Tizen Vehicle Information Web API Specification

Kevron Rees
April 16,2013

Contents

- Overview
- Use-Cases
- Examples
- Data Types
- Events
- Future plans

Overview

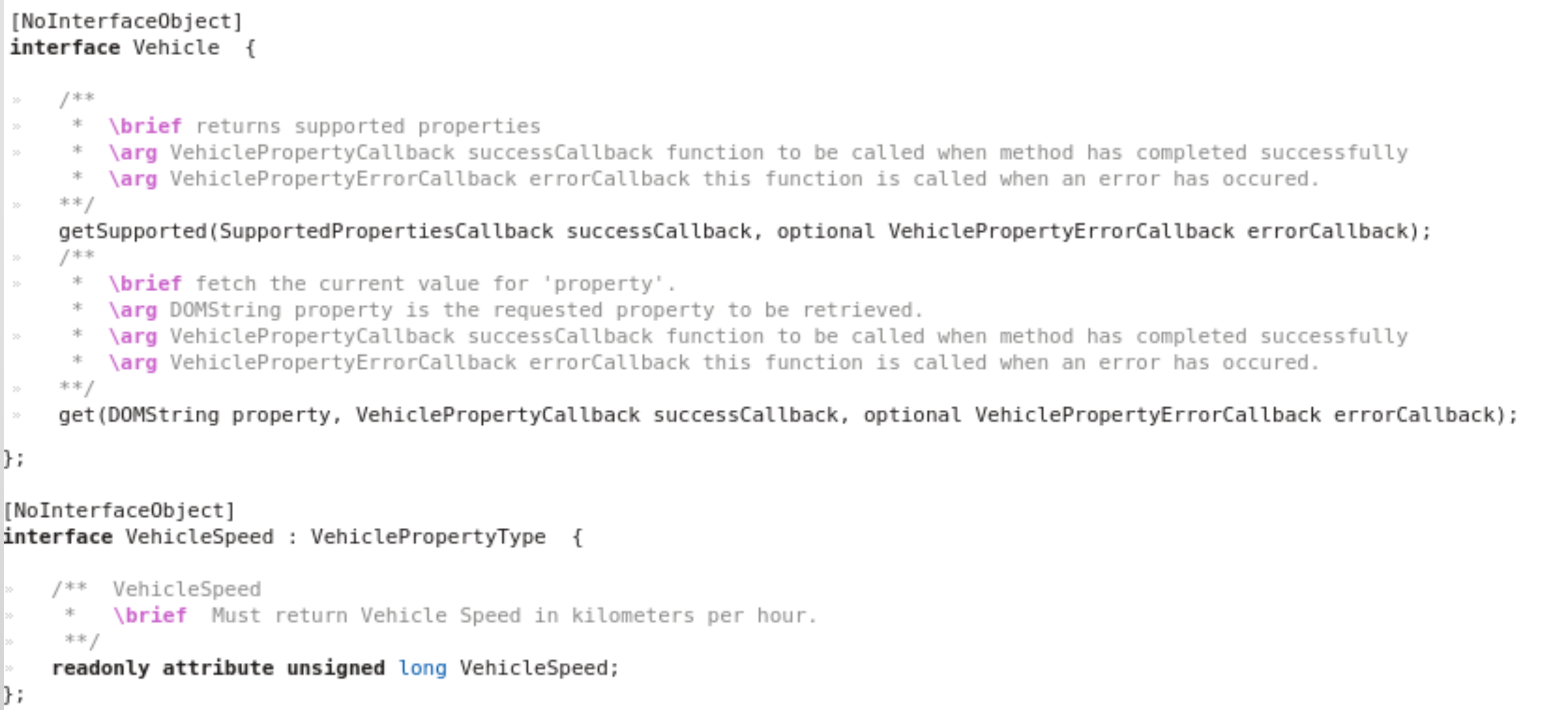
- Purpose:
  - to enable Tizen IVI developers with rich access to vehicle information

Use cases

1. Developer wants to write a tachometer application in html5
2. Developer wants to write an application that controls the HVAC system
3. Developer wants to retrieve the vehicle average economy analytics from last week

1. Developer wants to write a tachometer application in html5

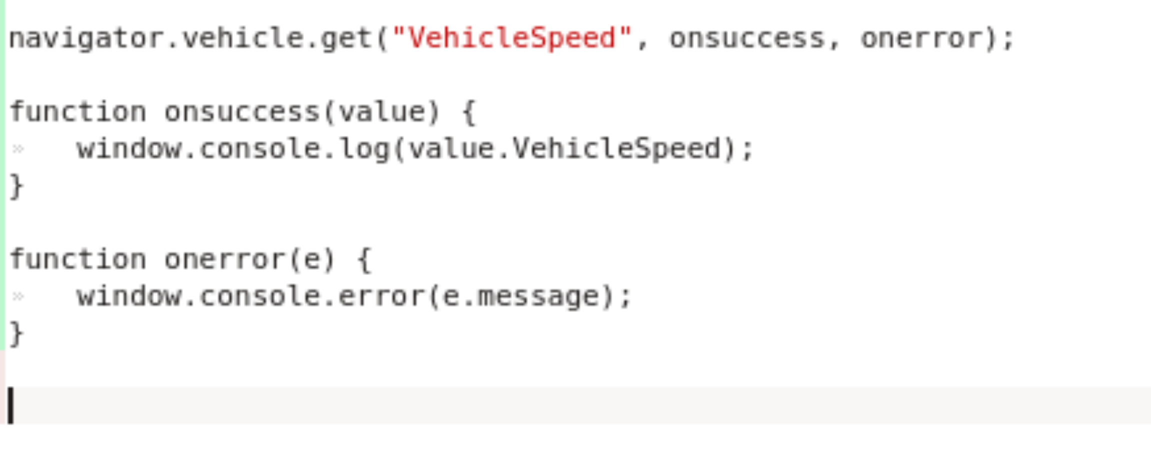
- get ()



```
[NoInterfaceObject]
interface Vehicle  {

    /**
     *  \brief returns supported properties
     *  \arg VehiclePropertyCallback successCallback function to be called when method has completed successfully
     *  \arg VehiclePropertyErrorCallback errorCallback this function is called when an error has occured.
    **/
    getSupported(SupportedPropertiesCallback successCallback, optional VehiclePropertyErrorCallback errorCallback);
    /**
     *  \brief fetch the current value for 'property'.
     *  \arg DOMString property is the requested property to be retrieved.
     *  \arg VehiclePropertyCallback successCallback function to be called when method has completed successfully
     *  \arg VehiclePropertyErrorCallback errorCallback this function is called when an error has occured.
    **/
    get(DOMString property, VehiclePropertyCallback successCallback, optional VehiclePropertyErrorCallback errorCallback);

};

[NoInterfaceObject]
interface VehicleSpeed : VehiclePropertyType  {

   /**  VehicleSpeed
    *   \brief  Must return Vehicle Speed in kilometers per hour.
    **/
   readonly attribute unsigned long VehicleSpeed;
};
```

Example

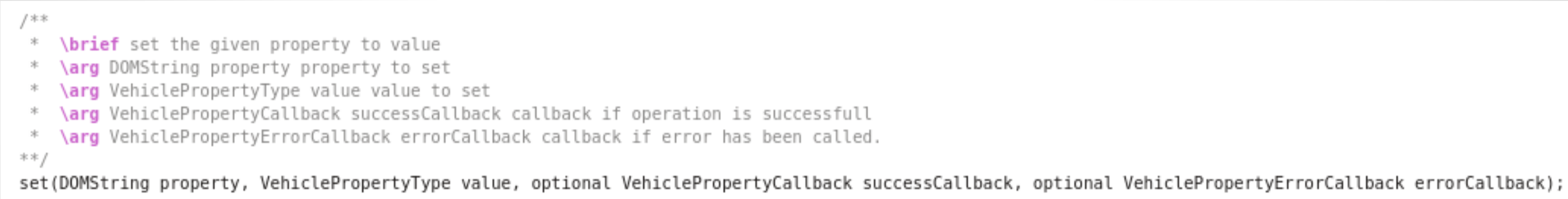
1. Developer wants to write a tachometer application in html5



```
navigator.vehicle.get("VehicleSpeed", onsuccess, onerror);

function onsuccess(value) {
    window.console.log(value.VehicleSpeed);
}

function onerror(e) {
    window.console.error(e.message);
}
```

2. Developer wants to write an application that controls the HVAC system

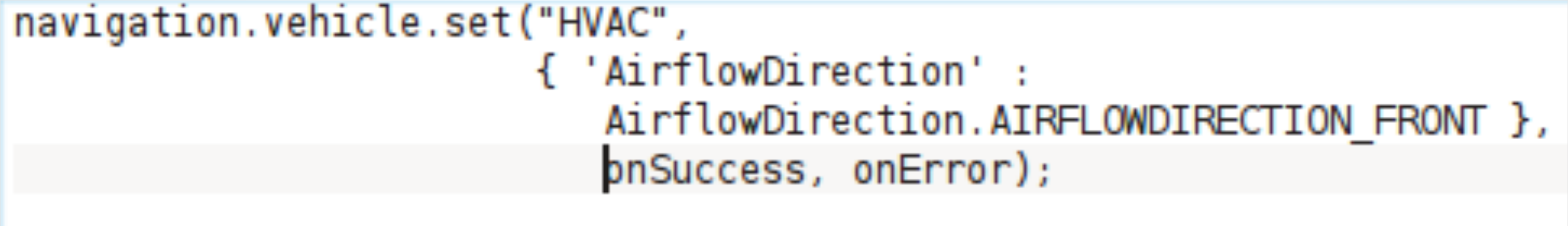
- set ()



```
/**
 *  \brief set the given property to value
 *  \arg DOMString property property to set
 *  \arg VehiclePropertyType value value to set
 *  \arg VehiclePropertyCallback successCallback callback if operation is successfull
 *  \arg VehiclePropertyErrorCallback errorCallback callback if error has been called.
**/
set(DOMString property, VehiclePropertyType value, optional VehiclePropertyCallback successCallback, optional VehiclePropertyErrorCallback errorCallback);
```

Example

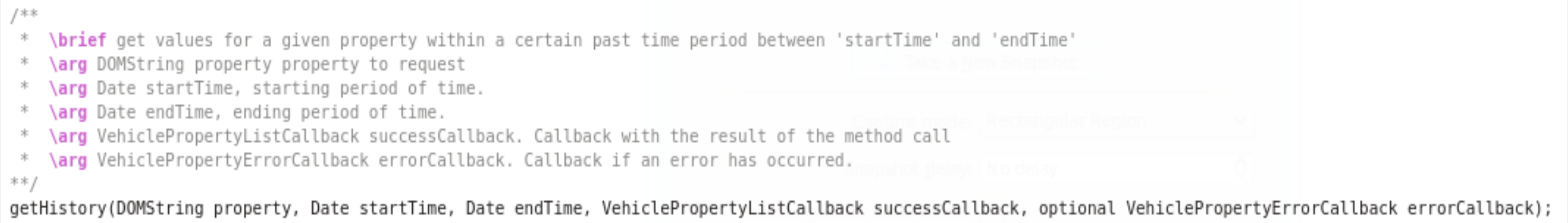
2. Developer wants to write an application that controls the HVAC system



```
navigation.vehicle.set("HVAC",
                       { 'AirflowDirection' :
                          AirflowDirection.AIRFLOWDIRECTION_FRONT },
                          onSuccess, onError);
```

3. Mechanic wants to retrieve the vehicle information from last week.

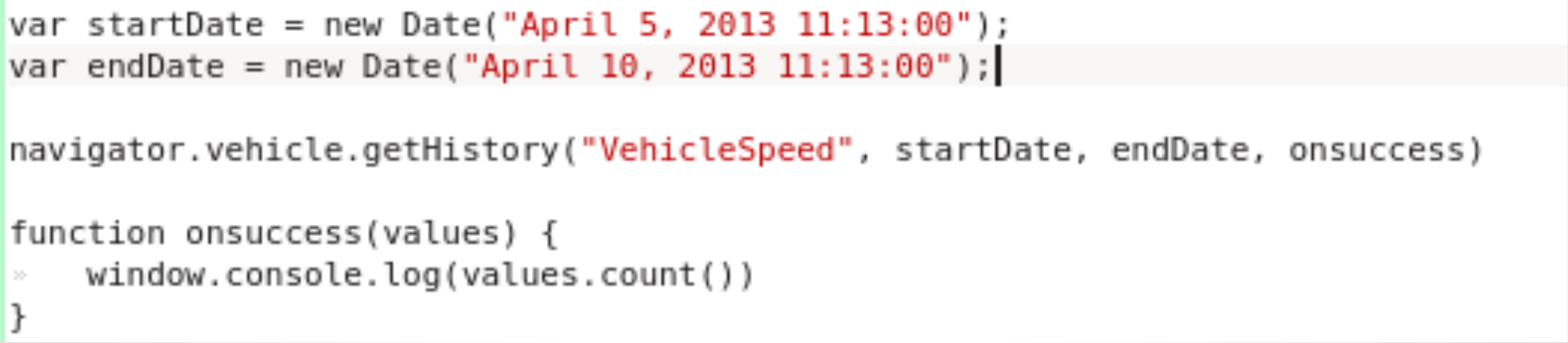
- getHistory ()



```
/**
 *  \brief get values for a given property within a certain past time period between 'startTime' and 'endTime'
 *  \arg DOMString property property to request
 *  \arg Date startTime, starting period of time.
 *  \arg Date endTime, ending period of time.
 *  \arg VehiclePropertyListCallback successCallback. Callback with the result of the method call
 *  \arg VehiclePropertyErrorCallback errorCallback. Callback if an error has occurred.
**/
getHistory(DOMString property, Date startTime, Date endTime, VehiclePropertyListCallback successCallback, optional VehiclePropertyErrorCallback errorCallback);
```

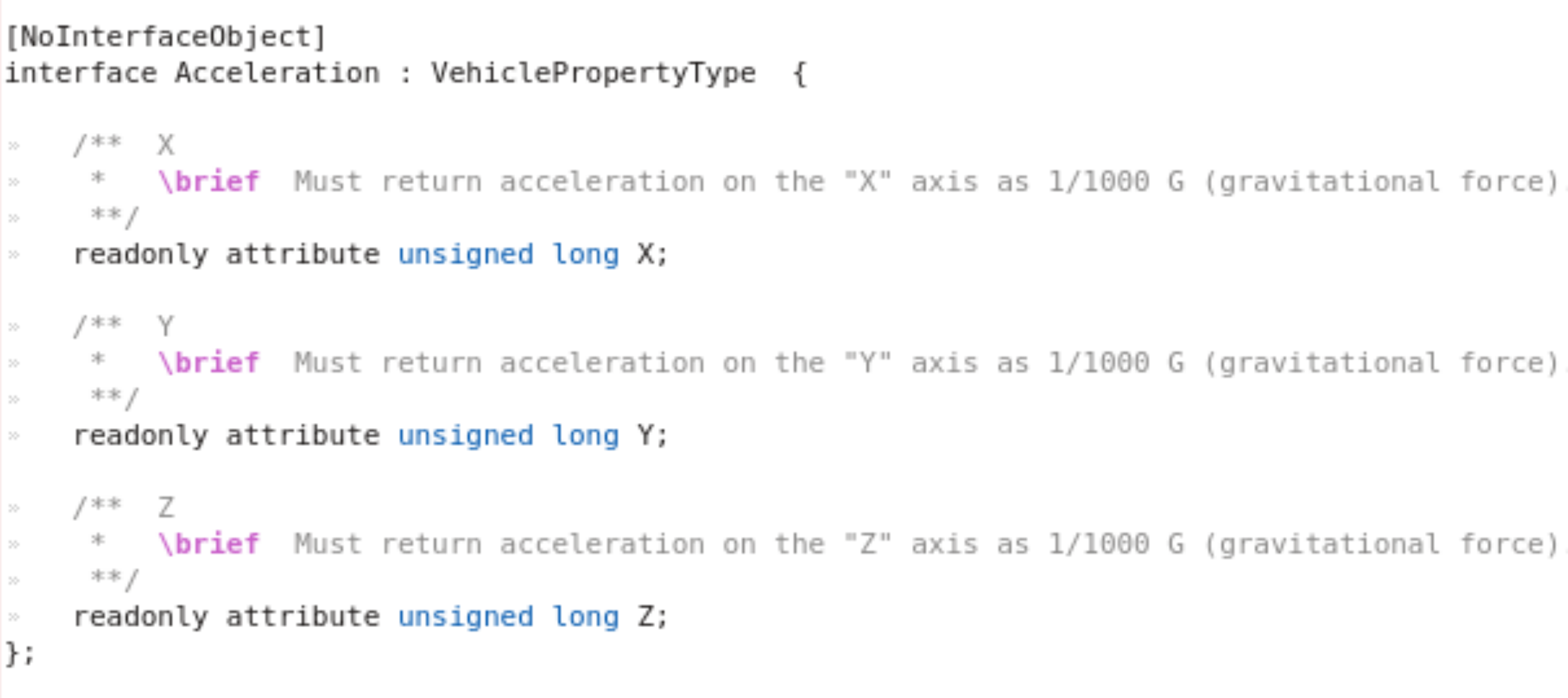
Example

3. Mechanic wants to retrieve the vehicle information from last week.



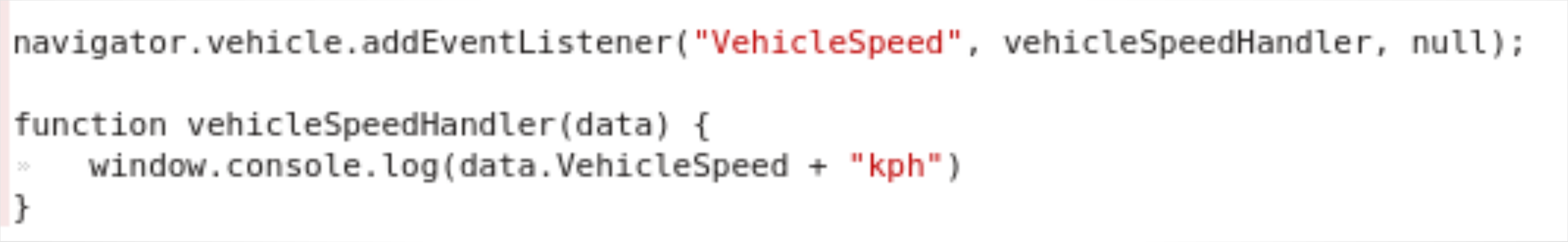
```
var startDate = new Date("April 5, 2013 11:13:00");
var endDate = new Date("April 10, 2013 11:13:00");

navigator.vehicle.getHistory("VehicleSpeed", startDate, endDate, onsuccess)

function onsuccess(values) {
    window.console.log(values.count())
}
```

Data Types



```
[NoInterfaceObject]
interface Acceleration : VehiclePropertyType  {

    /**  X
     *   \brief  Must return acceleration on the "X" axis as 1/1000 G (gravitational force)
     **/
    readonly attribute unsigned long X;

    /**  Y
     *   \brief  Must return acceleration on the "Y" axis as 1/1000 G (gravitational force)
     **/
    readonly attribute unsigned long Y;

    /**  Z
     *   \brief  Must return acceleration on the "Z" axis as 1/1000 G (gravitational force)
     **/
    readonly attribute unsigned long Z;
};
```

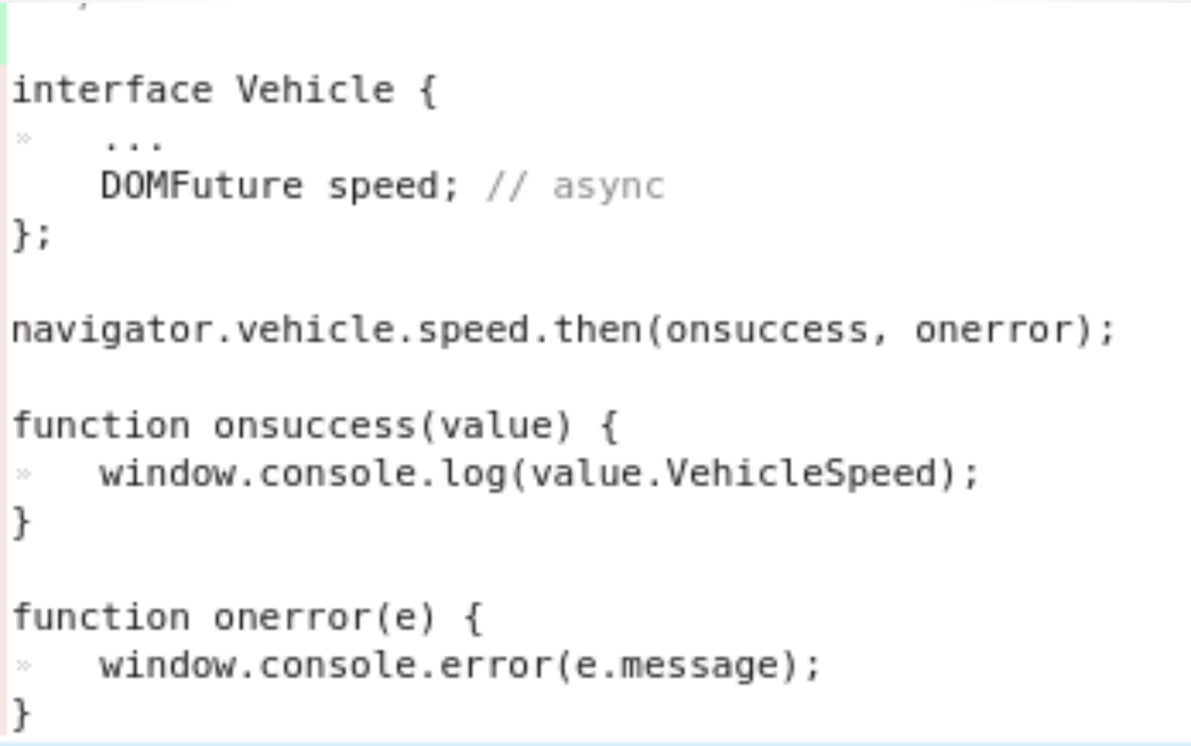
Events



```
navigator.vehicle.addEventListener("VehicleSpeed", vehicleSpeedHandler, null);

function vehicleSpeedHandler(data) {
    window.console.log(data.VehicleSpeed + "kph")
}
```

Future

- Transfer from WAC-style callbacks to W3C style
    - Use DOMFuture



```
interface Vehicle {
    ...
    DOMFuture speed; // async
};

navigator.vehicle.speed.then(onsuccess, onerror);

function onsuccess(value) {
    window.console.log(value.VehicleSpeed);
}

function onerror(e) {
    window.console.error(e.message);
}
```

Resources

Tizen Vehicle API draft specification:

http://otcshare.github.io/automotive-message-broker/docs/vehicle_spec.html

Draft WebIDL:

https://raw.github.com/otcshare/automotive-message-broker/master/docs/amb.idl

Contact:

Kevron Rees: kevron.m.rees@intel.com

Where do we start?

1. Use all data from all specs (worry about OBD-II later)
2. Normalize the data, agree on units, etc
3. ...
4. Profit