Open SDV Initiative

Open SDV API
全体概要とAPIコンセプト

バージョン：202503α
発行日：2025年3月31日

Open SDV Initiative

ドキュメントの位置付け

ドキュメントの目的

- このドキュメントは，Open SDV Initiativeで作成を進めている Open SDV API の全体概要とAPI設計のコンセプトを説明するものである

ドキュメントの完成度

- このドキュメントの現バージョンは完成度が低く，今後，大幅な改訂を行う可能性がある。また，仕様書の体裁も整っていない

変更履歴

バージョン	発行日	備考
202503α	2025年3月31日	初版

目次

OPEN SDV INITIATIVE の活動と現状

オープンSDVとOpen SDV Initiative

オープンSDVとは？

- サードパーティ(自動車メーカやその委託先以外の組織や個人)が開発したアプリケーションをインストールすることができる車両を，オープンSDVと呼ぶ

Open SDV Initiative の活動

- Open SDV Initiative は，近い将来にオープンSDVが実現・普及することを想定して，オープンSDVのためのビークルAPIとして “Open SDV API” を策定する
  - 策定したAPIは，標準化団体に提案する
- また，Open SDV API を評価するためのシミュレータの開発や実車を用いたPoCも行う

! 以下では，単にAPIと言えば，ビークルAPIを指すものとする

Open SDV Initiative の活動経緯

これまでの活動経緯

- 2024年6月20日：Open SDV Initiative の立ち上げを発表，参加企業の募集を開始
- 2024年10月：本格的な活動を開始
  - APIコンセプトWG，ボディWG，AD/ADAS WG，HMI WG，開発環境/シミュレーションWG，実車製作WG，調査WGを設置し，各分野の検討を開始。また，各WGのリーダからならリーダ会議を設置
  - 12月に，UXアイデア検討WGを追加設置
- 2025年3月：Open SDV API仕様の最初の版（バージョン：202503α）を公開
  - APIコンセプトWG，ボディWG，AD/ADAS WG，HMI WGのこの時点までの検討結果を取りまとめ

現時点での成果

Open SDV API仕様（バージョン：202503α）の完成度

- このバージョンは，4つのWGの2025年3月時点の検討成果を取りまとめたものである
  - 本格的な活動開始から半年しか経過しておらず，完成度は低い。4つのWGの検討成果間の不整合も残っており，今後，大幅な改訂を行う可能性もある
  - 知見のあるメンバがいなかったために，設置できなかったWGがあり，抜けも多い
  - 仕様の内容検討を優先したため，体裁も整っていない
- APIの策定が急がれる状況であり，他団体との協調のためにも，現時点の仕様を公表すべきと判断

残っている課題

検討できていないAPIの領域

- 以下の領域のAPIは，検討に着手できておらず，今後の検討課題として残っている（他にも抜けがある可能性）
  - バッテリー制御，エネルギー管理
  - ナビゲーションや自動運転に対するAPI（アプリケーション間連携）
- UXアイデア検討WGの成果の反映

仕様のテスト実装と評価

- シミュレータの開発とそれを用いた評価
- 実車を用いたPoC

特定の言語および（コンピュータの）OS向けの物理API

- 特定のプログラミング言語やOS上で Open SDV API を使用するための追加規程（バインディング）

API策定にあたっての考え方

基本的な考え方

APIの使用者と安全性

- サードパーティ（OEMやOEMから委託を受けたサプライヤ以外の組織や個人）とOEM（または委託先のサプライヤ）の両方が使うことを想定してAPIを策定する
- サードパーティに，危険な動作を起こす可能性があるAPIを使用させないために，OEMが，サードパーティに危険なAPIの使用を禁止するための機能（アクセス制御）を設ける
  - サードパーティ向けのAPIとOEM向けのAPIを別にするというアプローチは取らない
  - サードパーティが使用できるAPIを多くするために，策定するAPIは，可能な範囲で，ビークルOSにより危険を防止できるようなものとする

この考え方をとる理由

- あるAPIが危険であるかどうかは，車両の持つ安全機能によって異なる

  例）ウィンドウを閉めるAPIは，挟み込み防止機能を持つ車両では危険ではないが，持たない車両では危険である

- 現時点では危険なAPIが，技術の発展（車両の安全機能の進化）により，危険でないAPIになる可能性がある
- 危険なAPIをサードパーティに使わせて良いかどうかは，国や地域のレギュレーションによっても異なる

APIの策定にあたっての最重要視すべきこと

アプリケーションの開発容易性

- サードパーティのアプリケーション開発者にとっては，一度の開発で，多くの車両に適用できるアプリケーションが開発できることが重要
- APIで車両の違いを完全に隠蔽できなくても，アプリケーション側で車両の違いが吸収できれば良い
  - 例えば，車両によってウィンドウ（窓）の数が異なることに対しては，ウィンドウのリストをAPIを使って取得し，各ウィンドウに対して操作するAPIを呼び出す方法をとる
  - 車両の違いに依存せずに使えるAPI（例えば，すべてのウィンドウを操作するAPI）は，必要であれば，ライブラリ等で用意する
- 車両（主にハードウェア）の機能不足により，アプリケーションが動作できない場合があるのはやむを得ない

多様な車両に適用できるAPIとする

- 日系OEMの特徴／強みを活かせるAPIとするために，多様な車両に適用できるAPIとすることを目指す
  - 日系OEMの特徴/強み（の1つ）は，多様な車両を開発・販売していること
  - 多様な車両：パワートレインの違い，高級車／大衆車，乗用車以外の車両
- 車両の構成（configuration）の記述方法も策定する

策定するAPIの位置付け

どのようなアプリケーションを想定するか？

- 現時点で考えられるアプリケーションの例を後で示すが，想像しないようなキラーアプリケーションが出てくる可能性もあるため，汎用性・拡張性の高いAPIとすることを目指す
  - 見えているアプリケーションに特化したAPIとはしない

どのような車両を想定するか？

- まずは，一般の乗用車（オーナーカー，シェアリング用の車も含む）を想定する
- （可能な範囲で）サービスカー（バスやトラックなど）にも適用できるように配慮する

どのAPI(どの階層のAPI)を策定するか？

- Open SDV Initiative は，アプリケーションとビークルOSの間のインタフェースであるビークルAPIを策定対象とする
  - ビークルAPIには，次の2つの大きい目標がある
    - ✓車両によらず，同じアプリケーションが動作できる
    - ✓アプリケーションの開発が容易になる
  - この2つの目標が一致しない場合があることから，ビークルAPIを，前者の目標を達成するためのコアビークルAPIと，後者の目標を達成するための拡張ビークルAPIの2階層で構成する(詳しくは後述)
  - 実際には，想定するアプリケーションの作り方を検討し，APIを定義する階層を決めることが必要

どのAPI(どの階層のAPI)を策定するか？ – 続き

- 必要であれば，ビークルデバイスインタフェース(定義は後述)も策定するが，策定の優先度は低くする
- (コンピュータの)OSのAPIは，既存のもので十分な場合はそれを使えば良いため，策定スコープ外とする
  - 具体的には，AUTOSAR APやAndroid Automotiveが持っている機能のAPIは，それらで十分であれば，策定スコープ外とする
  - ただし，既存のAPIではビークルAPIとして不十分な場合には，この限りではない

APIの安全性に関する考え方

- ビークルAPIの中には，（対策をしないと）危険な動作を起こす可能性があるAPIが多数ある
- 策定するAPIは，可能な範囲で，ビークルOS（特にコアビークスサービス）により危険を防止できるようなものとする
- APIにより危険な動作を起こす場合に，ビークルOSは，それを防止する機能を持っていることが望ましい（必須ではない）
- ビークルOSで危険を防止できないようなAPIも残る
  - 同じAPI呼び出しに対しても，危険を防止できる車両と，できない車両がある

APIのアクセス制御

- どのAPIを実装するか，どのアプリケーションにどのAPIの使用を許すかは，OEMの判断
  - ただし，プライバシ情報に関わるAPIの使用を許すかは，ユーザに判断させる（スマホと同様）
- 安全性の確保，プライバシ保護，セキュリティ向上のために，APIのアクセス制御を導入する
- 安全性に関わるAPIを使用するサードパーティ製のアプリケーションは，安全性の審査を受けることが必要
  - 審査する安全性の例：危険な制御を行わないこと，ドライバデストラクションがないこと，など
  - 審査の結果として，アプリケーションに使用を許可するサービスコールに関する情報（これを「アクセス制御情報」と呼ぶ）を作成する

策定するAPIは，どのECUで使用するものか？（アプリケーションはどのECUで動作するか？）

- 基本方針：車両のE/Eアーキテクチャによらず使用できるAPIとする
- E/Eアーキテクチャ（ECUの構成）は車両によって異なるため，APIを使用するECUは限定しない
  - 下位層に分散プラットフォーム（AUTOSAR APやROSなど）があることを想定すれば，別のECUで実現されたサービスを呼び出すことは可能
  - 実際には，IVI ECUやADAS/AD ECUで使用すると思われる
  - アプリケーションによって異なるECUで動作させることや，1つのアプリケーションを複数のECUに分散して動作させることも可能

策定するのは，論理APIか物理APIか？

- まずは論理API（セマンティクスレベルのAPI）を策定する
  - 物理API（シンタックスレベルのAPI）は，プログラミング言語や（コンピュータの）OSに依存する。変わったとしても，アプリケーションの作り直しは機械的
  - ただし，APIを介して受け渡しされるデータの型については，物理レベル（具体的には，ビット長や単位）に踏み込んで規定する
    - ✓ビット長や単位（特に単位）が異なると，アプリケーションの修正が大きくなるため
- シミュレータやPoCの開発にあたっては，物理APIも定める

クラウド上のAPI（クラウド上やスマホ上のアプリケーションから，クラウドを介して車両の機能を操作するためのAPI）は策定しないのか？

- クラウド上のAPIも重要な課題であると認識している
- 車両の機能を操作するAPIは，車両内で使用するAPIもクラウド上のAPIも，以下を除いては大きい違いはないと考えられるため，まずは車両上のAPIにフォーカスする
  - クラウドには，車両に関する過去のデータやそれを集計したものを蓄積するため，クラウド上では，それらの蓄積データにアクセスAPIが必要になる可能性がある
- クラウド上のアプリケーションは，車両上のアプリケーションを介して，車両を操作する（＝車両上のAPIを使う）想定

アプリケーションの例

- 以下に示すのは，現時点で思いついているアプリケーションの例であり，想像しないようなキラーアプリケーションが出てくることを期待したい

ボディ系

- セントリーモード，洗車モード

自動運転系

- 自動運転（例：テスラのFull Self-Driving）
- 自動駐車，バレーパーキング

HMI系

- メーターのパーソナライズ

情報提供系

- ナビゲーション
- POI情報提供，旅行情報提供

エンターテイメント系

- ドライブ記録動画/アルバム作成
- 車両を活用したゲーム
- 自動運転中の車内エンターテイメント

プローブ系

- ドライブレコーダ
- テレマティクス保険，安全運転診断，運転能力診断
- リアルタイムストリートビュー
- 地図作成支援

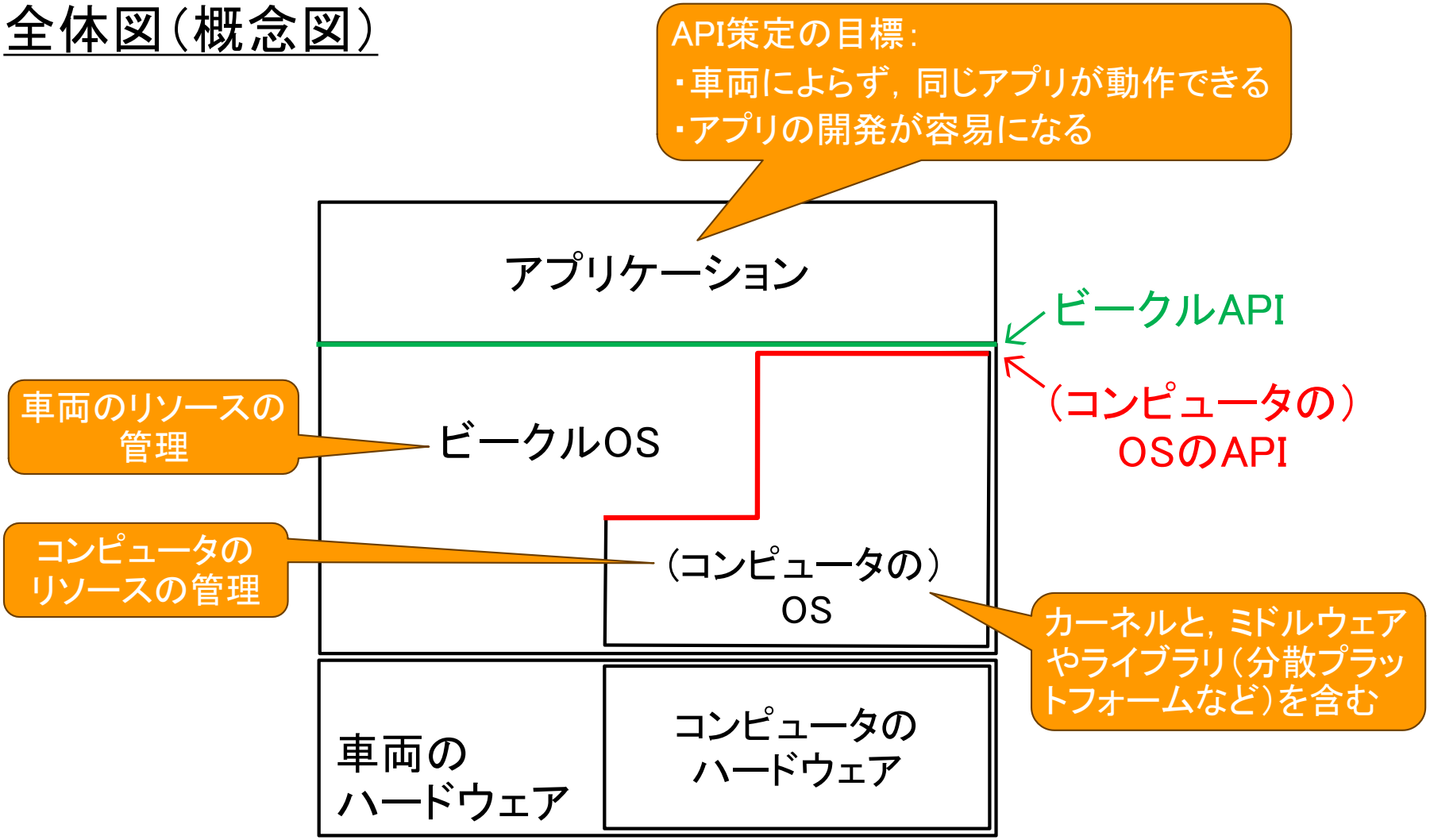
その他

- シェアリングカーの管理
- V2H（Vehicle to Home），V2G（Vehicle to Grid）

想定するシステム階層

想定するシステム階層（概念図）

全体図（概念図）

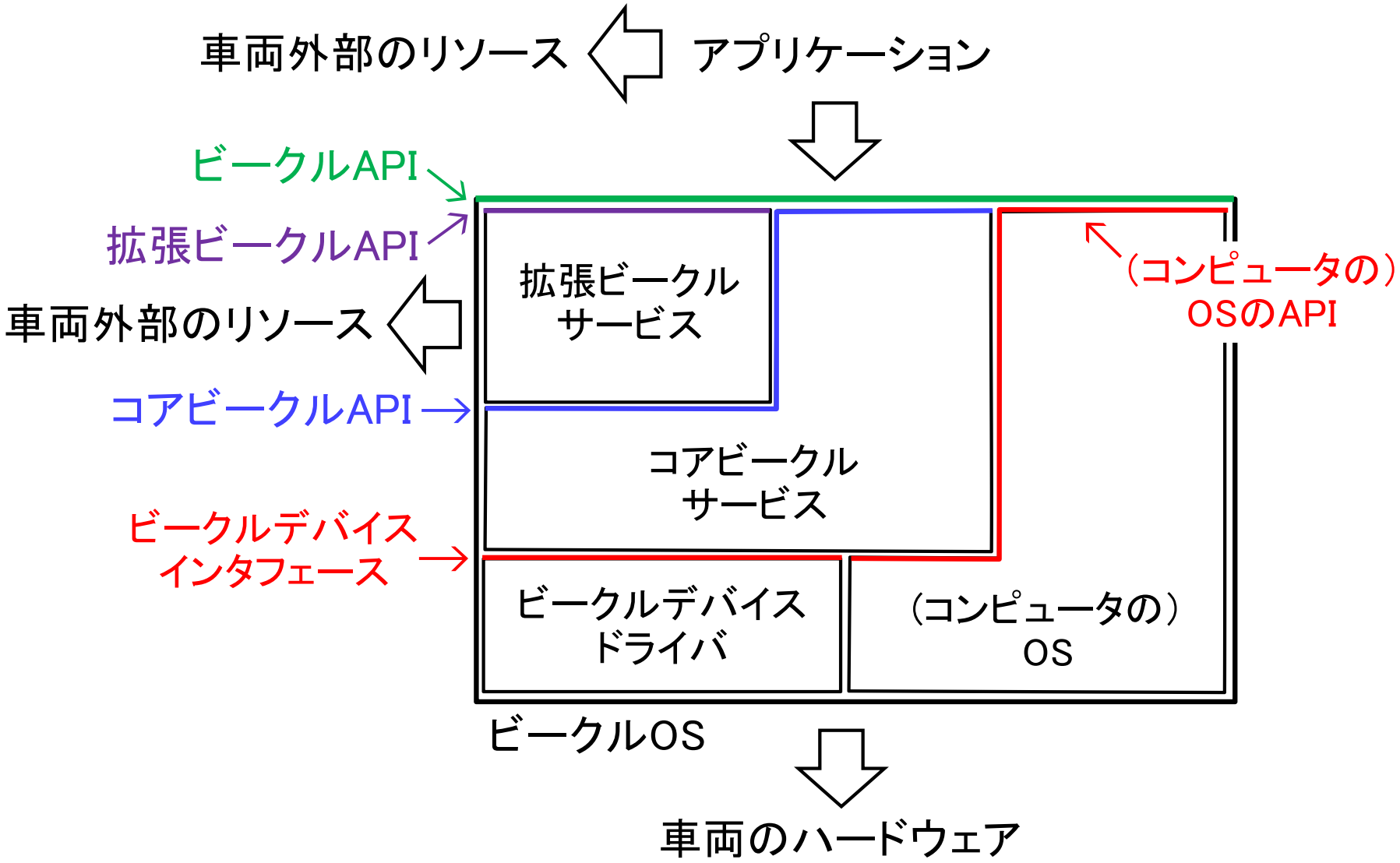


※ 実装構造を表現する図ではない

用語に関する補足説明

- ここでは，OSという用語を，ソフトウェアプラットフォーム（SPF）と同義で用いている（広義のOS）
- 「ビークルOS」は，「（コンピュータの）OS」も含む
  - コンピュータ（のハードウェア）も車両（のハードウェア）の一部分であるため
- 「（コンピュータの）OS」には，カーネルに加えて，ミドルウェアやライブラリを含んでいる
  - 分散プラットフォーム（AUTOSAR APのARA::COMやROSなど）を含んでいることを想定
  - AUTOSAR APのPlatform Foundation FCsに含まれるサービスは，ここに含まれるものと位置付ける

ビークルOS内部の構造（概念図）



コアビークルサービス

- 以下の3つの役割を持つソフトウェア
  - 車両のハードウェアの違いの吸収
  - 車両のハードウェアへのアクセスの調停
  - (可能な範囲での)安全性の確保
- 上の3つの役割を果たせる範囲で，コアビークルサービスの機能は最小限にする
  - 車両のハードウェア以外のリソースを管理する機能は，原則として持たせない

コアビークルAPI

- 上位階層のソフトウェアが，コアビークルサービスを使用するためのインタフェース
- 車両によらず同じアプリケーションを動作させるために鍵となるAPIであり，標準化・継続性が重要

拡張ビークルサービス

- アプリケーションの開発を容易化するために，コアビークルサービス上に実装されるソフトウェア
  - 車両のハードウェア以外のリソースを管理する機能は，拡張ビークルサービスで実現する
  - 例えば，地図を管理する機能は，拡張ビークルサービスで実現する。ダイナミックマップは，拡張ビークルサービスの位置付けとなる
- アプリケーションが拡張ビークルサービスの位置付けとなる場合もある
  - 具体的には，アプリケーションが他のアプリケーションから要求を受け付ける場合
  - 例えば，自動運転ソフトウェアやナビゲーションソフトウェアが典型例

拡張ビークルAPI

- 上位階層のソフトウェアが，拡張ビークルサービスを使用するためのインタフェース
- 類似の機能を持つ拡張ビークルサービスが，複数あっても良い
  - 車両外部のリソースを管理する機能に対して，1つのAPIに標準化することは難しいため
- アプリケーション開発者の視点で使いやすいものとすることが重要

ビークルサービス

- コアビークルサービスと拡張ビークルサービスを総称して，ビークルサービスと呼ぶ

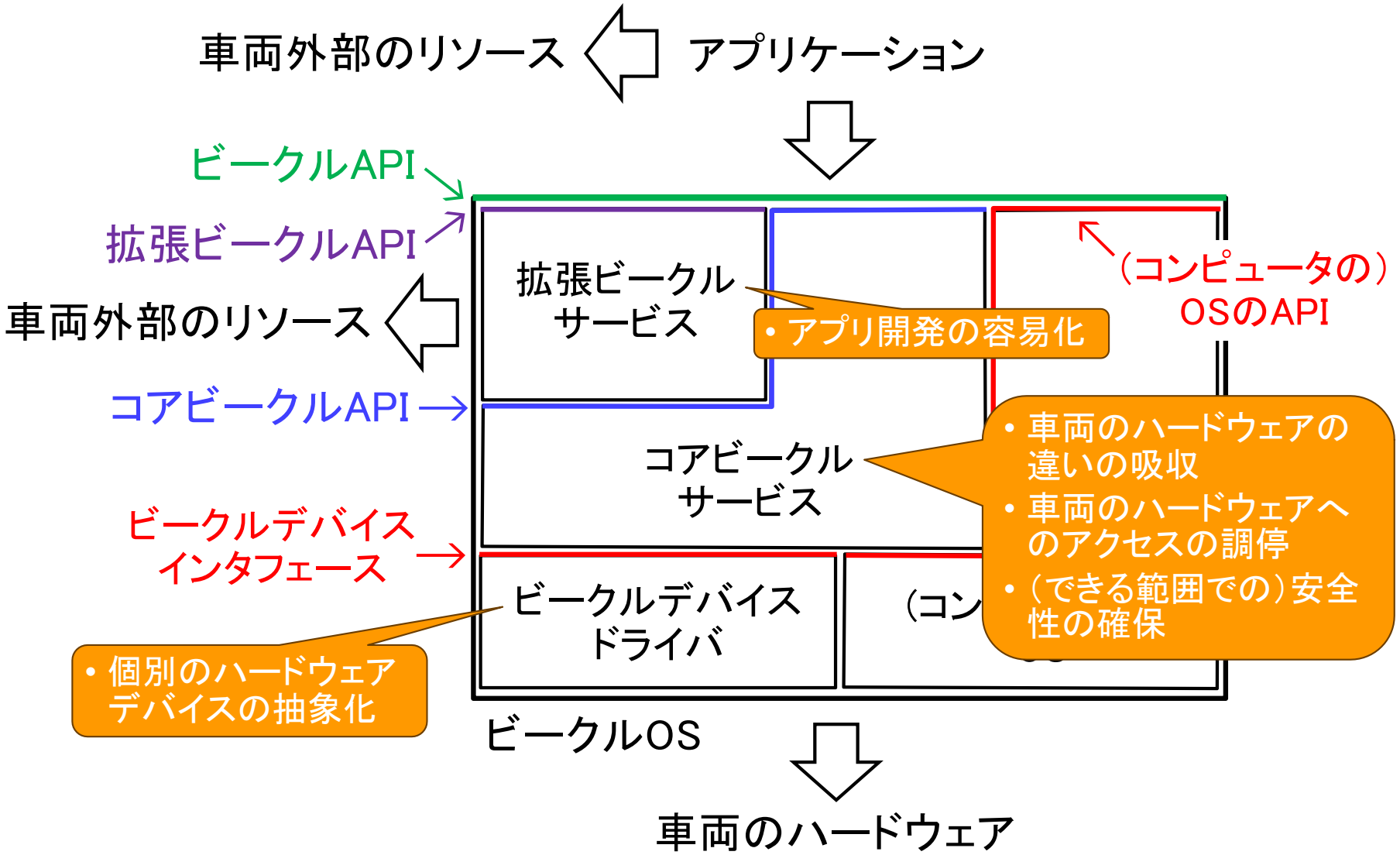
ビークルAPI

- コアビークルAPI，拡張ビークルAPI，（コンピュータの）OSのAPIで構成される

ビークルデバイスドライバ

- 車両のハードウェアデバイスを，それぞれ単独で抽象化するための階層
- ただし，何を1つのハードウェアデバイスと扱うか，どの程度抽象化するかは，一通りには定まらない
  - 例えば，HEVにおいて，「パワートレイン」を1つで扱う方法と，「エンジン」「モータ」「トランスミッション」を別々のハードウェアデバイスと扱う方法がある
  - 例えば，カメラに対する抽象化層は，カメラの生画像を提供する方法と，カメラ画像を認識した結果を提供する方法がある。なお，複数センサーの認識結果のフュージョン処理は，上位層の役割

ビークルOS内部の構造（役割注釈付き）



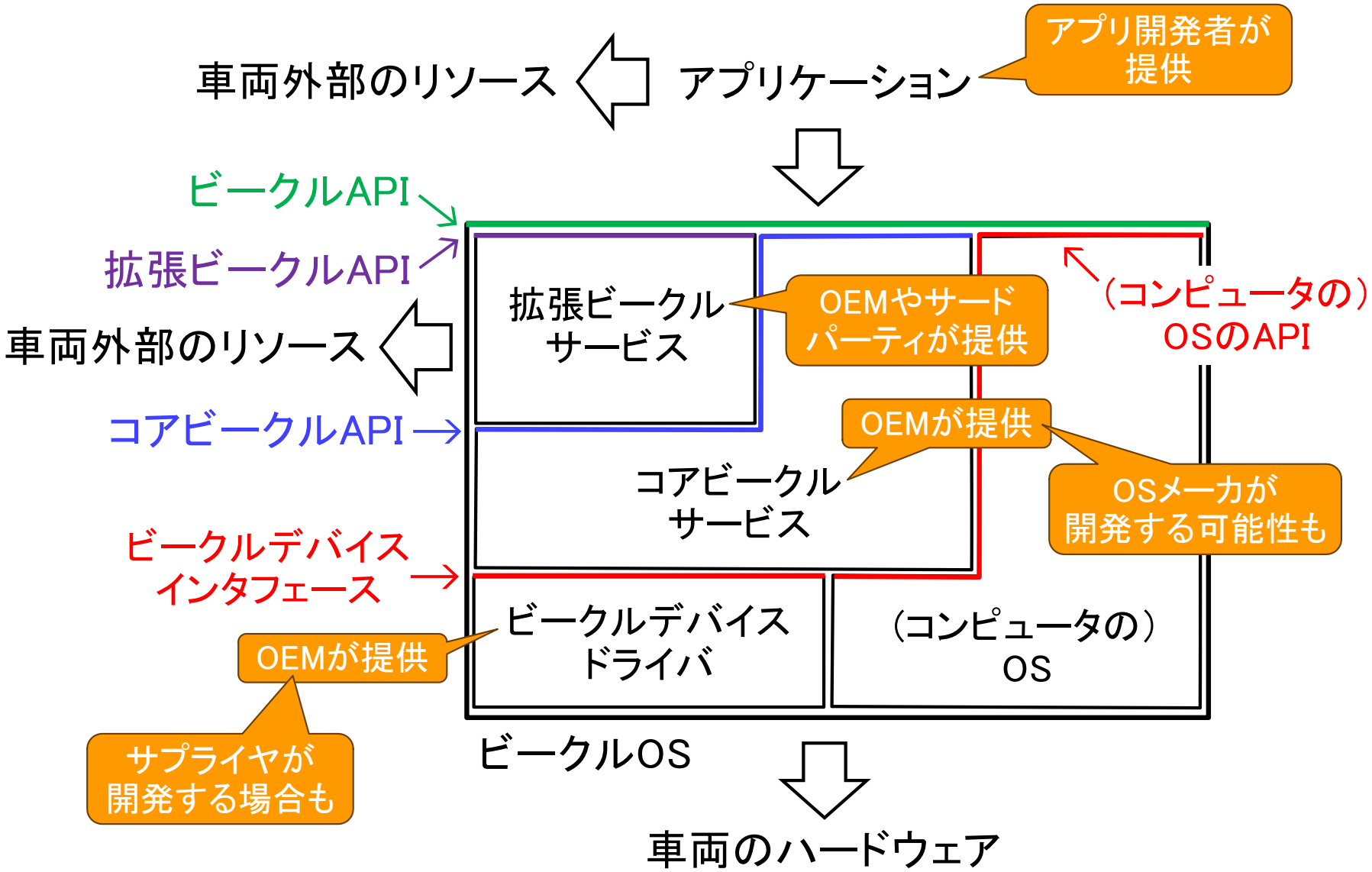
ソフトウェアのモジュール化の意義と階層分割の方針

- ソフトウェアのモジュール化（階層分割を含む）の目標の1つは，ハードウェアや周辺環境（交通規則，外部との通信仕様）の変更があった場合に，1つのモジュールのみの変更で対応できるようにすることである
  - 車両のハードウェアに変更があった場合には，ビークルOSのみの変更で対応できるのが望ましい
  - 交通規則（国・地域によって異なる），外部との通信仕様，地図の仕様の違いに対しては，1つの拡張ビークルサービスの差し替えのみで対応できるのが望ましい
- この目標に近づくように，階層分割を行う方針とする

各モジュールの提供者

- コアビークルサービス
  - OEMが提供
  - 開発は，OSメーカが行う（OSメーカが開発してOEMに提供する）可能性や，OSSを用いる（OEMがOSSを利用する）可能性もある
- 拡張ビークルサービス
  - OEMが提供する場合もあれば，サードパーティが提供する場合もある
  - OEM，サプライヤ，OSメーカ，アプリケーション開発者の誰が開発する可能性もある
- ビークルデバイスドライバ
  - OEMが提供
  - 開発は，サプライヤが担当することも多いと思われる

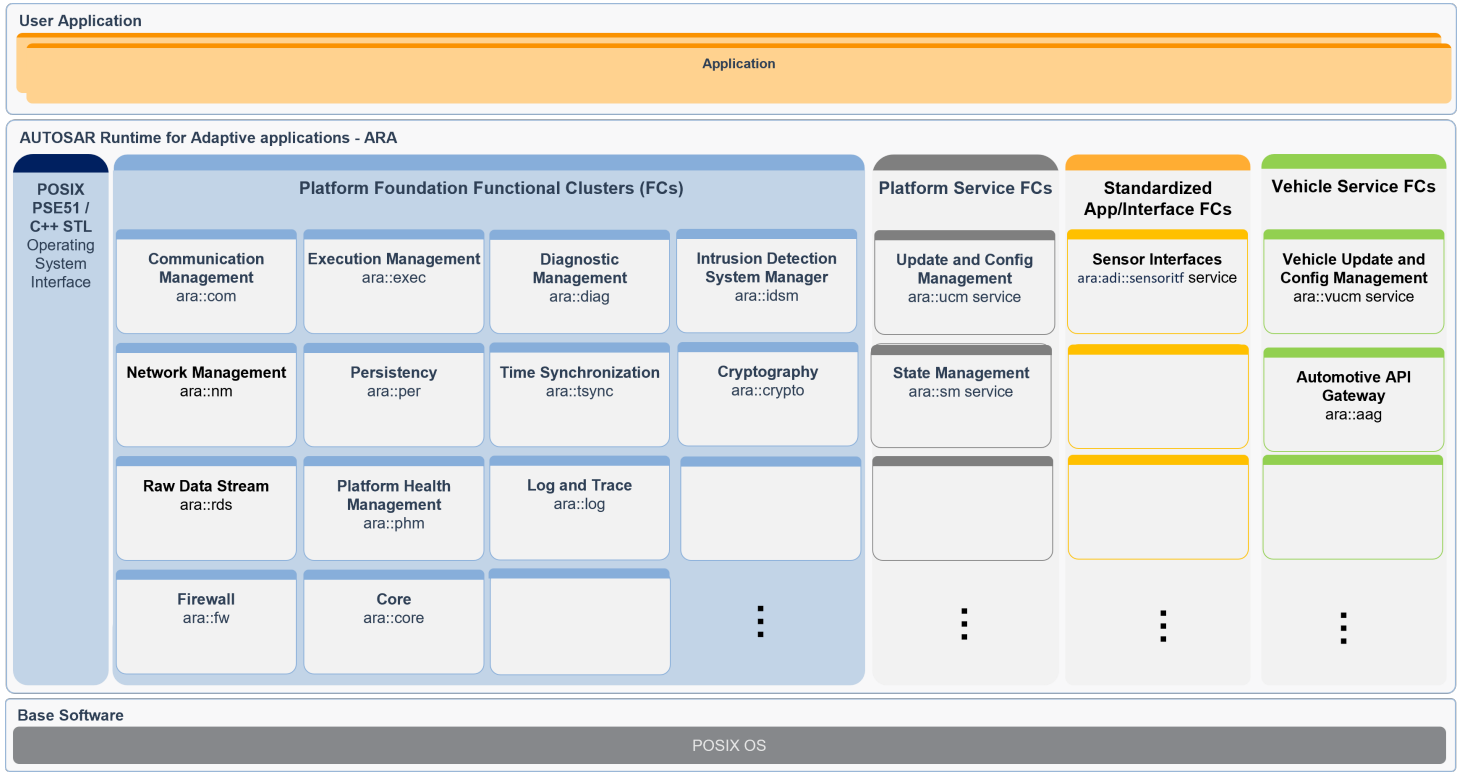
ビークルOS内部の構造（提供者注釈付き）



実装構造の例

AUTOSAR AP上に実装する場合

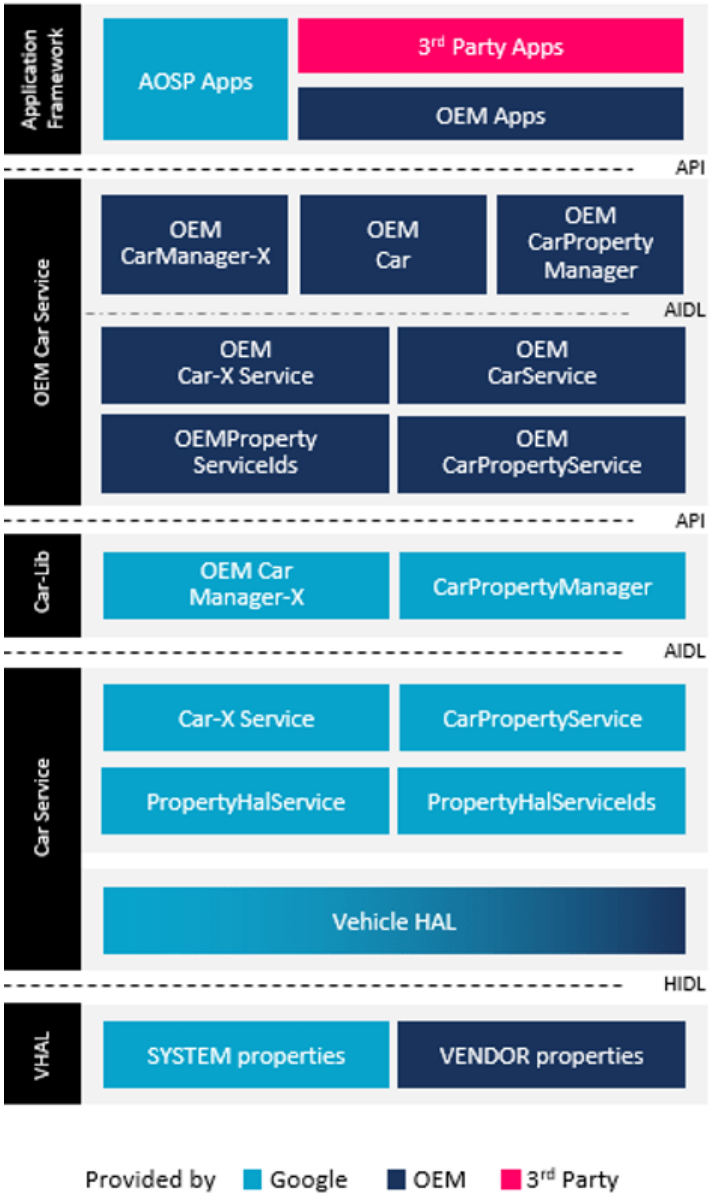
- 現在のAUTOSAR APの構成図（下図）では，User ApplicationとARAの間にService層を設け，ビークルサービスを，Service層で実現することになる



※ 図は，“Explanation of Adaptive Platform Design (R24-11)”より

Android Automotive上に実装する場合

- Android Automotive の構成図（右図）では，ビークルサービスを，“OEM Car Service” と “Vehicle HAL” で実現することになる



※ 図は，“Android Automotive OS Whitepaper”より

APIに関する用語・概念

用語の定義

オブジェクト

- ビークルOSが操作対象とするリソース(車両のデバイスや外部リソース)を，ビークルオブジェクト，または単にオブジェクトと呼ぶ
  - 何をオブジェクトとするかは，個別に検討する
- 同一種類のオブジェクトが複数ある場合には，必要に応じて，オブジェクトクラス(または単にクラス)，オブジェクトインスタンス(または単にインスタンス)と呼んで区別する
- COVESAでは，“item” という用語が使われており，その方が良いかもしれない

構成（configuration）と状態（status）

- オブジェクトの属性の中で，静的に決まるものを「構成（configuration）」，動的に変化するものを「状態（status）」と呼ぶ
- 両者の中間的な属性もあるため，「構成」が変わる場合もあるものとする
  - 例）トレーラーが牽引しているもの
  - 依存している「構成」が変わった場合には，アプリケーションの再起動が必要となっても良い

状態とイベント

- オブジェクトは，（一般に）状態を持つ
- オブジェクトの状態の（離散的な）変化のことを，イベントと呼ぶ
  - イベントには，状態変化の原因などに関する情報が付属している場合がある

標準制御

- OEMが自動車の出荷時に組み込んでいる制御ロジックを，標準制御と呼ぶ
  - 標準制御は，アプリケーション，ビークルOS，ハードウェアのいずれで実現されている場合もある
  - 標準制御の動作を止めることができるのを原則とするが，止められないものがあっても良い（ハードウェアで実現されている場合など）

車両の構成記述法

車両の構成記述の目的

- 車両の構成（configuration）記述は，次の2つを表現する
  - 車両の構成（車両の持つ装備や性能など）
  - APIで扱うオブジェクト

構成記述の策定方針

- 車両の構成を木構造で表現する
  - COVESA VSSをベースに，DVE+Uモデル[1]も参考に検討した結果，VSSとかなり違ったものとなっている
  - TODO：VSSに変更を加えた理由を明確にする
- 木構造の中間ノードが，APIの対象となるオブジェクトに対応する

[1] M. Gondo: “(The) Four Principles of SDV”, 15th AUTOSAR OPEN CONFERENCE, 2025年6月.

構成記述に対する要求・留意点

- 車両の構造・機能を，トップダウンに整理する
  - 例えば，洗車モードに遷移するというアプリケーションを作成する場合，水が入る可能性がある開口部をすべて数え挙げられることが必要
  - そのためには，キャビンの開口部に何があるか（例えば，ドア，ウィンドウ，ガラス）を整理し，各開口部の性質（例えば，ウィンドウは閉めれば水を通さない）を規定する必要がある
- 車両依存の拡張を考慮した記述方法とする
  - 車両の多様性を考えると，網羅的に規定するのは難しく，車両依存で拡張できることが必要なため
- 木構造の階層は，あまり深くならないようにする

車両の構成情報の取得

- 車両の構成記述ファイル
  - アプリケーション開発者は，ターゲットとする車両群の構成記述ファイルを参考する
  - 複数の車両に対応できるビークルOSは，構成記述ファイルを参照して動作を変える
  - アプリケーションが構成記述ファイルを参照しても良い
  - シミュレータや各種ツールが，構成記述ファイルを利用することも考えられる
  - TODO：構成記述ファイルの形式を定める
- 車両の構成を取得するためのサービスコール
  - 各オブジェクトに対して，そのオブジェクトのすべてのインスタンスの構成情報を返すgetConfig サービスコールを設ける

トップレベル

- VSSと同様に，Vehicle とする
  - DVE+Uモデルをベースに考えると，トップレベルを Vehicle，Driver，Environment の3つとする考え方もできるが，Driver は車両の運転者であること，Environment も車両の周辺環境であることを考えて，COVESA VSS と同様に，トップレベルは Vehicle だけとする

第2レベル

- DVE+Uモデルでの Vehicle は，VSS をベースに，階層をあまり深くしない，Vehicle 直下にオブジェクトを置かないなどの方針から，以下のようにする
  - Cabin … 乗員が乗る空間
  - CargoSpace … 荷物を乗せる空間（物流を考えて独立させる，VSSにはない）
  - Body … 主に車両の外側にある装備
  - Infotainment … 文字通り（VSSでは Cabin の下にある）
  - HMI … 文字通り（VSSでは Cabin.Infotainment の下にある）
  - Motion … 車両の運動制御（VSSにはない）
  - CurrentLocation … 車両の現在位置
  - Specification … 車両諸元（主に静的な情報，VSSにはない）

第2レベル – 続き

- Vehicle 関係の以下の第2レベルについては，十分に検討できておらず，現時点ではVSSのままとする
  - Powertrain … 情報を取るだけで制御はしない
  - Chassis … 情報を取るだけで制御はしない
  - ADAS … 現時点ではVSSのままとする
  - VehicleIdentification … 現時点ではVSSのままとする
- VSSでVehicle直下にある車両の移動に関する属性は Motion の下に，その他の車両諸元は Specification の下に置く

- DVE+Uモデルでの Driver は，運転者以外の乗員も含んでいるため，VSSと同様，次の2つを第2レベルとする
  - Driver … 運転者（遠隔運転の場合は，車内にいるとは限らない）
  - Occupant … 運転者以外の乗員
- DVE+Uモデルでの Environment は，ISO 23150のコンセプトを導入して，次の2つを第2レベルとする
  - SurroundModel … 車両の周辺モデル（ISO 23150で定義される情報）
  - Atmosphere … 車両の環境情報（VSSでは Exterior）
- TODO：VSSにあるその他の第2レベルについては，さらに検討が必要

OEM固有のオブジェクト／属性

- OEMは，固有のオブジェクトや属性を追加することができる
- TODO：標準のオブジェクト／属性名と衝突しないように，OEM固有のオブジェクト／属性の命名規則を定める

インスタンスの識別とID番号

- 同一種類のオブジェクトが複数ある場合には，そのオブジェクトクラスを表すノードの下に，オブジェクトインスタンスを識別するノードを置く
  - インスタンスを識別するノードは，“@”で始まる名称とする
- サービスコールのパラメータでオブジェクトを識別する場合には，オブジェクトのID番号を用いる
  - オブジェクトのID番号は，オブジェクトを示すノードの下に，“IdNumber”というノードを置いて，そこに格納する

車両構成記述のイメージ（ウィンドウの例）

- Vehicle.Cabin.Window.NumberOfWindows
  - ウィンドウの数（情報としては冗長）
- Vehicle.Cabin.Window.@FrontRight.IdNumber
  - “FrontRight”は，ウィンドウの識別名（任意の文字列）
  - IdNumberは，そのウィンドウを操作するサービスコールに指定するためのID番号（番号で指定する場合に）
- Vehicle.Cabin.Window.@FrontRight.位置
  - 車両におけるウィンドウの位置（位置の表現方法は要検討）
- Vehicle.Cabin.Window.@FrontRight.開閉属性
  - 移動方法の次元。サンルーフは移動方向が2つある
  - 手動開閉 or 自動開閉。手動開閉の場合，状態が取得できるかできないか

オブジェクトに付属するオブジェクト

- オブジェクトに付属するオブジェクトは，前者のオブジェクトのインスタンスの下に置くのではなく，独立したオブジェクトと扱う
  - 付属関係は，属性により表現する
- 例えば，ウィンドウやガラスに付属しているシェード（日除け）は，各種のシェードをすべて Cabin の下の Shade のインスタンスとし，属性を使ってウィンドウやガラスと紐付ける
  - Vehicle.Cabin.Shade.@Row2Right
  - Vehicle.Cabin.Shade.@Row2Right.対応するウィンドウ
  - Vehicle.Cabin.Shade.@Row2Right.対応するガラス
  - どのシェードに対しても同じサービスコールで制御する

APIの要素

サービスコール

- アプリケーションが，ビークルOSのサービスを呼び出すインタフェース
  - サービスコールは，対象となるオブジェクトの種類毎に定義する
- サービスコールは，パラメータとリターンパラメータを持つ
  - 対象となるオブジェクトが複数のインスタンスを持つ場合，第1パラメータを，インスタンスを識別するIDとする（オブジェクト指向言語などでは，このパラメータはメソッドのパラメータとはせず，メソッドを受け取るオブジェクトで表す）
  - リターンパラメータには，サービスコールが正常に実行されたかエラーになったか，エラーになった場合にはエラーの種類を示すものを含む

サービスコール – 続き

- API仕様では，サービスコールの名称と機能，パラメータとリターンパラメータの名称・データ型・意味，パラメータとリターンパラメータで使用する定数値の名称・意味などを規定する
- TODO：APIを記述するために，何らかの形式的な記述法（IDLのようなもの）を導入することを検討する
- ビークルサービスは，サービスコールを呼び出したアプリケーションのインスタンス（のID）を知ることができるものとする
  - それを用いて，調停やアクセス制御を行う

イベントキュー

- アプリケーションが，ビークルOS内で発生したイベントを受け取るためのインタフェース
  - アプリケーションは，複数のイベントキューを持つことができる。イベントキューはIDで識別する
  - イベントキューには，複数のイベントを入れることができる。イベントキューは，FIFO順でイベントを管理する。優先度管理をしたい場合には，複数のイベントキューを用いて，アプリケーション側で実現する
- IDで指定したイベントキューからイベントを取り出すサービスコールを用意する
  - TBD：イベントを取り出すサービスコールは，オブジェクト毎に用意するか，（オブジェクトによらず）1種類のみ用意する

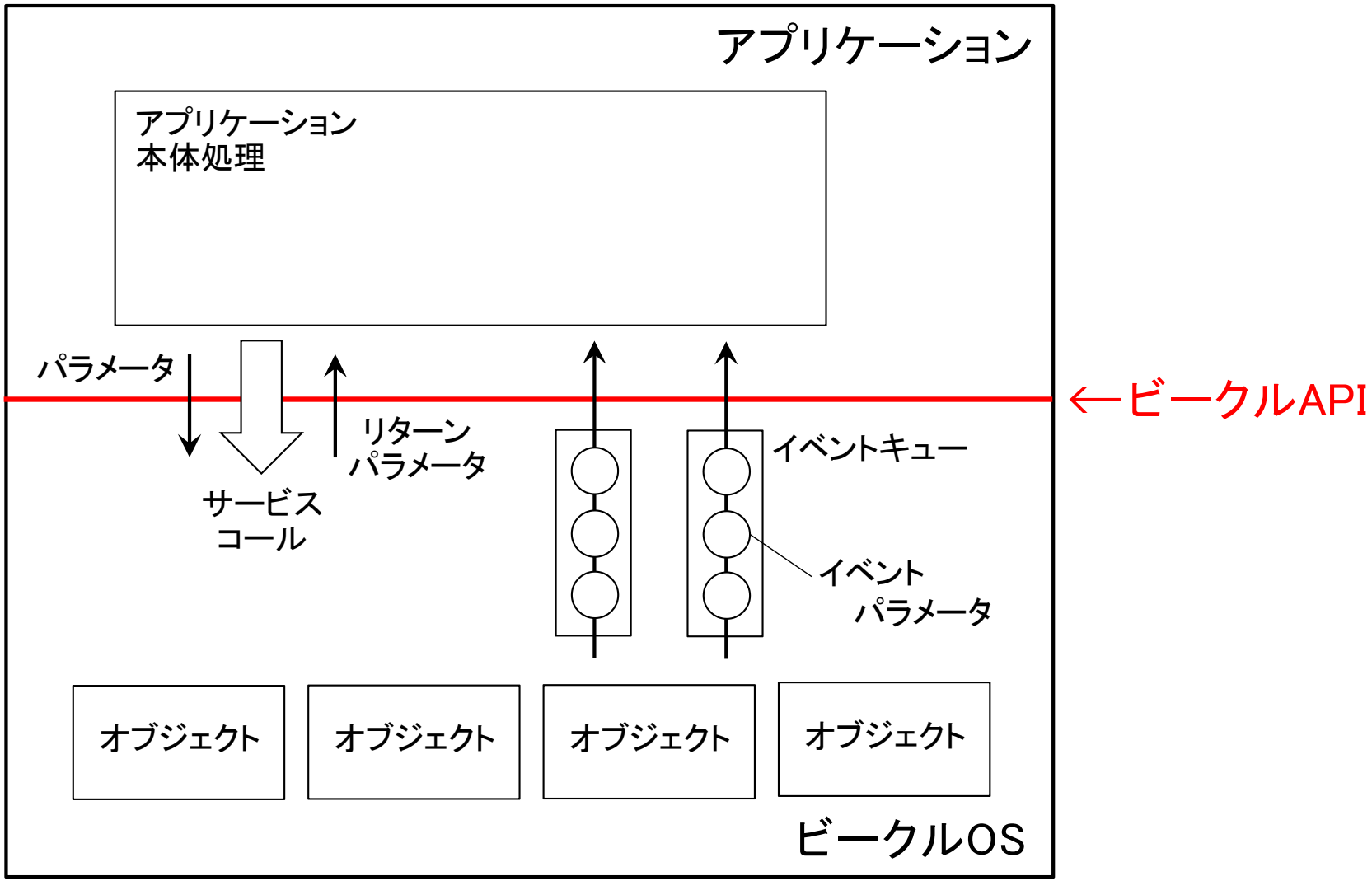
イベントキュー – 続き

- イベントを受け取る場合には，イベント通知を要求するサービスコール（通知要求サービスコール）を呼ぶ
  - 通知要求サービスコールは，対象となるオブジェクト毎に用意する（オブジェクトに対して複数の通知要求サービスコールを設けても良い）
  - 1つの通知要求サービスコールに対して，1つのイベントキューが生成される。通知要求サービスコールが正常実行されると，イベントキューのIDが返される
  - 受け取るイベントの種類を限定する場合には，通知要求サービスコールのパラメータで指定する
- イベント通知を停止する時は，イベントキューをクローズするサービスコールを呼ぶ
  - TBD：イベントキューをクローズするサービスコールは，オブジェクト毎に用意するか，1種類のみ用意する

イベントキュー – 続き

- イベントは，イベントの種類に加えて，付属情報を表すためのパラメータを持つ
- API仕様では，通知要求サービスコール（notify）の仕様，イベント（パラメータを含む）の名称・データ型・意味，パラメータで使用する定数値の名称・意味などを規定する

APIの要素（図示）


アプリケーション
アプリケーション
本体処理
パラメータ
サービス
コール
リターン
パラメータ
←ビークルAPI
イベントキュー
イベント
パラメータ
オブジェクト
オブジェクト
オブジェクト
オブジェクト
ビークルOS

データ型

- サービスコールのパラメータとリターンパラメータ，イベントのデータ型は，物理レベル（具体的には，ビット長や単位）に踏み込んで規定する
- TODO：データ型を記述するために，何らかの形式的な記述法を導入する
  - 既存のもので目的に合致するものがあれば使いたいが，現状では見つかっていない

データ型の具体化

- 基本データ型(ビット長は最低限を示す)
  - bool, char(不要かも)
  - int8, int16, int32, int64
  - uint8, uint16, uint32, uint64
  - float32, float64
- 文字列
- 配列
  - 固定長配列
  - 可変長配列
- 構造体
- 列挙型
- タグ付き共用体(バリアント型)
- オプショナル型 … 省略可能であることを示す

データ型の具体化 – 補足説明

- 可変長配列
  - C言語では，配列が置かれた領域の先頭番地と長さで実現する
  - C++では，std::vectorで実現しても良い（動的メモリ管理を行う多くのプログラミング言語でも同様）
- char，文字列
  - Unicodeの文字／文字列とする

(コンピュータの) OSに対する想定

(コンピュータの)OSが持つべき機能

- ビークルサービスを実現するベースとなる(コンピュータの)OSは，OSが一般的に持つ機能を備えており，アプリケーションはそれらの機能を利用できるものとする
- 特に，以下の機能を持つことを想定する
  - ECUを越えたサービス呼び出し機能とPub/Sub通信機能
  - 他のECU上で新しいプロセスを立ち上げる機能

サービスコールの設計方針

サービスコールの継続時間

- サービスコールは，短い時間で処理が完了するような設計とする
  - 例えば，ウィンドウを操作するサービスコールは，開閉動作が完了するまで待つのではなく，目標位置を指定して，開閉動作を開始するのみとする

サービスコールの呼び出し頻度／回数

- サービスコールを短い周期（具体的には，10ミリ秒未満）で呼び出す必要があるような設計としない
- サービスコールの粒度を大きくし，呼び出し回数が少なくなるような設計とする
  - サービスコールが，ネットワークを越えて呼び出されることが想定されるため

サービスコールを呼び出す順序

- サービスコールを特定の順序で呼び出さなければならないような設計にしない
  - 具体的には，POSIXにおけるファイル／デバイスのopen/close のようなサービスコールは設けない
- 以前の資料では，「ビークルOSのサービスは，なるべく状態を持たないように設計する」と記載していたが，ロック状態などは持たざるを得ないため，記載を変更した

制御の調停機能

基本方針

- 複数のアプリケーションが，同一のオブジェクトを制御（出力装置に出力）する場合の調停機能を，ビークルOSに持たせる
  - 調停に必要となるロックと操作優先度の概念を，ビークルAPIに導入する
  - コアビークルサービスには汎用的な調停機能のみを持たせ，アプリケーション特有の調停機能が必要な場合には，拡張ビークルサービスで実現する
- TODO：制御の衝突判定（同時に実行してはならないアプリケーションの検出）機能もビークルOSに持たせたいが，ロック機能で十分か要検討

ユースケース1）ウィンドウの制御

- ウィンドウを閉めるサービスコールが呼び出された後に，開くサービスコールが呼び出されると，最初のサービスコールによる制御は上書きされる
- 洗車モード中は，ウィンドウ開閉スイッチによるウィンドウの開閉を禁止する

ユースケース2）AD/ADAS

- 緊急ブレーキの動作中は，ACC（オートクルーズコントロール）による制御は受け付けない
  - ACCには，制御が受け付けられなかったことを通知する
- 自動運転とACCは，同時に実行してはならない

ユースケース3）音声出力

- アプリケーションAが音声出力中に，アプリケーションBが音声出力しようとした場合は，Aの音声出力が終了後に，Bの音声出力を行う
- アプリケーションAが音声出力中に，アプリケーションBが緊急性の高い音声を出力しようとした場合は，Aの音声出力を中断し，Bの音声出力を行う
  - アプリケーションAには，音声出力が中断されたことを通知する

標準的な調停機能

- 以下では，ビークルOSに持たせる標準的な調停機能（調停方法，ロック，操作優先度）を定義する
- 標準的な調停機能では不十分なオブジェクト（例えば，音声出力）に対しては，オブジェクト固有の調停機能を持たせる

後勝ちの原則

- 後に呼び出されたサービスコールによる制御が，前に呼び出されたサービスコールによる制御を上書きすることを原則とする
  - 短い時間で処理が完了するようにサービスコールを設計するとしたことから，この原則が自然
- 原則の適用例
  - ウィンドウを閉めるサービスコールが呼び出され，ウィンドウが閉じる動作中に，開くサービスコールが呼び出されると，最初のサービスコールによる制御はキャンセルされ，ウィンドウは開く動作を始める

ロック

- 他のアプリケーション（のインスタンス）によるオブジェクトの操作を禁止することを，ロックと呼ぶ
  - 必要なオブジェクトに対して，ロックの機能を持たせる
  - ロックは，機械的に行われる場合（例えば，ドアロック）と，ソフトウェア的にのみ行われる場合がある
- 利用例
  - 自動運転アプリケーションは，車両の運動制御（目標位置，目標速度）をロックし，他のアプリケーションが制御できないようにする

操作優先度

- 同じオブジェクトを複数のアプリケーションがロック／制御した場合に，どれを優先するかを決めるために，操作優先度の考え方を導入する
- TODO：操作優先度の意味と数値を定める
  - 例えば，緊急アプリ（16），特権ユーザ（32），特権アプリ（48），一般ユーザ（64），一般アプリ（80）
- 以下では，操作優先度の標準的な扱いを定義する
- オブジェクトによっては，操作優先度に対して追加の意味を持たせる場合もある

ロックと操作優先度

- オブジェクトをロックした操作優先度よりも高い操作優先度で操作した場合，ロックが強制解除され，操作を行うことができる
- 具体的な振る舞いとして，操作優先度*P*でロックした場合，次のようになる
  - *P*と同じか*P*より低い操作優先度での他のアプリケーションによる操作は禁止される
  - *P*より高い操作優先度での操作は禁止されず，ロックが強制解除される
  - *P*より高い操作優先度でロックされると，ロックを奪われる

LockableObject

- 標準的なロック機能を持つオブジェクトを，LockableObject と呼ぶ
- 具体的には，以下の機能を持つ
  - ロック／ロック解除のサービスコールを持つ
  - 状態参照によりロック状態を参照できる
  - ロック／ロック解除のイベントを持つ
- 「このオブジェクトは LockableObject である」と書くことで，標準的なロック機能を持つことが表現でき，各オブジェクトに対してロックの説明を記述する冗長性が避けられる
- LockableObject は親クラスのようなものであるが，実装を継承するとは限らないため，あくまでもインタフェースのインポートであると考える。多重インポートもあり

他の調停方法

- 先勝ち
  - 先勝ちの調停は，ロックを用いて実現できる
  - 例えば，ウィンドウを閉めるサービスコールを呼び出してから，ウィンドウが完全に閉まるまでは，他のアプリケーションによる制御を許したくない場合には，ウィンドウをロックした後にウィンドウを閉めるサービスコールを呼び出し，ウィンドウが完全に閉まったら(イベントで捕まえることができる)ロックを解除すれば良い
- 操作停止
  - TODO:矛盾した操作をした場合には，操作を止める調停方法が必要という意見がある。具体的なユースケースを探す

アクセス制御

基本方針（再掲）

- どのAPIを実装するか，どのアプリケーションにどのAPIの使用を許すかは，OEMの判断
  - ただし，プライバシ情報に関わるAPIの使用を許すかは，ユーザに判断させる（スマホと同様）
- 安全性の確保，プライバシ保護，セキュリティ向上のために，APIのアクセス制御を導入する
- 安全性に関わるAPIを使用するサードパーティ製のアプリケーションは，安全性の審査を受けることが必要
  - 審査する安全性の例：危険な制御を行わないこと，ドライバデストラクションがないこと，など
  - 審査の結果として，アプリケーションに使用を許可するサービスコールに関する情報（これを「アクセス制御情報」と呼ぶ）を作成する

アクセス制御の必要性

- アクセス制御の必要性は，以下の3つの観点から出てくる

安全性確保

- 安全性に関わる操作を，アプリケーションに許可するかどうかを制御する
- アクセスの可否は，OEM（またはOEMから委託された者）が，アプリケーションを審査することによって決定する

プライバシ保護

- プライバシに関わる情報へのアクセスを，アプリケーションに許可するかどうかを制御する
- アクセスの可否は，ユーザが決定する

セキュリティ向上

- アプリケーションが侵害された場合に備えて，必要なサービスコール以外は呼び出せないようにする

プライバシ保護のためのアクセス制御

- スマホ向けのOSと同様の機能で良いと思われる

アクセス制御方法の概要

- プライバシに関わる情報にアクセスするサービスコールは，あらかじめ特定しておく（OSDVI API仕様書で？）
- アプリケーションが該当するサービスコールを最初に呼び出した時に，ビークルOSは，アクセスを許可するかどうかをユーザに問い合わせる
- アプリケーション毎のアクセス制御情報は，運転者個人に紐づいたユーザプリファレンスデータ内に格納し，「設定」機能により設定変更できる
  - 同一の車両でも，ユーザによって設定は異なる
  - ユーザは，複数の車両に対して同じ設定を使用できる

OEMの純正アプリケーション

- OEMの純正アプリケーション（標準制御を実現するアプリケーションを含む）については，アプリケーション毎のユーザによる許可をスキップできるようにする
  - OEMは，車両を販売する時点で，ユーザから包括的な許可をもらうことを想定

安全性確保のためのアクセス制御

- 安全性に関わる機能を持つビークルOSに特有の機能になると思われる

アクセス制御方法の概要

- 安全性に関わる操作を行う可能性のあるサービスコールは，あらかじめ特定しておく（OSDVI API仕様書で）
- 該当するサービスコールを使用するアプリケーションは，使用するサービスコールを申告して，OEM（またはOEMから委託された者。以下，審査機関と呼ぶ）の審査を受ける
- 審査の結果として，アプリケーションに使用を許可するサービスコールに関する情報を作成する
  - これを，「アクセス制御情報」と呼ぶ
  - これにより，セキュリティ向上のためのアクセス制御も実現する

アクセス制御方法の概要 – 続き

- 「アクセス制御情報」は，アプリケーションと一緒に配布する（審査機関による署名付きで）
- ビークルOSは，アプリケーションが「アクセス制御情報」で許可されていないサービスコールを呼び出した場合，エラー（アクセス制御違反）とする

「アクセス制御情報」に関する要求

- 「アクセス制御情報」は，車両に依存しないものとする
  - 審査工数が大きくなるのを防ぐために，車両ごとに審査するのを避けるため

ユースケース1）操作対象オブジェクトと車両状態への依存性

- 運転者の前（インパネ）に設置されたディスプレイ内の（一部分の）表示領域を考える
- 「アクセス制御情報」では，以下のようなことを表現したい
  - アプリケーションA（例えば，カーナビ）は，運転者が運転操作に注力する必要がある車両状態でも，その表示領域を使用できる
  - アプリケーションB（例えば，ゲーム）は，運転者が運転操作に注力する必要がない車両状態でしか，その表示領域を使用できない

→ サービスコールの使用可否は，操作対象のオブジェクトインスタンスと，車両状態に依存して変化する

- ただし，車両によって持っているオブジェクトインスタンスは異なるため，「アクセス制御情報」中でインスタンスを指定する方法は採れない

ユースケース2）車両の安全機能への依存性

- ウィンドウに挟み込み防止機能を持つ車両Xと，持たない車両Yがある場合を考える
- 「アクセス制御情報」では，以下のようなことを表現したい
  - ウィンドウを操作するサービスコールを，アプリケーションAは，車両Xでは使用できるが，車両Yでは使用できない
  - アプリケーションBは，どちらの車両でも使用できる

→ サービスコールの使用可否は，車両（ビークルOS）側で安全性が確保されるかに依存して変化する

- 車両の構成記述に，（操作により安全性に関わる可能性のある）オブジェクト（の各インスタンス）に対して，車両側で安全性が確保されるかどうかの情報を持つことが必要

ユースケース3）車両状態に依存する安全性

- 運転席のシートを移動させるサービスコールを考える
  - 車両が駐車中は，このサービスコールを呼び出しても安全である
  - 車両が走行中（手動運転中）にこのサービスコールを安全に呼び出せる機能を備えている車両と備えていない車両があることが考えられる

→ 車両の構成記述に持つ「車両側で安全性が確保されるかどうかの情報」は，車両状態に依存する

! *例があまり良くない。納得性の高い例があれば差し換える*

車両の構成記述への追加情報

- 車両の構成記述には，操作により安全性に関わる可能性のあるオブジェクト(の各インスタンス)に対して，安全性クラスを追加する
- どのオブジェクトにどのような安全性クラスがあるかは，OSDVI API仕様書で規定する
- 例えば，ウィンドウに対しては，以下の2つの安全性クラスがある
  - 車両側で安全性が確保される(挟み込み防止機能を持つ)
  - 車両側で安全性が確保されない(挟み込み防止機能を持たない)

車両の構成記述への追加情報 – 続き

- 例えば，シートに対しては，以下の2つの安全性クラスがある
  - 運転中に移動させると危険につながる可能性がある（運転席のシート）
  - 移動させても危険につながらない（その他のシート）
- 例えば，ディスプレイ（内の表示領域）に対しては，以下の4つの安全性クラスがある
  - 運転者が視線を移動せずに見ることができる（ヘッドアップディスプレイ）
  - 運転者が少しの視線移動で見ることができる（インパネ内のディスプレイ）
  - 運転者が大きい視線移動で見ることができる（センターディスプレイなど）
  - 運転者が見ることができない（助手席や後席向けのディスプレイなど）

「アクセス制御情報」の内容

- 使用する各サービスコール・(サービスコールの操作対象オブジェクトの) 各安全性クラス毎に，次の情報をもつ
  - サービスコールを呼び出せる車両状態の条件
  - 使用できる操作優先度の最高値
- 安全性クラスを持たないオブジェクトに対するサービスコールの場合は，各サービスコール毎に上の情報を持つ
- アクセス制御情報中に，適用される安全性クラスに関する情報がない場合，そのサービスコールは呼び出せないものとする
  - API仕様がバージョンアップした場合に，このような状況が起こる

補足説明

- 「アクセス制御情報」で使用が許可されているサービスコールでも，エラーになる場合もある
  - 例えば，標準の車両状態で，使用可否を十分に表現できない場合
  - どのエラーコードとするか（アクセス制御違反か別のエラーか）は要検討

アクセス制御のための車両状態

車両状態の定義方針

- 車両の状態（IG ON，ACC ON，OFFや，自動運転のレベルなど）によって，使用できるAPIやAPIの振る舞いは変化する
  - 一般に，車両状態は，車両（特にパワートレイン）による違いが大きく，標準化は難しいと考えられている
- ビークルAPI仕様では，アクセス制御に使用するという目的に絞って，車両状態を定義する

車両状態の定義

- 車両状態は，次の3つのサブ状態の組み合わせで表現する
  - 走行状態
  - 運転状態
  - 走行環境（オプション）
- さらに，以下のサブ状態の導入について検討する
  - バッテリ状態（バッテリ残量のレベル）
  - 乗員状態（乗員が乗っているか否か）
  - 異常状態（故障，事故，それらの深刻度）

走行状態

- 走行中
  - 走行している状態（速度がゼロでない状態）
  - 運転者は運転に対して何らかの義務を負っている（運転状態がドライバレスの場合を除く）
- 停止中
  - 走行途中に一時的に止まっている状態（速度がゼロの状態）
  - 運転者は運転席を離れてはならず，運転者は周囲の状況に注意を払う必要がある（運転状態がドライバレスの場合を除く）
  - ドアを開けても良い（要検討）

走行状態 – 続き

- 駐車中
  - シフトレバーがパーキング位置にある状態。または，パーキングブレーキを掛けている状態
  - 運転者は運転席を離れても良い
- 電源OFF
  - パワートレインの電源が切れている状態
    - ✓いわゆるアクセサリ（ACC）モードは，ここに含める

運転状態

- 手動運転(レベル1, ハンズオフ可能でないレベル2)
  - 運転者が何らかの運動制御を行う状態。運転者は, 安全運転に注力する義務がある
- ハンズオフ(ハンズオフ可能なレベル2)
  - 車両の運動制御は自動化されているが, 運転者は常に周囲の状況を監視し, システムが対応できない状況に備える必要がある状態
- アイズオフ(レベル3)
  - 車両の運動制御は完全に自動化されており, 運転者は運転操作に注力する必要がない状態。ただし, 運転者は運転席に座っている必要があり, システムからの操作引き継ぎ要求に速やかに対応する義務がある
- ドライバレス(レベル4～5, ブレインオフ)
  - 車両の運動制御は完全に自動化されており, 運転者がいなくても車両が安全に運行できる状態

走行状態と運転状態の依存性

- 走行状態が「駐車中」または「電源OFF」の場合，運転状態は「ドライバレス」とする
  - これにより，サービスコールを使用できる条件をシンプルに記述できるようになる

走行環境（オプション）

- 公道走行状態
- 非公道走行状態（通称：サーキットモード）
  - 公道でない道路を走行している場合に使用できるAPIはあるものと思われる
- 走行環境の状態を持たない車両があってもよい
  - 走行環境の状態を持たない車両においては，走行環境に対する条件があるサービスコールは，許可されない（言い換えると，すべての走行環境で呼び出せるサービスコールのみ仕様できる）
- 走行環境をより細かく分類することも考えられる

  例）自動車専用道と一般道

  例）走行している国・地域（適用される交通ルール）

その他の車両状態

- バッテリ状態（バッテリ残量のレベル）
  - 審査の結果，あるAPIを，バッテリ残量によっては使用できない（または使用できる）という状況があるか？
- 乗員状態（乗員が乗っているか否か）
  - 審査の結果，あるAPIを，乗員が乗っている状態（または乗っていない状態）では使用できない（または使用できる）という状況があるか？
- 異常状態（故障，事故，それらの深刻度）
  - 審査の結果，あるAPIを，異常状態では使用できない（または使用できる）という状況があるか？
  - 導入する場合には，異常状態のレベルを設ける（2～3段階程度？）

共通機能

すべてのオブジェクトに用意する機能

オブジェクトの構成参照サービスコール

- 名称：getConfig
- パラメータ
  - なし
- リターンパラメータ
  - 正常終了 or エラーの種類
  - 対象オブジェクトのすべてのインスタンスの構成情報（対象オブジェクトの構成情報の可変長配列）
- 機能
  - 対象オブジェクトのすべてのインスタンスの構成情報を参照する

オブジェクトの状態参照サービスコール

- 名称：getStatus
- パラメータ
  - *xxx*Id：操作対象のオブジェクトID（複数のインスタンスを持たないオブジェクトに対しては，このパラメータは省く）
- リターンパラメータ
  - 正常終了 or エラーの種類
  - オブジェクトの状態
- 機能
  - 操作対象のオブジェクト（のインスタンス）に関するすべての状態を参照する

LockableObject の機能

ロックサービスコール

- 名称：lock
- パラメータ
  - *xxx*Id：ロック対象のオブジェクトID（複数のインスタンスを持たないオブジェクトに対しては，このパラメータは省く）
  - priority：操作優先度
- リターンパラメータ
  - 正常終了 or エラーの種類
- 機能
  - ロック対象のオブジェクト（のインスタンス）を，指定した操作優先度でロックする

LockableObject の機能

オブジェクト状態

- LockableObject は，次の状態を持つ
  - ロック状態
    - ✓「ロックされている」「ロックされていない」のいずれか
  - [ロックしている場合]ロックしたアプリケーションの（インスタンスのID）
  - [ロックしている場合]ロック操作優先度
    - ✓ロック操作を行なった操作優先度
- オブジェクト状態を参照するサービスコール（getStatus）により，上記の状態を参照することができる

ロック解除サービスコール

- 名称：unlock
- パラメータ
  - *xxx*Id：ロック解除対象のオブジェクトID（複数のインスタンスを持たないオブジェクトに対しては，このパラメータは省く）
- リターンパラメータ
  - 正常終了 or エラーの種類
- 機能
  - ロック解除対象のオブジェクト（のインスタンス）を，ロック解除する

イベント

- LockableObject は，次のイベントを持つ
  - ロックイベント
    - ✓オブジェクトが，他のアプリケーション（のインスタンス）によってロックされた
  - ロック解除イベント
    - ✓オブジェクトが，他のアプリケーション（のインスタンス）によってロック解除された

MovableObject の機能

- API仕様の記述を簡潔にするために，LockableObject と同様の位置付けのものとして，MovableObject を定義する
  - 参考：COVESAでは，MovableItem という名称になっている
- 詳細はTBD

イベントキュー操作機能

- TBD