Document Title	Explanation of Automotive API
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	1121

Document Status	published
Part of AUTOSAR Standard	Adaptive Platform
Part of Standard Release	R24-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2024-11-27	R24-11	AUTOSAR Release Management	• Initial release

Disclaimer

Contents

1 Introduction

1.1 Background

The lifecycles of the Intelligent and Connected Vehicles depend on the ability of diagnosing the vehicle and its functions remotely and updating its software modules to keep the vehicle roadworthy, safe and secure. Vehicle manufacturers also provide the access to vehicle data and functions through their proprietary APIs to selected service providers and to the authorities for sovereign tasks.

At the same time the users, the customers have adopted a new way of interacting with their connected devices, their digitalized life. This interaction is through applications in their smart devices using cloud services to access and control the connected devices to satisfy the desire for individually configured information, entertainment and comfort. This is increasingly the default user experience standard also for the vehicle customers of the IoT era.

The AUTOSAR standard is well equipped for connected software components and automotive applications within the vehicle in the vehicle internal E/E architecture. AUTOSAR standard also supports the updateability of the vehicle software. Today there is not yet support in the standard for safely and securely exposing the data and functions produced by the AUTOSAR applications to non-AUTOSAR clients, whether on-board or outside of the vehicle system boundaries. The Automotive API is to provide this support in the standard.

1.2 Objective

The objective of the Automotive API is to establish a standardized access to read and write vehicle data from both on-board and vehicle external applications (e.g., a cloud server), while simultaneously letting the data and services have their project-, OEM-specific definitions and representations in the in-vehicle network. Crucially, the new interface is intended to be useable by individuals and organizations that are not members of AUTOSAR. This approach will contribute to the connected vehicle ecosystem, driving innovation, further development and compatibility across all vehicles and applications using the new interface.

1.3 Definition of the Automotive API

The Automotive API is an interface that allows data-centric communication with the vehicle. It defines how other systems can access selected vehicle data securely and independently of the in-vehicle representation using a standardized interface across vehicle types and manufacturers. The Functional Cluster that enables this communication is the Automotive API Gateway. It represents all the required Adaptive

Platform functionality for the Automotive API. It is introduced to the Adaptive Platform as a new functional cluster in the Vehicle Service Functional Clusters group.

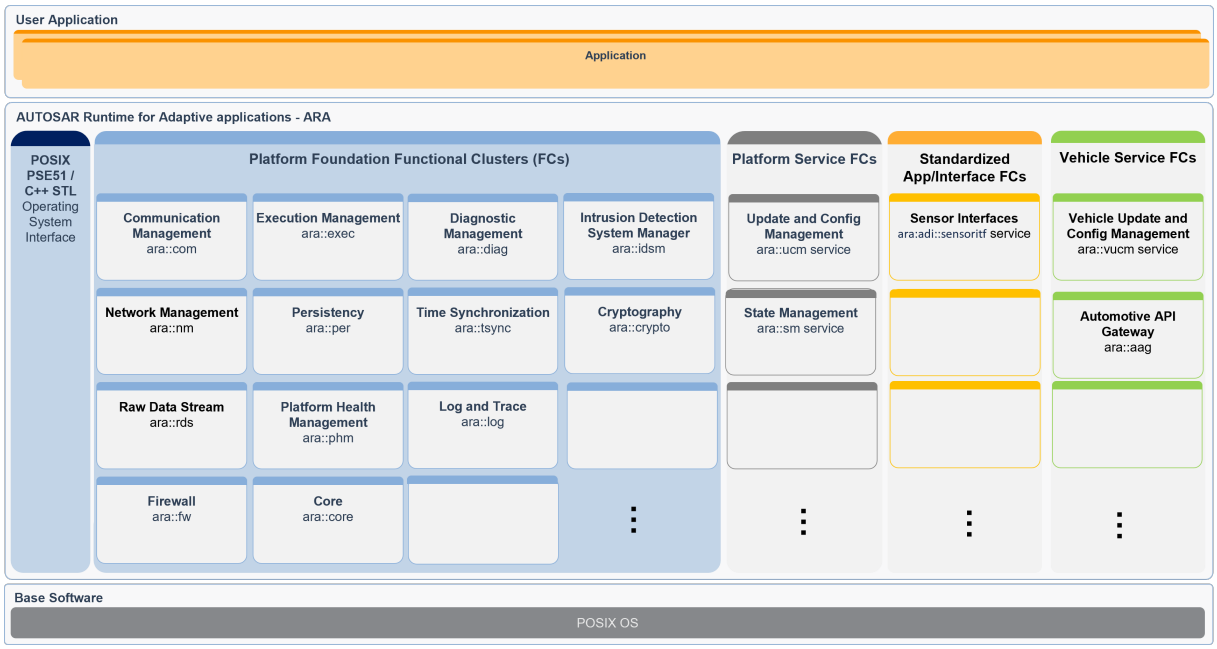

Figure 1.1: Automotive API Gateway as a new Vehicle Service Functional Cluster in the Adaptive Platform architecture.

2 Definition of terms and acronyms

The glossary below includes acronyms and abbreviations relevant to the explanation of the Automotive API.

2.1 Acronyms and abbreviations

Abbreviation / Acronym:	Description:
ara::com	C++ namespace of the AUTOSAR Adaptive Platform functional cluster Communication Management that abstracts communication within the Adaptive Platform. It is also used as a shorthand for the functional cluster. [1]
ARXML	Autosar Extensible Markup Language File; Modeling format for AUTOSAR
ASP	Automotive Service Provider
COVESA	The Connected Vehicle Systems Alliance
DDS	Data Distribution Service, a middleware protocol and API standard from the Object Management Group (OMG)
ECF	External Concept Framework
HTTP	Hypertext Transfer Protocol, an application layer protocol in the Internet protocol suite model for distributed, collaborative, hypermedia information systems
IoT	Internet of things
IPsec	Internet Protocol security, end-to-end security scheme within third layer of Open Systems Interconnection (OSI) model
MACsec	Media Access Control security, a common reference to network security standard IEEE 802.1AE
MPL	The Mozilla Public License, a free and open-source weak copyleft license for most Mozilla Foundation software such as Firefox and Thunderbird. The MPL license is developed and maintained by Mozilla, as an effort to reconsile the concerns of both open-source and proprietary developers.
MQTT	MQ Telemetry Transport, a lightweight, publish-subscribe, machine to machine network protocol for message queue/message queuing service
OEM	Original Equipment Manufacturer
SecOC	Security On-board Communication, for details check [2]
SWC	Software Component
TLS	Transport Layer Security, a cryptographic protocol designed to provide communications security over a computer network.
W3C	World Wide Web Consortium, the main international standards organization for the World Wide Web
VISS	The Vehicle Information Service Specification; This document refers to the version 2 of VISS as defined in [3] and [4].
VSS	Vehicle Signal Specification as defined in [5]

Table 2.1: Acronyms and abbreviations used in the scope of this Document

2.2 Definition of terms

Terms:	Description:
Automotive API Gateway	The functional cluster that offers the Automotive API and can thus connect VISS clients to vehicle internal data.
PassThroughSwConnector	See clause 4.3.1 in [6]. May check also [7] for further details in context of Adaptive Platform.

Table 2.2: Definition of terms in the scope of this Document

3 Related Documentation

3.1 Input documents & related standards and norms

[1] Explanation of ara::com API
AUTOSAR_AP_EXP_ARAComAPI

[2] Specification of Secure Onboard Communication Protocol
AUTOSAR_FO_PRS_SecOcProtocol

[3] Vehicle Information Service Specification - Core
https://github.com/COVESA/vehicle-information-service-specification/releases/download/v2.0/VISSv2_Core.pdf

[4] Vehicle Information Service Specification - Transport
https://github.com/COVESA/vehicle-information-service-specification/releases/download/v2.0/VISSv2_Transport.pdf

[5] Vehicle Signal Specification
https://covesa.github.io/vehicle_signal_specification/

[6] Software Component Template
AUTOSAR_CP_TPS_SoftwareComponentTemplate

[7] Specification of Manifest
AUTOSAR_AP_TPS_ManifestSpecification

[8] Technical Report on VSS Representation
AUTOSAR_AP_TR_VSSRepresentation

[9] ISO/TS 20077-3:2024 – Road vehicles – Extended vehicle (ExVe) methodology – Part 3: Upstream process to develop services
https://www.iso.org

4 Solution Strategy

4.1 Introduction

This Chapter explains how the vehicle data is described in a standardized, manufacturer- and vehicle-independent way. Then, using this description, it is defined how a user can communicate with the vehicle using the Automotive API to access the desired data.

This chapter also gives an introduction to translating incoming requests to Adaptive Platform internal representations, meaning a set of ARXML Service Interface definitions. The definitions are used to describe how to access the required data within the vehicle. Implementors are free to choose how and where the required information is accessed and possibly converted or processed. Such flexibility allows implementors to connect to all sorts of currently existing sources of data.

Finally, this chapter also describes how the connection from incoming requests to internal representations can be configured efficiently in common scenarios.

4.2 VSS: The Common Vehicle Data Model

The vehicle data model defines syntax and semantics for the data that is accessible through the Automotive API. This model needs to be a common one, to allow unified interaction with the Automotive API across vehicles and manufacturers regardless of the internal representation of the data. The Vehicle Signal Specification (VSS) was chosen as the common vehicle data model. VSS is an initiative by COVESA (`https://covesa.global/`) and licensed under MPL-2.0. It defines a syntax that is used to describe individual signals in a structured way. Using this syntax, a standardized catalog of signals is created within VSS. This VSS standard catalog is used as the common vehicle data model for the Automotive API. With that, data inside vehicles can be described in a completely vendor- and vehicle-independent way. The Automotive APIs of two different vehicles are only interoperable to the extent that the catalogs they deploy share a common subset. For that reason, the standard catalog defined by COVESA plays a central role by providing this common ground.

The VSS standard catalog can be extended and adapted to vendor-specific needs while keeping its core compatible with the standard. This allows flexibility wherever it is required.

A VSS catalog consists of a collection of defined signals. Individual signals defined by VSS are leaves of a tree structure and can be uniquely identified by their qualified names.

Figure 4.1 shows the definition of the signal *Vehicle.Speed* in the VSS standard catalog. The Speed leaf has the type sensor and is a child of the Vehicle branch (not shown here) with the fully qualified name *Vehicle.Speed*. The signals can be further refined by

using additional attributes that VSS provides, like minimum and maximum allowed values or a default value. VSS also offers a wide variety of supported datatypes including strings, arrays, and structs.

Using the individual signal definitions structured by the branches in a hierarchical tree formation, an extensive standard catalog is defined by the VSS. It allows a common vocabulary for these vehicle signals across OEMs and vehicle types.

```
Speed:
  datatype: float
  type: sensor
  unit: km/h
  description: Vehicle speed.
```

Figure 4.1: Example: VSS Signal Vehicle Speed

Further information can be found in the VSS documentation in [5].

4.3 VISS: The Communication Protocol

To facilitate access to the Automotive API a network interface, a communication protocol is required. The interface is to use the VSS vehicle data model. For that purpose, the Vehicle Information Service Specification version 2 (VISS) was chosen. VISS was originally developed by W3C and was then moved to COVESA.

VISS allows users to get, set, and subscribe to data that is defined in a VSS catalog. The VISS specification is split into the CORE [3] that describes the messaging API, and the TRANSPORT [4] specification that details protocol bindings for WebSocket, HTTP, and MQTT. This makes VISS flexible regarding the desired protocol as well as enabling future extensions to the list of supported protocols.

VISS supports secure communication through the TLS v1.2. It is a mandatory requirement for all VISS transports. The access control between a VISS client and a server is another mandatory requirement in VISS. Access control can be provided by an access token-based approach. As an extension of the access control VISS supports (non-normatively) the External Concept Framework (ECF) with the capability of the VISS server to enforce consent results. The consent resolution is delegated to the ECF. Both the secure communication and the access control on the VISS interface are essential enablers for safe, secure and legitimate access to data through the Automotive API. How the external interface security & access control are provided for the Automotive API and how they are mapped to the AUTOSAR in-vehicle security are outside the current scope of the Automotive API standard.

4.4 Components Overview

4.4.1 Introduction

Users can access the Vehicle using the VISS protocol through the Automotive API Gateway. The Gateway in turn uses ara::com to communicate with the vehicle internals to access the underlying data as requested by the users. In the first release of the Automotive API the focus is on off-board clients. In principle the Automotive API can serve both on- and off-board clients.

Since the data representation in the vehicle is not standardized and since it can be very different between concrete vehicles, the details of the ara::com interface of the source Adaptive Application(s) that provide the actual data cannot be standardized. Instead, specific solutions are required. The concept aims to support the simple creation of such solutions. (see Section 4.4.4)

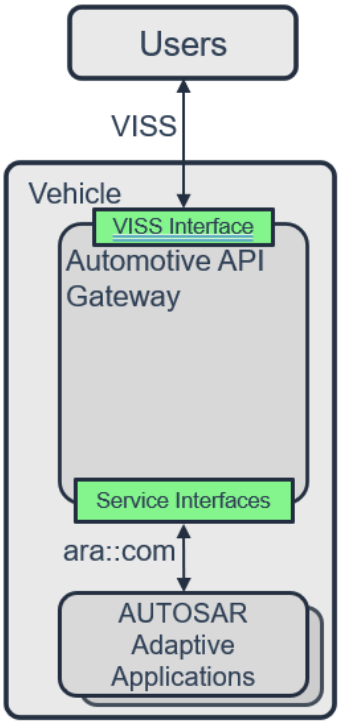

Figure 4.2: Automotive API Gateway.

4.4.2 Automotive API Gateway

The Automotive API Gateway (or just *Gateway* in the following) offers a VISS interface using a specified VSS catalog to the users of the Automotive API. To answer the VISS requests, it uses Service Interfaces that need to be provided within the AUTOSAR Adaptive Platform. How data that resides in systems other than AUTOSAR can be connected to the Gateway is not described in this release.

The Gateway is responsible for implementing a VISS server and offering it to its users. The VISS server is expected to perform access control duties for the requests and offer all required transport protocols. Although the VISS server supports filtering of the requests, there is no support for it in the ara::com service interface.

It is possible to deploy multiple Automotive API Gateways in one vehicle.

4.4.3 Definition of the VSS Derived Service Interfaces

The VSS derived Service Interfaces are a set of Service Interfaces derived from VSS data ("service") described using ARXML. The Automotive API Gateway provides an AUTOSAR compliant interface to provide VSS through VISS. Implementors must implement the provided Service Interfaces by either implementing the generated Skeletons or creating a declarative mapping in the ARXML model. See [8] for details.

This step should be done by tooling that ensures efficiency in the process and consistency between the used VSS catalog and the ARXML.

The VSS derived service interfaces are stable internal interfaces of the Automotive API Gateway as long as there are no changes in the offered VSS catalog. The Gateway uses this interface to define the access to the vehicle data for VISS requests.

The transformation from the VSS catalog to the VSS derived Service Interfaces is described in such a way that a generator can be created for it. Thus, a complete Gateway executable could be created in four steps:

1. Import of the VSS catalog into ARXML
2. Configuration of the required mapping in ARXML
3. Implementation of mapping that is not modeled in ARXML
4. Generation of the Gateway

In general, each VSS leaf is represented as a field and a method of the Gateway Interface. VISS requests are then mapped to corresponding actions on the field and method.

4.4.4 The Service Interfaces of the Gateway

This part of the document introduces ARXML model compliant ways of connecting the VSS derived Service Interfaces of the Gateway to already existing vehicle-internal data. Some common scenarios are covered.

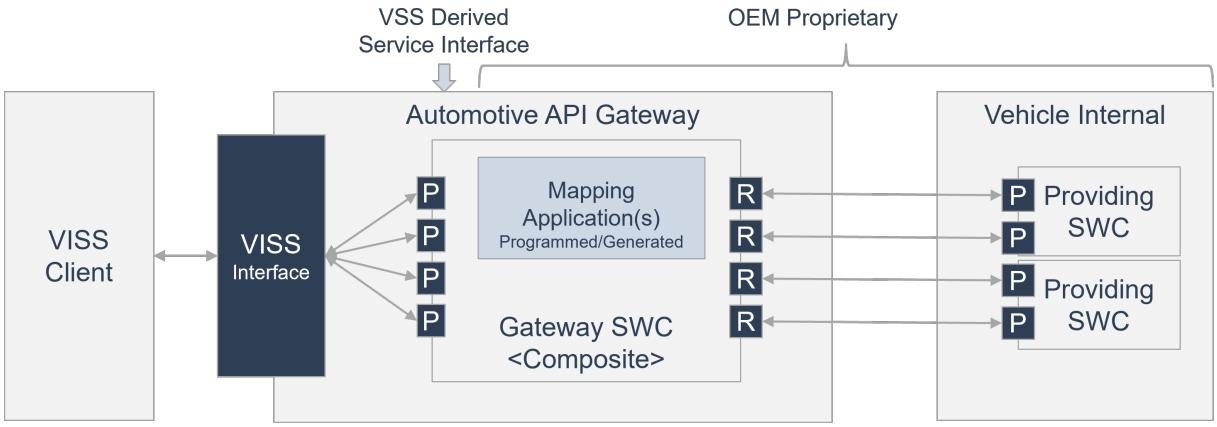

Figure 4.3: The Automotive API Gateway Connecting VSS data to in-vehicle services.

In general, it can be assumed that most, if not all data is already available within the vehicle. That means that there are existing Vehicle Internal Services with OEM proprietary Interface definitions. They have to be connected to the VSS derived Service Interfaces of the Automotive API Gateway. This requires mapping. Since the definitions of the vehicle internal Service Interfaces are proprietary, the mapping can also only be a proprietary solution. However, this concept aims to support the mapping efforts. These approaches can be mixed, so that for each signal the appropriate solution can be chosen. Although in the above figure the mapping is shown as part of the Gateway Composite SWC, it can also be deployed outside of the Gateway.

4.4.4.1 Data Exists in Identical Format

If a field or method defined by a VSS derived Service Interface is the same as the one defined by a Service Interface that an existing Adaptive Application offers or if a compatible Adaptive Application is created, then the Gateway can be configured to expose an Rport using the identical definition of the field/method. Then a trivial PassThroughSwConnector can be used to connect the two. This way, access to data that is explicitly offered for the Automotive API or that is already present identically within the vehicle can easily be made accessible via the Automotive API. It is currently assumed that this will be a rare occasion.

Although the AUTOSAR modelling supports this, there is no specification of the semantics of the PassThroughSwConnector in the context of the Automotive API Gateway in this release. This means the situation can be modeled but from that model an automatic code generation is unlikely to be possible. As long as this caveat applies the approach outlined in the first bullet point in Section 4.4.4.3 should be used.

4.4.4.2 Data Exists in Compatible Format

Data is likely to exist within the vehicle in a different format. For example, the datatype or unit might not be identical but compatible, or the data might have a different structure

or names. In this case the conversion capabilities of the PassThroughSwConnector can be leveraged. With them it is expected that many of the signals defined by the VSS catalog can be mapped to already existing OEM-defined interfaces.

The same caveat regarding the PassThroughSwConnector as in Section 4.4.4.1 applies.

4.4.4.3 Data Exists in Incompatible Format

The desired data defined by the VSS derived Service Interface is likely to be available within the current vehicle system. However, it is also anticipated that the data is represented so differently that the capabilities of the PassThroughSwConnector are insufficient. To solve this issue, the data must be converted in another way. Examples for this could be situations where data must be accumulated, distributed, processed, filtered, smoothed etc.

Since this can get extremely complex, it was decided to solve these conversions in code rather than modelling them in ARXML. However, in the future, there might be further conversion capabilities added to the modelling if there is demand for that.

This approach could also be used in case the conversion with PassThroughSwConnectors is not desired.

The required adaptations can be implemented in different locations, depending on the context:

- The conversion can be made as part of the Automotive API Gateway. For that the VSS derived Service Interfaces are used to generate C++ Skeletons that are used to implement the desired functionality.

  This approach can also be used in case the conversion with PassThroughSwConnector is not desired or not (yet) available.

- The required conversion can be made part of the Adaptive Application that currently provides access to the unconverted data. That Adaptive Application would then just have to provide an additional field/method in an existing or new Service Interface that it provides. Then, the field/method can be used as outlined in Section 4.4.4.1

- One or multiple separate Adaptive Applications with the purpose of converting the data as required can be created. They would require all Interfaces that allow access to the data in its original representation and provide Interfaces offering the data as defined by the VSS derived Interfaces.

4.4.4.4 Data Exists in Classic Platform

Using the signal to service translation capabilities of AUTOSAR, data that is present in a Classic Platform environment can be made accessible to the Adaptive Platform and then connected to the Gateway like in the previous scenarios.

5 Use Cases

This chapter explains use cases for the Automotive API. We can split them orthogonally addressing following aspects of the concept:

- Types of the clients - in-vehicle, and remote device (it has further impact on the Automotive API Gateway placement),
- Types of the interface for data providers/consumers - AUTOSAR Adaptive Platform (AP), AUTOSAR Classic (CP) and non-AUTOSAR platform as data providers,
- Deployment scenarios - fully in-vehicle, partially in-vehicle with cloud-based front-end,
- Time criticality.

5.1 Remote clients

In this scenario, Automotive API Gateway deployment depends on the type of the client:

Connectivity between vehicle and remote devices via short-range communication methods

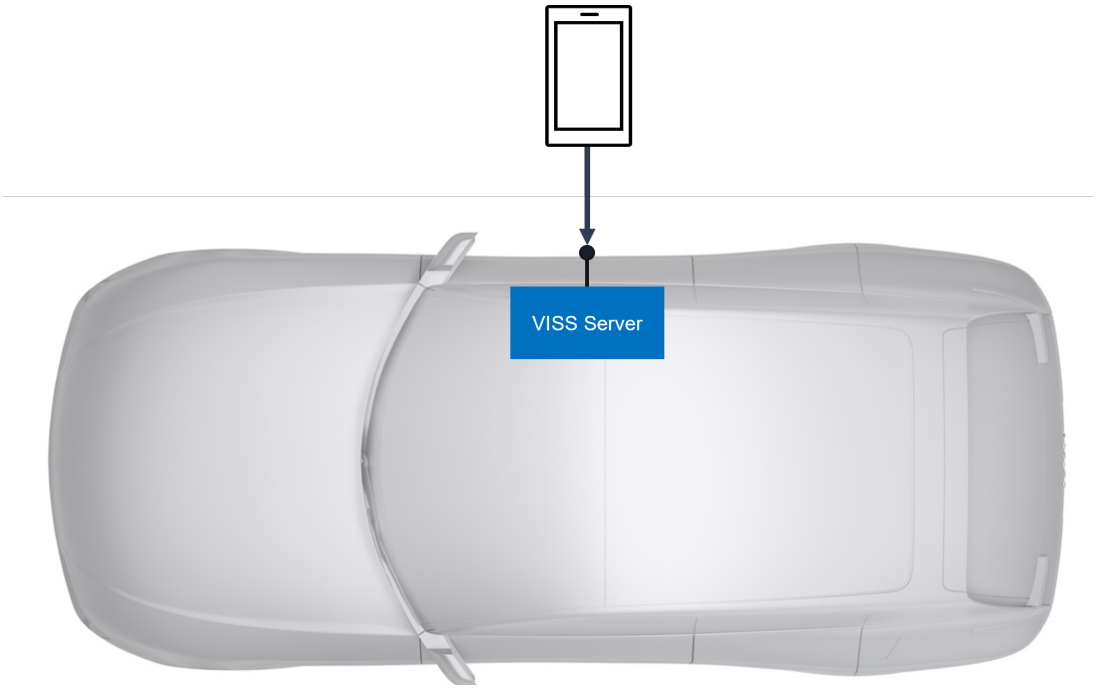

Figure 5.1: Connectivity of vehicle and remote devices with short-range communication technologies

Connectivity between vehicle and remote devices using short-range communication technologies is is the reality today. For example, connecting the device and the vehicle using Bluetooth or using vehicles as a Wi-Fi access point. Direct access (omitting cloud servers) from smart devices to the vehicle can decrease latencies and improve user experience. Automotive API Gateway provides access to the vehicle signals but hides all specifics of automotive technologies, making it easier to take benefits of such communication for mobile application developers.

Vehicle as a connected device provides data through a cloud service

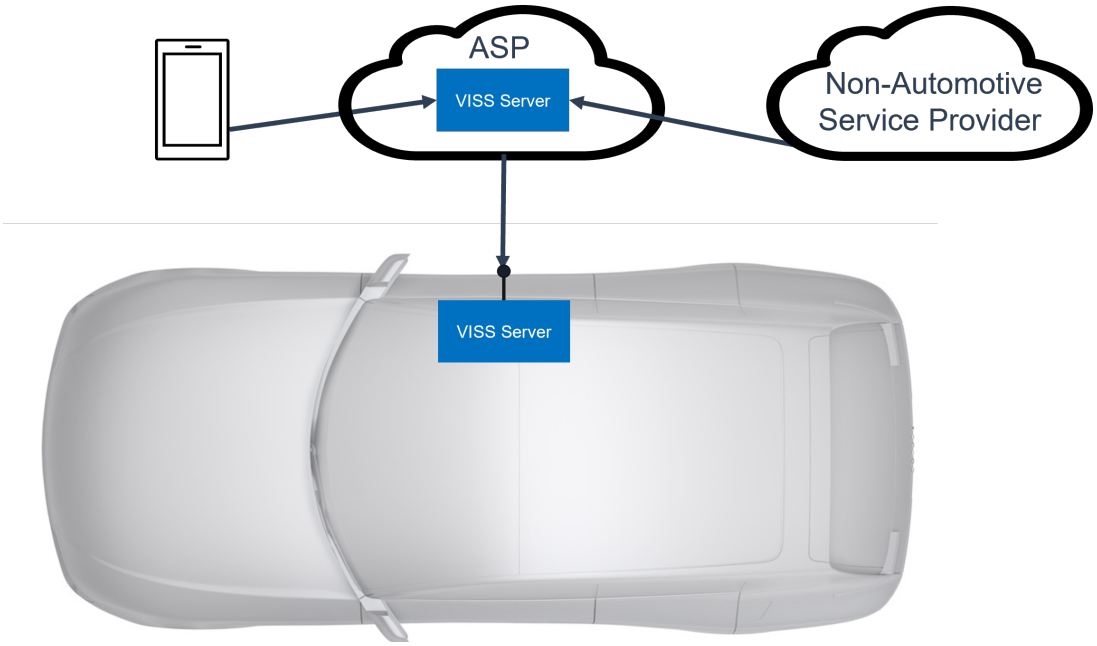

Figure 5.2: Vehicle provides data via the Cloud Service

Connectivity between a vehicle and smart devices or data processors, on long distance requires cloud-based infrastructure. In that case, Automotive API Gateway is partially deployed in-vehicle, which provides data to the `VISS` server in a cloud (cloud-based Automotive API Gateway front-end).

5.2 In-vehicle clients

In this scenario Automotive API Gateway is deployed fully in-vehicle. The client of the data can be:

- Adaptive Applications,
- Non-AUTOSAR system in E/E network like Android-based Infotainment system.

Modern vehicles are complex systems where different subsystems coexist, e.g., Android and AUTOSAR-based platforms. It might be challenging to connect these ecosystems due to inherent incompatibilities. The Automotive API can be used by other subsystems to get simplified and unified access to AUTOSAR subsystems.

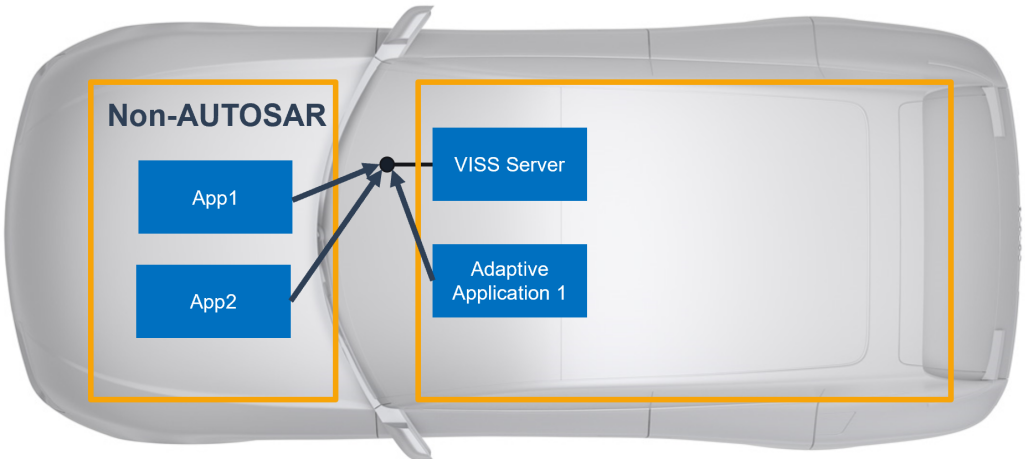

Figure 5.3: In-Vehicle Application Requesting VSS Data

5.3 Interfacing data providers/consumers components

The following interfaces domains can be distinguished:

- ara::com-based
  - VSS derived Service Interfaces may be utilized OnBoard
  - Existing OnBoard Service Interfaces provided through the AUTOSAR Methodology Workflow
  - Signal-based communication can be realized with usage of Signal2Service mapping
- raw data stream interface
  - This interface can also be used to connect to providers/consumers
- other interfaces
  - POSIX Sockets API
    Direct networked or memory based communication
  - Custom
    As this is very generic use case, often involving proprietary protocols, there is no detail to present in this document.

Interaction between the Adaptive Platform and the Classic Platform requires additional adaptation. Adaptive AUTOSAR already defines such adaptation as Signal2Service translation.

5.4 Time criticality

VISS as a text-based protocol is by no means optimized for time critical use case. Even the time stamp, that is a part of the VISS protocol, cannot currently support the original capture time of the value within the vehicle but only the capture time of the VISS network binding. Any time critical use case with VISS should be tested via benchmarking

6 Deployment Scenarios

In any deployment scenario the practical arrangements to make the Automotive API available to customers, for example, non-automotive service providers who want to interact with the vehicles through the Automotive API, are ultimately a task for the OEM and they are not addressed further in this document. For example the customer needs to know how to connect with the API (for example, the FQDN of the Automotive API "point of presence", required credentials, etc.) The network interface is expected to comply with the VISS transport. *ISO TR 20077-3* [9] can be used as a reference for an applicable process.

6.1 Cloud Server requesting VSS data

This is a typical "IoT scenario" where the end users access a service in the cloud, and the service itself requires certain vehicle data exposed in the Automotive API. A standardized API for accessing the vehicle data across the model ranges and OEMs provides economies of scale advantage to service providers and supports the development of the service ecosystem. This applies to both automotive and non-automotive service providers. Potential services include aftermarket services for monitoring, tailoring, interacting with the vehicle, fleet management, car insurance, and shared mobility services.

Although the Automotive API provides a standard based interface for vehicle data, the service provider still needs a business relationship and a contract with the involved OEMs or a data marketplace operator in order to gain access to the vehicle data. *ISO Technical Specification 20077-3 "Road vehicles - Extended vehicle (ExVe) methodology - Part 3: Upstream process to develop services"* [9] attempts to formalize the process to initiate and facilitate the communication between a service provider and an OEM to access the required vehicle data.

How exactly to arrange the connectivity between the end user and the vehicle application holding the data in question vary and will also depend on the relationship between the service provider and the OEM. For certain service scenarios the web services interface of the Extended Vehicle framework of ISO applies [ISO 20078 series of standards] while for certain other scenarios a more real time interaction may be required.

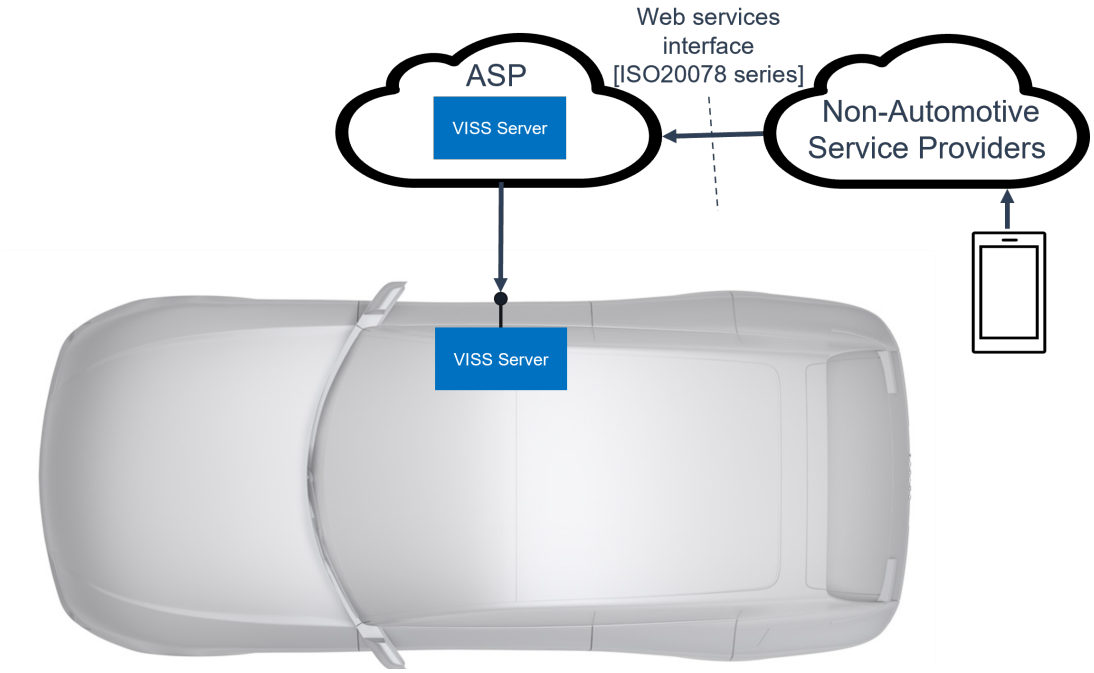

Figure 6.1: A cloud server requesting VSS data through a web services interface

6.2 Multiple Automotive API Gateways in a Vehicle

A vehicle provides different types of data. The data can be grouped to data sets based on the intended clients and their roles, safety criticality, sensitivity, etc.
The diagnostic station requires much more insights about in-vehicle hardware than telemetric data for data centers. Automotive API integrators can separate data regarding domains, severity and/or function collections and integrate them into several different Automotive API Gateways. In this use-case the Automotive API Service Provider authenticates and authorizes the clients, finally forwarding requests to the concrete in-vehicle endpoint. VISS Proxy Servers can perform additional filtering i.e., due to business model or security some clients have accesses only to limited VSS branches. Another benefit is that this deployment decreases workload of the ECU with Automotive APIs Gateways because filtering and more sophisticated authorization and authentication are performed out of the vehicle.

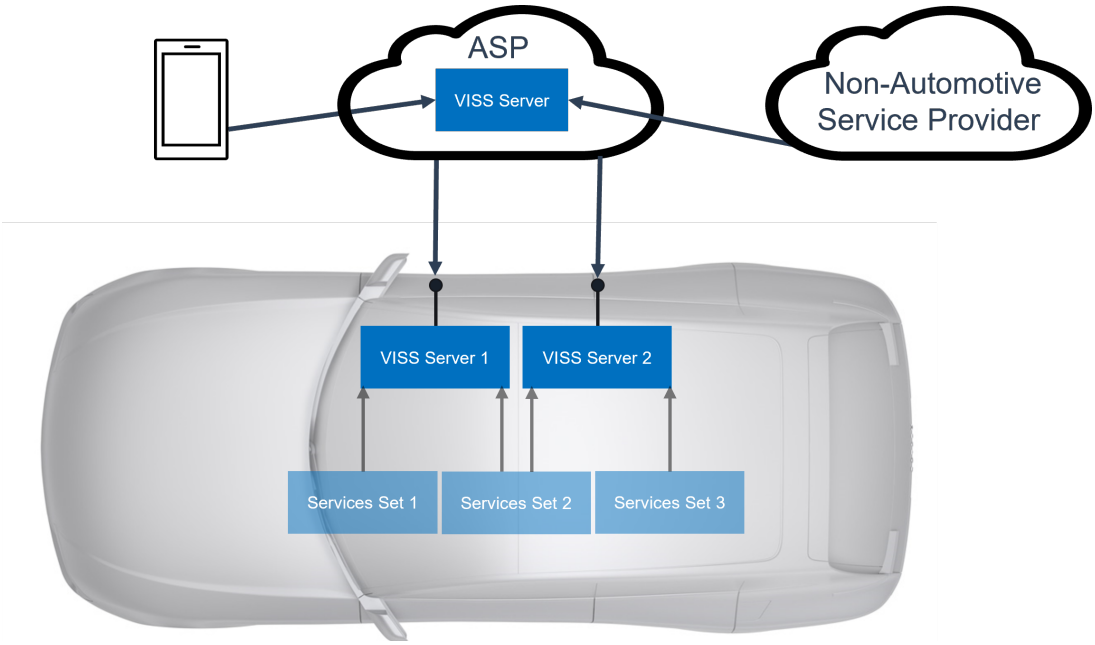

Figure 6.2: Multiple Automotive API Gateway deployment

6.3 In-vehicle application (non-AUTOSAR platform) requesting VSS data

Modern vehicles are complex systems where different subsystems coexist, e.g., Android and AUTOSAR-based platforms. It might be challenging to connect these ecosystems due to inherent incompatibilities.
The Automotive API can be used by other subsystems to get simplified and unified access to AUTOSAR subsystems.

6.4 Cascaded VISS Servers

Multiple VISS Servers can be cascaded to form the Automotive API Gateway. E.g., there could be one server in the Automotive Service Provider's (ASP) cloud that answers some of the requests directly by using a data source in the ASP cloud.
Other requests are forwarded to another VISS server that is deployed in the vehicle. That server then again decides where to retrieve the information from. It could be connected to a VISS server that offers data from an Android Automotive system as well as another server that is responsible for offering the data that resides within an AUTOSAR ecosystem within the vehicle. Each subsequent server could then have an accordingly narrowed VSS catalog that it offers. Clients could be made to always connect to the outermost VISS server as shown in figure 6.3. Alternatively, some clients could also directly connect to another VISS server. In the case of a mobile device that is in the vehicle this could enable interaction with the Automotive API even if there is no connection to the internet.

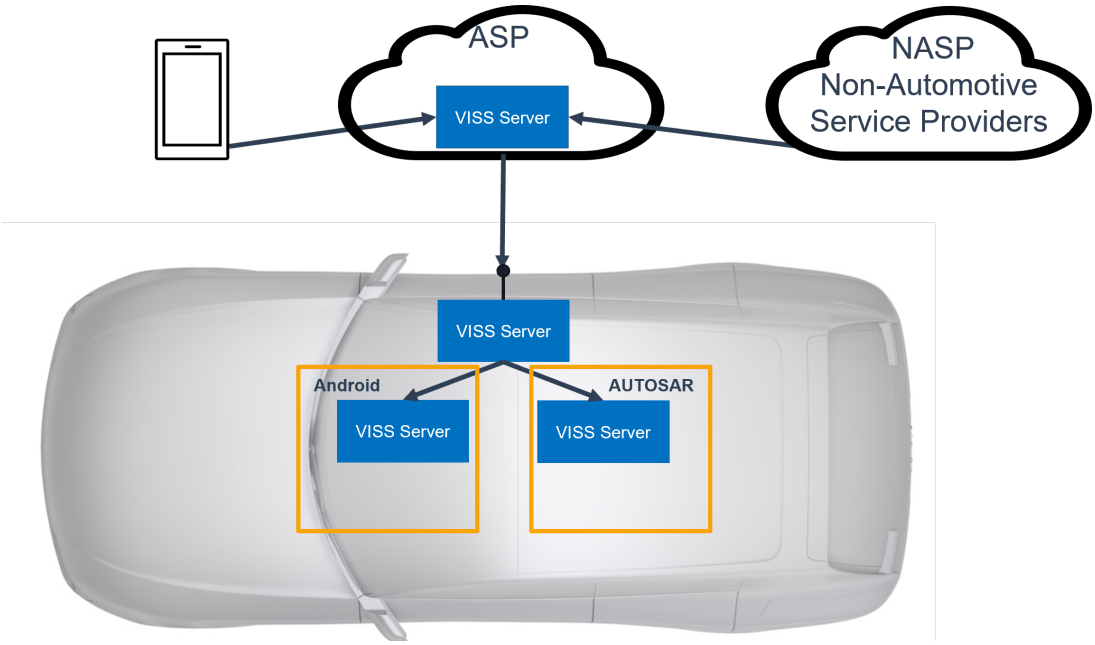

Figure 6.3: Example of Cascaded VISS server deployment

7 Security

The Automotive API is an additional entry point to the vehicle. It makes in-vehicle data and capabilities accessible and allows to change their state. Therefore it needs to protect confidentiality, integrity, and authenticity of the communication and control the system performance. The VISS protects confidentiality of the sensitive data requiring TLS v1.2 (but without any specific cipher suites, leaving the decision to the integrators of such solution) and access control.

At this moment Automotive API Gateway has no defined configuration of the TLS as it depends on the Automotive API Gateway implementation. It is possible that an Automotive API Gateway will be implemented on such way that it will use open-source solution for VISS which does not use AUTOSAR Crypto API. Maybe in the next releases such configuration will be added.

TLS configuration for the in-vehicle network and external interfaces can be different.

The VISS protocol designers propose the infrastructure and message flow for access control in a non-normative section of the documentation. The diagram below presents the border between the mandatory and the non-mandatory elements:

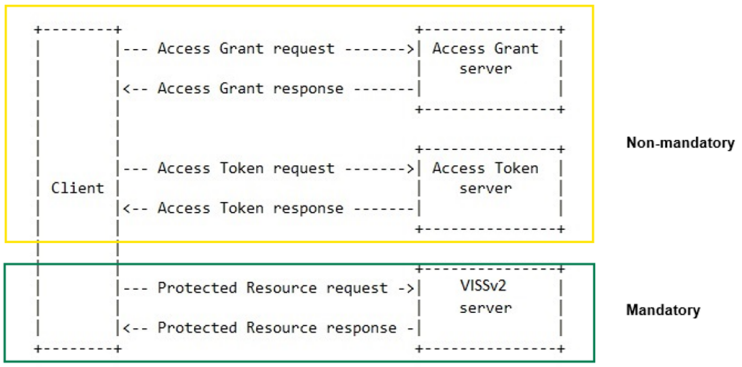

Figure 7.1: The Abstract protocol flow. (Source: [3]; Colored boxes are added)

An important aspect of such a systems that is not shown in the diagram is a consent management which controls the lifetime of the user consents for access/change data. The VISS specification proposes the External Consent Framework for consent management, but it does not specify how it obtains consent status.

For tampering protection and for protecting against man-in-the-middle attacks on the in-vehicle communication between the Automotive API Gateway and AUTOSAR applications the following technologies can be used. They are all configurable using AUTOSAR methodology:

- MACsec,
- IPsec,
- SecOC,

- TLS,
- DDS has built-in security features.

The Automotive API Gateway need to mitigate the risk of overloading the system. For that it can limit e.g., the maximal number of the clients, the requests count and the overall message count. For the cloud interactions the deployment of cascaded VISS servers can simplify this process as the gateway will then interact with only one client who performs load balancing, additional security checks and other actions requiring more computational power.