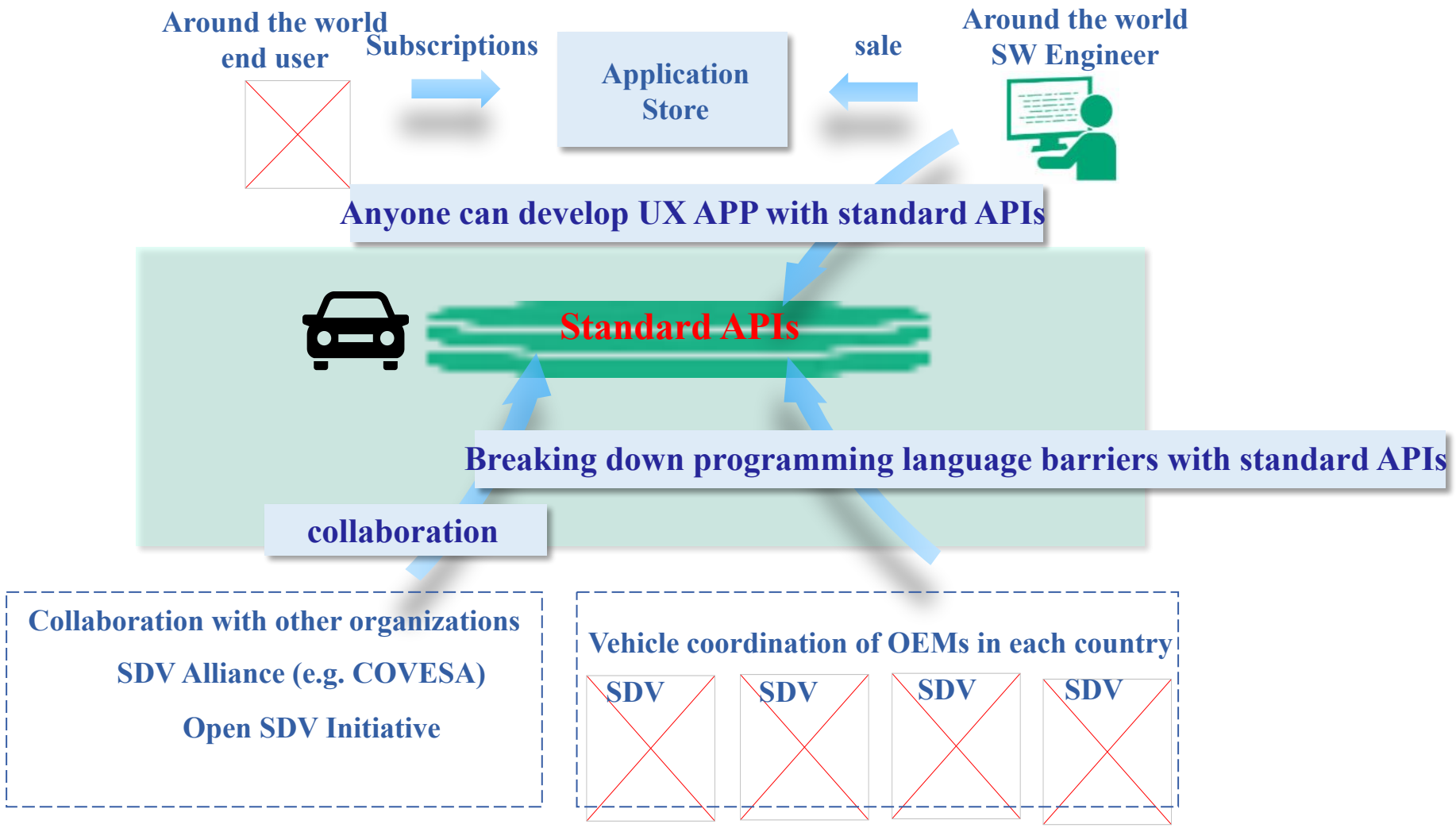
Quickly adopt the latest technology with OTAs to maintain and improve the competitiveness of your products

It can be customized according to the user's needs, providing a more comfortable and convenient travel experience

New players related to software development will enter the market, and the economy will be stimulated by creative services and products.

- **SDV will create new business opportunities for the entire automotive industry**
- **Collaboration/Standardization is so important to realize the SDV**

Standardization of APIs for SDV



To realize SDV, it is essential to standardize the APIs, which is the common language for SDV.

Structure of the API Standardization WG

Leadership

WG Management | WG Supervision

- **Tsubouchi (Honda)**
- **Ando (Toyota)**
- **Miwa (Nissan)**

Admin

Dev. Env. Mgmt. | License Mgmt.

- Administrative tasks

WG-PMO

Group Activity Mgmt. | Coordi-nation

- Project Management

Alliance

Alliance with other organizations
・Morita (Bosch)

1) Strategy and Planning

Reference Arch. | Rule & process | Review Deliverables

- **Define reference architecture** diagram, API design rules, design process, and PoC plan
- Participants:
  - Board member companies

2) Use Case Base Evaluation

Use case Ideation | Prioritize APIs | System Engineering

- **Make use cases** by research and ideation
- System analysis and request APIs
- Participants:
  - Companies who plan to utilize the standardized APIs

3) Design and Development

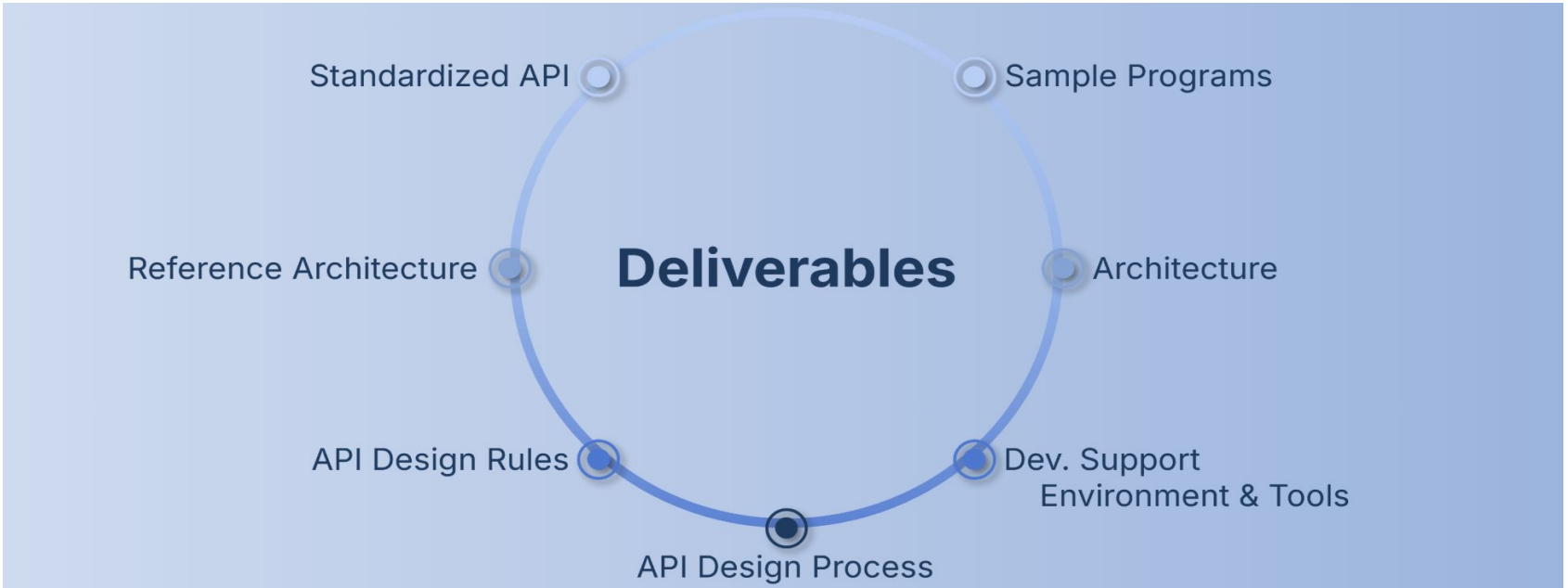
Define JASPAR VSS | SWE Engineering | Reference vehicle and app

- Update VSS / vspec
- Develop dadabroker, code generator for API and reference applications
- Participants:
  - Companies with experience in implementing in-vehicle SW

- **More than 100 software engineers (from 38 companies) are actively working on the project**

Contributors

What will the JASPAR API WG provide?



Standardized API
Source code and specifications for the standard APIs

Reference Architecture
A logical reference architecture for integrating the standard APIs

API Design Rules
Concrete design rules to ensure consistency, reusability, and safety

API Design Process
An efficient process covering use-case definition through API implementation and validation

Dev. Support Environment & Tools
Software development environment and tooling

Architecture
Yoriito VSS, Yoriito VISS, Yoriito VISS Server (**Yoriito Kura**)

Sample Programs
Headless app, GUI app, PAL implementation samples

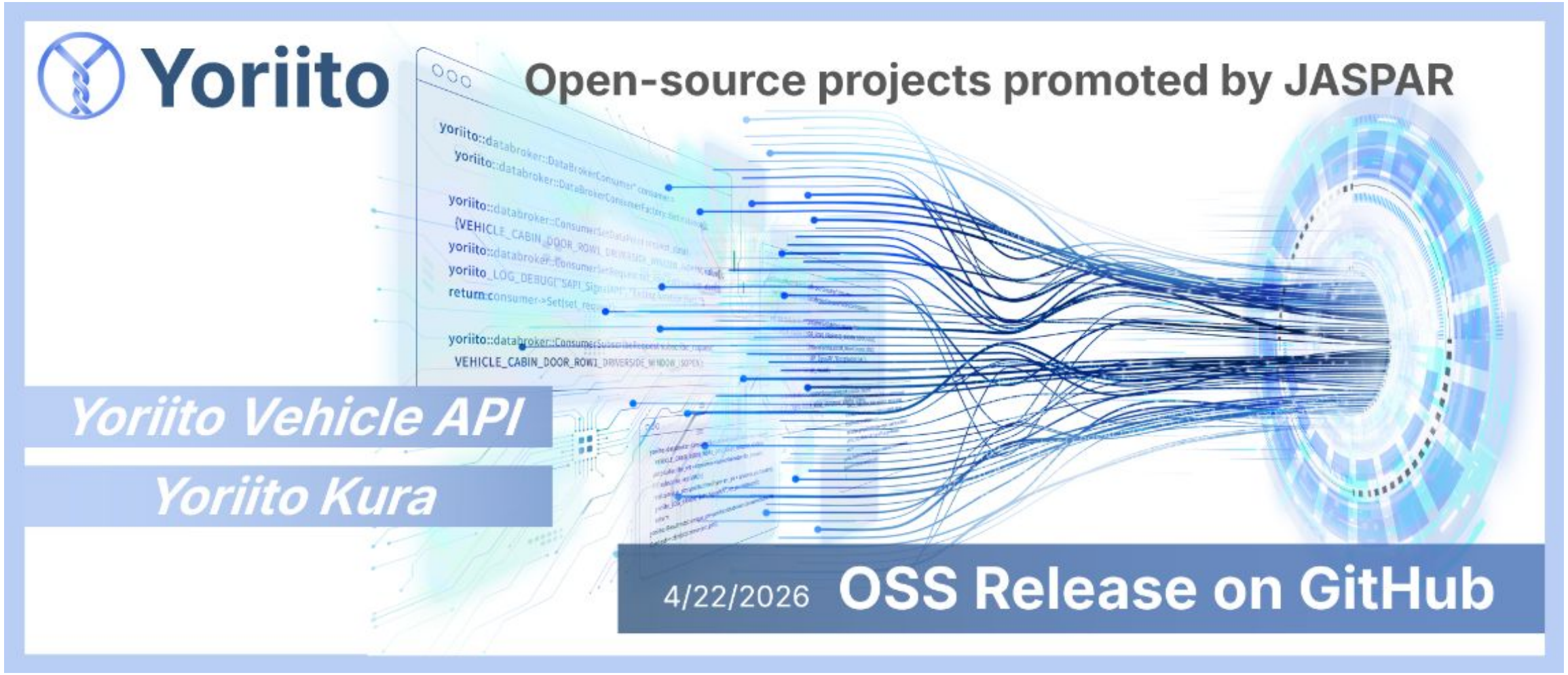
Achievement in FY2025

- Based on COVESA VSS, JAPSAR implemented APIs that allows comprehensive control of vehicle signals
- Prepared tutorials and various documentation so that new developers can easily start app development

Theme		Activities	Deliverables
API Planning		•Define vehicle functions based on COVESA VSS •Investigated APIs to meet the use-cases	•Define Vehicle Signals：631 •Define Use cases： 117
API Development	**Design/ Implementation**	•Design and implement Signal APIs	**•Signal APIs： 10,132 APIs** **•API code： implement 375,773 LOC Test code 515,629 LOC**
	User support/ Quality control	•Make tutorials and sample applications for onboarding new developers •Define quality criteria	•HeadlessApplication code： 3,375 LOC •UI code： 2,114 LOC **•Tutorial Documents: Many**
Architecture/ Infrastructure		•Architecture design, design guidelines •Build Development Infrastructure (SDK、Runtime、CLI、CI/CD、libraries)	•JASPAR Databroker code： 97,000 LOC •Libraries **•Documentation** **- Design Doc** **- Specification Doc** **- Design/Implementation Guidelines**

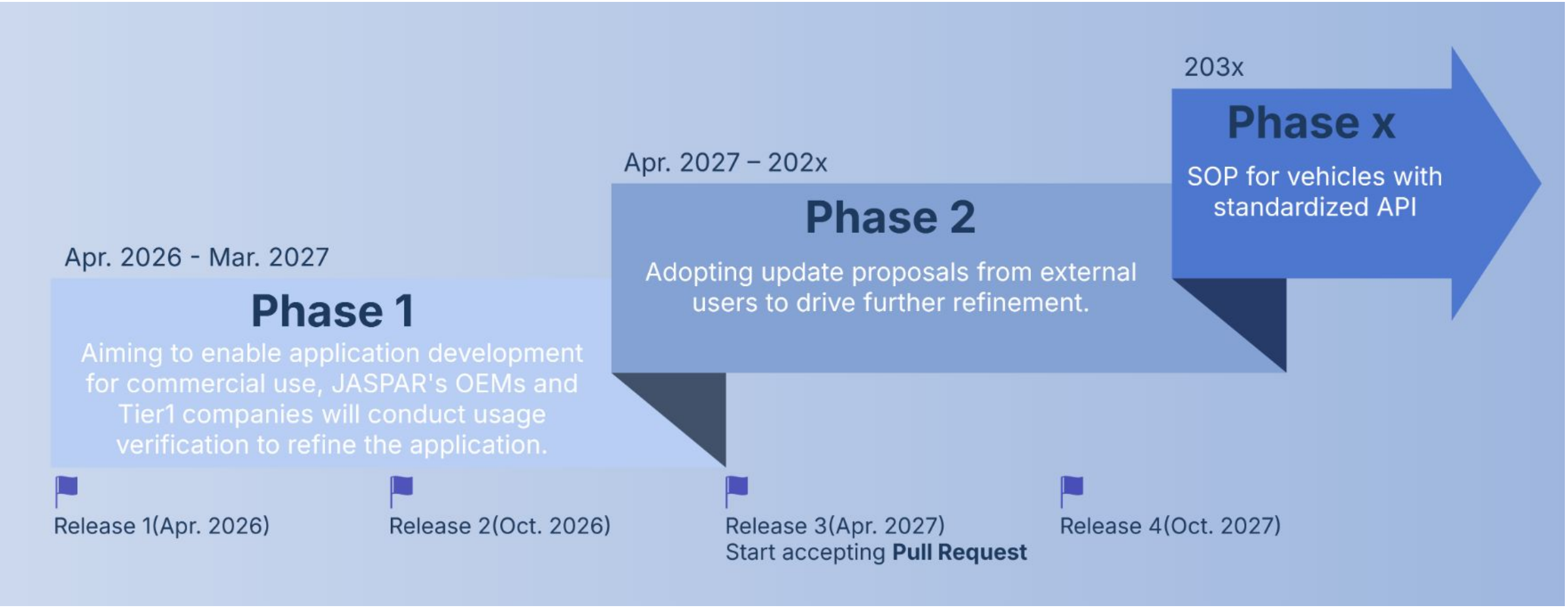
Many accomplishments in just one year!! Thanks to strong collaboration!

1st OSS Release Today!!



More open. More free. Automotive software—built by everyone

Roadmap



Architecture and scope

* Under development

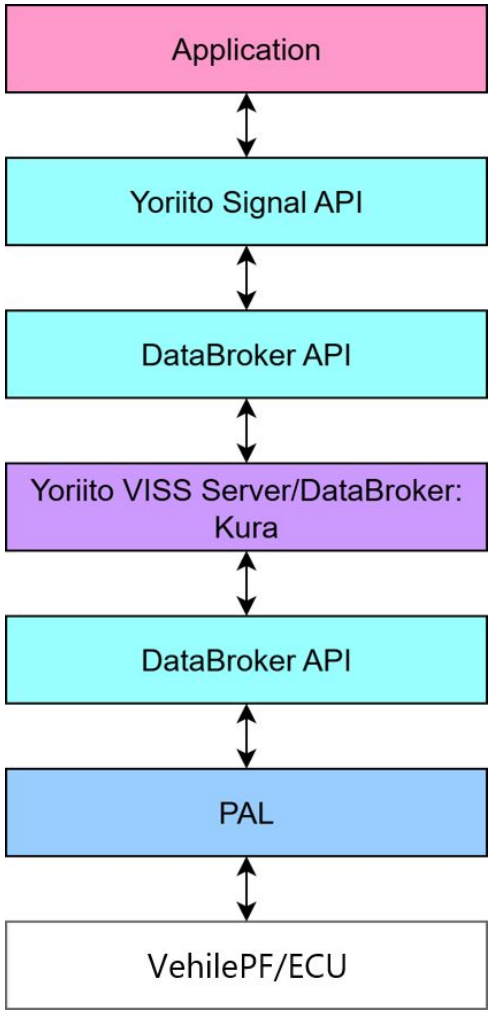
*

Out-of-scope

- **Authentication and authorization methods**: The methods for authentication and authorization (e.g., ID management, token/certificate handling, permission models) are out of scope for the Yoriito Vehicle API. This API defines its interfaces and behavior assuming it is called within a properly authenticated and authorized environment.
- **In-vehicle OS, middleware, and network stacks**: The specifications and implementation methods for the in-vehicle OS, middleware, and network stacks (e.g., CAN, SOME/IP) are out of scope. The Yoriito Vehicle API focuses on the specifications of the abstract APIs (Signal API, Access API, PAL) that operate on top of them.
- **Vehicle Functions and Safety Concepts**: The presence or absence of the vehicle functions themselves, their design policies, and safety concepts (e.g., HARA, ASIL) are out of scope. The scope of Yoriito Vehicle API is limited to how existing or company-defined vehicle functions are abstracted and provided as APIs.
- **UX design**: UX designs such as application specifications, screen designs, and user operation flows are out of scope. The Yoriito Vehicle API focuses on defining common vehicle-function interfaces that are usable by applications.
- **Company-specific operational and lifecycle management**: Company-specific operational and lifecycle management such as OTA update methods, application distribution channels, and auditing/logging practices are out of scope. The Yoriito Vehicle API only addresses the minimal items directly related to interoperability such as API versioning policies.

Key Components

Key Components of the Yoriito Vehicle API Basic Architecture are as follows:



Yoriito Signal API:
An implementation of a vehicle API that can be commonly used across OEMs with different vehicle signal definitions
This API is defined based on COVESA's VSS as its underlying data model.

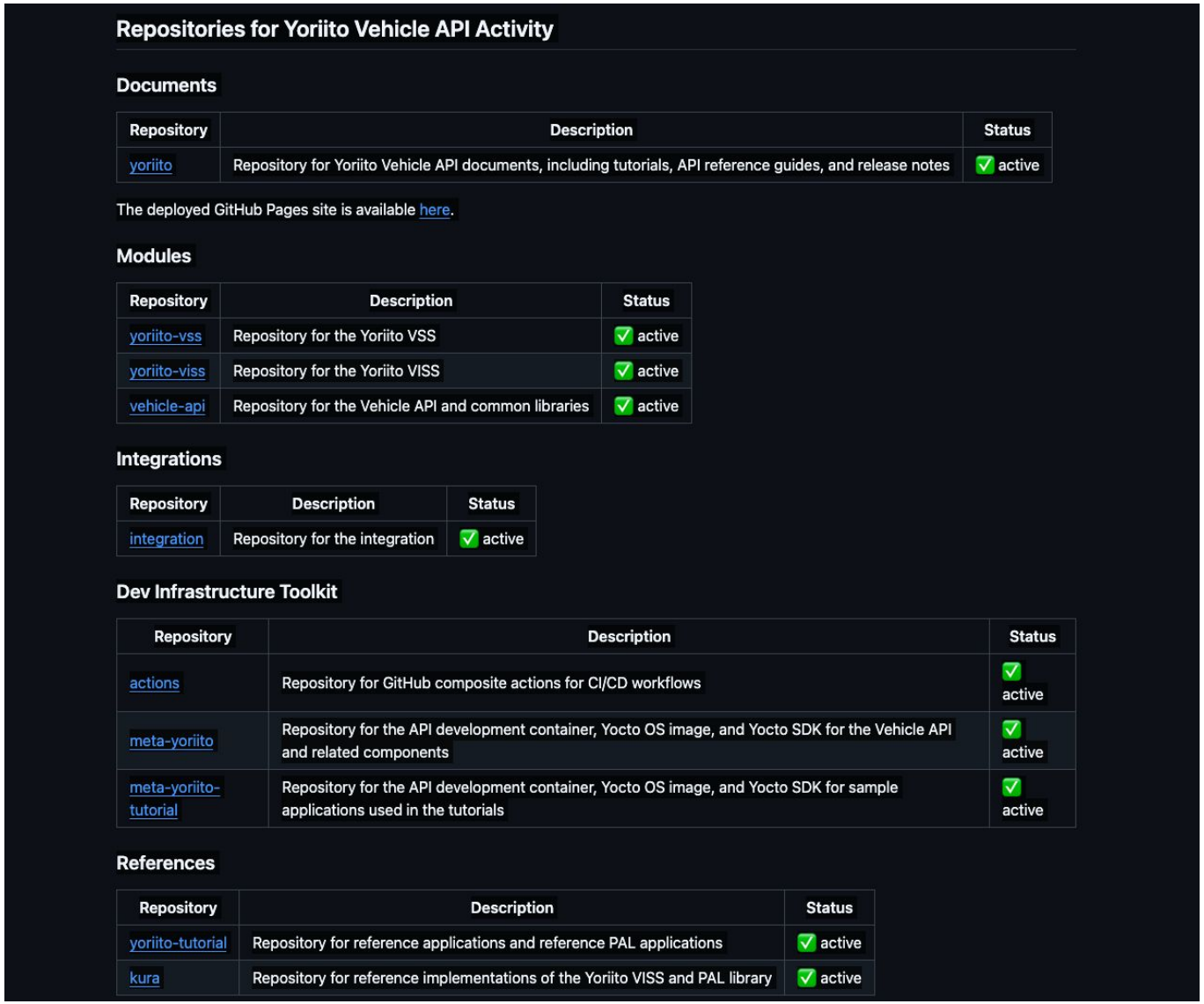
DataBroker API:
A low-level client API for communicating with the Yoriito VISS Server/DataBroker

Yoriito VISS Server/DataBroker:
A data broker that serves as the core of communication between the application and PAL
Yoriito provides *Kura* as the reference implementation.
Kura is based on COVESA's VISS v3, with several additional functions added by JASPAR that are considered necessary for in-vehicle use.
To enable an early launch of YoriiTo, it is currently implemented with proprietary extensions; however, JASPAR will continue discussions with COVESA going forward.

PAL (Platform Abstraction Layer):
Software that absorbs and abstracts differences in vehicle platforms

For more detailed information, please refer to the following
https://docs.yoriito.dev/design/architecture_overview/

Repositories



Repositories for Yoriito Vehicle API Activity

Documents

Repository	Description	Status
yoriito	Repository for Yoriito Vehicle API documents, including tutorials, API reference guides, and release notes	✅ active

The deployed GitHub Pages site is available here.

Modules

Repository	Description	Status
yoriito-vss	Repository for the Yoriito VSS	✅ active
yoriito-viss	Repository for the Yoriito VISS	✅ active
vehicle-api	Repository for the Vehicle API and common libraries	✅ active

Integrations

Repository	Description	Status
integration	Repository for the integration	✅ active

Dev Infrastructure Toolkit

Repository	Description	Status
actions	Repository for GitHub composite actions for CI/CD workflows	✅ active
meta-yoriito	Repository for the API development container, Yocto OS image, and Yocto SDK for the Vehicle API and related components	✅ active
meta-yoriito-tutorial	Repository for the API development container, Yocto OS image, and Yocto SDK for sample applications used in the tutorials	✅ active

References

Repository	Description	Status
yoriito-tutorial	Repository for reference applications and reference PAL applications	✅ active
kura	Repository for reference implementations of the Yoriito VISS and PAL library	✅ active

https://github.com/yoriito/

Documents
https://github.com/yoriito/yoriito

Modules
https://github.com/yoriito/yoriito-vss
https://github.com/yoriito/yoriito-viss
https://github.com/yoriito/vehicle-api

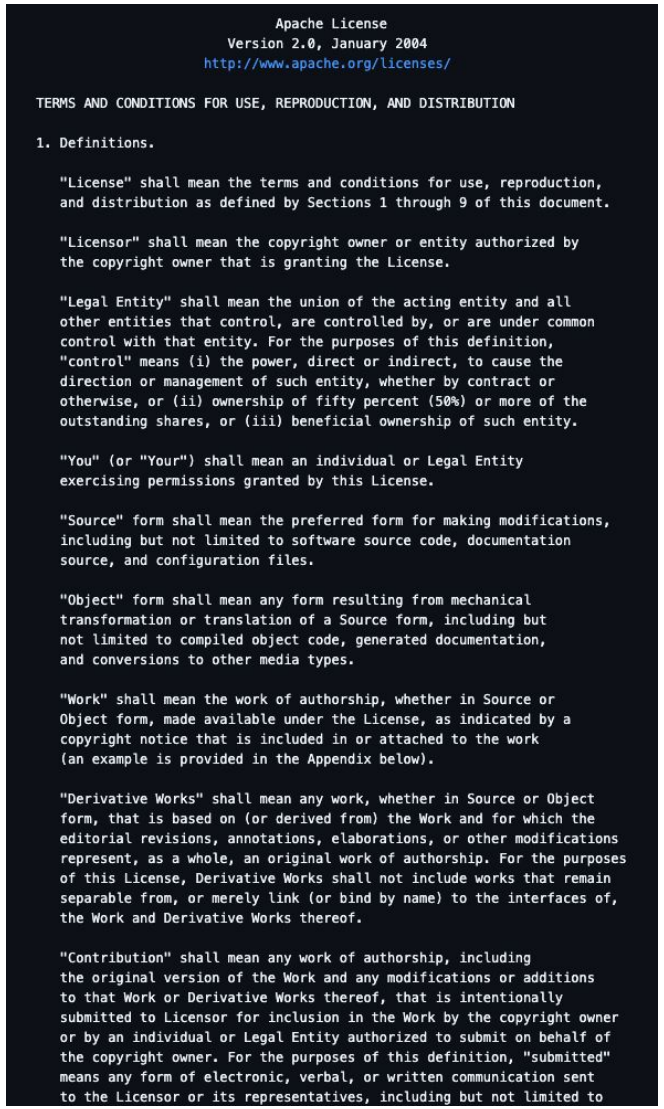
Integrations
https://github.com/yoriito/integration

Dev Infrastructure Toolkit
https://github.com/yoriito/actions
https://github.com/yoriito/meta-yoriito
https://github.com/yoriito/meta-yoriito-tutorial

Reference
https://github.com/yoriito/yoriito-tutorial
https://github.com/yoriito/kura

License, NOTICE and Security Policy

License is Apache-2.0



Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

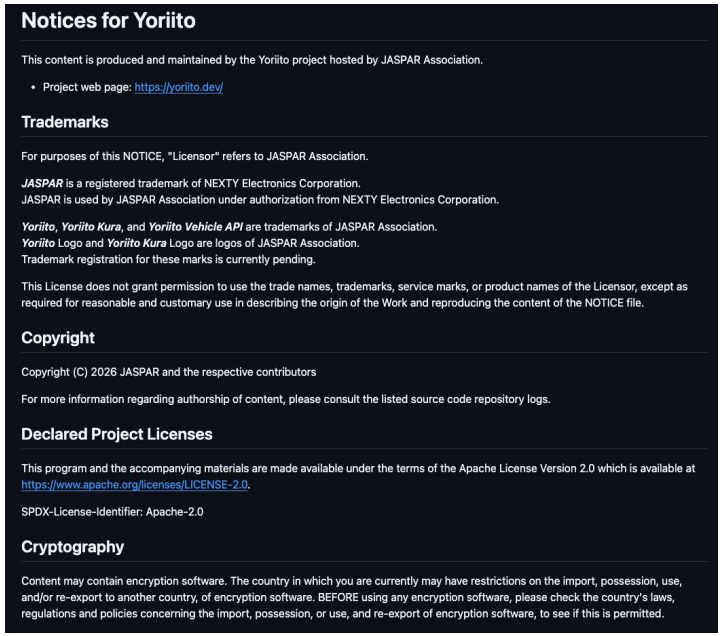
"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to

Copyright is JASPAR



Notices for Yoriito

This content is produced and maintained by the Yoriito project hosted by JASPAR Association.

- Project web page: https://yoriito.dev/

Trademarks

For purposes of this NOTICE, "Licensor" refers to JASPAR Association.

JASPAR is a registered trademark of NEXTY Electronics Corporation.
JASPAR is used by JASPAR Association under authorization from NEXTY Electronics Corporation.

Yoriito, ***Yoriito Kura***, and ***Yoriito Vehicle API*** are trademarks of JASPAR Association.
Yoriito Logo and ***Yoriito Kura*** Logo are logos of JASPAR Association.
Trademark registration for these marks is currently pending.

This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

Copyright

Copyright (C) 2026 JASPAR and the respective contributors

For more information regarding authorship of content, please consult the listed source code repository logs.

Declared Project Licenses

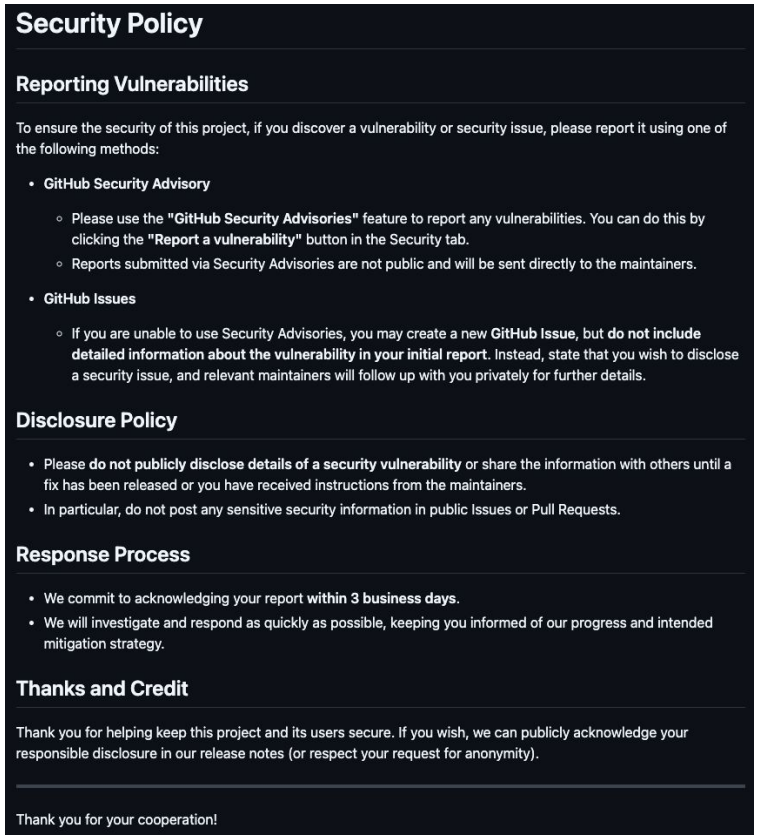
This program and the accompanying materials are made available under the terms of the Apache License Version 2.0 which is available at https://www.apache.org/licenses/LICENSE-2.0.

SPDX-License-Identifier: Apache-2.0

Cryptography

Content may contain encryption software. The country in which you are currently may have restrictions on the import, possession, use, and/or re-export to another country, of encryption software. BEFORE using any encryption software, please check the country's laws, regulations and policies concerning the import, possession, or use, and re-export of encryption software, to see if this is permitted.

Please support for security



Security Policy

Reporting Vulnerabilities

To ensure the security of this project, if you discover a vulnerability or security issue, please report it using one of the following methods:

- **GitHub Security Advisory**
  - Please use the **"GitHub Security Advisories"** feature to report any vulnerabilities. You can do this by clicking the **"Report a vulnerability"** button in the Security tab.
  - Reports submitted via Security Advisories are not public and will be sent directly to the maintainers.
- **GitHub Issues**
  - If you are unable to use Security Advisories, you may create a new **GitHub Issue**, but **do not include detailed information about the vulnerability in your initial report**. Instead, state that you wish to disclose a security issue, and relevant maintainers will follow up with you privately for further details.

Disclosure Policy

- Please **do not publicly disclose details of a security vulnerability** or share the information with others until a fix has been released or you have received instructions from the maintainers.
- In particular, do not post any sensitive security information in public Issues or Pull Requests.

Response Process

- We commit to acknowledging your report **within 3 business days**.
- We will investigate and respond as quickly as possible, keeping you informed of our progress and intended mitigation strategy.

Thanks and Credit

Thank you for helping keep this project and its users secure. If you wish, we can publicly acknowledge your responsible disclosure in our release notes (or respect your request for anonymity).

Thank you for your cooperation!

Issues can be created.
However, non-security issues may be deprioritized because this release is still a draft / candidate.

Quality Assessment

Checked below, refer to ***quality*** directory for more detail of each repository

Category	vehicle-api	kura	yoriito-tutorial
Unit Test	✅ passed	✅ passed	-
Integration Test	✅ passed	-	-
Coding Style	✅ passed	✅ passed	-
Coding Metrics	✅ passed	-	-
Static Analysis	✅ passed	-	-
Dynamic Analysis	-	-	✅ passed
Performance	-	-	✅ passed

For SCA (Risk Analysis), SBOM, and startup check, please refer to the quality assessment page of this repository.

Vehicle-api

vehicle-api v0.1.0

Signal API

All target items passed. See below for details.

Category	Result	Reference
Unit Test	All 17,067 normal and 23,205 abnormal test cases passed.	unit-test-results.zip
Integration Test	All 171 normal and 98 abnormal test cases passed.	integration-test-results.zip
Linter	Checked by cpplint. No critical issue found.	cpplint_summary
Code Metrics	Checked by lizard. No critical issue found.	lizard-result.zip
SAST	Checked by Coverity. No critical issue found.	coverity-result.zip

Logger

All target items passed. See below for details.

Category	Result	Reference
Unit Test	Successful DLT logging [build option: ENABLE_LOG_BACKEND_DLT = ON]	TestResult_dlt
	Successful stdout logging [build option: ENABLE_LOG_BACKEND_DLT = OFF]	TestResult_stdout
Linter	Checked by cpplint. No critical issue found.	cpplint_summary
Code Metrics	Checked by lizard. No critical issue found.	lizard-result.zip
SAST	Checked by Coverity. No critical issue found.	coverity-result.zip

yoriito

yoriito v0.1.0

All target items passed. See below for details.

Category	Result	Reference
SCA	Checked by Polaris. No critical issues found.	SCA_report
SBOM	Generated by Polaris. Copyleft licenses are limited to sample implementations only.	License_analysis_report
Startup check	Success	Startup log

kura

kura v0.1.0

All target items passed. See below for details.

Category	Result	Reference
Unit Test	All 184 test cases for Consumer IF passed.	Consumer IF
	All 231 test cases for Producer IF passed.	Producer IF
Linter	Fixed by cargo fmt and clippy.	cargo

Yoriito-tutorial

yoriito-tutorial v0.1.0

All target items passed. See below for details.

Category	Result	Reference
DAST	Checked by Sanitizer and no errors found.	[1]
Performance	CPU Usage: max 100% (All cores), Memory Usage: approx. 180 MB	[2]

[1]
Target device

- Dev container on WSL.

Precondition

- Running tutorial app with `sanitizer.mp4` in the assets folder.

[2]
Target device:

- Raspberry Pi 4B
  - Broadcom:BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC 1.5GHz
  - Memory 4GB

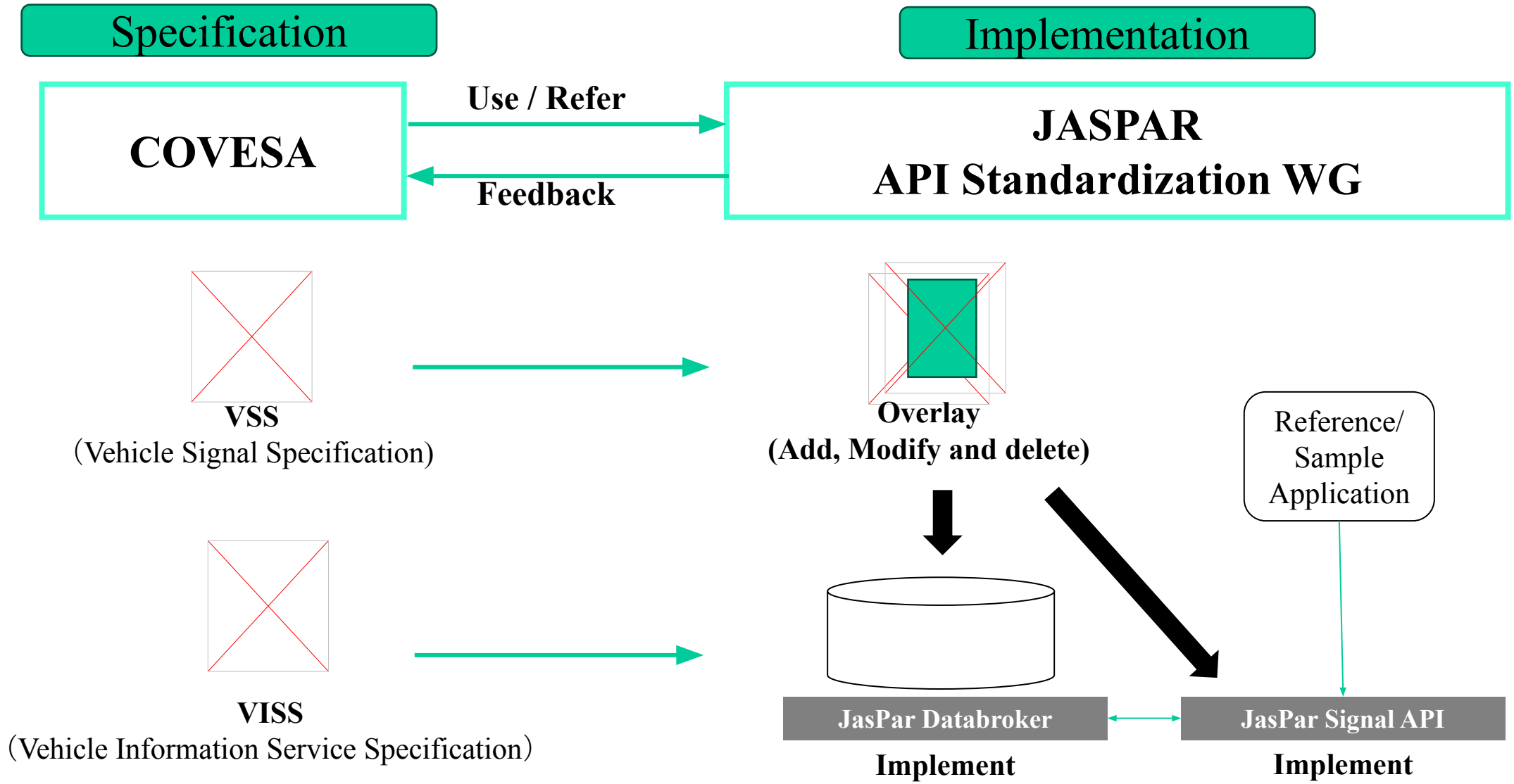
Preconditions:

- Using Top command.
- Running the tutorial app with `performance.mp4` in the assets folder.

Agenda

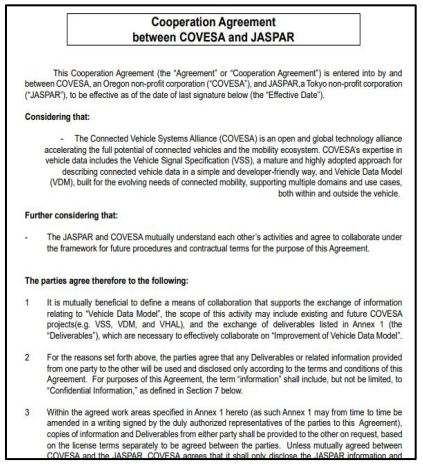
1. JASPAR Overview
2. API Standardization in JASPAR
3. Collaboration with COVESA
4. Conclusion

COVESA – JASPAR relationship



COVESA – JASPAR Collaboration

- Signed MoU in 2025 Dec
- Kicked-off Joint WG in 2026 March, started regular meetings



Cooperation Agreement between COVESA and JASPAR

This Cooperation Agreement (the "Agreement" or "Cooperation Agreement") is entered into by and between COVESA, an Oregon non-profit corporation ("COVESA"), and JASPAR,a Tokyo non-profit corporation ("JASPAR"), to be effective as of the date of last signature below (the "Effective Date").

Considering that:

- The Connected Vehicle Systems Alliance (COVESA) is an open and global technology alliance accelerating the full potential of connected vehicles and the mobility ecosystem. COVESA's expertise in vehicle data includes the Vehicle Signal Specification (VSS), a mature and highly adopted approach for describing connected vehicle data in a simple and developer-friendly way, and Vehicle Data Model (VDM), built for the evolving needs of connected mobility, supporting multiple domains and use cases, both within and outside the vehicle.

Further considering that:

- The JASPAR and COVESA mutually understand each other's activities and agree to collaborate under the framework for future procedures and contractual terms for the purpose of this Agreement.

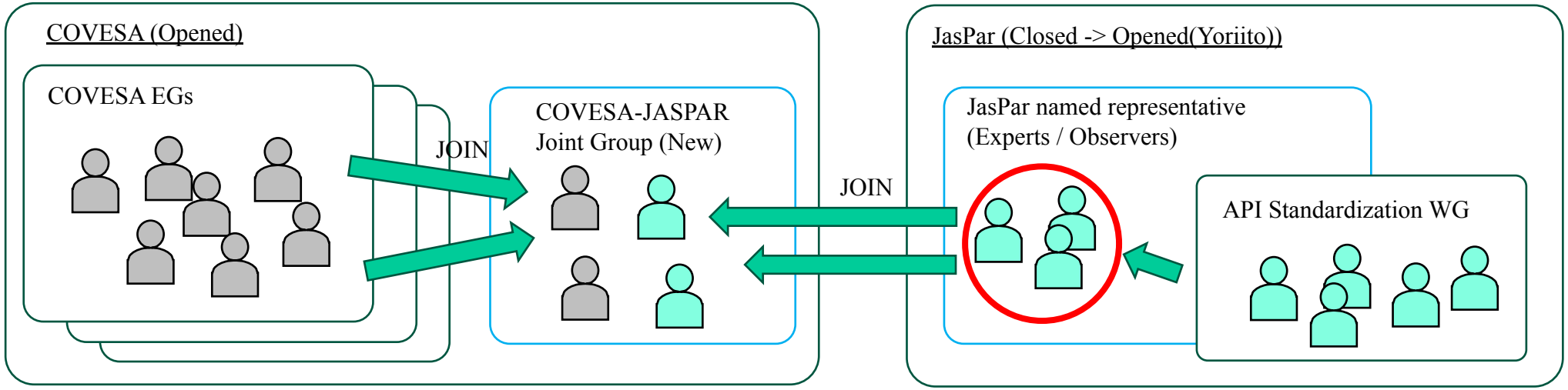
The parties agree therefore to the following:

1 It is mutually beneficial to define a means of collaboration that supports the exchange of information relating to "Vehicle Data Model", the scope of this activity may include existing and future COVESA projects(e.g. VSS, VDM, and VHAL), and the exchange of deliverables listed in Annex 1 (the "Deliverables"), which are necessary to effectively collaborate on "Improvement of Vehicle Data Model".

2 For the reasons set forth above, the parties agree that any Deliverables or related information provided from one party to the other will be used and disclosed only according to the terms and conditions of this Agreement. For purposes of this Agreement, the term "information" shall include, but not be limited, to "Confidential Information," as defined in Section 7 below.

3 Within the agreed work areas specified in Annex 1 hereto (as such Annex 1 may from time to time be amended in a writing signed by the duly authorized representatives of the parties to this Agreement), copies of information and Deliverables from either party shall be provided to the other on request, based on the license terms separately to be agreed between the parties. Unless mutually agreed between COVESA and the JASPAR, COVESA agrees that it shall only disclose the JASPAR information and

MoU



Combined efforts and complementary each other to create better, more widely adopted standard!

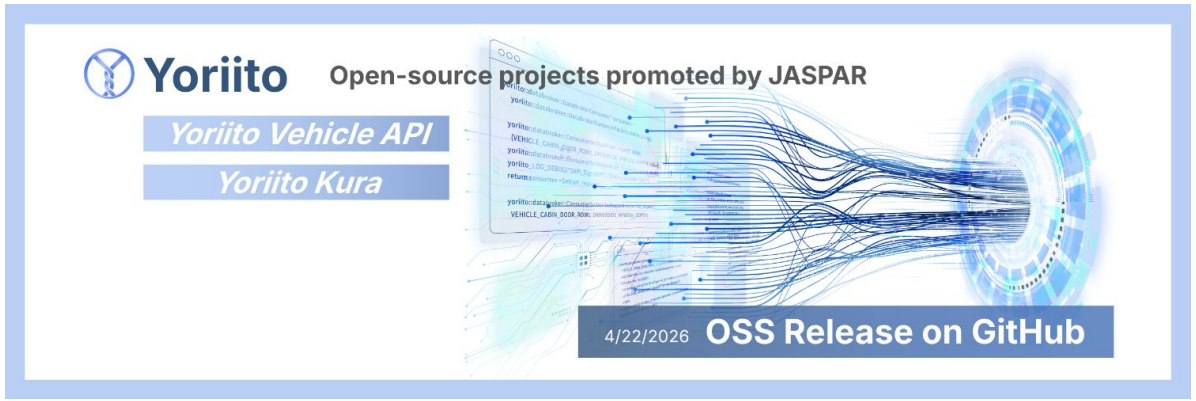
Agenda

1. JASPAR Overview
2. API Standardization in JASPAR
3. Collaboration with COVESA
4. Conclusion

Conclusion

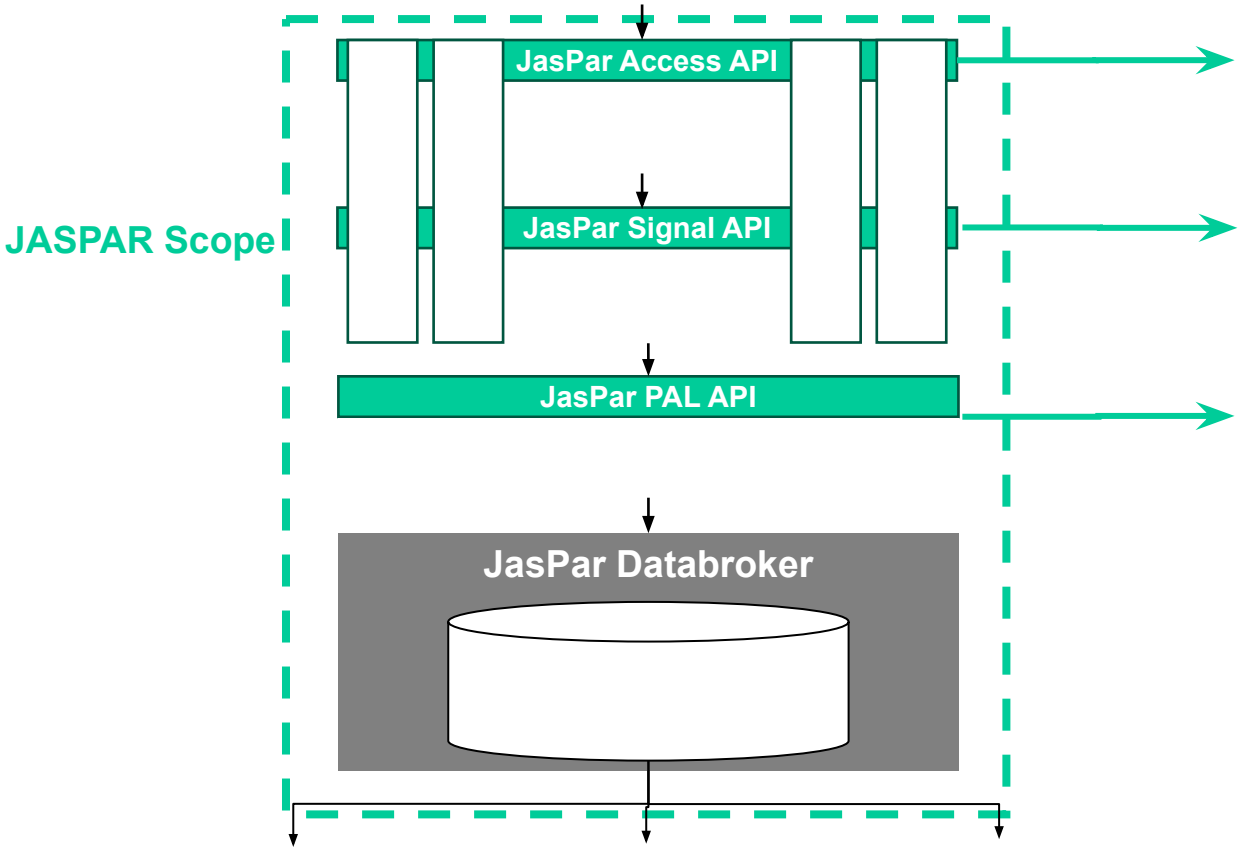
- JASPAR has started API WG activities involving Japan's leading automobile-related companies since 2025 Feb
- Started Signal APIs development (incl. databroker and development tools) based on COVESA VSS
- JASPAR has just derivered the first set of APIs in 2026 as OSS
- To promote broader adoption of the APIs, JASPAR is willing to collaborate with various open-source communities and standardization bodies.

27



Scan the QR code to visit JASPAR and Yoriito website

Architecture Overview



Access API (Under discussion)

- Abstracted APIs (Abstract data structure (vehicle signals) and provide more user-friendly APIs)

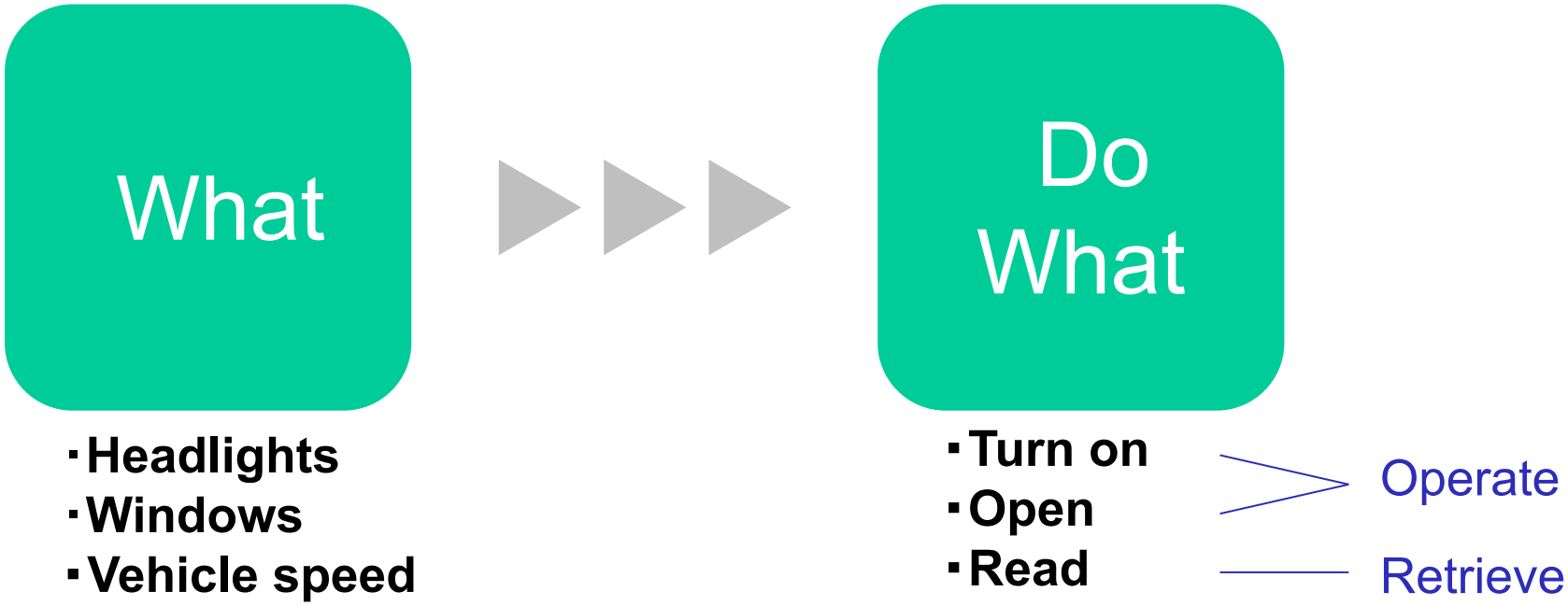
Signal API

- Access Vehicle Signals
- Vehicle Signals are created based on COVESA VSS (standardize signals for vehicle sensor/actuator)

PAL API

- PAL: Platform Abstraction Layer
- Abstract OEMs Middleware/OS

Basic Structure of the API



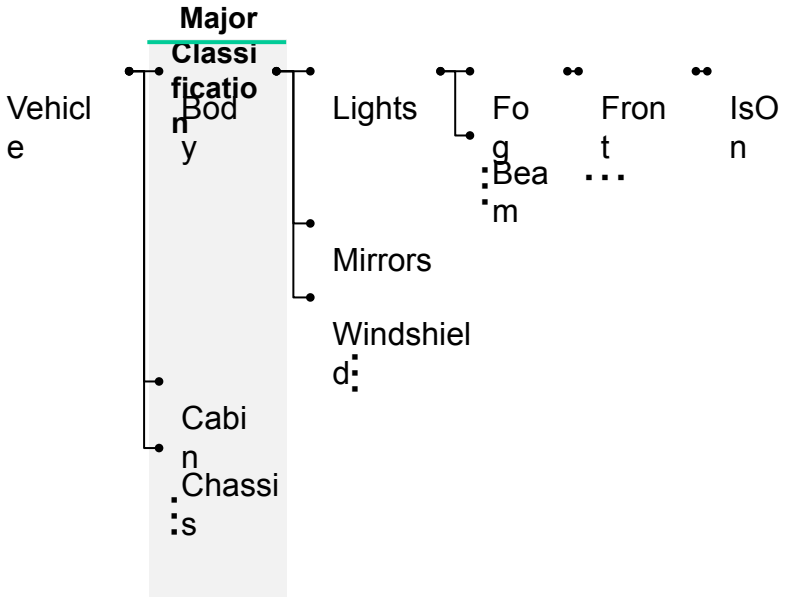
Vehicle equipment and functions vary widely
⇒ Clean up similar/duplicated items based on COVESA VSS
(e.g. defroster, de-icer, defogger...)

"**Operate**" and "**Retrieve**"

API's "What" ... Operate/Retrieve

- VSS[1] : Classifies vehicle equipment and functions with **tree structure** (e.g., Body::Light::Fog Light::Front::ON state)
  ⇒ Based on this VSS data structure, define APIs that can operate/retrieve the VSS data

VSS Classification (Tree Structure)



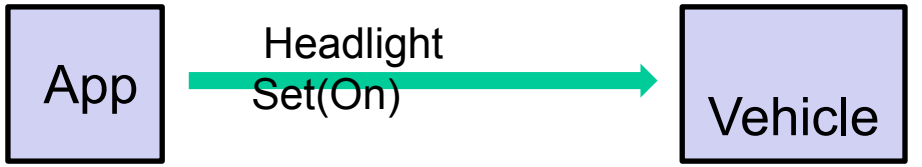
Major Category	Overview	Number of Subcategories
Body	Body shape, hood/trunk, horn, rain sensor, windows, lighting system, side mirrors, etc.	14
Cabin	Shades, Air Conditioning, Roof/Doors, Rearview Mirror, Interior Lights, Seats, etc.	30
Occupant	Occupant identification for each seat, head and eye position, etc.	3
LowVoltageBattery	(Auxiliary Battery) Nominal (fully charged) voltage/capacity, current voltage, current balance (current)	1
ADAS	Level, CC, LK, Obstacle Detection, ABS, TCS, ESC, EBD, DMS, etc.	11
Exterior	Outside temperature, humidity, light intensity, etc.	1
Driver	User identification information, distracted driving detection, hands-off detection, information receptivity, fatigue level, heart rate, etc.	2
Diagnostics	Retrieving active diagnostic trouble codes	1
CurrentLocation	Retrieve location information, timestamp, etc.	3
Trailer	Retrieve whether a trailer is connected	1
AngularVelocity	Acquire acceleration in the roll, pitch, and yaw directions of the vehicle	1
Acceleration	Acquire vehicle acceleration in forward/backward, left/right, and up/down directions	1
Connectivity	Determines network connection status with cloud / Availability of offboard functions	1
VehicleIdentification	VIN, WMI, brand, model, year, body type, number, manufacturing date, color, etc.	1
Service	Maintenance service requirement, distance/time to service (dealer?)	1
Chassis	Accelerator, brake, steering, parking brake, etc.	8
Powertrain	Engine/motor signals, fuel/battery signals, transmission signals, etc.	22

Vehicle Signal Specification:

https://covesa.github.io/vehicle_signal_specification/

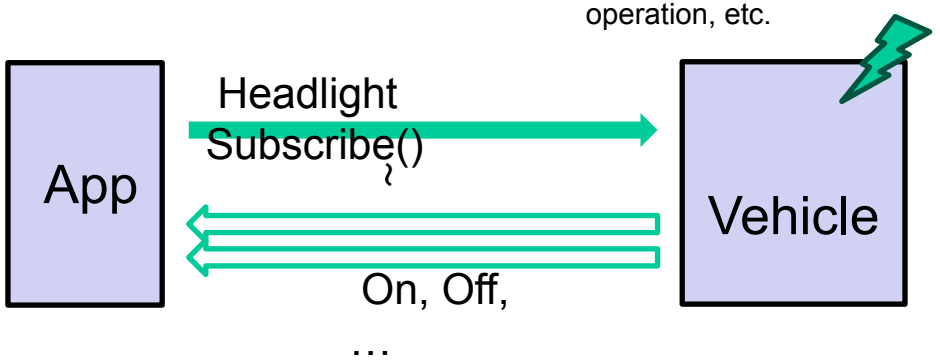
API "How to" ... Operations and Retrieval Methods

- Operate Method
  - Set
    (Application performs the operation
    at any desired time)



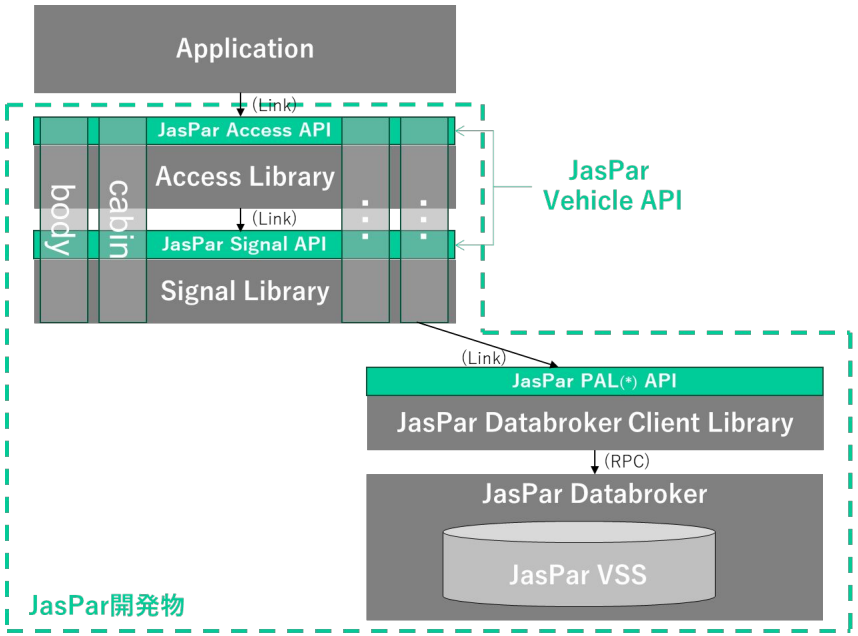
- Retrieval Method
  - Get
    (App retrieves data at any time)

  - Subscribe
    (Retrieve when the value changes)



API Hierarchy: JasPar Signal API and JasPar Access API

- To balance comprehensiveness and convenience, we define an Access API to enhance usability on top of the Signal API that utilizes VSS



Overall Architecture

✅ JasPar Access API (under detailed consideration)

Ex. TurnHeadLight(HeadLight::High)
...Turn headlights to high beam

- ✔ Application developer can directly calls the Access APIs
- ✔ Bundling Signal APIs to enhance usability

✅ JasPar Signal API

Ex. jaspar::vehicle::body::lights::beam::high::ison::Actuator::Set(true)

Adopting the VSS tree structure — VSS node types

... Turns on the headlight high beams

- ✔ Use same name and type as VSS so that developer can easily find API

API Standardization Approach

(1) Derivalables

- **API**
  - I/F Spec Documentation (incl. behavior definitions)
  - Source Code
  - API Evaluation Program
  - Sample Code
- **SW Development Process and Support Tools**
  - API Design & Development Process
  - SDK, Runtime, and API Management Infrastructure

(2) Scope of Standardization

- **From Interface to API Implementation**
  - Ensure the consistent behavior of APIs by standardizing the Middleware/HAL implementation.
  - Accelerate the development process by utilizing the software assets of participating companies.

(3) OSS

- **Release the Deliverables as OSS**
  - By making them OSS, we aim to ensure that our standardized APIs will be widely used.
  - Promote collaborations with other OSS initiatives (such as COVESA, AGL and Eclipse) and ecosystems.

Through these 3 perspectives, we will promote the standardization of APIs and address related topics

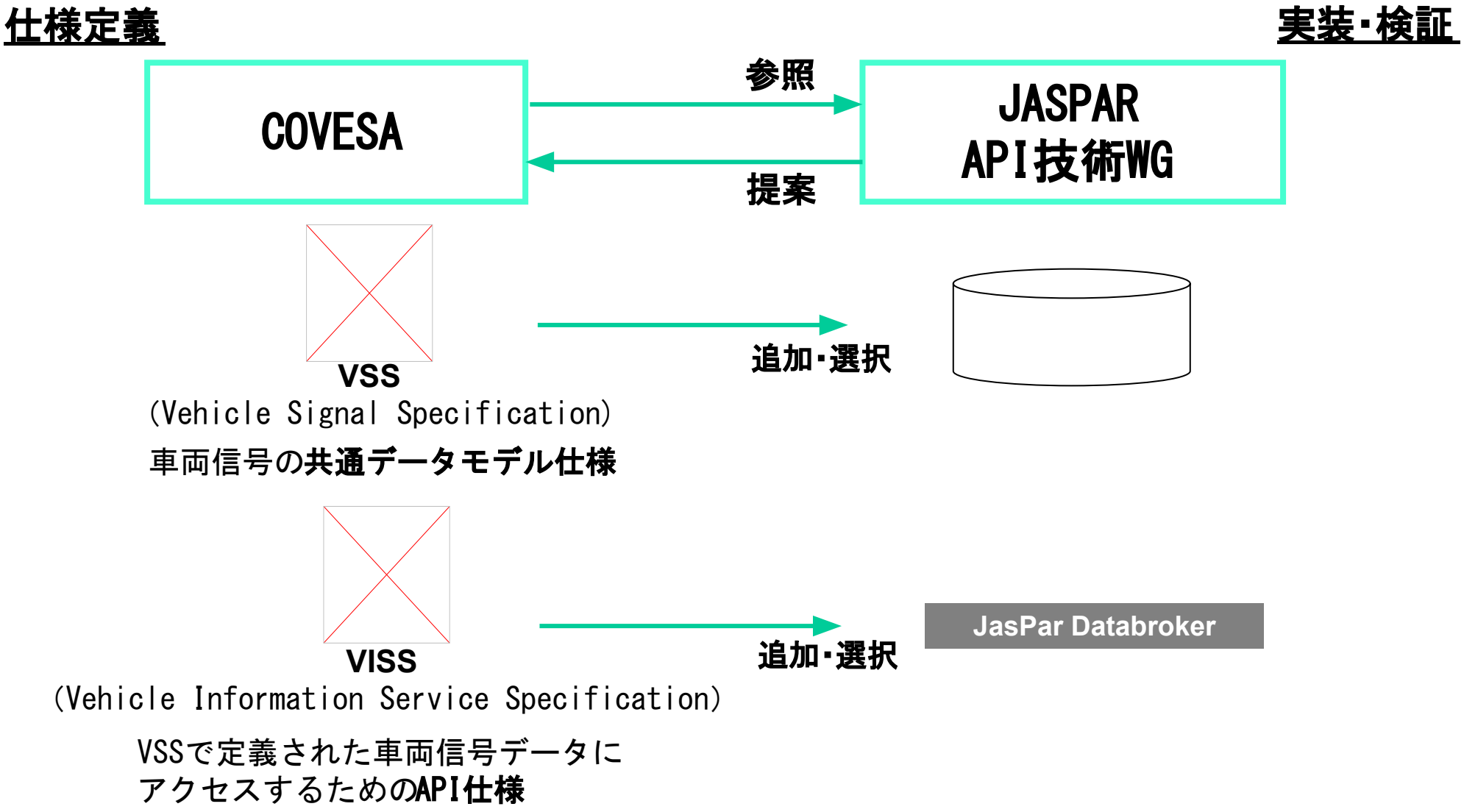
34

Roadmap

- In the MVP phase, establish minimal APIs and infrastructure
- For the Adoption phase, extend and promote the APIs so that many developer can use

	2025/04 – 2025/12	2026/01 – 2026/12
	MVP Phase	**Adoption Phase**
Activities	**Develop minimal APIs and basis for the development** • Develop primitive APIs for accessing vehicle data based on COVESA VSS. • Make design rules and development process. Set up development infrastructure.	**Develop SDK to promote API adoption**
		Make process to support API adoption (e.g. maintenance process, release process)
	API evaluation (PoC) • Evaluate the APIs through application development	**API improvement/extension** (e.g. support non-functional requirements, more abstracted APIs)
	OSS preparation • Make processes for OSS development	**OSS execution** • Release deliverables as OSS. • **Collaborate with other OSS community**
Release Plan		**●R1.0 ('26/xx)**

②API開発：COVESAとの関係

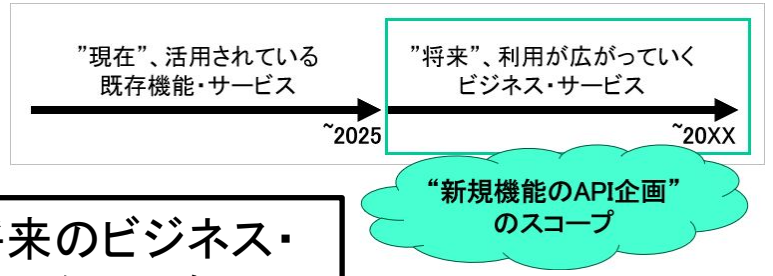


仕様策定はCOVESA、JASPARは利用、実装、Feedbackを行う

③API企画

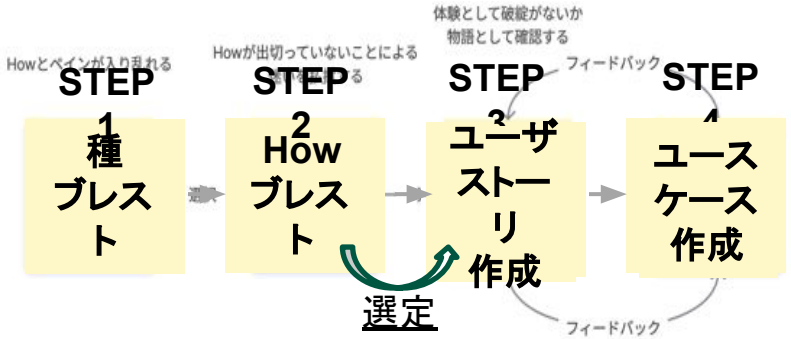
- **“将来”、利用が広がっていくビジネス・サービスに必要となりそうな　APIの探索**
- **「アイディエーション」** と **「サービス調査」** の2本立てで推進

API企画全体のスコープ



達成目標

クルマを使ったビジネス・サービス案から VSS発の検討で生まれない APIの探索

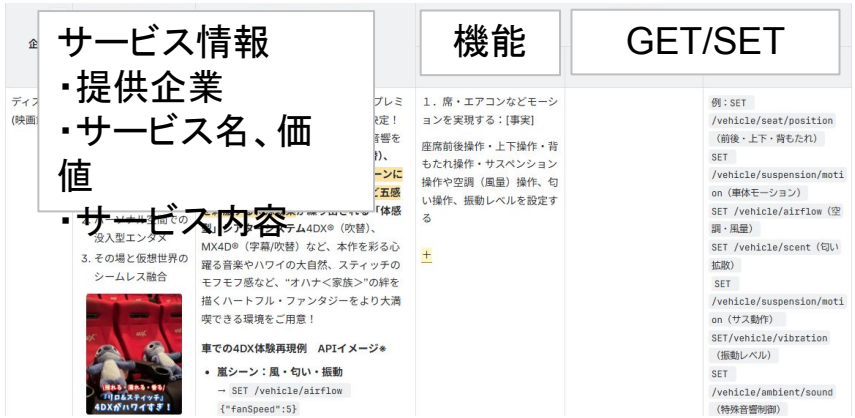


実現したい機能やユーザペイン、あったらいいなという自由な発想 でアイデアの種、Howをブレストにて創出

現状のJASPAR方針の「まずBody系」や「2030年前後のSOPへの採用を目標」等を意識し、**比較的 現実的なアイディアを選定**（選定しなかったものはアイディアバンクへ）

今回選定しなかったアイディア例
- ・現状、検知方式・センサが無い
- ・現状、車両機構・アクチュエータが無い
- ・Body以外 (ex. AD/ADAS系)
- ・標準化の対象から外れそう・・・ (競争領域案件)

JASPARの標準APIが将来のビジネス・サービス実現に不足ないか の確認



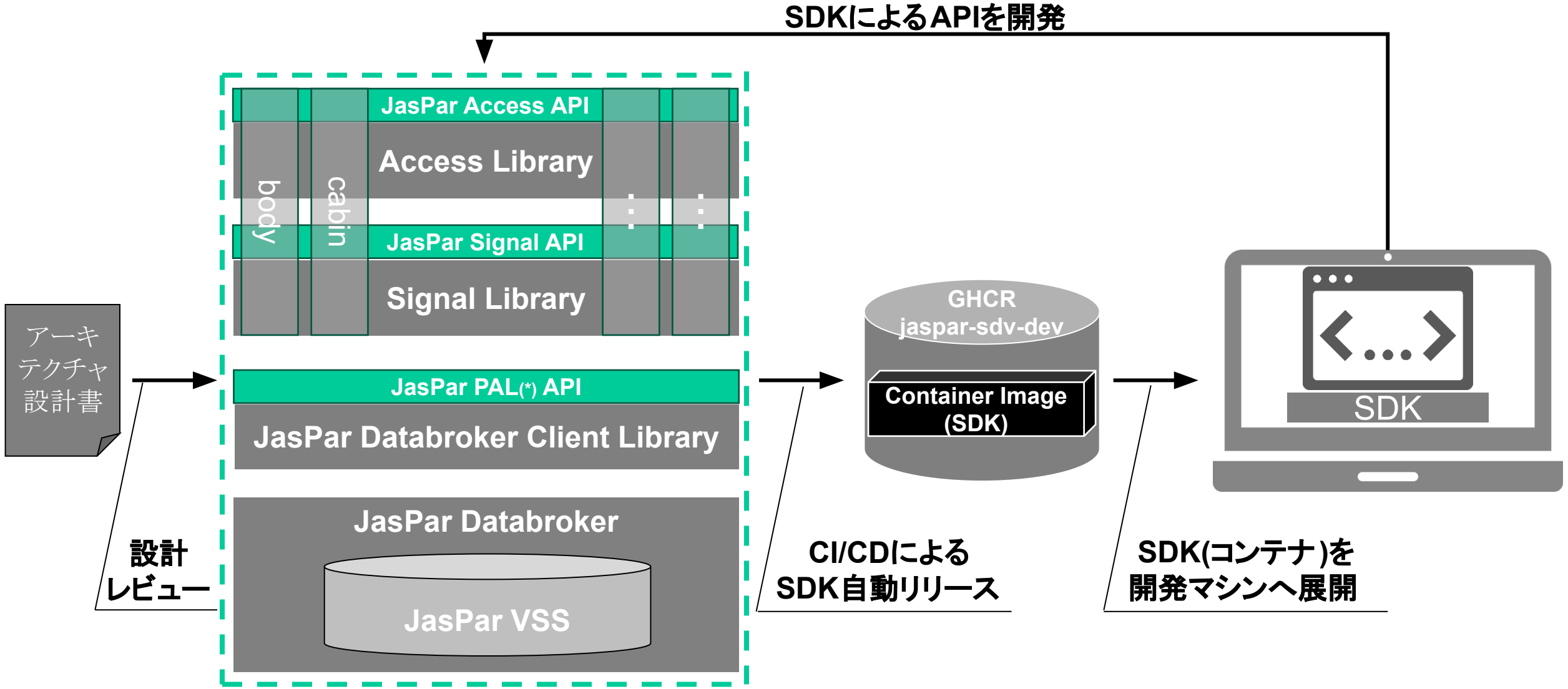
まずは、浅く広く調査を実施
⇒注力先を検討中

(本格調査はFY26実施予定)

あらゆる業界(自動車業界も含む)のサービスを調査中
- ・開発インフラ
- ・フリート/公共交通
- ・インフラ
- ・エンタメ
- ・データ/サーバー
- ・保険
- ・販売
- ・・・

どの業界を深掘りするのが良いか議論中

④開発環境：API開発の流れ



API開発者向けにSDKを作成して、効率よい開発を推進。 26年度には、API利用者向けを開発開始

2026年1月末までのアウトプット

COVESA VSSを基に、車両信号を網羅的に操作できるAPIを実装。また、利用者が”すぐ使える”ようにするため、チュートリアルや各種ドキュメントも整備した。

活動テーマ		実施内容	想定成果物	現状の実績、主なアウトプット
API企画		•車両機能網羅性を担保するため、 **COVESA VSSの信号をもとに車両機能を定義** •上記のVSS企画以外に、 **車両利用者ニーズに対応するAPIの調査、企画** を実施	•定義した車両信号一覧 •サービス調査結果 •ユーザーストーリー •ユースケース	•定義した車両信号数：631 •調査サービス数： 83件 •ユーザーストーリー： 23件 •ユースケース： 117件
API開発	設計実装	•定義された機能を基に**SignalAPIの設計および実装**	• Signal API	•**Signal API本数： 10,132 /10,132** •APIソースコード行数： 実装コード 375,773 / **テストコード 515,629**
	利用検証品質管理	•**API利用促進に向けたチュートリアル作成、 サンプルアプリ実装** •品質基準の定義、利用検証	•HeadlessApp/UIフレームワーク •チュートリアルドキュメント	•HeadlessAppのソースコード行数： 3,375 •UIのソースコード行数： 2,114 •チュートリアルドキュメント
アーキ開発環境		•アーキテクチャ（JASPAR Databroker）の設計・構築、開発ガイドラインの作成 •開発環境（SDK、シュミレーター、Runtime、CLI、CI/CD、各種ライブラリ）の検討	•JASPAR Databroker •各種ライブラリ（loggin機能やPAL等） •各種設計書・実装ガイドライン •開発環境の構想書	•JASPAR Databroker実装コード行数： 97,000 •各種ライブラリ •各種ドキュメント - 設計書 - 仕様書 - 実装ガイドライン

計画通り推進している

今後の活動

成果物は26年度内に公開を目指す。公開後は、OEMやTier1企業を中心にAPIの利用検証を進め、得られたFBを基に機能拡充や品質向上を図り、半年ごとに更新していく予定。

	API・関連ドキュメントの公開	API/基盤の拡張、ブラッシュアップ
目指す状態	APIを利用可能な状態で公開し、APIの存在を多くの企業や関係者に広く認知させ、利用していただく	OEMやTier1企業を中心にAPIの利用検証を進め、得られたFBを基に品質向上および機能拡充を図る
公開対象	公開対象 選定の考え方 利用者が迷わず使えるようにAPIのプログラムだけでなく、チュートリアル等のドキュメントも拡充させて公開する 公開成果物 • Signal API • Kura（DataBroker）/JASPAR VSS • 各種ライブラリ(loggong機能やPAL等) • HeadpessApp/UIフレームワーク(Flutter) • チュートリアル • 仕様書等のドキュメント	公開対象 選定の考え方 OEMやTier1企業を中心にAPIの利用検証を行い、アップデートしたものを公開する 公開成果物 • Signal API (アップデート版) • Kura（Data Broker）/JASPAR VSS(アップデート版)

詳細・追加情報は
3月4日JASPAR活動報告会にて報告