Web API for Vehicle Data

Web API for Vehicle Data

1. Introduction

The Web API for Vehicle Data defines the interfaces for web applications in IVI system to access vehicle data delivered via vehicle bus. The types of supported data are based on the survey of GENIVI OEMs. Currently, Only P1 and P2 Vehicle APIs are included in this specification.

2. Concepts and Considerations

Vehicle data varies depending on vehicles and the policies of OEM. Considering that, it is not easy to decide which types of vehicle data are mandatory. So, if the most common parts are agreed to be included in the standardization, just few types of data might be allowed. Instead, a different approach is required. For better utilization, it is better to include various OEMs vehicle data points as optional. It can fit not only for an application which uses limited types of vehicle data, but also for HMI which requires extensive vehicle data.

For flexibility, it needs to reduce a number of APIs by using more generalized APIs (irrespective of data types) using polymorphism.

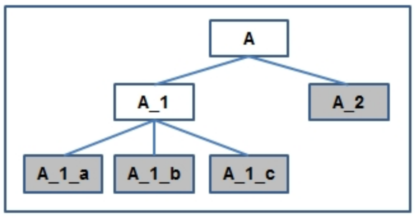
In addition, to handle many types of data effectively, a hierarchical access for a group of multiple data is provided. It can increase the difficulty in implementing user agent which provides an interface for web applications. But, it will help to make web application more effectively.

Another important characteristic of vehicle data is that most of vehicle data is a transient status. So, "get" operation alone is not sufficient for vehicle data. Additionally, vehicle data monitoring APIs (based on Publish-subscribe pattern) are required. JavaScript supports it by event handler and all interfaces that contain vehicle data are designed to be inherited from DOM Event interface.

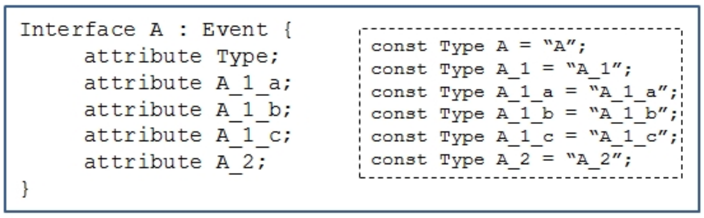
Apart from that, some vehicle data related to speed or positioning can come very frequently from vehicle bus. It can cause serious overhead on the entire system if it is delivered as it is. Therefore the value update frequency must be adjusted by data provider (plug-in, service, browser component, etc.) in accordance with the system performance and purpose.

3. API Description

3.1. The method for handling multiple data at a time in JavaScript



Tree representation of data structure



```
Interface A : Event {
     attribute Type;
     attribute A_1_a;
     attribute A_1_b;
     attribute A_1_c;
     attribute A_2;
}
```

```
const Type A = “A”;
const Type A_1 = “A_1”;
const Type A_1_a = “A_1_a”;
const Type A_1_b = “A_1_b”;
const Type A_1_c = “A_1_c”;
const Type A_2 = “A_2”;
```

Interface definition of data structure

- All real data (attribute) are located at leaf nodes; A_1_a, A_1_b, A_1_c, A_2
- A node having one or more child nodes is defined as an ID to indicate a group of data; A, A_1
- All node are defined as a corresponding const Type ID
- Special attribute "Type" is used as an ID to identify the intended type and the range of validity of data
    - If Type is "A_1_b", the attribute A_1_b is only valid
    - If Type is "A_1", it means that attributes A_1_a, A_1_b and A_1_c are valid
    - If Type is "A", it means that all 4 attributes are valid
- When exchanging data, the whole interface (object A) is handed over. But we'll use it as various virtual type by own rules.
    - If a sender sets Type as "A_1", the receiver treats it as a type "A_1" object as Type is indicating.
    - JavaScript don't require type-casting. The only thing the web application has to do is accessing only valid attributes.
    - Even though the other attributes are accessible, the validity of values is not guaranteed. It can have the last value or garbage value.
    - If a web application tries to access A_2 attribute although the Type is "A_1", it doesn't cause any error but its value is invalid.

- Example Code

```
function handleInterfaceA(objA) {
  if (objA.type == "A_1") {
   console.log("value A_1_a = "+objA.A_1_a); // It's valid.
   console.log("value A_1_b = "+objA.A_1_b); // It's valid.
   console.log("value A_1_c = "+objA.A_1_c); // It's valid.
   console.log("value A_2 = "+objA.A_2);  // It's possible but the value is invalid in our
rules.
  }
  else if (objA.type == "A_2") {
   console.log("value A_2 = "+objA.A_2);  // It's valid.
  }
 }
```

3.2. Common rule

```
[NoInterfaceObject]
interface VehicleEvent : Event {
};

interface VehicleInfoEvent : VehicleEvent {};
interface RunningStatusEvent : VehicleEvent {};
interface MaintenanceEvent : VehicleEvent {};
interface PersonalizationEvent : VehicleEvent {};
interface DrivingSafetyEvent : VehicleEvent {};
interface VisionSystemEvent : VehicleEvent {};
interface ParkingEvent : VehicleEvent {};
interface ClimateEnvironmentEvent : VehicleEvent {};
```

All interfaces for data exchange are defined to inherit **`VehicleEvent`** interface. And all vehicle data belong to a type of **`VehicleEvent`** and can be accessed as an attribute of that. It's for handling various vehicle data in uniform way, and also for allowing JavaScript event handler to access vehicle data because **`VehicleEvent`** is inherited from **`Event`** interface.

Even though we want to pass only a piece of data type, we have to pass the whole **`VehicleEvent`** interface type object which contains the data. It can cause tens of bytes overhead. It's the trade-off for some convenience. Another issue is how we can know the exact purpose of passing data. To do that, special purpose of attribute is provided and that's **`VehicleEventType`**.

3.3. Get/Set operation

```
[NoInterfaceObject]
interface VehicleInterface : EventTarget {
 void get(VehicleEventType type, VehicleDataHandler handler, ErrorCallback errorCB);
 void set(VehicleEventType type, VehicleEvent data, SuccessCallback successCB, ErrorCallback
errorCB);
};
```

3.3.1. Get a single vehicle data

To get a certain specific type of data, web application can use a detailed **`VehicleEventType`**.
Vehicle data is delivered via an unified object (**`JSObject`**) containing a set of vehicle data. But only the requested data is guaranteed of its validity.

- Example Code

```
// Define constants for transmissionGearType
var TRANSMISSIONGEARTYPE_AUTO = 1;
var TRANSMISSIONGEARTYPE_MANUAL = 2;

// Get a transmission gear type from VehicleInfoEvent
vehicle.get("vehicleinfo_transmissiongeartype", handleVehicleData, handleError)
function handleVehicleData(data) {
 if (data.transmissionGearType == TRANSMISSIONGEARTYPE_AUTO) {
  console.log("Automatic transmission equipped");
 } else if (data.transmissionGearType == TRANSMISSIONGEARTYPE_MANUAL) {
  console.log("Manual transmission equipped");
 }
}
```

3.3.2. Get multiple vehicle data at a time

If web applications want to get multiple vehicle data in a certain category at a time, it can use the upper level **`VehicleEventType`**.
If any of vehicle events under the upper level **`VehicleEventType`** has any type of trouble, an error callback is called.

- Example Code

```
// Define constants for tirePressureStatus
var TIREPRESSURESTATUS_NORMAL = 0;
var TIREPRESSURESTATUS_LOW = 1;
var TIREPRESSURESTATUS_HIGH = 2;

// Get tire pressure status information from MaintenanceEvent
vehicle.get("maintenance_tirepressurestatus", handleVehicleData, handleError)
function handleVehicleData(data) {
 if ((data.tirePressureStatusFrontLeft != 0) || (data.tirePressureStatusFrontRight != 0) ||
(data.tirePressureStatusRearLeft != 0) || (data.tirePressureStatusRearRight != 0)) {
  console.log("Check your tire pressure!");
 }
}
```

In the example code, using **`"maintenance_tirepressurestatus"`** is used for upper level **`VehicleEventType`** of **`"_FRONTLEFT"`**, **`"_FRONTRIGHT"`**, **`"_REARLEFT"`**, and **`"_REARRIGHT"`**.
The delievered data contains all four types of lower level vehicle data.

```
...
  const VehicleEventType MAINTENANCE_TIREPRESSURESTATUS = "maintenance_tirepressurestatus";
  const VehicleEventType MAINTENANCE_TIREPRESSURESTATUS_FRONTLEFT =
"maintenance_tirepressurestatus_frontleft";
  const VehicleEventType MAINTENANCE_TIREPRESSURESTATUS_FRONTRIGHT =
"maintenance_tirepressurestatus_frontright";
  const VehicleEventType MAINTENANCE_TIREPRESSURESTATUS_REARLEFT =
"maintenance_tirepressurestatus_rearleft";
  const VehicleEventType MAINTENANCE_TIREPRESSURESTATUS_REARRIGHT =
"maintenance_tirepressurestatus_rearright";
...
```

3.3.3. Set a single vehicle data

Setting value means controlling vehicle functionality. It depends on OEM's decision what kinds of data are allowed to be set.
Of course it's very critical function and need to be used carefully. It's recommended to allow it only to trusted web application.
The result of set operation comes as a callback - handle Success.

> The success callback means only that the set command is transferred to vehicle bus successfully, not being accepted by the target ECU and the real value is changed.
> To know the real result, web application need to check the vehicle data value again whether it's changed or not.

- Example Code

```
// Define constants for drivingMode
var DRIVINGMODE_COMFORT = 1;
var DRIVINGMODE_AUTO = 2;
var DRIVINGMODE_SPORT = 3;
var DRIVINGMODE_ECO = 4;
var DRIVINGMODE_MANUAL = 5;
// Monitor door open status from DrivingSafetyEvent
data = new Object();
data.drivingMode = DRIVINGMODE_SPORT;
vehicle.set("personalization_drivingmode", data, handleSuccess, handleError);
function handleSuccess() {
 console.log("Setting driving mode command is successfully sent to the vehicle bus.");
}
```

3.3.4. Set multiple vehicle data at a time

If the proper **`VehicleEventType`** is used, it can be also allowed to set a group of data at a time.

> **Consideration for OEMs**
> It is recommended to provide this feature only when attributes are associated with one real vehicle control message.
> It means that the granularity of set operation needs to be aligned with the real vehicle message.

3.4. Using Event Handler to monitor vehicle data

Usual way of using JavaScript event handler is supported in the same manner.
It also use the upper level of **`VehicleEventType`** to monitor a group of vehicle data.

- Example Code

```
// Monitor door open status from DrivingSafetyEvent
vehicle.addEventListener("drivingsafety_dooropenstatus", handleVehicleData, false);
// This function is called when any of door open status is changed
function handleVehicleData(data) {
 if (data.type == "drivingsafety_dooropenstatus_driver") {
  if (data.doorOpenStatusDriver) {
   console.log("The driver door is opened.");
  } else {
   console.log("The driver door is closed.");
  }
 } else if (data.type == "drivingsafety_dooropenstatus_passenger") {
 }
}
```

3.5. getSupportedEventTypes

Since most of attributes of vehicle data will be supported optionally depending on OEMs, we need to provide a way to check whether a certain data is supported or not for web applications.
Web Applications can know it if **`UNKNOWN`** error is returned when it requests a certain data.

```
[NoInterfaceObject]
interface VehicleInterface : EventTarget {
 VehicleEventType[] getSupportedEventTypes(VehicleEventType type, boolean writable);
};
```

The other way is to use **`getSupportedEventTypes`** method.

If the method is called with a **VehicleEventType** parameter, it returns an array of all **VehicleEventType** objects which belong to the type. It means all sub-nodes of the type.
It allows web applications to know which types are supported in the system and to work in accordance with the result.

Another parameter **writable** indicates whether the types are writable or readable.
Since the same **VehicleEventType** is used for both set and get, getSupportedEventTypes needs to be called with this parameter.

> Even though the web application doesn't have a proper permission to access the attributes, **getSupportedEventTypes** could return it as supported.
> Depending on how to handle this case, the result could be different as UNKNOWN or ACCESS_DENIED when the web application tries to access the data.

- Example Code 1: Query by single VehicleEventType

```
// Define constants for vehicleType
var VEHICLETYPE_SEDAN = 1;
var VEHICLETYPE_COUPE = 2;
var VEHICLETYPE_CABRIOLET = 3;
var VEHICLETYPE_ROADSTER = 4;
var VEHICLETYPE_SUV = 5;
var VEHICLETYPE_TRUCK = 6;

var supportedEventType = vehicle.getSupportedEventTypes("vehicleinfo_vehicletype", false);
if (supportedEventType == "") {
 // Show the default vehicle image when the vehicle type is not supported
 document.getElementById("imgVehicle").src = "img/vehicleDefault.jpg";
 }
} else {
    vehicle.get("vehicleinfo_vehicletype", handleVehicleData, handleError);
    handleVehicleData(data) {
        if (data.vehicleType == 1) {
             document.getElementById("imgVehicle").src = "img/vehicleSedan.jpg";
        } else if (data.vehicleType == 2) {
             document.getElementById("imgVehicle").src = "img/vehicleCoupe.jpg";
        } else if (data.vehicleType == 3) {
             document.getElementById("imgVehicle").src = "img/vehicleCabriole.jpg";
        } else if (data.vehicleType == 4) {
             document.getElementById("imgVehicle").src = "img/vehicleRoadster.jpg";
        } else if (data.vehicleType == 5) {
             document.getElementById("imgVehicle").src = "img/vehicleSUV.jpg";
        } else if (data.vehicleType == 6) {
             document.getElementById("imgVehicle").src = "img/vehicleTruck.jpg";
        } else {
             document.getElementById("imgVehicle").src = "img/vehicleDefault.jpg";
        }
    }
}
```

- Example Code 2: Query by group VehicleEventType

```
var supportedEventType = vehicle.getSupportedEventTypes("vehicleinfo_doortype", false);
for (itr = 0; itr < supportedEventType.length; itr++) {
 if (supportedEventType[itr] == "vehicleinfo_doortype_3rdrow") {
  // Do the things for the vehicle with 3rd row doors
  console.log("This vehicle has 3rd row doors.");
 }
}
```

4. Interfaces

4.1 General

4.1.1 VehicleError

```
interface VehicleError : Error {
 const short ACCESS_DENIED = 1;
 const short NOT_AVAILABLE = 2;
 const short UNKNOWN = 0;
};
```

The **`VehicleError`** interface encapsulates all errors in accessing of **`VehicleEvent`** objects.

4.1.1.1 Constants

`ACCESS_DENIED` of type short
Access to the requested method was denied at the implementation or by the user.

`NOT_AVAILABLE` of type short
Access to the data was not available temporary.

`UNKNOWN` of type short
An unknown error occurred, or the requested method is not supported by the current implementation.

4.1.2 VehicleEvent

```
[NoInterfaceObject]
interface VehicleEvent : Event {
};
```

The interface defines a generic event for vehicle data specific events.

4.1.3 SuccessCallback

```
[NoInterfaceObject]
callback interface SuccessCallback {
 void onSuccess();
};
```

4.1.3.1 Methods

`onSuccess`
This is the wrapper interface for callbacks indicating success of the **`set()`** operation.
No parameters.
Return type: void

4.1.4 ErrorCallback

```
[NoInterfaceObject]
callback interface ErrorCallback {
 void onError(VehicleError error);
};
```

4.1.4.1 Methods

`onError`
This is the wrapper interface for callbacks indicating failure of the **`get()`** or **`set()`** operation.

Parameter	Type	Nullable	Optional	Description
error	VehicleError	X	X	The vehicle access related error object of an unsuccessful asynchronous operation

Return type: void

4.1.5 VehicleDataHandler

```
interface VehicleDataHandler {
 void handleVehicleData(VehicleEvent data);
};
```

The interface defines the callback method to access vehicle data asynchronously.

4.1.6 Vehicle

```
genivi implements Vehicle;
```

The **`Vehicle`** interface is exposed on the **`genivi`** object.

```
[NoInterfaceObject]
interface Vehicle {
 readonly attribute VehicleInterface vehicle;
};
```

All instances of the **`genivi`** type are defined to also implement the **`Vehicle`** interface.

4.1.6.1 Attributes

`vehicle` of type **`VehicleInterface`**, readonly
The object from which vehicle data is accessed.

4.1.7 VehicleInterface

```
[NoInterfaceObject]
interface VehicleInterface : EventTarget {
 void get(VehicleEventType type, VehicleDataHandler handler, ErrorCallback errorCB);
 void set(VehicleEventType type, VehicleEvent data, SuccessCallback successCB, ErrorCallback
errorCB);
 VehicleEventType[] getSupportedEventTypes(VehicleEventType type, boolean writable);
};
```

The interface defines the object, where the event listener for accessing **`VehicleEvent`** type data can be registered.
The interface is accessible through the **`genivi.vehicle`** object.

4.1.7.1 Methods

`get`
Get a certain data or group of data.
This method takes three arguments.

Parameter	Type	Nullable	Optional	Description
type	VehicleEventType	X	X	ID(data type) of a certain data (or group of data) to retrieve
handler	VehicleDataHandler	X	X	Function to receive the result data when the operation completes successfully
errorCB	ErrorCallback	X	X	Function to call when any type of error occurs

Return type: **`void`**

For details, find the chapter 3. API Description above.

`set`
Set a certain data or group of data.
This method takes four arguments.

Parameter	Type	Nullable	Optional	Description
type	VehicleEventType	X	X	ID(data type) of a certain data (or group of data) to write(or to send to the vehicle bus)
data	VehicleEvent	X	X	A certain data (or group of data) to write(or to send to the vehicle bus)
successCB	SuccessCallback	X	X	Function to call when the operation completes successfully
errorCB	ErrorCallback	X	X	Function to call when any type of error occurs

Return type: **`void`**

For details, find the chapter 3. API Description above.

`getSupportedEventTypes`
Returns an array of all VehicleEventType objects which belong to the type.
This method takes one argument.

Parameter	Type	Nullable	Optional	Description
type	VehicleEventType	X	X	ID(data type) of a certain data (or group of data) to check whether supported or not
writable	boolean	X	X	true for getting writable types supported, false for getting readable types supported

Return type: **`VehicleEventType[]`**

For details, find the chapter 3. API Description above.

4.2 VehicleEvent Type

4.2.1 VehicleInfoEvent

```
interface VehicleInfoEvent : VehicleEvent {
 const VehicleEventType VEHICLEINFO = "vehicleinfo";
 const VehicleEventType VEHICLEINFO_WMI = "vehicleinfo_wmi";
 const VehicleEventType VEHICLEINFO_VIN = "vehicleinfo_vin";
 const VehicleEventType VEHICLEINFO_VEHICLETYPE = "vehicleinfo_vehicletype";
 const VehicleEventType VEHICLEINFO_DOORTYPE = "vehicleinfo_doortype";
 const VehicleEventType VEHICLEINFO_DOORTYPE_1STROW = "vehicleinfo_doortype_1strow";
 const VehicleEventType VEHICLEINFO_DOORTYPE_2NDROW = "vehicleinfo_doortype_2ndrow";
 const VehicleEventType VEHICLEINFO_DOORTYPE_3RDROW = "vehicleinfo_doortype_3rdrow";
 const VehicleEventType VEHICLEINFO_FUELTYPE = "vehicleinfo_fueltype";
 const VehicleEventType VEHICLEINFO_TRANSMISSIONGEARTYPE = "vehicleinfo_transmissiongeartype";
 const VehicleEventType VEHICLEINFO_WHEELINFO = "vehicleinfo_wheelinfo";
 const VehicleEventType VEHICLEINFO_WHEELINFO_RADIUS = "vehicleinfo_wheelinfo_radius";
 const VehicleEventType VEHICLEINFO_WHEELINFO_TRACK = "vehicleinfo_wheelinfo_track";

 const unsigned short VEHICLETYPE_SEDAN = 1;
 const unsigned short VEHICLETYPE_COUPE = 2;
 const unsigned short VEHICLETYPE_CABRIOLET = 3;
 const unsigned short VEHICLETYPE_ROADSTER = 4;
 const unsigned short VEHICLETYPE_SUV = 5;
 const unsigned short VEHICLETYPE_TRUCK = 6;

 const octet FUELTYPE_GASOLINE = 0x01;
 const octet FUELTYPE_METHANOL= 0x02;
 const octet FUELTYPE_ETHANOL = 0x03;
 const octet FUELTYPE_DIESEL= 0x04;
 const octet FUELTYPE_LPG = 0x05;
 const octet FUELTYPE_CNG = 0x06;
 const octet FUELTYPE_PROPANE = 0x07;
 const octet FUELTYPE_ELECTRIC = 0x08;
 const octet FUELTYPE_BIFUELRUNNINGGASOLINE = 0x09;
 const octet FUELTYPE_BIFUELRUNNINGMETHANOL = 0x0A;
 const octet FUELTYPE_BIFUELRUNNINGETHANOL = 0x0B;
 const octet FUELTYPE_BIFUELRUNNINGLPG = 0x0C;
 const octet FUELTYPE_BIFUELRUNNINGCNG = 0x0D;
 const octet FUELTYPE_BIFUELRUNNINGPROP = 0x0E;
 const octet FUELTYPE_BIFUELRUNNINGELECTRICITY = 0x0F;
 const octet FUELTYPE_BIFUELMIXEDGASELECTRIC= 0x10;
 const octet FUELTYPE_HYBRIDGASOLINE = 0x11;
 const octet FUELTYPE_HYBRIDETHANOL = 0x12;
 const octet FUELTYPE_HYBRIDDIESEL = 0x13;
 const octet FUELTYPE_HYBRIDELECTRIC = 0x14;
 const octet FUELTYPE_HYBRIDMIXEDFUEL = 0x15;
 const octet FUELTYPE_HYBRIDREGENERATIVE = 0x16;

 const unsigned short TRANSMISSIONGEARTYPE_AUTO = 1;
 const unsigned short TRANSMISSIONGEARTYPE_MANUAL = 2;
 const unsigned short TRANSMISSIONGEARTYPE_CVT = 3;

 readonly attribute VehicleEventType type;
 readonly attribute DOMString wmi;
 readonly attribute DOMString vin;
 readonly attribute unsigned short? vehicleType;
 readonly attribute unsigned short? doorType1stRow;
 readonly attribute unsigned short? doorType2ndRow;
 readonly attribute unsigned short? doorType3rdRow;
 readonly attribute octet? fuelType;
 readonly attribute unsigned short? transmissionGearType;
 readonly attribute double? wheelInfoRadius;
 readonly attribute double? wheelInfoTrack;
};
```

4.2.1.1 Attributes

`type` of type VehicleEventType, readonly
This attribute represents a intended type of this VehicleInfoEvent.

`wmi` of type DOMString, readonly
This attribute contains the WMI(World Manufacturer Identifier as defined by SAE) information for this vehicle. It's 3 bytes long characters.

`vin` of type DOMString, readonly
This attribute contains the VIN(Vehicle Identification Number as defined by ISO 3779) information for this vehicle. It's 17 bytes long characters.

`vehicleType` of type unsigned short, readonly, nullable
This attribute indicates what type of this vehicle. The value is one of the following values.

```
const unsigned short VEHICLETYPE_SEDAN = 1;
const unsigned short VEHICLETYPE_COUPE = 2;
const unsigned short VEHICLETYPE_CABRIOLET = 3;
const unsigned short VEHICLETYPE_ROADSTER = 4;
const unsigned short VEHICLETYPE_SUV = 5;
const unsigned short VEHICLETYPE_TRUCK = 6;
```

`doorType1stRow` of type unsigned short, readonly, nullable
This attribute contains number of doors in first row.

`doorType2ndRow` of type unsigned short, readonly, nullable
This attribute contains number of doors in second row.

`doorType3rdRow` of type unsigned short, readonly, nullable
This attribute contains number of doors in third row.

`fuelType` of type octet, readonly, nullable
This attribute indicates what type of fuel this vehicle uses. Types are defined by OBD-II.

`transmissionGearType` of type unsigned short, readonly, nullable
This attribute indicates the transmission gear type of this vehicle. The value is one of the following values.

```
const unsigned short TRANSMISSIONGEARTYPE_AUTO = 1;
const unsigned short TRANSMISSIONGEARTYPE_MANUAL = 2;
const unsigned short TRANSMISSIONGEARTYPE_CVT = 3;
```

`wheelInfoRadius` of type double, readonly, nullable
This attribute contains the radius of Wheel. It's information for GPS Dead Reckoning.

`wheelInfoTrack` of type double, readonly, nullable
This attribute contains the wheel track. It's information for GPS Dead Reckoning.

4.2.2 RunningStatusEvent

```
interface RunningStatusEvent : VehicleEvent {
 const VehicleEventType RUNNINGSTATUS = "runningstatus";
 const VehicleEventType RUNNINGSTATUS_VEHICLEPOWERMODE = "runningstatus_vehiclepowermode";
 const VehicleEventType RUNNINGSTATUS_SPEEDOMETER = "runningstatus_speedometer";
 const VehicleEventType RUNNINGSTATUS_ENGINESPEED = "runningstatus_enginespeed";
 const VehicleEventType RUNNINGSTATUS_TRIPMETER = "runningstatus_tripmeter";
 const VehicleEventType RUNNINGSTATUS_TRIPMETER_1 = "runningstatus_tripmeter_1";
 const VehicleEventType RUNNINGSTATUS_TRIPMETER_2 = "runningstatus_tripmeter_2";
 const VehicleEventType RUNNINGSTATUS_TRIPMETER_1_MILEAGE = "runningstatus_tripmeter_1_mileage";
 const VehicleEventType RUNNINGSTATUS_TRIPMETER_2_MILEAGE = "runningstatus_tripmeter_2_mileage";
 const VehicleEventType RUNNINGSTATUS_TRIPMETER_1_AVERAGESPEED =
"runningstatus_tripmeter_1_averagespeed";
 const VehicleEventType RUNNINGSTATUS_TRIPMETER_2_AVERAGESPEED =
"runningstatus_tripmeter_2_averagespeed";
 const VehicleEventType RUNNINGSTATUS_TRIPMETER_1_FUELCONSUMPTION =
"runningstatus_tripmeter_1_fuelconsumption";
 const VehicleEventType RUNNINGSTATUS_TRIPMETER_2_FUELCONSUMPTION =
"runningstatus_tripmeter_2_fuelconsumption";
 const VehicleEventType RUNNINGSTATUS_TRANSMISSIONGEARSTATUS =
"runningstatus_transmissiongearstatus";
 const VehicleEventType RUNNINGSTATUS_CRUISECONTROL = "runningstatus_cruisecontrol";
 const VehicleEventType RUNNINGSTATUS_CRUISECONTROL_STATUS = "runningstatus_cruisecontrol_status";
 const VehicleEventType RUNNINGSTATUS_CRUISECONTROL_SPEED = "runningstatus_cruisecontrol_speed";
 const VehicleEventType RUNNINGSTATUS_WHEELBRAKE = "runningstatus_wheelbrake";
 const VehicleEventType RUNNINGSTATUS_LIGHTSSTATUS = "runningstatus_lightsstatus";
 const VehicleEventType RUNNINGSTATUS_LIGHTSSTATUS_HEAD = "runningstatus_lightsstatus_head";
 const VehicleEventType RUNNINGSTATUS_LIGHTSSTATUS_HIGHBEAM =
"runningstatus_lightsstatus_highbeam";
 const VehicleEventType RUNNINGSTATUS_LIGHTSSTATUS_TURNLEFT =
"runningstatus_lightsstatus_turnleft";
 const VehicleEventType RUNNINGSTATUS_LIGHTSSTATUS_TURNRIGHT =
"runningstatus_lightsstatus_turnright";
 const VehicleEventType RUNNINGSTATUS_LIGHTSSTATUS_BRAKE = "runningstatus_lightsstatus_brake";
 const VehicleEventType RUNNINGSTATUS_LIGHTSSTATUS_FOGFRONT =
"runningstatus_lightsstatus_fogfront";
 const VehicleEventType RUNNINGSTATUS_LIGHTSSTATUS_FOGREAR = "runningstatus_lightsstatus_fogrear";
 const VehicleEventType RUNNINGSTATUS_LIGHTSSTATUS_HAZARD = "runningstatus_lightsstatus_hazard";
 const VehicleEventType RUNNINGSTATUS_LIGHTSSTATUS_PARKING = "runningstatus_lightsstatus_parking";
 const VehicleEventType RUNNINGSTATUS_INTERIORLIGHTSSTATUS = "runningstatus_interiorlightsstatus";
 const VehicleEventType RUNNINGSTATUS_INTERIORLIGHTSSTATUS_DRIVER =
"runningstatus_interiorlightsstatus_driver";
 const VehicleEventType RUNNINGSTATUS_INTERIORLIGHTSSTATUS_PASSENGER =
"runningstatus_interiorlightsstatus_passenger";
 const VehicleEventType RUNNINGSTATUS_INTERIORLIGHTSSTATUS_CENTER =
"runningstatus_interiorlightsstatus_center";
 const VehicleEventType RUNNINGSTATUS_AUTOMATICHEADLIGHTS = "runningstatus_automaticheadlights";
 const VehicleEventType RUNNINGSTATUS_DYNAMICHIGHBEAM = "runningstatus_dynamichighbeam";
 const VehicleEventType RUNNINGSTATUS_HORN = "runningstatus_horn";
 const VehicleEventType RUNNINGSTATUS_CHIME = "runningstatus_chime";
 const VehicleEventType RUNNINGSTATUS_FUEL = "runningstatus_fuel";
 const VehicleEventType RUNNINGSTATUS_ESTIMATEDRANGE = "runningstatus_estimatedrange";
 const VehicleEventType RUNNINGSTATUS_ENGINEOIL = "runningstatus_engineoil";
 const VehicleEventType RUNNINGSTATUS_ENGINEOIL_REMAINING = "runningstatus_engineoil_remaining";
 const VehicleEventType RUNNINGSTATUS_ENGINEOIL_CHANGE = "runningstatus_engineoil_change";
 const VehicleEventType RUNNINGSTATUS_ENGINEOIL_TEMP = "runningstatus_engineoil_temp";
 const VehicleEventType RUNNINGSTATUS_ENGINECOOLANT = "runningstatus_enginecoolant";
 const VehicleEventType RUNNINGSTATUS_ENGINECOOLANT_LEVEL = "runningstatus_enginecoolant_level";
 const VehicleEventType RUNNINGSTATUS_ENGINECOOLANT_TEMP = "runningstatus_enginecoolant_temp";
 const VehicleEventType RUNNINGSTATUS_STEERINGWHEELANGLE = "runningstatus_steeringwheelangle";

 const unsigned short VEHICLEPOWERMODE_OFF = 1;
 const unsigned short VEHICLEPOWERMODE_ACC = 2;
 const unsigned short VEHICLEPOWERMODE_RUN = 3;
 const unsigned short VEHICLEPOWERMODE_IGNITION = 4;

 const unsigned short TRANSMISSIONGEARSTATUS_NEUTRAL = 0;
 const unsigned short TRANSMISSIONGEARSTATUS_MANUAL1 = 1;
 const unsigned short TRANSMISSIONGEARSTATUS_MANUAL2 = 2;
 const unsigned short TRANSMISSIONGEARSTATUS_MANUAL3 = 3;
 const unsigned short TRANSMISSIONGEARSTATUS_MANUAL4 = 4;
 const unsigned short TRANSMISSIONGEARSTATUS_MANUAL5 = 5;
 const unsigned short TRANSMISSIONGEARSTATUS_MANUAL6 = 6;
 const unsigned short TRANSMISSIONGEARSTATUS_MANUAL7 = 7;
 const unsigned short TRANSMISSIONGEARSTATUS_MANUAL8 = 8;
 const unsigned short TRANSMISSIONGEARSTATUS_MANUAL9 = 9;
 const unsigned short TRANSMISSIONGEARSTATUS_MANUAL10 = 10;
 const unsigned short TRANSMISSIONGEARSTATUS_AUTO1 = 11;
 const unsigned short TRANSMISSIONGEARSTATUS_AUTO2 = 12;
 const unsigned short TRANSMISSIONGEARSTATUS_AUTO3 = 13;
 const unsigned short TRANSMISSIONGEARSTATUS_AUTO4 = 14;
 const unsigned short TRANSMISSIONGEARSTATUS_AUTO5 = 15;
 const unsigned short TRANSMISSIONGEARSTATUS_AUTO6 = 16;
 const unsigned short TRANSMISSIONGEARSTATUS_AUTO7 = 17;
 const unsigned short TRANSMISSIONGEARSTATUS_AUTO8 = 18;
 const unsigned short TRANSMISSIONGEARSTATUS_AUTO9 = 19;
 const unsigned short TRANSMISSIONGEARSTATUS_AUTO10 = 20;
 const unsigned short TRANSMISSIONGEARSTATUS_DRIVE = 32;
 const unsigned short TRANSMISSIONGEARSTATUS_PARKING = 64;
 const unsigned short TRANSMISSIONGEARSTATUS_REVERSE = 128;

 const unsigned short WHEELBRAKE_IDLE = 1;
 const unsigned short WHEELBRAKE_ENGAGED = 2;
 const unsigned short WHEELBRAKE_MALFUNCTION = 3;

 const unsigned short ENGINECOOLANTLEVEL_NORMAL = 0;
 const unsigned short ENGINECOOLANTLEVEL_LOW = 1;

 readonly attribute VehicleEventType type;
 readonly attribute unsigned short? vehiclePowerMode;
 readonly attribute unsigned short speedometer;
 readonly attribute unsigned short? engineSpeed;
 readonly attribute unsigned long? tripMeter1Mileage;
 readonly attribute unsigned long? tripMeter2Mileage;
 readonly attribute unsigned short? tripMeter1AverageSpeed;
 readonly attribute unsigned short? tripMeter2AverageSpeed;
 readonly attribute unsigned long? tripMeter1FuelConsumption;
 readonly attribute unsigned long? tripMeter2FuelConsumption;
 readonly attribute unsigned short? transmissionGearStatus;
 readonly attribute boolean? cruiseControlStatus;
 readonly attribute unsigned short? cruiseControlSpeed;
 readonly attribute unsigned short? wheelBrake;
 readonly attribute boolean? lightsStatusHead;
 readonly attribute boolean? lightsStatusHighBeam;
 readonly attribute boolean? lightsStatusTurnLeft;
 readonly attribute boolean? lightsStatusTurnRight;
 readonly attribute boolean? lightsStatusBrake;
 readonly attribute boolean? lightsStatusFogFront;
 readonly attribute boolean? lightsStatusFogRear;
 readonly attribute boolean? lightsStatusHazard;
 readonly attribute boolean? lightsStatusParking;
 readonly attribute boolean? interiorLightsStatusDriver;
 readonly attribute boolean? interiorLightsStatusPassenger;
 readonly attribute boolean? interiorLightsStatusCenter;
 readonly attribute boolean? automaticHeadlights;
 readonly attribute boolean? dynamicHighBeam;
 readonly attribute boolean? horn;
 readonly attribute boolean? chime;
 readonly attribute unsigned short fuel;
 readonly attribute unsigned long? estimatedRange;
 readonly attribute unsigned short? engineOilRemaining;
 readonly attribute boolean? engineOilChange;
 readonly attribute short? engineOilTemp;
 readonly attribute unsigned short? engineCoolantLevel;
```

```
 readonly attribute short? engineCoolantTemp;
 readonly attribute short? steeringWheelAngle;
};
```

4.2.2.1 Attributes

`type` of type VehicleEventType, readonly
This attribute represents a intended type of this RunningStatusEvent.

`vehiclePowerMode` of type unsigned short, readonly, nullable
This attribute exposes the current vehicle power mode.
It can represents the key position in the key cylinder or any type of power mode in the IVI system.
The value is one of the following values.

```
const unsigned short VEHICLEPOWERMODE_OFF = 1;
const unsigned short VEHICLEPOWERMODE_ACC = 2;
const unsigned short VEHICLEPOWERMODE_RUN = 3;
const unsigned short VEHICLEPOWERMODE_IGNITION = 4;
```

`speedometer` of type unsigned short, readonly
This attribute contains the current speed of this vehicle.
The unit of value is km/h or mph, which can be checked by measurementSystem attribute of PersonalizationEvent.

`engineSpeed` of type unsigned short, readonly, nullable
This attribute contains the current speed of engine. The unit is RPM.

`tripMeter1Mileage` of type unsigned long, readonly, nullable
This attribute contains the current mileage of tripmeter 1. The unit of value is km or mile.

`tripMeter2Mileage` of type unsigned long, readonly, nullable
This attribute contains the current mileage of tripmeter 2. The unit of value is km or mile.

`tripMeter1AverageSpeed` of type unsigned short, readonly, nullable
This attribute contains the current average speed of tripmeter 1. The unit of value is km/h or mph.

`tripMeter2AverageSpeed` of type unsigned short, readonly, nullable
This attribute contains the current average speed of tripmeter 2. The unit of value is km/h or mph.

`tripMeter1FuelConsumption` of type unsigned long, readonly, nullable
This attribute contains the current fuel consumption(efficiency) of tripmeter 1. The unit of value is l/100, mpg or km/l.

`tripMeter2FuelConsumption` of type unsigned long, readonly, nullable
This attribute contains the current fuel consumption(efficiency) of tripmeter 2. The unit of value is l/100, mpg or km/l.

`transmissionGearStatus` of type unsigned short, readonly, nullable
This attribute contains the current status of transmission gear.
The value is one of the following values.

```
const unsigned short TRANSMISSIONGEARSTATUS_NEUTRAL = 0;
const unsigned short TRANSMISSIONGEARSTATUS_MANUAL1 = 1;
const unsigned short TRANSMISSIONGEARSTATUS_MANUAL2 = 2;
const unsigned short TRANSMISSIONGEARSTATUS_MANUAL3 = 3;
const unsigned short TRANSMISSIONGEARSTATUS_MANUAL4 = 4;
const unsigned short TRANSMISSIONGEARSTATUS_MANUAL5 = 5;
const unsigned short TRANSMISSIONGEARSTATUS_MANUAL6 = 6;
const unsigned short TRANSMISSIONGEARSTATUS_MANUAL7 = 7;
const unsigned short TRANSMISSIONGEARSTATUS_MANUAL8 = 8;
const unsigned short TRANSMISSIONGEARSTATUS_MANUAL9 = 9;
const unsigned short TRANSMISSIONGEARSTATUS_MANUAL10 = 10;
const unsigned short TRANSMISSIONGEARSTATUS_AUTO1 = 11;
const unsigned short TRANSMISSIONGEARSTATUS_AUTO2 = 12;
const unsigned short TRANSMISSIONGEARSTATUS_AUTO3 = 13;
const unsigned short TRANSMISSIONGEARSTATUS_AUTO4 = 14;
const unsigned short TRANSMISSIONGEARSTATUS_AUTO5 = 15;
const unsigned short TRANSMISSIONGEARSTATUS_AUTO6 = 16;
const unsigned short TRANSMISSIONGEARSTATUS_AUTO7 = 17;
const unsigned short TRANSMISSIONGEARSTATUS_AUTO8 = 18;
const unsigned short TRANSMISSIONGEARSTATUS_AUTO9 = 19;
const unsigned short TRANSMISSIONGEARSTATUS_AUTO10 = 20;
const unsigned short TRANSMISSIONGEARSTATUS_DRIVE = 32;
const unsigned short TRANSMISSIONGEARSTATUS_PARKING = 64;
const unsigned short TRANSMISSIONGEARSTATUS_REVERSE = 128;
```

`cruiseControlStatus` of type boolean, readonly, nullable
This attribute indicates the status of cruise control.
'true' means cruise control is activated. 'false' means it is deactivated.

`cruiseControlSpeed` of type unsigned short, readonly, nullable
This attribute contains the setting value of desired speed of cruise control. The unit of value is km/h or mph.

`wheelBrake` of type unsigned short, readonly, nullable
This attribute contains the current status of wheel brake.
The value is one of the following values.

```
const unsigned short WHEELBRAKE_IDLE = 1;
const unsigned short WHEELBRAKE_ENGAGED = 2;
const unsigned short WHEELBRAKE_MALFUNCTION = 3;
```

`lightsStatusHead` of type boolean, readonly, nullable
This attribute indicates the status of headlight.
'true' means headlight is on. 'false' means it is off.

`lightsStatusHighBeam` of type boolean, readonly, nullable
This attribute indicates whether headlight is high or low.
'true' means headlight is high. 'false' means it is low.

`lightsStatusTurnLeft` of type boolean, readonly, nullable
This attribute indicates the status of left turn signal.
'true' means left turn signal is on. 'false' means it is off.

`lightsStatusTurnRight` of type boolean, readonly, nullable
This attribute indicates the status of right turn signal.
'true' means right turn signal is on. 'false' means it is off.

`lightsStatusBrake` of type boolean, readonly, nullable
This attribute indicates the status of brake(stop) light. Usually, it corresponds with the status of wheelBrake.
'true' means brake light is on. 'false' means it is off.

`lightsStatusFogFront` of type boolean, readonly, nullable
This attribute indicates the status of front fog light.
'true' means front fog light is on. 'false' means it is off.

`lightsStatusFogRear` of type boolean, readonly, nullable
This attribute indicates the status of rear fog light.
'true' means rear fog light is on. 'false' means it is off.

`lightsStatusHazard` of type boolean, readonly, nullable
This attribute indicates the status of hazard(emergency) light.
'true' means hazard light is on. 'false' means it is off.

`lightsStatusParking` of type boolean, readonly, nullable
This attribute indicates the status of parking light.

'true' means parking light is on. 'false' means it is off.

`interiorLightsStatusDriver` of type boolean, readonly, nullable
This attribute indicates the status of interior(courtesy) light at driver's side.
'true' means driver's interior light is on. 'false' means it is off.

`interiorLightsStatusPassenger` of type boolean, readonly, nullable
This attribute indicates the status of interior(courtesy) light at passenger's side.
'true' means passenger's interior light is on. 'false' means it is off.

`interiorLightsStatusCenter` of type boolean, readonly, nullable
This attribute indicates the status of central interior(courtesy) light.
'true' means central interior light is on. 'false' means it is off.

`automaticHeadlights` of type boolean, readonly, nullable
This attribute indicates whether the automatic headlight control function is activated or not.
'true' means automatic headlight control is activated. 'false' means it is deactivated.

`dynamicHighBeam` of type boolean, readonly, nullable
This attribute indicates whether the dynamic highbeam control function is activated or not.
'true' means dynamic highbeam control is activated. 'false' means it is deactivated.

`horn` of type boolean, readonly, nullable
This attribute indicates the status of horn.
'true' means horn is on. 'false' means it is off.

`chime` of type boolean, readonly, nullable
This attribute indicates the status of chime.
'true' means chime is on. 'false' means it is off.

`fuel` of type unsigned short, readonly
This attribute contains the current percentage of remaining fuel over total capacity. The unit is %.

`estimatedRange` of type unsigned long, readonly, nullable
This attribute represents estimated reachable distance with the available fuel. The unit of value is km or mile.

`engineOilRemaining` of type unsigned short, readonly, nullable
This attribute contains the current percentage of remaining engine oil life.
The value 0% means starvation. 100% means normal.

`engineOilChange` of type boolean, readonly, nullable
This attribute indicates whether the engine oil should be changed soon or not.
'true' means engine oil should be changed soon. 'false' means engine oil is normal now.

`engineOilTemp` of type short, readonly, nullable
This attribute contains the current temperature of engine oil. The unit of value is °C or °F.

`engineCoolantLevel` of type unsigned short, readonly, nullable
This attribute contains the level of engine coolant.
The value is one of the following values.

```
const unsigned short ENGINECOOLANTLEVEL_NORMAL = 0;
const unsigned short ENGINECOOLANTLEVEL_LOW = 1;
```

`engineCoolantTemp` of type short, readonly, nullable
This attribute contains the current temperature of engine oil coolant. The unit of value is °C or °F.

`steeringWheelAngle` of type short, readonly, nullable
This attribute contains the current angle of steering wheel. The range of value is from -400 to +400 degrees.

4.2.3 MaintenanceEvent

```
interface MaintenanceEvent : VehicleEvent {
 const VehicleEventType MAINTENANCE = "maintenance";
 const VehicleEventType MAINTENANCE_ODOMETER = "maintenance_odometer";
 const VehicleEventType MAINTENANCE_TRANSMISSIONOIL = "maintenance_transmissionoil";
 const VehicleEventType MAINTENANCE_TRANSMISSIONOIL_LIFELEVEL =
"maintenance_transmissionoil_lifelevel";
 const VehicleEventType MAINTENANCE_TRANSMISSIONOIL_TEMP = "maintenance_transmissionoil_temp";
 const VehicleEventType MAINTENANCE_BRAKEFLUIDLEVEL = "maintenance_brakefluidlevel";
 const VehicleEventType MAINTENANCE_WASHERFLUIDLEVEL = "maintenance_washerfluidlevel";
 const VehicleEventType MAINTENANCE_MALFUNCTIONINDICATORLAMP =
"maintenance_malfunctionindicatorlamp";
 const VehicleEventType MAINTENANCE_BATTERY = "maintenance_battery";
 const VehicleEventType MAINTENANCE_BATTERY_VOLTAGE = "maintenance_battery_voltage";
 const VehicleEventType MAINTENANCE_BATTERY_CURRENT = "maintenance_battery_current";
 const VehicleEventType MAINTENANCE_TIREPRESSURE = "maintenance_tirepressure";
 const VehicleEventType MAINTENANCE_TIREPRESSURE_FRONTLEFT = "maintenance_tirepressure_frontleft";
 const VehicleEventType MAINTENANCE_TIREPRESSURE_FRONTRIGHT =
"maintenance_tirepressure_frontright";
 const VehicleEventType MAINTENANCE_TIREPRESSURE_REARLEFT = "maintenance_tirepressure_rearleft";
 const VehicleEventType MAINTENANCE_TIREPRESSURE_REARRIGHT = "maintenance_tirepressure_rearright";
 const VehicleEventType MAINTENANCE_TIREPRESSURESTATUS = "maintenance_tirepressurestatus";
 const VehicleEventType MAINTENANCE_TIREPRESSURESTATUS_FRONTLEFT =
"maintenance_tirepressurestatus_frontleft";
 const VehicleEventType MAINTENANCE_TIREPRESSURESTATUS_FRONTRIGHT =
"maintenance_tirepressurestatus_frontright";
 const VehicleEventType MAINTENANCE_TIREPRESSURESTATUS_REARLEFT =
"maintenance_tirepressurestatus_rearleft";
 const VehicleEventType MAINTENANCE_TIREPRESSURESTATUS_REARRIGHT =
"maintenance_tirepressurestatus_rearright";

 const unsigned short TIREPRESSURESTATUS_NORMAL = 0;
 const unsigned short TIREPRESSURESTATUS_LOW = 1;
 const unsigned short TIREPRESSURESTATUS_HIGH = 2;

 readonly attribute VehicleEventType type;
 readonly attribute unsigned long odometer;
 readonly attribute boolean? transmissionOilLifeLevel;
 readonly attribute short? transmissionOilTemp;
 readonly attribute boolean? brakeFluidLevel;
 readonly attribute boolean? washerFluidLevel;
 readonly attribute boolean? malfunctionIndicatorLamp;
 readonly attribute unsigned short? batteryVoltage;
 readonly attribute unsigned short? batteryCurrent;
 readonly attribute unsigned short? tirePressureFrontLeft;
 readonly attribute unsigned short? tirePressureFrontRight;
 readonly attribute unsigned short? tirePressureRearLeft;
 readonly attribute unsigned short? tirePressureRearRight;
 readonly attribute unsigned short? tirePressureStatusFrontLeft;
 readonly attribute unsigned short? tirePressureStatusFrontRight;
 readonly attribute unsigned short? tirePressureStatusRearLeft;
 readonly attribute unsigned short? tirePressureStatusRearRight;
};
```

4.2.3.1 Attributes

`type` of type VehicleEventType, readonly
This attribute represents a intended type of this MaintenanceEvent.

`odometer` of type unsigned long, readonly
This attribute contains the current odometer in this vehicle. The unit of value is km or mile.

`transmissionOilLifeLevel` of type boolean, readonly, nullable
This attribute indicates whether the transmission oil should be checked or not.
'true' means the level of transmission oil is low, so should be checked. 'false' means it is normal now.

`transmissionOilTemp` of type unsigned short, readonly, nullable
This attribute contains the current temperature of transmission oil. The unit of value is °C or °F.

`brakeFluidLevel` of type boolean, readonly, nullable
This attribute indicates whether the brake fluid should be checked or not.
'true' means the level of brake fluid is low, so should be checked. 'false' means it is normal now.

`washerFluidLevel` of type boolean, readonly, nullable

This attribute indicates whether the washer fluid should be refilled or not.
'true' means the level of washer fluid is low, so should be refilled. 'false' means it is normal now.

`malfunctionIndicatorLamp` of type boolean, readonly, nullable
This attribute indicates whether the engine malfunction indicator lamp is on or not.
'true' means the indicator lamp is on, the engine should be checked. 'false' means it is normal now.

`batteryVoltage` of type unsigned short, readonly, nullable
This attribute contains the voltage value of battery. The unit of value is V.

`batteryCurrent` of type unsigned short, readonly, nullable
This attribute contains the current value of battery. The unit of value is AH.

`tirePressureFrontLeft` of type unsigned short, readonly, nullable
`tirePressureFrontRight` of type unsigned short, readonly, nullable
`tirePressureRearLeft` of type unsigned short, readonly, nullable
`tirePressureRearRight` of type unsigned short, readonly, nullable
These attributes contain the current pressure of each tire. The unit of value is PSI.

`tirePressureStatusFrontLeft` of type unsigned short, readonly, nullable
`tirePressureStatusFrontRight` of type unsigned short, readonly, nullable
`tirePressureStatusRearLeft` of type unsigned short, readonly, nullable
`tirePressureStatusRearRight` of type unsigned short, readonly, nullable
These attributes represent the status of each tire.
The value is one of the following values.

```
const unsigned short TIREPRESSURESTATUS_NORMAL = 0;
const unsigned short TIREPRESSURESTATUS_LOW = 1;
const unsigned short TIREPRESSURESTATUS_HIGH = 2;
```

4.2.4 PersonalizationEvent

```
interface PersonalizationEvent : VehicleEvent {
 const VehicleEventType PERSONALIZATION = "personalization";
 const VehicleEventType PERSONALIZATION_KEYID = "personalization_keyid";
 const VehicleEventType PERSONALIZATION_LANGUAGE = "personalization_language";
 const VehicleEventType PERSONALIZATION_MEASUREMENTSYSTEM = "personalization_measurementsystem";
 const VehicleEventType PERSONALIZATION_MEASUREMENTSYSTEMSTRING =
"personalization_measurementsystemstring";
 const VehicleEventType PERSONALIZATION_MEASUREMENTSYSTEMSTRING_FUEL =
"personalization_measurementsystemstring_fuel";
 const VehicleEventType PERSONALIZATION_MEASUREMENTSYSTEMSTRING_DISTANCE =
"personalization_measurementsystemstring_distance";
 const VehicleEventType PERSONALIZATION_MEASUREMENTSYSTEMSTRING_SPEED =
"personalization_measurementsystemstring_speed";
 const VehicleEventType PERSONALIZATION_MEASUREMENTSYSTEMSTRING_CONSUMPTION =
"personalization_measurementsystemstring_consumption";
 const VehicleEventType PERSONALIZATION_MIRROR = "personalization_mirror";
 const VehicleEventType PERSONALIZATION_MIRROR_DRIVER = "personalization_mirror_driver";
 const VehicleEventType PERSONALIZATION_MIRROR_PASSENGER = "personalization_mirror_passenger";
 const VehicleEventType PERSONALIZATION_MIRROR_INSIDE = "personalization_mirror_inside";
 const VehicleEventType PERSONALIZATION_STEERINGWHEELPOSITION =
"personalization_steeringwheelposition";
 const VehicleEventType PERSONALIZATION_STEERINGWHEELPOSITION_SLIDE =
"personalization_steeringwheelposition_slide";
 const VehicleEventType PERSONALIZATION_STEERINGWHEELPOSITION_TILT =
"personalization_steeringwheelposition_tilt";
 const VehicleEventType PERSONALIZATION_DRIVINGMODE = "personalization_drivingmode";
 const VehicleEventType PERSONALIZATION_DRIVERSEATPOSITION = "personalization_driverseatposition";
 const VehicleEventType PERSONALIZATION_DRIVERSEATPOSITION_RECLINESEATBACK =
"personalization_driverseatposition_reclineseatback";
 const VehicleEventType PERSONALIZATION_DRIVERSEATPOSITION_SLIDE =
"personalization_driverseatposition_slide";
 const VehicleEventType PERSONALIZATION_DRIVERSEATPOSITION_CUSHIONHEIGHT =
"personalization_driverseatposition_cushionheight";
 const VehicleEventType PERSONALIZATION_DRIVERSEATPOSITION_HEADREST =
"personalization_driverseatposition_headrest";
 const VehicleEventType PERSONALIZATION_DRIVERSEATPOSITION_BACKCUSHION =
"personalization_driverseatposition_backcushion";
 const VehicleEventType PERSONALIZATION_DRIVERSEATPOSITION_SIDECUSHION =
"personalization_driverseatposition_sidecushion";
 const VehicleEventType PERSONALIZATION_PASSENGERSEATPOSITION =
"personalization_passengerseatposition";
 const VehicleEventType PERSONALIZATION_PASSENGERSEATPOSITION_RECLINESEATBACK =
"personalization_passengerseatposition_reclineseatback";
 const VehicleEventType PERSONALIZATION_PASSENGERSEATPOSITION_SLIDE =
"personalization_passengerseatposition_slide";
 const VehicleEventType PERSONALIZATION_PASSENGERSEATPOSITION_CUSHIONHEIGHT =
"personalization_passengerseatposition_cushionheight";
 const VehicleEventType PERSONALIZATION_PASSENGERSEATPOSITION_HEADREST =
"personalization_passengerseatposition_headrest";
 const VehicleEventType PERSONALIZATION_PASSENGERSEATPOSITION_BACKCUSHION =
"personalization_passengerseatposition_backcushion";
 const VehicleEventType PERSONALIZATION_PASSENGERSEATPOSITION_SIDECUSHION =
"personalization_passengerseatposition_sidecushion";
 const VehicleEventType PERSONALIZATION_DASHBOARDILLUMINATION =
"personalization_dashboardillumination";
 const VehicleEventType PERSONALIZATION_GENERATEDVEHICLESOUNDMODE =
"personalization_generatedvehiclesoundmode";

 const unsigned short LANGUAGE_ENGLISH = 1;
 const unsigned short LANGUAGE_SPANISH = 2;
 const unsigned short LANGUAGE_FRENCH = 3;

 const unsigned short DRIVINGMODE_COMFORT = 1;
 const unsigned short DRIVINGMODE_AUTO = 2;
 const unsigned short DRIVINGMODE_SPORT = 3;
 const unsigned short DRIVINGMODE_ECO = 4;
 const unsigned short DRIVINGMODE_MANUAL = 5;

 const unsigned short GENERATEDVEHICLESOUNDMODE_NORMAL = 1;
 const unsigned short GENERATEDVEHICLESOUNDMODE_QUIET = 2;
 const unsigned short GENERATEDVEHICLESOUNDMODE_SPORTIVE = 3;

 readonly attribute VehicleEventType type;
 readonly attribute DOMString? keyId;
 readonly attribute unsigned short? language;
 readonly attribute boolean? measurementSystem;
 readonly attribute DOMString? measurementSystemStringFuel;
 readonly attribute DOMString? measurementSystemStringDistance;
 readonly attribute DOMString? measurementSystemStringSpeed;
 readonly attribute DOMString? measurementSystemStringConsumption;
 readonly attribute unsigned short? mirrorDriver;
 readonly attribute unsigned short? mirrorPassenger;
 readonly attribute unsigned short? mirrorInside;
 readonly attribute unsigned short? steeringWheelPositionSlide;
 readonly attribute unsigned short? steeringWheelPositionTilt;
 readonly attribute unsigned short? drivingMode;
 readonly attribute unsigned short? driverSeatPositionReclineSeatback;
 readonly attribute unsigned short? driverSeatPositionSlide;
 readonly attribute unsigned short? driverSeatPositionCushionHeight;
 readonly attribute unsigned short? driverSeatPositionHeadrest;
 readonly attribute unsigned short? driverSeatPositionBackCushion;
 readonly attribute unsigned short? driverSeatPositionSideCushion;
 readonly attribute unsigned short? passengerSeatPositionReclineSeatback;
 readonly attribute unsigned short? passengerSeatPositionSlide;
 readonly attribute unsigned short? passengerSeatPositionCushionHeight;
 readonly attribute unsigned short? passengerSeatPositionHeadrest;
 readonly attribute unsigned short? passengerSeatPositionBackCushion;
 readonly attribute unsigned short? passengerSeatPositionSideCushion;
 readonly attribute unsigned short? dashboardIllumination;
 readonly attribute unsigned short? generatedVehicleSoundMode;
};
```

4.2.4.1 Attributes

`type` of type VehicleEventType, readonly
This attribute represents a intended type of this PersonalizationEvent.

`keyId` of type DOMString, readonly, nullable
This attribute contains the string representation of ID of remote control key. It can be used to identify the driver and to apply his personal settings.

`language` of type unsigned short, readonly, nullable
This attribute represents the setting of language used by the system currently.
The value is one of the following values.

```
const unsigned short LANGUAGE_ENGLISH = 1;
const unsigned short LANGUAGE_SPANISH = 2;
const unsigned short LANGUAGE_FRENCH = 3;
```

`measurementSystem` of type boolean, readonly, nullable
This attribute indicates whether the current measurement system is km(litter) or mile(gallon).
'true' means the current measurement system is km(litter). 'false' means it is mile(gallon).

`measurementSystemStringFuel` of type DOMString, readonly, nullable
This attribute contains the string representation of fuel unit in current measurement system.
The value is one of both "litter" and "gallon".

`measurementSystemStringDistance` of type DOMString, readonly, nullable
This attribute contains the string representation of distance unit in current measurement system.
The value is one of both "km" and "mile".

`measurementSystemStringSpeed` of type DOMString, readonly, nullable
This attribute contains the string representation of speed unit in current measurement system.
The value is one of both "km/h" and "mph".

`measurementSystemStringConsumption` of type DOMString, readonly, nullable
This attribute contains the string representation of fuel consumption unit in current measurement system.
The value is one of following values: "l/100", "mpg", "km/l".

`mirrorDriver` of type unsigned short, readonly, nullable
`mirrorPassenger` of type unsigned short, readonly, nullable
`mirrorInside` of type unsigned short, readonly, nullable
These attributes represent the position setting value of each mirror.
The value is 2 bytes compound type where the first byte is tilt value and the last byte is pan value.

`steeringWheelPositionSlide` of type unsigned short, readonly, nullable
This attribute represents the position setting of steering wheel sliding.

`steeringWheelPositionTilt` of type unsigned short, readonly, nullable
This attribute represents the position setting of steering wheel tilting.

`drivingMode` of type unsigned short, readonly, nullable
This attribute represents the setting of current driving mode.
The value is one of the following values.

```
const unsigned short DRIVINGMODE_COMFORT = 1;
const unsigned short DRIVINGMODE_AUTO = 2;
const unsigned short DRIVINGMODE_SPORT = 3;
const unsigned short DRIVINGMODE_ECO = 4;
const unsigned short DRIVINGMODE_MANUAL = 5;
```

`driverSeatPositionReclineSeatback` of type unsigned short, readonly, nullable
This attribute represents the position setting of driver seatback reclining.

`driverSeatPositionSlide` of type unsigned short, readonly, nullable
This attribute represents the position setting of driver seat sliding.

`driverSeatPositionCushionHeight` of type unsigned short, readonly, nullable
This attribute represents the height setting of driver seat cushion.

driverSeatPositionHeadrest of type unsigned short, readonly, nullable
This attribute represents the position setting of driver seat headrest.

driverSeatPositionBackCushion of type unsigned short, readonly, nullable
This attribute represents the position setting of driver seat back cushion.

driverSeatPositionSideCushion of type unsigned short, readonly, nullable
This attribute represents the position setting of driver seat side cushion.

passengerSeatPositionReclineSeatback of type unsigned short, readonly, nullable
This attribute represents the position setting of passenger seatback reclining.

passengerSeatPositionSlide of type unsigned short, readonly, nullable
This attribute represents the position setting of passenger seat sliding.

passengerSeatPositionCushionHeight of type unsigned short, readonly, nullable
This attribute represents the height setting of passenger seat cushion.

passengerSeatPositionHeadrest of type unsigned short, readonly, nullable
This attribute represents the position setting of passenger seat headrest.

passengerSeatPositionBackCushion of type unsigned short, readonly, nullable
This attribute represents the position setting of passenger seat back cushion.

passengerSeatPositionSideCushion of type unsigned short, readonly, nullable
This attribute represents the position setting of passenger seat side cushion.

dashboardIllumination of type unsigned short, readonly, nullable
This attribute contains the current percentage of dashboard illumination brightness.
The value 0% means darkest. 100% means brightest.

generatedVehicleSoundMode of type unsigned short, readonly, nullable
This attribute represents the setting of generated vehicle sound mode. It is usually supported in electric vehicles.
The value is one of the following values.

```
const unsigned short GENERATEDVEHICLESOUNDMODE_NORMAL = 1;
const unsigned short GENERATEDVEHICLESOUNDMODE_QUIET = 2;
const unsigned short GENERATEDVEHICLESOUNDMODE_SPORTIVE = 3;
```

4.2.5 DrivingSafetyEvent

```


interface DrivingSafetyEvent : VehicleEvent {
 const VehicleEventType DRIVINGSAFETY = "drivingsafety";
 const VehicleEventType DRIVINGSAFETY_ANTILOCKBRAKINGSYSTEM =
"drivingsafety_antilockbrakingsystem";
 const VehicleEventType DRIVINGSAFETY_TRACTIONCONTROLSYSTEM =
"drivingsafety_tractioncontrolsystem";
 const VehicleEventType DRIVINGSAFETY_ELECTRONICSTABILITYCONTROL =
"drivingsafety_electronicstabilitycontrol";
 const VehicleEventType DRIVINGSAFETY_VEHICLETOPSPEEDLIMIT = "drivingsafety_vehicletopspeedlimit";
 const VehicleEventType DRIVINGSAFETY_AIRBAGSTATUS = "drivingsafety_airbagstatus";
 const VehicleEventType DRIVINGSAFETY_AIRBAGSTATUS_DRIVER = "drivingsafety_airbagstatus_driver";
 const VehicleEventType DRIVINGSAFETY_AIRBAGSTATUS_PASSENGER =
"drivingsafety_airbagstatus_passenger";
 const VehicleEventType DRIVINGSAFETY_AIRBAGSTATUS_SIDE = "drivingsafety_airbagstatus_side";
 const VehicleEventType DRIVINGSAFETY_DOOROPENSTATUS = "drivingsafety_dooropenstatus";
 const VehicleEventType DRIVINGSAFETY_DOOROPENSTATUS_DRIVER =
"drivingsafety_dooropenstatus_driver";
 const VehicleEventType DRIVINGSAFETY_DOOROPENSTATUS_PASSENGER =
"drivingsafety_dooropenstatus_passenger";
 const VehicleEventType DRIVINGSAFETY_DOOROPENSTATUS_REARLEFT =
"drivingsafety_dooropenstatus_rearleft";
 const VehicleEventType DRIVINGSAFETY_DOOROPENSTATUS_REARRIGHT =
"drivingsafety_dooropenstatus_rearright";
 const VehicleEventType DRIVINGSAFETY_DOOROPENSTATUS_TRUNK = "drivingsafety_dooropenstatus_trunk";
 const VehicleEventType DRIVINGSAFETY_DOOROPENSTATUS_FUELFILTERCAP =
"drivingsafety_dooropenstatus_fuelfiltercap";
 const VehicleEventType DRIVINGSAFETY_DOOROPENSTATUS_HOOD = "drivingsafety_dooropenstatus_hood";
 const VehicleEventType DRIVINGSAFETY_DOORLOCKSTATUS = "drivingsafety_doorlockstatus";
 const VehicleEventType DRIVINGSAFETY_DOORLOCKSTATUS_DRIVER =
"drivingsafety_doorlockstatus_driver";
 const VehicleEventType DRIVINGSAFETY_DOORLOCKSTATUS_PASSENGER =
"drivingsafety_doorlockstatus_passenger";
```

```
 const VehicleEventType DRIVINGSAFETY_DOORLOCKSTATUS_REARLEFT =
"drivingsafety_doorlockstatus_rearleft";
 const VehicleEventType DRIVINGSAFETY_DOORLOCKSTATUS_REARRIGHT =
"drivingsafety_doorlockstatus_rearright";
 const VehicleEventType DRIVINGSAFETY_CHILDSAFETYLOCK = "drivingsafety_childsafetylock";
 const VehicleEventType DRIVINGSAFETY_OCCUPANTSSTATUS = "drivingsafety_occupantsstatus";
 const VehicleEventType DRIVINGSAFETY_OCCUPANTSSTATUS_DRIVER =
"drivingsafety_occupantsstatus_driver";
 const VehicleEventType DRIVINGSAFETY_OCCUPANTSSTATUS_PASSENGER =
"drivingsafety_occupantsstatus_passenger";
 const VehicleEventType DRIVINGSAFETY_OCCUPANTSSTATUS_REARLEFT =
"drivingsafety_occupantsstatus_rearleft";
 const VehicleEventType DRIVINGSAFETY_OCCUPANTSSTATUS_REARRIGHT =
"drivingsafety_occupantsstatus_rearright";
 const VehicleEventType DRIVINGSAFETY_SEATBELT = "drivingsafety_seatbelt";
 const VehicleEventType DRIVINGSAFETY_SEATBELT_DRIVER = "drivingsafety_seatbelt_driver";
 const VehicleEventType DRIVINGSAFETY_SEATBELT_PASSENGER = "drivingsafety_seatbelt_passenger";
 const VehicleEventType DRIVINGSAFETY_SEATBELT_REARLEFT = "drivingsafety_seatbelt_rearleft";
 const VehicleEventType DRIVINGSAFETY_SEATBELT_REARRIGHT = "drivingsafety_seatbelt_rearright";
 const VehicleEventType DRIVINGSAFETY_WINDOWLOCK = "drivingsafety_windowlock";
 const VehicleEventType DRIVINGSAFETY_WINDOWLOCK_DRIVER = "drivingsafety_windowlock_driver";
 const VehicleEventType DRIVINGSAFETY_WINDOWLOCK_PASSENGER = "drivingsafety_windowlock_passenger";
 const VehicleEventType DRIVINGSAFETY_WINDOWLOCK_REARLEFT = "drivingsafety_windowlock_rearleft";
 const VehicleEventType DRIVINGSAFETY_WINDOWLOCK_REARRIGHT = "drivingsafety_windowlock_rearright";
 const VehicleEventType DRIVINGSAFETY_OBSTACLEDISTANCE = "drivingsafety_obstacledistance";
 const VehicleEventType DRIVINGSAFETY_OBSTACLEDISTANCE_SENSORSTATUS =
"drivingsafety_obstacledistance_sensorstatus";
 const VehicleEventType DRIVINGSAFETY_OBSTACLEDISTANCE_FRONTCENTER =
"drivingsafety_obstacledistance_frontcenter";
 const VehicleEventType DRIVINGSAFETY_OBSTACLEDISTANCE_REARCENTER =
"drivingsafety_obstacledistance_rearcenter";
 const VehicleEventType DRIVINGSAFETY_OBSTACLEDISTANCE_FRONTLEFT =
"drivingsafety_obstacledistance_frontleft";
 const VehicleEventType DRIVINGSAFETY_OBSTACLEDISTANCE_FRONTRIGHT =
"drivingsafety_obstacledistance_frontright";
 const VehicleEventType DRIVINGSAFETY_OBSTACLEDISTANCE_MIDDLELEFT =
"drivingsafety_obstacledistance_middleleft";
 const VehicleEventType DRIVINGSAFETY_OBSTACLEDISTANCE_MIDDLERIGHT =
"drivingsafety_obstacledistance_middleright";
 const VehicleEventType DRIVINGSAFETY_OBSTACLEDISTANCE_REARLEFT =
"drivingsafety_obstacledistance_rearleft";
 const VehicleEventType DRIVINGSAFETY_OBSTACLEDISTANCE_REARRIGHT =
"drivingsafety_obstacledistance_rearright";
 const VehicleEventType DRIVINGSAFETY_FRONTCOLLISIONDETECTION =
"drivingsafety_frontcollisiondetection";
 const VehicleEventType DRIVINGSAFETY_FRONTCOLLISIONDETECTION_STATUS =
"drivingsafety_frontcollisiondetection_status";
 const VehicleEventType DRIVINGSAFETY_FRONTCOLLISIONDETECTION_DISTANCE =
"drivingsafety_frontcollisiondetection_distance";
 const VehicleEventType DRIVINGSAFETY_FRONTCOLLISIONDETECTION_TIME =
"drivingsafety_frontcollisiondetection_time";

 const unsigned short ANTILOCKBRAKINGSYSTEM_AVAILABLE = 1;
 const unsigned short ANTILOCKBRAKINGSYSTEM_IDLE = 2;
 const unsigned short ANTILOCKBRAKINGSYSTEM_ENGAGED = 3;

 const unsigned short TRACTIONCONTROLSYSTEM_AVAILABLE = 1;
 const unsigned short TRACTIONCONTROLSYSTEM_IDLE = 2;
 const unsigned short TRACTIONCONTROLSYSTEM_ENGAGED = 3;

 const unsigned short ELECTRONICSTABILITYCONTROL_AVAILABLE = 1;
 const unsigned short ELECTRONICSTABILITYCONTROL_IDLE = 2;
 const unsigned short ELECTRONICSTABILITYCONTROL_ENGAGED = 3;

 const unsigned short AIRBAGSTATUS_ACTIVATE = 1;
 const unsigned short AIRBAGSTATUS_DEACTIVATE = 2;
 const unsigned short AIRBAGSTATUS_DEPLOYMENT = 3;

 const unsigned short DOOROPENSTATUS_OPEN = 1;
 const unsigned short DOOROPENSTATUS_AJAR = 2;
 const unsigned short DOOROPENSTATUS_CLOSE = 3;

 const unsigned short OCCUPANTSSTATUS_ADULT = 1;
 const unsigned short OCCUPANTSSTATUS_CHILD = 2;
 const unsigned short OCCUPANTSSTATUS_VACANT = 3;
```

```
readonly attribute VehicleEventType type;
readonly attribute unsigned short? antilockBrakingSystem;
readonly attribute unsigned short? tractionControlSystem;
readonly attribute unsigned short? electronicStabilityControl;
readonly attribute unsigned short? vehicleTopSpeedLimit;
readonly attribute unsigned short? airbagStatusDriver;
readonly attribute unsigned short? airbagStatusPassenger;
readonly attribute unsigned short? airbagStatusSide;
readonly attribute unsigned short? doorOpenStatusDriver;
readonly attribute unsigned short? doorOpenStatusPassenger;
readonly attribute unsigned short? doorOpenStatusRearLeft;
readonly attribute unsigned short? doorOpenStatusRearRight;
readonly attribute unsigned short? doorOpenStatusTrunk;
readonly attribute unsigned short? doorOpenStatusFuelFilterCap;
readonly attribute unsigned short? doorOpenStatusHood;
readonly attribute boolean? doorLockStatusDriver;
readonly attribute boolean? doorLockStatusPassenger;
readonly attribute boolean? doorLockStatusRearLeft;
readonly attribute boolean? doorLockStatusRearRight;
readonly attribute boolean? childSafetyLock;
readonly attribute unsigned short? occupantsStatusDriver;
readonly attribute unsigned short? occupantsStatusPassenger;
readonly attribute unsigned short? occupantsStatusRearLeft;
readonly attribute unsigned short? occupantsStatusRearRight;
readonly attribute boolean? seatBeltDriver;
readonly attribute boolean? seatBeltPassenger;
readonly attribute boolean? seatBeltRearLeft;
readonly attribute boolean? seatBeltRearRight;
readonly attribute boolean? windowLockDriver;
readonly attribute boolean? windowLockPassenger;
readonly attribute boolean? windowLockRearLeft;
readonly attribute boolean? windowLockRearRight;
readonly attribute boolean? obstacleDistanceSensorStatus;
readonly attribute unsigned short? obstacleDistanceFrontCenter;
readonly attribute unsigned short? obstacleDistanceRearCenter;
readonly attribute unsigned short? obstacleDistanceFrontLeft;
readonly attribute unsigned short? obstacleDistanceFrontRight;
readonly attribute unsigned short? obstacleDistanceMiddleLeft;
readonly attribute unsigned short? obstacleDistanceMiddleRight;
readonly attribute unsigned short? obstacleDistanceRearLeft;
readonly attribute unsigned short? obstacleDistanceRearRight;
readonly attribute boolean? frontCollisionDetectionStatus;
```

```
  readonly attribute unsigned long? frontCollisionDetectionDistance;
  readonly attribute unsigned long? frontCollisionDetectionTime;
 };
```

4.2.5.1 Attributes

`type` of type VehicleEventType, readonly
This attribute represents a intended type of this DrivingSafetyEvent.

`antilockBrakingSystem` of type unsigned short, readonly, nullable
This attribute represents the status of ABS(Antilock Braking System).
The value is one of the following values.

```
const unsigned short ANTILOCKBRAKINGSYSTEM_AVAILABLE = 1;
const unsigned short ANTILOCKBRAKINGSYSTEM_IDLE = 2;
const unsigned short ANTILOCKBRAKINGSYSTEM_ENGAGED = 3;
```

`tractionControlSystem` of type unsigned short, readonly, nullable
This attribute represents the status of TCS(Traction Control System).
The value is one of the following values.

```
const unsigned short TRACTIONCONTROLSYSTEM_AVAILABLE = 1;
const unsigned short TRACTIONCONTROLSYSTEM_IDLE = 2;
const unsigned short TRACTIONCONTROLSYSTEM_ENGAGED = 3;
```

`electronicStabilityControl` of type unsigned short, readonly, nullable
This attribute represents the status of ESC(Electronic Stability Control).
The value is one of the following values.

```
const unsigned short ELECTRONICSTABILITYCONTROL_AVAILABLE = 1;
const unsigned short ELECTRONICSTABILITYCONTROL_IDLE = 2;
const unsigned short ELECTRONICSTABILITYCONTROL_ENGAGED = 3;
```

`vehicleTopSpeedLimit` of type unsigned short, readonly, nullable
This attribute contains the setting value of desired limit speed of this vehicle. The unit of value is km/h or mph.

`airbagStatusDriver` of type unsigned short, readonly, nullable
`airbagStatusPassenger` of type unsigned short, readonly, nullable
`airbagStatusSide` of type unsigned short, readonly, nullable
These attributes represent the status of each airbag.
The value is one of the following values.

```
const unsigned short AIRBAGSTATUS_ACTIVATE = 1;
const unsigned short AIRBAGSTATUS_DEACTIVATE = 2;
const unsigned short AIRBAGSTATUS_DEPLOYMENT = 3;
```

`doorOpenStatusDriver` of type unsigned short, readonly, nullable
`doorOpenStatusPassenger` of type unsigned short, readonly, nullable
`doorOpenStatusRearLeft` of type unsigned short, readonly, nullable
`doorOpenStatusRearRight` of type unsigned short, readonly, nullable
`doorOpenStatusTrunk` of type unsigned short, readonly, nullable
`doorOpenStatusFuelFilterCap` of type unsigned short, readonly, nullable
`doorOpenStatusHood` of type unsigned short, readonly, nullable
These attributes represent the status of each door whether it is open or not.
The value is one of the following values.

```
const unsigned short DOOROPENSTATUS_OPEN = 1;
const unsigned short DOOROPENSTATUS_AJAR = 2;
const unsigned short DOOROPENSTATUS_CLOSE = 3;
```

`doorLockStatusDriver` of type boolean, readonly, nullable
`doorLockStatusPassenger` of type boolean, readonly, nullable
`doorLockStatusRearLeft` of type boolean, readonly, nullable
`doorLockStatusRearRight` of type boolean, readonly, nullable
These attributes indicate whether each door is locked or not.
'true' means the door is locked. 'false' means it is unlocked.

`childSafetyLock` of type boolean, readonly, nullable
This attribute indicates whether child safety lock is on or not.
'true' means child safety lock is on. 'false' means it is off.

`occupantsStatusDriver` of type unsigned short, readonly, nullable
`occupantsStatusPassenger` of type unsigned short, readonly, nullable
`occupantsStatusRearLeft` of type unsigned short, readonly, nullable
`occupantsStatusRearRight` of type unsigned short, readonly, nullable
These attributes represent the status of each seat whether it is occupied or not.
The value is one of the following values.

```
const unsigned short OCCUPANTSSTATUS_ADULT = 1;
const unsigned short OCCUPANTSSTATUS_CHILD = 2;
const unsigned short OCCUPANTSSTATUS_VACANT = 3;
```

`seatBeltDriver` of type boolean, readonly, nullable
`seatBeltPassenger` of type boolean, readonly, nullable
`seatBeltRearLeft` of type boolean, readonly, nullable
`seatBeltRearRight` of type boolean, readonly, nullable
These attributes indicate the seat belt status of each seat whether it is fasten or not.
'true' means the seat belt is fasten. 'false' means it is unfasten.

`windowLockDriver` of type boolean, readonly, nullable
`windowLockPassenger` of type boolean, readonly, nullable
`windowLockRearLeft` of type boolean, readonly, nullable
`windowLockRearRight` of type boolean, readonly, nullable
These attributes indicate whether each window is locked or not.
'true' means the window is locked. 'false' means it is unlocked.

`obstacleDistanceSensorStatus` of type boolean, readonly, nullable
This attribute indicates whether obstacle distance sensor is activated or not.
'true' means obstacle distance sensor is activated. 'false' means it is deactivated.

`obstacleDistanceFrontCenter` of type unsigned short, readonly, nullable
`obstacleDistanceRearCenter` of type unsigned short, readonly, nullable
`obstacleDistanceFrontLeft` of type unsigned short, readonly, nullable
`obstacleDistanceFrontRight` of type unsigned short, readonly, nullable
`obstacleDistanceMiddleLeft` of type unsigned short, readonly, nullable
`obstacleDistanceMiddleRight` of type unsigned short, readonly, nullable
`obstacleDistanceRearLeft` of type unsigned short, readonly, nullable
`obstacleDistanceRearRight` of type unsigned short, readonly, nullable
These attributes contain the distance to a barrier from each sensor position. The unit of value is cm or in.

`frontCollisionDetectionStatus` of type boolean, readonly, nullable
This attribute indicates whether front collision detection is activated or not.
'true' means front collision detection is activated. 'false' means it is deactivated.

`frontCollisionDetectionDistance` of type unsigned long, readonly, nullable
This attribute contains the distance to a front barrier from this vehicle. The unit of value is m or feet.

`frontCollisionDetectionTime` of type unsigned long, readonly, nullable
This attribute contains the estimated remaining time to crash into a front barrier. The unit of value is ms.

4.2.6 VisionSystemEvent

```
interface VisionSystemEvent : VehicleEvent {
 const VehicleEventType VISIONSYSTEM = "visionsystem";
 const VehicleEventType VISIONSYSTEM_LANEDEPARTUREDETECTIONSTATUS =
"visionsystem_lanedeparturedetectionstatus";
 const VehicleEventType VISIONSYSTEM_LANEDEPARTED = "visionsystem_lanedeparted";

 readonly attribute VehicleEventType type;
 readonly attribute boolean? laneDepartureDetectionStatus;
 readonly attribute boolean? laneDeparted;
};
```

4.2.6.1 Attributes

`type` of type VehicleEventType, readonly
This attribute represents a intended type of this VisionSystemEvent.

`laneDepartureDetectionStatus` of type boolean, readonly, nullable
This attribute indicates whether lane departure detection is activated or not.

'true' means lane departure detection is activated. 'false' means it is deactivated.

laneDeparted of type boolean, readonly, nullable
This attribute indicates whether this vehicle is departed from its lane or not.
'true' means this vehicle is departed from its lane. 'false' means it is not departed.

4.2.7 ParkingEvent

```
interface ParkingEvent : VehicleEvent {
 const VehicleEventType PARKING = "parking";
 const VehicleEventType PARKING_SECURITYALERT = "parking_securityalert";
 const VehicleEventType PARKING_PARKINGBRAKE = "parking_parkingbrake";
 const VehicleEventType PARKING_PARKINGLIGHTS = "parking_parkinglights";

 const unsigned short SECURITYALERT_AVAILABLE = 1;
 const unsigned short SECURITYALERT_IDLE = 2;
 const unsigned short SECURITYALERT_ACTIVATED = 3;
 const unsigned short SECURITYALERT_ALARM_DETECTED = 4;

 readonly attribute VehicleEventType type;
 readonly attribute unsigned short? securityAlert;
 readonly attribute boolean? parkingBrake;
 readonly attribute boolean? parkingLights;
};
```

4.2.7.1 Attributes

type of type VehicleEventType, readonly
This attribute represents a intended type of this ParkingEvent.

securityAlert of type unsigned short, readonly, nullable
This attribute represents the status of security alert.
The value is one of the following values.

```
const unsigned short SECURITYALERT_AVAILABLE = 1;
const unsigned short SECURITYALERT_IDLE = 2;
const unsigned short SECURITYALERT_ACTIVATED = 3;
const unsigned short SECURITYALERT_ALARM_DETECTED = 4;
```

parkingBrake of type boolean, readonly, nullable
This attribute indicates whether parking brake is engaged or not.
'true' means parking brake is engaged. 'false' means it is not engaged.

parkingLights of type boolean, readonly, nullable
This attribute indicates whether parking light function is set to be activated or not. It's not about whether the light is on or off currently.
'true' means parking light is activated. 'false' means it is deactivated.

4.2.8 ClimateEnvironmentEvent

```
interface ClimateEnvironmentEvent : VehicleEvent {
 const VehicleEventType CLIMATEENVIRONMENT = "climateenvironment";
 const VehicleEventType CLIMATEENVIRONMENT_INTERIORTEMP = "climateenvironment_interiortemp";
 const VehicleEventType CLIMATEENVIRONMENT_EXTERIORTEMP = "climateenvironment_exteriortemp";
 const VehicleEventType CLIMATEENVIRONMENT_EXTERIORBRIGHTNESS =
"climateenvironment_exteriorbrightness";
 const VehicleEventType CLIMATEENVIRONMENT_RAINSENSOR = "climateenvironment_rainsensor";
 const VehicleEventType CLIMATEENVIRONMENT_WINDSHIELDWIPER = "climateenvironment_windshieldwiper";
 const VehicleEventType CLIMATEENVIRONMENT_REARWIPER = "climateenvironment_rearwiper";
 const VehicleEventType CLIMATEENVIRONMENT_HVACFAN = "climateenvironment_hvacfan";
 const VehicleEventType CLIMATEENVIRONMENT_HVACFAN_DIRECTION =
"climateenvironment_hvacfan_direction";
 const VehicleEventType CLIMATEENVIRONMENT_HVACFAN_SPEED = "climateenvironment_hvacfan_speed";
 const VehicleEventType CLIMATEENVIRONMENT_HVACFAN_TARGETTEMP =
"climateenvironment_hvacfan_targettemp";
 const VehicleEventType CLIMATEENVIRONMENT_AIRCONDITIONING = "climateenvironment_airconditioning";
 const VehicleEventType CLIMATEENVIRONMENT_AIRRECIRCULATION =
"climateenvironment_airrecirculation";
 const VehicleEventType CLIMATEENVIRONMENT_HEATER = "climateenvironment_heater";
 const VehicleEventType CLIMATEENVIRONMENT_DEFROST = "climateenvironment_defrost";
 const VehicleEventType CLIMATEENVIRONMENT_DEFROST_WINDSHIELD =
"climateenvironment_defrost_windshield";
 const VehicleEventType CLIMATEENVIRONMENT_DEFROST_REARWINDOW =
"climateenvironment_defrost_rearwindow";
 const VehicleEventType CLIMATEENVIRONMENT_DEFROST_SIDEMIRRORS =
"climateenvironment_defrost_sidemirrors";
 const VehicleEventType CLIMATEENVIRONMENT_STEERINGWHEELHEATER =
"climateenvironment_steeringwheelheater";
 const VehicleEventType CLIMATEENVIRONMENT_SEATHEATER = "climateenvironment_seatheater";
 const VehicleEventType CLIMATEENVIRONMENT_SEATCOOLER = "climateenvironment_seatcooler";
 const VehicleEventType CLIMATEENVIRONMENT_WINDOW = "climateenvironment_window";
 const VehicleEventType CLIMATEENVIRONMENT_WINDOW_DRIVER = "climateenvironment_window_driver";
 const VehicleEventType CLIMATEENVIRONMENT_WINDOW_PASSENGER =
"climateenvironment_window_passenger";
 const VehicleEventType CLIMATEENVIRONMENT_WINDOW_REARLEFT = "climateenvironment_window_rearleft";
 const VehicleEventType CLIMATEENVIRONMENT_WINDOW_REARRIGHT =
"climateenvironment_window_rearright";
 const VehicleEventType CLIMATEENVIRONMENT_SUNROOF = "climateenvironment_sunroof";
 const VehicleEventType CLIMATEENVIRONMENT_SUNROOF_OPENNESS =
"climateenvironment_sunroof_openness";
 const VehicleEventType CLIMATEENVIRONMENT_SUNROOF_TILT = "climateenvironment_sunroof_tilt";
 const VehicleEventType CLIMATEENVIRONMENT_CONVERTIBLEROOF = "climateenvironment_convertibleroof";

 const unsigned short RAINSENSOR_NORAIN = 0;
 const unsigned short RAINSENSOR_LEVEL1 = 1;
 const unsigned short RAINSENSOR_LEVEL2 = 2;
 const unsigned short RAINSENSOR_LEVEL3 = 3;
 const unsigned short RAINSENSOR_LEVEL4 = 4;
 const unsigned short RAINSENSOR_LEVEL5 = 5;
 const unsigned short RAINSENSOR_LEVEL6 = 6;
 const unsigned short RAINSENSOR_LEVEL7 = 7;
 const unsigned short RAINSENSOR_LEVEL8 = 8;
 const unsigned short RAINSENSOR_LEVEL9 = 9;
 const unsigned short RAINSENSOR_HEAVIESTRAIN = 10;

 const unsigned short WIPER_OFF = 0;
 const unsigned short WIPER_ONCE = 1;
 const unsigned short WIPER_SLOWEST = 2;
 const unsigned short WIPER_SLOW = 3;
 const unsigned short WIPER_FAST = 4;
 const unsigned short WIPER_FASTEST = 5;
 const unsigned short WIPER_AUTO = 10;

 const unsigned short HVACFAN_DIRECTION_FRONTPANEL = 1;
 const unsigned short HVACFAN_DIRECTION_FLOORDUCT = 2;
 const unsigned short HVACFAN_DIRECTION_FRONTFLOOR = 3;
 const unsigned short HVACFAN_DIRECTION_DEFROSTERFLOOR = 4;

 readonly attribute VehicleEventType type;
 readonly attribute short? interiorTemp;
 readonly attribute short? exteriorTemp;
 readonly attribute unsigned long? exteriorBrightness;
 readonly attribute unsigned short? rainSensor;
 readonly attribute unsigned short? windshieldWiper;
 readonly attribute unsigned short? rearWiper;
 readonly attribute unsigned short? hvacFanDirection;
 readonly attribute unsigned short? hvacFanSpeed;
 readonly attribute short? hvacFanTargetTemp;
 readonly attribute boolean? airConditioning;
 readonly attribute boolean? airRecirculation;
 readonly attribute boolean? heater;
 readonly attribute boolean? defrostWindshield;
 readonly attribute boolean? defrostRearWindow;
 readonly attribute boolean? defrostSideMirrors;
 readonly attribute boolean? steeringWheelHeater;
 readonly attribute boolean? seatHeater;
 readonly attribute boolean? seatCooler;
 readonly attribute unsigned short? windowDriver;
 readonly attribute unsigned short? windowPassenger;
 readonly attribute unsigned short? windowRearLeft;
 readonly attribute unsigned short? windowRearRight;
 readonly attribute unsigned short? sunroofOpenness;
 readonly attribute unsigned short? sunroofTilt;
 readonly attribute boolean? convertibleRoof;
};
```

4.2.8.1 Attributes

`type` of type VehicleEventType, readonly
This attribute represents a intended type of this ClimateEnvironmentEvent.

`interiorTemp` of type short, readonly, nullable
This attribute contains the current temperature of the inside vehicle.
The unit of value is 0.1°C or 0.1°F. e.g. value 205 means 20.5°C

`exteriorTemp` of type short, readonly, nullable
This attribute contains the current temperature of the outside vehicle.
The unit of value is 0.1°C or 0.1°F. e.g. value 205 means 20.5°C

`exteriorBrightness` of type unsigned long, readonly, nullable
This attribute contains the brightness of the outside vehicle. The unit of value is lux.

`rainSensor` of type unsigned short, readonly, nullable
This attribute represents the level of rain intensity.
The value is one of the following values.

```
const unsigned short RAINSENSOR_NORAIN = 0;
const unsigned short RAINSENSOR_LEVEL1 = 1;
const unsigned short RAINSENSOR_LEVEL2 = 2;
const unsigned short RAINSENSOR_LEVEL3 = 3;
const unsigned short RAINSENSOR_LEVEL4 = 4;
const unsigned short RAINSENSOR_LEVEL5 = 5;
const unsigned short RAINSENSOR_LEVEL6 = 6;
const unsigned short RAINSENSOR_LEVEL7 = 7;
const unsigned short RAINSENSOR_LEVEL8 = 8;
const unsigned short RAINSENSOR_LEVEL9 = 9;
const unsigned short RAINSENSOR_HEAVIESTRAIN = 10;
```

`windshieldWiper` of type unsigned short, readonly, nullable
`rearWiper` of type unsigned short, readonly, nullable
This attribute represents the current setting of each wiper.
The value is one of the following values.

```
const unsigned short WIPER_OFF = 0;
const unsigned short WIPER_ONCE = 1;
const unsigned short WIPER_SLOWEST = 2;
const unsigned short WIPER_SLOW = 3;
const unsigned short WIPER_FAST = 4;
const unsigned short WIPER_FASTEST = 5;
const unsigned short WIPER_AUTO = 10;
```

`hvacFanDirection` of type unsigned short, readonly, nullable
This attribute represents the direction setting of HVAC fan.
The value is one of the following values.

```
const unsigned short HVACFAN_DIRECTION_FRONTPANEL = 1;
const unsigned short HVACFAN_DIRECTION_FLOORDUCT = 2;
const unsigned short HVACFAN_DIRECTION_FRONTFLOOR = 3;
const unsigned short HVACFAN_DIRECTION_DEFROSTERFLOOR = 4;
```

`hvacFanSpeed` of type unsigned short, readonly, nullable
This attribute represents the level of HVAC fan speed. The range of value is from 0(weakest) to 7(strongest).

`hvacFanTargetTemp` of type short, readonly, nullable
This attribute contains the desired target temperature of the inside vehicle.
The unit of value is 0.1°C or 0.1°F. e.g. value 205 means 20.5°C

`airConditioning` of type boolean, readonly, nullable
This attribute indicates whether air conditioner is on or off.
'true' means air conditioner is on. 'false' means it is off.

`airRecirculation` of type boolean, readonly, nullable

This attribute indicates whether air recirculation system is on or off.
'true' means air recirculation system is on. 'false' means it is off.

`heater` of type boolean, readonly, nullable
This attribute indicates whether heater is on or off.
'true' means heater is on. 'false' means it is off.

`defrostWindshield` of type boolean, readonly, nullable
`defrostRearWindow` of type boolean, readonly, nullable
`defrostSideMirrors` of type boolean, readonly, nullable
These attributes indicate whether defrost per each position is on or off.
'true' means defrost is on. 'false' means it is off.

`steeringWheelHeater` of type boolean, readonly, nullable
This attribute indicates whether steering wheel heater is on or off.
'true' means steering wheel heater is on. 'false' means it is off.

`seatHeater` of type boolean, readonly, nullable
This attribute indicates whether seat heater is on or off.
'true' means seat heater is on. 'false' means it is off.

`seatCooler` of type boolean, readonly, nullable
This attribute indicates whether seat cooler is on or off.
'true' means seat cooler is on. 'false' means it is off.

`windowDriver` of type unsigned short, readonly, nullable
`windowPassenger` of type unsigned short, readonly, nullable
`windowRearLeft` of type unsigned short, readonly, nullable
`windowRearRight` of type unsigned short, readonly, nullable
These attributes contain the percentage of openness per each window.
The unit of value is %. 0% means it's completely closed, 100% means it's fully opened.

`sunroofOpenness` of type unsigned short, readonly, nullable
This attribute contains the percentage of openness of sunroof.
The unit of value is %. 0% means sunroof is completely closed, 100% means it's fully opened.

`sunroofTilt` of type unsigned short, readonly, nullable
This attribute contains the percentage of current degrees of sunroof tilting.
The unit of value is %. 0% means sunroof is completely closed, 100% means it's tilted to maximum.

`convertibleRoof` of type boolean, readonly, nullable
This attribute indicates whether convertible roof is opened or closed.
'true' means roof is opened. 'false' means it is closed.

5. Reference Implementation

Reference implementation of Web API plug-in for accessing Vehicle Data

6. Full Web IDL

```
module vehicle {
 typedef DOMString VehicleEventType;

 interface VehicleError : Error {
  const short ACCESS_DENIED = 1;
  const short NOT_AVAILABLE = 2;
  const short UNKNOWN = 0;
 };

 [NoInterfaceObject]
 interface VehicleEvent : Event {
 };

 [NoInterfaceObject]
 callback interface SuccessCallback {
  void onSuccess();
 };

 [NoInterfaceObject]
 callback interface ErrorCallback {
  void onError(VehicleError error);
 };
 interface VehicleDataHandler {
  void handleVehicleData(VehicleEvent data);
 };

 [NoInterfaceObject]
 interface Vehicle {
  readonly attribute VehicleInterface vehicle;
 };
 genivi implements Vehicle;

 [NoInterfaceObject]
 interface VehicleInterface : EventTarget {
  void get(VehicleEventType type, VehicleDataHandler handler, ErrorCallback errorCB);
  void set(VehicleEventType type, VehicleEvent data, SuccessCallback successCB, ErrorCallback
errorCB);
  VehicleEventType[] getSupportedEventTypes(VehicleEventType type, boolean writable);
 };

 interface VehicleInfoEvent : VehicleEvent {
  const VehicleEventType VEHICLEINFO = "vehicleinfo";
  const VehicleEventType VEHICLEINFO_WMI = "vehicleinfo_wmi";
  const VehicleEventType VEHICLEINFO_VIN = "vehicleinfo_vin";
  const VehicleEventType VEHICLEINFO_VEHICLETYPE = "vehicleinfo_vehicletype";
  const VehicleEventType VEHICLEINFO_DOORTYPE = "vehicleinfo_doortype";
  const VehicleEventType VEHICLEINFO_DOORTYPE_1STROW = "vehicleinfo_doortype_1strow";
  const VehicleEventType VEHICLEINFO_DOORTYPE_2NDROW = "vehicleinfo_doortype_2ndrow";
  const VehicleEventType VEHICLEINFO_DOORTYPE_3RDROW = "vehicleinfo_doortype_3rdrow";
  const VehicleEventType VEHICLEINFO_FUELTYPE = "vehicleinfo_fueltype";
  const VehicleEventType VEHICLEINFO_TRANSMISSIONGEARTYPE = "vehicleinfo_transmissiongeartype";
  const VehicleEventType VEHICLEINFO_WHEELINFO = "vehicleinfo_wheelinfo";
  const VehicleEventType VEHICLEINFO_WHEELINFO_RADIUS = "vehicleinfo_wheelinfo_radius";
  const VehicleEventType VEHICLEINFO_WHEELINFO_TRACK = "vehicleinfo_wheelinfo_track";

  const unsigned short VEHICLETYPE_SEDAN = 1;
  const unsigned short VEHICLETYPE_COUPE = 2;
  const unsigned short VEHICLETYPE_CABRIOLET = 3;
  const unsigned short VEHICLETYPE_ROADSTER = 4;
  const unsigned short VEHICLETYPE_SUV = 5;
  const unsigned short VEHICLETYPE_TRUCK = 6;

  const octet FUELTYPE_GASOLINE = 0x01;
  const octet FUELTYPE_METHANOL= 0x02;
  const octet FUELTYPE_ETHANOL = 0x03;
  const octet FUELTYPE_DIESEL= 0x04;
  const octet FUELTYPE_LPG = 0x05;
  const octet FUELTYPE_CNG = 0x06;
  const octet FUELTYPE_PROPANE = 0x07;
  const octet FUELTYPE_ELECTRIC = 0x08;
  const octet FUELTYPE_BIFUELRUNNINGGASOLINE = 0x09;
  const octet FUELTYPE_BIFUELRUNNINGMETHANOL = 0x0A;
  const octet FUELTYPE_BIFUELRUNNINGETHANOL = 0x0B;
  const octet FUELTYPE_BIFUELRUNNINGLPG = 0x0C;
  const octet FUELTYPE_BIFUELRUNNINGCNG = 0x0D;
  const octet FUELTYPE_BIFUELRUNNINGPROP = 0x0E;
  const octet FUELTYPE_BIFUELRUNNINGELECTRICITY = 0x0F;
  const octet FUELTYPE_BIFUELMIXEDGASELECTRIC= 0x10;
  const octet FUELTYPE_HYBRIDGASOLINE = 0x11;
  const octet FUELTYPE_HYBRIDETHANOL = 0x12;
  const octet FUELTYPE_HYBRIDDIESEL = 0x13;
  const octet FUELTYPE_HYBRIDELECTRIC = 0x14;
  const octet FUELTYPE_HYBRIDMIXEDFUEL = 0x15;
  const octet FUELTYPE_HYBRIDREGENERATIVE = 0x16;

  const unsigned short TRANSMISSIONGEARTYPE_AUTO = 1;
  const unsigned short TRANSMISSIONGEARTYPE_MANUAL = 2;
  const unsigned short TRANSMISSIONGEARTYPE_CVT = 3;

  readonly attribute VehicleEventType type;
  readonly attribute DOMString wmi;
  readonly attribute DOMString vin;
  readonly attribute unsigned short? vehicleType;
  readonly attribute unsigned short? doorType1stRow;
  readonly attribute unsigned short? doorType2ndRow;
  readonly attribute unsigned short? doorType3rdRow;
  readonly attribute octet? fuelType;
  readonly attribute unsigned short? transmissionGearType;
  readonly attribute double? wheelInfoRadius;
  readonly attribute double? wheelInfoTrack;
 };

 interface RunningStatusEvent : VehicleEvent {
  const VehicleEventType RUNNINGSTATUS = "runningstatus";
  const VehicleEventType RUNNINGSTATUS_VEHICLEPOWERMODE = "runningstatus_vehiclepowermode";
  const VehicleEventType RUNNINGSTATUS_SPEEDOMETER = "runningstatus_speedometer";
  const VehicleEventType RUNNINGSTATUS_ENGINESPEED = "runningstatus_enginespeed";
  const VehicleEventType RUNNINGSTATUS_TRIPMETER = "runningstatus_tripmeter";
  const VehicleEventType RUNNINGSTATUS_TRIPMETER_1 = "runningstatus_tripmeter_1";
  const VehicleEventType RUNNINGSTATUS_TRIPMETER_2 = "runningstatus_tripmeter_2";
  const VehicleEventType RUNNINGSTATUS_TRIPMETER_1_MILEAGE = "runningstatus_tripmeter_1_mileage";
  const VehicleEventType RUNNINGSTATUS_TRIPMETER_2_MILEAGE = "runningstatus_tripmeter_2_mileage";
  const VehicleEventType RUNNINGSTATUS_TRIPMETER_1_AVERAGESPEED =
"runningstatus_tripmeter_1_averagespeed";
  const VehicleEventType RUNNINGSTATUS_TRIPMETER_2_AVERAGESPEED =
"runningstatus_tripmeter_2_averagespeed";
  const VehicleEventType RUNNINGSTATUS_TRIPMETER_1_FUELCONSUMPTION =
"runningstatus_tripmeter_1_fuelconsumption";
  const VehicleEventType RUNNINGSTATUS_TRIPMETER_2_FUELCONSUMPTION =
"runningstatus_tripmeter_2_fuelconsumption";
  const VehicleEventType RUNNINGSTATUS_TRANSMISSIONGEARSTATUS =
"runningstatus_transmissiongearstatus";
  const VehicleEventType RUNNINGSTATUS_CRUISECONTROL = "runningstatus_cruisecontrol";
  const VehicleEventType RUNNINGSTATUS_CRUISECONTROL_STATUS =
"runningstatus_cruisecontrol_status";
  const VehicleEventType RUNNINGSTATUS_CRUISECONTROL_SPEED = "runningstatus_cruisecontrol_speed";
  const VehicleEventType RUNNINGSTATUS_WHEELBRAKE = "runningstatus_wheelbrake";
  const VehicleEventType RUNNINGSTATUS_LIGHTSSTATUS = "runningstatus_lightsstatus";
  const VehicleEventType RUNNINGSTATUS_LIGHTSSTATUS_HEAD = "runningstatus_lightsstatus_head";
  const VehicleEventType RUNNINGSTATUS_LIGHTSSTATUS_HIGHBEAM =
"runningstatus_lightsstatus_highbeam";
  const VehicleEventType RUNNINGSTATUS_LIGHTSSTATUS_TURNLEFT =
"runningstatus_lightsstatus_turnleft";
  const VehicleEventType RUNNINGSTATUS_LIGHTSSTATUS_TURNRIGHT =
"runningstatus_lightsstatus_turnright";
  const VehicleEventType RUNNINGSTATUS_LIGHTSSTATUS_BRAKE = "runningstatus_lightsstatus_brake";
  const VehicleEventType RUNNINGSTATUS_LIGHTSSTATUS_FOGFRONT =
"runningstatus_lightsstatus_fogfront";
  const VehicleEventType RUNNINGSTATUS_LIGHTSSTATUS_FOGREAR =
"runningstatus_lightsstatus_fogrear";
  const VehicleEventType RUNNINGSTATUS_LIGHTSSTATUS_HAZARD = "runningstatus_lightsstatus_hazard";
  const VehicleEventType RUNNINGSTATUS_LIGHTSSTATUS_PARKING =
"runningstatus_lightsstatus_parking";
  const VehicleEventType RUNNINGSTATUS_INTERIORLIGHTSSTATUS =
"runningstatus_interiorlightsstatus";
  const VehicleEventType RUNNINGSTATUS_INTERIORLIGHTSSTATUS_DRIVER =
"runningstatus_interiorlightsstatus_driver";
  const VehicleEventType RUNNINGSTATUS_INTERIORLIGHTSSTATUS_PASSENGER =
"runningstatus_interiorlightsstatus_passenger";
  const VehicleEventType RUNNINGSTATUS_INTERIORLIGHTSSTATUS_CENTER =
"runningstatus_interiorlightsstatus_center";
  const VehicleEventType RUNNINGSTATUS_AUTOMATICHEADLIGHTS = "runningstatus_automaticheadlights";
  const VehicleEventType RUNNINGSTATUS_DYNAMICHIGHBEAM = "runningstatus_dynamichighbeam";
  const VehicleEventType RUNNINGSTATUS_HORN = "runningstatus_horn";
  const VehicleEventType RUNNINGSTATUS_CHIME = "runningstatus_chime";
  const VehicleEventType RUNNINGSTATUS_FUEL = "runningstatus_fuel";
  const VehicleEventType RUNNINGSTATUS_ESTIMATEDRANGE = "runningstatus_estimatedrange";
  const VehicleEventType RUNNINGSTATUS_ENGINEOIL = "runningstatus_engineoil";
  const VehicleEventType RUNNINGSTATUS_ENGINEOIL_REMAINING = "runningstatus_engineoil_remaining";
  const VehicleEventType RUNNINGSTATUS_ENGINEOIL_CHANGE = "runningstatus_engineoil_change";
  const VehicleEventType RUNNINGSTATUS_ENGINEOIL_TEMP = "runningstatus_engineoil_temp";
  const VehicleEventType RUNNINGSTATUS_ENGINECOOLANT = "runningstatus_enginecoolant";
  const VehicleEventType RUNNINGSTATUS_ENGINECOOLANT_LEVEL = "runningstatus_enginecoolant_level";
  const VehicleEventType RUNNINGSTATUS_ENGINECOOLANT_TEMP = "runningstatus_enginecoolant_temp";
  const VehicleEventType RUNNINGSTATUS_STEERINGWHEELANGLE = "runningstatus_steeringwheelangle";

  const unsigned short VEHICLEPOWERMODE_OFF = 1;
  const unsigned short VEHICLEPOWERMODE_ACC = 2;
  const unsigned short VEHICLEPOWERMODE_RUN = 3;
  const unsigned short VEHICLEPOWERMODE_IGNITION = 4;
  const unsigned short TRANSMISSIONGEARSTATUS_NEUTRAL = 0;
  const unsigned short TRANSMISSIONGEARSTATUS_MANUAL1 = 1;
  const unsigned short TRANSMISSIONGEARSTATUS_MANUAL2 = 2;
  const unsigned short TRANSMISSIONGEARSTATUS_MANUAL3 = 3;
  const unsigned short TRANSMISSIONGEARSTATUS_MANUAL4 = 4;
  const unsigned short TRANSMISSIONGEARSTATUS_MANUAL5 = 5;
  const unsigned short TRANSMISSIONGEARSTATUS_MANUAL6 = 6;
  const unsigned short TRANSMISSIONGEARSTATUS_MANUAL7 = 7;
  const unsigned short TRANSMISSIONGEARSTATUS_MANUAL8 = 8;
  const unsigned short TRANSMISSIONGEARSTATUS_MANUAL9 = 9;
  const unsigned short TRANSMISSIONGEARSTATUS_MANUAL10 = 10;
  const unsigned short TRANSMISSIONGEARSTATUS_AUTO1 = 11;
  const unsigned short TRANSMISSIONGEARSTATUS_AUTO2 = 12;
  const unsigned short TRANSMISSIONGEARSTATUS_AUTO3 = 13;
  const unsigned short TRANSMISSIONGEARSTATUS_AUTO4 = 14;
  const unsigned short TRANSMISSIONGEARSTATUS_AUTO5 = 15;
  const unsigned short TRANSMISSIONGEARSTATUS_AUTO6 = 16;
  const unsigned short TRANSMISSIONGEARSTATUS_AUTO7 = 17;
  const unsigned short TRANSMISSIONGEARSTATUS_AUTO8 = 18;
  const unsigned short TRANSMISSIONGEARSTATUS_AUTO9 = 19;
  const unsigned short TRANSMISSIONGEARSTATUS_AUTO10 = 20;
  const unsigned short TRANSMISSIONGEARSTATUS_DRIVE = 32;
  const unsigned short TRANSMISSIONGEARSTATUS_PARKING = 64;
  const unsigned short TRANSMISSIONGEARSTATUS_REVERSE = 128;

  const unsigned short WHEELBRAKE_IDLE = 1;
  const unsigned short WHEELBRAKE_ENGAGED = 2;
  const unsigned short WHEELBRAKE_MALFUNCTION = 3;

  const unsigned short ENGINECOOLANTLEVEL_NORMAL = 0;
  const unsigned short ENGINECOOLANTLEVEL_LOW = 1;

  readonly attribute VehicleEventType type;
  readonly attribute unsigned short? vehiclePowerMode;
  readonly attribute unsigned short speedometer;
  readonly attribute unsigned short? engineSpeed;
  readonly attribute unsigned long? tripMeter1Mileage;
  readonly attribute unsigned long? tripMeter2Mileage;
  readonly attribute unsigned short? tripMeter1AverageSpeed;
  readonly attribute unsigned short? tripMeter2AverageSpeed;
  readonly attribute unsigned long? tripMeter1FuelConsumption;
  readonly attribute unsigned long? tripMeter2FuelConsumption;
  readonly attribute unsigned short? transmissionGearStatus;
  readonly attribute boolean? cruiseControlStatus;
  readonly attribute unsigned short? cruiseControlSpeed;
  readonly attribute unsigned short? wheelBrake;
  readonly attribute boolean? lightsStatusHead;
  readonly attribute boolean? lightsStatusHighBeam;
  readonly attribute boolean? lightsStatusTurnLeft;
  readonly attribute boolean? lightsStatusTurnRight;
  readonly attribute boolean? lightsStatusBrake;
  readonly attribute boolean? lightsStatusFogFront;
  readonly attribute boolean? lightsStatusFogRear;
  readonly attribute boolean? lightsStatusHazard;
  readonly attribute boolean? lightsStatusParking;
  readonly attribute boolean? interiorLightsStatusDriver;
  readonly attribute boolean? interiorLightsStatusPassenger;
  readonly attribute boolean? interiorLightsStatusCenter;
  readonly attribute boolean? automaticHeadlights;
  readonly attribute boolean? dynamicHighBeam;
  readonly attribute boolean? horn;
  readonly attribute boolean? chime;
  readonly attribute unsigned short fuel;
  readonly attribute unsigned long? estimatedRange;
  readonly attribute unsigned short? engineOilRemaining;
  readonly attribute boolean? engineOilChange;
  readonly attribute short? engineOilTemp;
  readonly attribute unsigned short? engineCoolantLevel;
  readonly attribute short? engineCoolantTemp;
  readonly attribute short? steeringWheelAngle;
 };

 interface MaintenanceEvent : VehicleEvent {
  const VehicleEventType MAINTENANCE = "maintenance";
  const VehicleEventType MAINTENANCE_ODOMETER = "maintenance_odometer";
```

```
  const VehicleEventType MAINTENANCE_TRANSMISSIONOIL = "maintenance_transmissionoil";
  const VehicleEventType MAINTENANCE_TRANSMISSIONOIL_LIFELEVEL =
"maintenance_transmissionoil_lifelevel";
  const VehicleEventType MAINTENANCE_TRANSMISSIONOIL_TEMP = "maintenance_transmissionoil_temp";
  const VehicleEventType MAINTENANCE_BRAKEFLUIDLEVEL = "maintenance_brakefluidlevel";
  const VehicleEventType MAINTENANCE_WASHERFLUIDLEVEL = "maintenance_washerfluidlevel";
  const VehicleEventType MAINTENANCE_MALFUNCTIONINDICATORLAMP =
"maintenance_malfunctionindicatorlamp";
  const VehicleEventType MAINTENANCE_BATTERY = "maintenance_battery";
  const VehicleEventType MAINTENANCE_BATTERY_VOLTAGE = "maintenance_battery_voltage";
  const VehicleEventType MAINTENANCE_BATTERY_CURRENT = "maintenance_battery_current";
  const VehicleEventType MAINTENANCE_TIREPRESSURE = "maintenance_tirepressure";
  const VehicleEventType MAINTENANCE_TIREPRESSURE_FRONTLEFT =
"maintenance_tirepressure_frontleft";
  const VehicleEventType MAINTENANCE_TIREPRESSURE_FRONTRIGHT =
"maintenance_tirepressure_frontright";
  const VehicleEventType MAINTENANCE_TIREPRESSURE_REARLEFT = "maintenance_tirepressure_rearleft";
  const VehicleEventType MAINTENANCE_TIREPRESSURE_REARRIGHT =
"maintenance_tirepressure_rearright";
  const VehicleEventType MAINTENANCE_TIREPRESSURESTATUS = "maintenance_tirepressurestatus";
  const VehicleEventType MAINTENANCE_TIREPRESSURESTATUS_FRONTLEFT =
"maintenance_tirepressurestatus_frontleft";
  const VehicleEventType MAINTENANCE_TIREPRESSURESTATUS_FRONTRIGHT =
"maintenance_tirepressurestatus_frontright";
  const VehicleEventType MAINTENANCE_TIREPRESSURESTATUS_REARLEFT =
"maintenance_tirepressurestatus_rearleft";
  const VehicleEventType MAINTENANCE_TIREPRESSURESTATUS_REARRIGHT =
"maintenance_tirepressurestatus_rearright";

  const unsigned short TIREPRESSURESTATUS_NORMAL = 0;
  const unsigned short TIREPRESSURESTATUS_LOW = 1;
  const unsigned short TIREPRESSURESTATUS_HIGH = 2;

  readonly attribute VehicleEventType type;
  readonly attribute unsigned long odometer;
  readonly attribute boolean? transmissionOilLifeLevel;
  readonly attribute short? transmissionOilTemp;
  readonly attribute boolean? brakeFluidLevel;
  readonly attribute boolean? washerFluidLevel;
  readonly attribute boolean? malfunctionIndicatorLamp;
  readonly attribute unsigned short? batteryVoltage;
  readonly attribute unsigned short? batteryCurrent;
  readonly attribute unsigned short? tirePressureFrontLeft;
  readonly attribute unsigned short? tirePressureFrontRight;
  readonly attribute unsigned short? tirePressureRearLeft;
  readonly attribute unsigned short? tirePressureRearRight;
  readonly attribute unsigned short? tirePressureStatusFrontLeft;
  readonly attribute unsigned short? tirePressureStatusFrontRight;
  readonly attribute unsigned short? tirePressureStatusRearLeft;
  readonly attribute unsigned short? tirePressureStatusRearRight;
 };

 interface PersonalizationEvent : VehicleEvent {
  const VehicleEventType PERSONALIZATION = "personalization";
  const VehicleEventType PERSONALIZATION_KEYID = "personalization_keyid";
  const VehicleEventType PERSONALIZATION_LANGUAGE = "personalization_language";
  const VehicleEventType PERSONALIZATION_MEASUREMENTSYSTEM = "personalization_measurementsystem";
  const VehicleEventType PERSONALIZATION_MEASUREMENTSYSTEMSTRING =
"personalization_measurementsystemstring";
  const VehicleEventType PERSONALIZATION_MEASUREMENTSYSTEMSTRING_FUEL =
"personalization_measurementsystemstring_fuel";
  const VehicleEventType PERSONALIZATION_MEASUREMENTSYSTEMSTRING_DISTANCE =
"personalization_measurementsystemstring_distance";
  const VehicleEventType PERSONALIZATION_MEASUREMENTSYSTEMSTRING_SPEED =
"personalization_measurementsystemstring_speed";
  const VehicleEventType PERSONALIZATION_MEASUREMENTSYSTEMSTRING_CONSUMPTION =
"personalization_measurementsystemstring_consumption";
  const VehicleEventType PERSONALIZATION_MIRROR = "personalization_mirror";
  const VehicleEventType PERSONALIZATION_MIRROR_DRIVER = "personalization_mirror_driver";
  const VehicleEventType PERSONALIZATION_MIRROR_PASSENGER = "personalization_mirror_passenger";
  const VehicleEventType PERSONALIZATION_MIRROR_INSIDE = "personalization_mirror_inside";
  const VehicleEventType PERSONALIZATION_STEERINGWHEELPOSITION =
"personalization_steeringwheelposition";
  const VehicleEventType PERSONALIZATION_STEERINGWHEELPOSITION_SLIDE =
"personalization_steeringwheelposition_slide";
```

```
  const VehicleEventType PERSONALIZATION_STEERINGWHEELPOSITION_TILT =
"personalization_steeringwheelposition_tilt";
  const VehicleEventType PERSONALIZATION_DRIVINGMODE = "personalization_drivingmode";
  const VehicleEventType PERSONALIZATION_DRIVERSEATPOSITION =
"personalization_driverseatposition";
  const VehicleEventType PERSONALIZATION_DRIVERSEATPOSITION_RECLINESEATBACK =
"personalization_driverseatposition_reclineseatback";
  const VehicleEventType PERSONALIZATION_DRIVERSEATPOSITION_SLIDE =
"personalization_driverseatposition_slide";
  const VehicleEventType PERSONALIZATION_DRIVERSEATPOSITION_CUSHIONHEIGHT =
"personalization_driverseatposition_cushionheight";
  const VehicleEventType PERSONALIZATION_DRIVERSEATPOSITION_HEADREST =
"personalization_driverseatposition_headrest";
  const VehicleEventType PERSONALIZATION_DRIVERSEATPOSITION_BACKCUSHION =
"personalization_driverseatposition_backcushion";
  const VehicleEventType PERSONALIZATION_DRIVERSEATPOSITION_SIDECUSHION =
"personalization_driverseatposition_sidecushion";
  const VehicleEventType PERSONALIZATION_PASSENGERSEATPOSITION =
"personalization_passengerseatposition";
  const VehicleEventType PERSONALIZATION_PASSENGERSEATPOSITION_RECLINESEATBACK =
"personalization_passengerseatposition_reclineseatback";
  const VehicleEventType PERSONALIZATION_PASSENGERSEATPOSITION_SLIDE =
"personalization_passengerseatposition_slide";
  const VehicleEventType PERSONALIZATION_PASSENGERSEATPOSITION_CUSHIONHEIGHT =
"personalization_passengerseatposition_cushionheight";
  const VehicleEventType PERSONALIZATION_PASSENGERSEATPOSITION_HEADREST =
"personalization_passengerseatposition_headrest";
  const VehicleEventType PERSONALIZATION_PASSENGERSEATPOSITION_BACKCUSHION =
"personalization_passengerseatposition_backcushion";
  const VehicleEventType PERSONALIZATION_PASSENGERSEATPOSITION_SIDECUSHION =
"personalization_passengerseatposition_sidecushion";
  const VehicleEventType PERSONALIZATION_DASHBOARDILLUMINATION =
"personalization_dashboardillumination";
  const VehicleEventType PERSONALIZATION_GENERATEDVEHICLESOUNDMODE =
"personalization_generatedvehiclesoundmode";

  const unsigned short LANGUAGE_ENGLISH = 1;
  const unsigned short LANGUAGE_SPANISH = 2;
  const unsigned short LANGUAGE_FRENCH = 3;

  const unsigned short DRIVINGMODE_COMFORT = 1;
  const unsigned short DRIVINGMODE_AUTO = 2;
  const unsigned short DRIVINGMODE_SPORT = 3;
  const unsigned short DRIVINGMODE_ECO = 4;
  const unsigned short DRIVINGMODE_MANUAL = 5;

  const unsigned short GENERATEDVEHICLESOUNDMODE_NORMAL = 1;
  const unsigned short GENERATEDVEHICLESOUNDMODE_QUIET = 2;
  const unsigned short GENERATEDVEHICLESOUNDMODE_SPORTIVE = 3;

  readonly attribute VehicleEventType type;
  readonly attribute DOMString? keyId;
  readonly attribute unsigned short? language;
  readonly attribute boolean? measurementSystem;
  readonly attribute DOMString? measurementSystemStringFuel;
  readonly attribute DOMString? measurementSystemStringDistance;
  readonly attribute DOMString? measurementSystemStringSpeed;
  readonly attribute DOMString? measurementSystemStringConsumption;
  readonly attribute unsigned short? mirrorDriver;
  readonly attribute unsigned short? mirrorPassenger;
  readonly attribute unsigned short? mirrorInside;
  readonly attribute unsigned short? steeringWheelPositionSlide;
  readonly attribute unsigned short? steeringWheelPositionTilt;
  readonly attribute unsigned short? drivingMode;
  readonly attribute unsigned short? driverSeatPositionReclineSeatback;
  readonly attribute unsigned short? driverSeatPositionSlide;
  readonly attribute unsigned short? driverSeatPositionCushionHeight;
  readonly attribute unsigned short? driverSeatPositionHeadrest;
  readonly attribute unsigned short? driverSeatPositionBackCushion;
  readonly attribute unsigned short? driverSeatPositionSideCushion;
  readonly attribute unsigned short? passengerSeatPositionReclineSeatback;
  readonly attribute unsigned short? passengerSeatPositionSlide;
  readonly attribute unsigned short? passengerSeatPositionCushionHeight;
  readonly attribute unsigned short? passengerSeatPositionHeadrest;
  readonly attribute unsigned short? passengerSeatPositionBackCushion;
  readonly attribute unsigned short? passengerSeatPositionSideCushion;
  readonly attribute unsigned short? dashboardIllumination;
  readonly attribute unsigned short? generatedVehicleSoundMode;
 };

 interface DrivingSafetyEvent : VehicleEvent {
  const VehicleEventType DRIVINGSAFETY = "drivingsafety";
  const VehicleEventType DRIVINGSAFETY_ANTILOCKBRAKINGSYSTEM =
"drivingsafety_antilockbrakingsystem";
  const VehicleEventType DRIVINGSAFETY_TRACTIONCONTROLSYSTEM =
"drivingsafety_tractioncontrolsystem";
  const VehicleEventType DRIVINGSAFETY_ELECTRONICSTABILITYCONTROL =
"drivingsafety_electronicstabilitycontrol";
  const VehicleEventType DRIVINGSAFETY_VEHICLETOPSPEEDLIMIT =
"drivingsafety_vehicletopspeedlimit";
  const VehicleEventType DRIVINGSAFETY_AIRBAGSTATUS = "drivingsafety_airbagstatus";
  const VehicleEventType DRIVINGSAFETY_AIRBAGSTATUS_DRIVER = "drivingsafety_airbagstatus_driver";
  const VehicleEventType DRIVINGSAFETY_AIRBAGSTATUS_PASSENGER =
"drivingsafety_airbagstatus_passenger";
  const VehicleEventType DRIVINGSAFETY_AIRBAGSTATUS_SIDE = "drivingsafety_airbagstatus_side";
  const VehicleEventType DRIVINGSAFETY_DOOROPENSTATUS = "drivingsafety_dooropenstatus";
  const VehicleEventType DRIVINGSAFETY_DOOROPENSTATUS_DRIVER =
"drivingsafety_dooropenstatus_driver";
  const VehicleEventType DRIVINGSAFETY_DOOROPENSTATUS_PASSENGER =
"drivingsafety_dooropenstatus_passenger";
  const VehicleEventType DRIVINGSAFETY_DOOROPENSTATUS_REARLEFT =
"drivingsafety_dooropenstatus_rearleft";
  const VehicleEventType DRIVINGSAFETY_DOOROPENSTATUS_REARRIGHT =
"drivingsafety_dooropenstatus_rearright";
  const VehicleEventType DRIVINGSAFETY_DOOROPENSTATUS_TRUNK =
"drivingsafety_dooropenstatus_trunk";
  const VehicleEventType DRIVINGSAFETY_DOOROPENSTATUS_FUELFILTERCAP =
"drivingsafety_dooropenstatus_fuelfiltercap";
  const VehicleEventType DRIVINGSAFETY_DOOROPENSTATUS_HOOD = "drivingsafety_dooropenstatus_hood";
  const VehicleEventType DRIVINGSAFETY_DOORLOCKSTATUS = "drivingsafety_doorlockstatus";
  const VehicleEventType DRIVINGSAFETY_DOORLOCKSTATUS_DRIVER =
"drivingsafety_doorlockstatus_driver";
  const VehicleEventType DRIVINGSAFETY_DOORLOCKSTATUS_PASSENGER =
"drivingsafety_doorlockstatus_passenger";
  const VehicleEventType DRIVINGSAFETY_DOORLOCKSTATUS_REARLEFT =
"drivingsafety_doorlockstatus_rearleft";
  const VehicleEventType DRIVINGSAFETY_DOORLOCKSTATUS_REARRIGHT =
"drivingsafety_doorlockstatus_rearright";
  const VehicleEventType DRIVINGSAFETY_CHILDSAFETYLOCK = "drivingsafety_childsafetylock";
  const VehicleEventType DRIVINGSAFETY_OCCUPANTSSTATUS = "drivingsafety_occupantsstatus";
  const VehicleEventType DRIVINGSAFETY_OCCUPANTSSTATUS_DRIVER =
"drivingsafety_occupantsstatus_driver";
  const VehicleEventType DRIVINGSAFETY_OCCUPANTSSTATUS_PASSENGER =
"drivingsafety_occupantsstatus_passenger";
  const VehicleEventType DRIVINGSAFETY_OCCUPANTSSTATUS_REARLEFT =
"drivingsafety_occupantsstatus_rearleft";
  const VehicleEventType DRIVINGSAFETY_OCCUPANTSSTATUS_REARRIGHT =
"drivingsafety_occupantsstatus_rearright";
  const VehicleEventType DRIVINGSAFETY_SEATBELT = "drivingsafety_seatbelt";
  const VehicleEventType DRIVINGSAFETY_SEATBELT_DRIVER = "drivingsafety_seatbelt_driver";
  const VehicleEventType DRIVINGSAFETY_SEATBELT_PASSENGER = "drivingsafety_seatbelt_passenger";
  const VehicleEventType DRIVINGSAFETY_SEATBELT_REARLEFT = "drivingsafety_seatbelt_rearleft";
  const VehicleEventType DRIVINGSAFETY_SEATBELT_REARRIGHT = "drivingsafety_seatbelt_rearright";
  const VehicleEventType DRIVINGSAFETY_WINDOWLOCK = "drivingsafety_windowlock";
  const VehicleEventType DRIVINGSAFETY_WINDOWLOCK_DRIVER = "drivingsafety_windowlock_driver";
  const VehicleEventType DRIVINGSAFETY_WINDOWLOCK_PASSENGER =
"drivingsafety_windowlock_passenger";
  const VehicleEventType DRIVINGSAFETY_WINDOWLOCK_REARLEFT = "drivingsafety_windowlock_rearleft";
  const VehicleEventType DRIVINGSAFETY_WINDOWLOCK_REARRIGHT =
"drivingsafety_windowlock_rearright";
  const VehicleEventType DRIVINGSAFETY_OBSTACLEDISTANCE = "drivingsafety_obstacledistance";
  const VehicleEventType DRIVINGSAFETY_OBSTACLEDISTANCE_SENSORSTATUS =
"drivingsafety_obstacledistance_sensorstatus";
  const VehicleEventType DRIVINGSAFETY_OBSTACLEDISTANCE_FRONTCENTER =
"drivingsafety_obstacledistance_frontcenter";
  const VehicleEventType DRIVINGSAFETY_OBSTACLEDISTANCE_REARCENTER =
"drivingsafety_obstacledistance_rearcenter";
  const VehicleEventType DRIVINGSAFETY_OBSTACLEDISTANCE_FRONTLEFT =
"drivingsafety_obstacledistance_frontleft";
  const VehicleEventType DRIVINGSAFETY_OBSTACLEDISTANCE_FRONTRIGHT =
"drivingsafety_obstacledistance_frontright";
  const VehicleEventType DRIVINGSAFETY_OBSTACLEDISTANCE_MIDDLELEFT =
"drivingsafety_obstacledistance_middleleft";
  const VehicleEventType DRIVINGSAFETY_OBSTACLEDISTANCE_MIDDLERIGHT =
"drivingsafety_obstacledistance_middleright";
  const VehicleEventType DRIVINGSAFETY_OBSTACLEDISTANCE_REARLEFT =
"drivingsafety_obstacledistance_rearleft";
  const VehicleEventType DRIVINGSAFETY_OBSTACLEDISTANCE_REARRIGHT =
"drivingsafety_obstacledistance_rearright";
  const VehicleEventType DRIVINGSAFETY_FRONTCOLLISIONDETECTION =
"drivingsafety_frontcollisiondetection";
  const VehicleEventType DRIVINGSAFETY_FRONTCOLLISIONDETECTION_STATUS =
"drivingsafety_frontcollisiondetection_status";
  const VehicleEventType DRIVINGSAFETY_FRONTCOLLISIONDETECTION_DISTANCE =
"drivingsafety_frontcollisiondetection_distance";
  const VehicleEventType DRIVINGSAFETY_FRONTCOLLISIONDETECTION_TIME =
"drivingsafety_frontcollisiondetection_time";

  const unsigned short ANTILOCKBRAKINGSYSTEM_AVAILABLE = 1;
  const unsigned short ANTILOCKBRAKINGSYSTEM_IDLE = 2;
  const unsigned short ANTILOCKBRAKINGSYSTEM_ENGAGED = 3;

  const unsigned short TRACTIONCONTROLSYSTEM_AVAILABLE = 1;
  const unsigned short TRACTIONCONTROLSYSTEM_IDLE = 2;
  const unsigned short TRACTIONCONTROLSYSTEM_ENGAGED = 3;

  const unsigned short ELECTRONICSTABILITYCONTROL_AVAILABLE = 1;
  const unsigned short ELECTRONICSTABILITYCONTROL_IDLE = 2;
  const unsigned short ELECTRONICSTABILITYCONTROL_ENGAGED = 3;

  const unsigned short AIRBAGSTATUS_ACTIVATE = 1;
  const unsigned short AIRBAGSTATUS_DEACTIVATE = 2;
  const unsigned short AIRBAGSTATUS_DEPLOYMENT = 3;

  const unsigned short DOOROPENSTATUS_OPEN = 1;
  const unsigned short DOOROPENSTATUS_AJAR = 2;
  const unsigned short DOOROPENSTATUS_CLOSE = 3;

  const unsigned short OCCUPANTSSTATUS_ADULT = 1;
  const unsigned short OCCUPANTSSTATUS_CHILD = 2;
  const unsigned short OCCUPANTSSTATUS_VACANT = 3;

  readonly attribute VehicleEventType type;
  readonly attribute unsigned short? antilockBrakingSystem;
  readonly attribute unsigned short? tractionControlSystem;
  readonly attribute unsigned short? electronicStabilityControl;
  readonly attribute unsigned short? vehicleTopSpeedLimit;
  readonly attribute unsigned short? airbagStatusDriver;
  readonly attribute unsigned short? airbagStatusPassenger;
  readonly attribute unsigned short? airbagStatusSide;
  readonly attribute unsigned short? doorOpenStatusDriver;
  readonly attribute unsigned short? doorOpenStatusPassenger;
  readonly attribute unsigned short? doorOpenStatusRearLeft;
  readonly attribute unsigned short? doorOpenStatusRearRight;
  readonly attribute unsigned short? doorOpenStatusTrunk;
  readonly attribute unsigned short? doorOpenStatusFuelFilterCap;
  readonly attribute unsigned short? doorOpenStatusHood;
  readonly attribute boolean? doorLockStatusDriver;
  readonly attribute boolean? doorLockStatusPassenger;
  readonly attribute boolean? doorLockStatusRearLeft;
  readonly attribute boolean? doorLockStatusRearRight;
  readonly attribute boolean? childSafetyLock;
  readonly attribute unsigned short? occupantsStatusDriver;
  readonly attribute unsigned short? occupantsStatusPassenger;
  readonly attribute unsigned short? occupantsStatusRearLeft;
  readonly attribute unsigned short? occupantsStatusRearRight;
  readonly attribute boolean? seatBeltDriver;
  readonly attribute boolean? seatBeltPassenger;
  readonly attribute boolean? seatBeltRearLeft;
  readonly attribute boolean? seatBeltRearRight;
  readonly attribute boolean? windowLockDriver;
  readonly attribute boolean? windowLockPassenger;
  readonly attribute boolean? windowLockRearLeft;
  readonly attribute boolean? windowLockRearRight;
  readonly attribute boolean? obstacleDistanceSensorStatus;
  readonly attribute unsigned short? obstacleDistanceFrontCenter;
  readonly attribute unsigned short? obstacleDistanceRearCenter;
  readonly attribute unsigned short? obstacleDistanceFrontLeft;
  readonly attribute unsigned short? obstacleDistanceFrontRight;
  readonly attribute unsigned short? obstacleDistanceMiddleLeft;
  readonly attribute unsigned short? obstacleDistanceMiddleRight;
  readonly attribute unsigned short? obstacleDistanceRearLeft;
  readonly attribute unsigned short? obstacleDistanceRearRight;
  readonly attribute boolean? frontCollisionDetectionStatus;
  readonly attribute unsigned long? frontCollisionDetectionDistance;
  readonly attribute unsigned long? frontCollisionDetectionTime;
 };

 interface VisionSystemEvent : VehicleEvent {
  const VehicleEventType VISIONSYSTEM = "visionsystem";
  const VehicleEventType VISIONSYSTEM_LANEDEPARTUREDETECTIONSTATUS =
"visionsystem_lanedeparturedetectionstatus";
  const VehicleEventType VISIONSYSTEM_LANEDEPARTED = "visionsystem_lanedeparted";

  readonly attribute VehicleEventType type;
  readonly attribute boolean? laneDepartureDetectionStatus;
  readonly attribute boolean? laneDeparted;
 };

 interface ParkingEvent : VehicleEvent {
  const VehicleEventType PARKING = "parking";
  const VehicleEventType PARKING_SECURITYALERT = "parking_securityalert";
  const VehicleEventType PARKING_PARKINGBRAKE = "parking_parkingbrake";
  const VehicleEventType PARKING_PARKINGLIGHTS = "parking_parkinglights";

  const unsigned short SECURITYALERT_AVAILABLE = 1;
  const unsigned short SECURITYALERT_IDLE = 2;
  const unsigned short SECURITYALERT_ACTIVATED = 3;
  const unsigned short SECURITYALERT_ALARM_DETECTED = 4;

  readonly attribute VehicleEventType type;
  readonly attribute unsigned short? securityAlert;
  readonly attribute boolean? parkingBrake;
  readonly attribute boolean? parkingLights;
 };

 interface ClimateEnvironmentEvent : VehicleEvent {
  const VehicleEventType CLIMATEENVIRONMENT = "climateenvironment";
  const VehicleEventType CLIMATEENVIRONMENT_INTERIORTEMP = "climateenvironment_interiortemp";
  const VehicleEventType CLIMATEENVIRONMENT_EXTERIORTEMP = "climateenvironment_exteriortemp";
  const VehicleEventType CLIMATEENVIRONMENT_EXTERIORBRIGHTNESS =
"climateenvironment_exteriorbrightness";
  const VehicleEventType CLIMATEENVIRONMENT_RAINSENSOR = "climateenvironment_rainsensor";
  const VehicleEventType CLIMATEENVIRONMENT_WINDSHIELDWIPER =
"climateenvironment_windshieldwiper";
  const VehicleEventType CLIMATEENVIRONMENT_REARWIPER = "climateenvironment_rearwiper";
  const VehicleEventType CLIMATEENVIRONMENT_HVACFAN = "climateenvironment_hvacfan";
  const VehicleEventType CLIMATEENVIRONMENT_HVACFAN_DIRECTION =
"climateenvironment_hvacfan_direction";
  const VehicleEventType CLIMATEENVIRONMENT_HVACFAN_SPEED = "climateenvironment_hvacfan_speed";
  const VehicleEventType CLIMATEENVIRONMENT_HVACFAN_TARGETTEMP =
"climateenvironment_hvacfan_targettemp";
  const VehicleEventType CLIMATEENVIRONMENT_AIRCONDITIONING =
"climateenvironment_airconditioning";
  const VehicleEventType CLIMATEENVIRONMENT_AIRRECIRCULATION =
"climateenvironment_airrecirculation";
  const VehicleEventType CLIMATEENVIRONMENT_HEATER = "climateenvironment_heater";
  const VehicleEventType CLIMATEENVIRONMENT_DEFROST = "climateenvironment_defrost";
  const VehicleEventType CLIMATEENVIRONMENT_DEFROST_WINDSHIELD =
"climateenvironment_defrost_windshield";
  const VehicleEventType CLIMATEENVIRONMENT_DEFROST_REARWINDOW =
"climateenvironment_defrost_rearwindow";
  const VehicleEventType CLIMATEENVIRONMENT_DEFROST_SIDEMIRRORS =
"climateenvironment_defrost_sidemirrors";
  const VehicleEventType CLIMATEENVIRONMENT_STEERINGWHEELHEATER =
"climateenvironment_steeringwheelheater";
  const VehicleEventType CLIMATEENVIRONMENT_SEATHEATER = "climateenvironment_seatheater";
  const VehicleEventType CLIMATEENVIRONMENT_SEATCOOLER = "climateenvironment_seatcooler";
  const VehicleEventType CLIMATEENVIRONMENT_WINDOW = "climateenvironment_window";
  const VehicleEventType CLIMATEENVIRONMENT_WINDOW_DRIVER = "climateenvironment_window_driver";
  const VehicleEventType CLIMATEENVIRONMENT_WINDOW_PASSENGER =
"climateenvironment_window_passenger";
  const VehicleEventType CLIMATEENVIRONMENT_WINDOW_REARLEFT =
"climateenvironment_window_rearleft";
  const VehicleEventType CLIMATEENVIRONMENT_WINDOW_REARRIGHT =
"climateenvironment_window_rearright";
  const VehicleEventType CLIMATEENVIRONMENT_SUNROOF = "climateenvironment_sunroof";
  const VehicleEventType CLIMATEENVIRONMENT_SUNROOF_OPENNESS =
"climateenvironment_sunroof_openness";
  const VehicleEventType CLIMATEENVIRONMENT_SUNROOF_TILT = "climateenvironment_sunroof_tilt";
  const VehicleEventType CLIMATEENVIRONMENT_CONVERTIBLEROOF =
"climateenvironment_convertibleroof";

  const unsigned short RAINSENSOR_NORAIN = 0;
  const unsigned short RAINSENSOR_LEVEL1 = 1;
  const unsigned short RAINSENSOR_LEVEL2 = 2;
  const unsigned short RAINSENSOR_LEVEL3 = 3;
  const unsigned short RAINSENSOR_LEVEL4 = 4;
  const unsigned short RAINSENSOR_LEVEL5 = 5;
  const unsigned short RAINSENSOR_LEVEL6 = 6;
  const unsigned short RAINSENSOR_LEVEL7 = 7;
  const unsigned short RAINSENSOR_LEVEL8 = 8;
  const unsigned short RAINSENSOR_LEVEL9 = 9;
  const unsigned short RAINSENSOR_HEAVIESTRAIN = 10;

  const unsigned short WIPER_OFF = 0;
  const unsigned short WIPER_ONCE = 1;
  const unsigned short WIPER_SLOWEST = 2;
  const unsigned short WIPER_SLOW = 3;
  const unsigned short WIPER_FAST = 4;
  const unsigned short WIPER_FASTEST = 5;
  const unsigned short WIPER_AUTO = 10;

  const unsigned short HVACFAN_DIRECTION_FRONTPANEL = 1;
  const unsigned short HVACFAN_DIRECTION_FLOORDUCT = 2;
  const unsigned short HVACFAN_DIRECTION_FRONTFLOOR = 3;
  const unsigned short HVACFAN_DIRECTION_DEFROSTERFLOOR = 4;

  readonly attribute VehicleEventType type;
  readonly attribute short? interiorTemp;
  readonly attribute short? exteriorTemp;
  readonly attribute unsigned long? exteriorBrightness;
  readonly attribute unsigned short? rainSensor;
  readonly attribute unsigned short? windshieldWiper;
  readonly attribute unsigned short? rearWiper;
  readonly attribute unsigned short? hvacFanDirection;
  readonly attribute unsigned short? hvacFanSpeed;
  readonly attribute short? hvacFanTargetTemp;
  readonly attribute boolean? airConditioning;
  readonly attribute boolean? airRecirculation;
  readonly attribute boolean? heater;
  readonly attribute boolean? defrostWindshield;
  readonly attribute boolean? defrostRearWindow;
  readonly attribute boolean? defrostSideMirrors;
  readonly attribute boolean? steeringWheelHeater;
  readonly attribute boolean? seatHeater;
  readonly attribute boolean? seatCooler;
  readonly attribute unsigned short? windowDriver;
  readonly attribute unsigned short? windowPassenger;
  readonly attribute unsigned short? windowRearLeft;
  readonly attribute unsigned short? windowRearRight;
  readonly attribute unsigned short? sunroofOpenness;
  readonly attribute unsigned short? sunroofTilt;
  readonly attribute boolean? convertibleRoof;
 };
}
```