***Overall Interface Structure***. It looks like we should clean up the overall interface hierarchy, to avoid many situation where methods do not apply to specific derived interfaces.

Why do we have the *Navigation* interface? Why is *Vehicle* not sufficient?

May it be that an interface as general as *VehicleInterface* may not make much sense? Here are problems with it:

* Many derived interfaces are describing vehicle configuration, not subject to any change – so listeners and setters are not really meaningful
* For really dynamic ones subscription, availability and setting (if any) should be done at individual attribute level
* Zoning applied only to few objects in the spec

So an alternative approach would have simplified *VehicleConfigurationInterface* and *VehicleZonedConfigurationInterface*, with only getters supported. The rest of them could fall as single attributes under *VehicleParameterInterface* or *VehicleZonedParameterInterface*, supporting all necessary methods, including subscriptions (and setters, if we ever find a settable parameter).

***Availability***. Availability depends on both the interface and the operation. So the *available()* method should be split into *isAvailableForGet()* and *isAvailableForSet()*, the former also applying to subscriptions, the latter only supported for Parameter interfaces.

 I’d also make some minor re-wording to achieve [what I see as] superior clarity. So the codes would be “not\_supported\_due\_to\_security\_policy” and “not\_supported\_due\_to\_business\_policy”, with corresponding changes in method names. As a variant of the former I’d also add “available\_to\_system\_apps\_only”, for platforms that support such a notion (usually meaning apps signed with the same key as system code).

***Identification***. As many methods of driver identification may be used, some in current while other in future vehicles, it would make sense to support both *driverName* (instead of DriverID, which is usually refers to a document issued by authorities) and *identificationType* attributes. On the contrary, I’d suggest to withhold internal IDs such as pertaining to key fobs, on privacy/security grounds.

*identificationType* could assume values of “selected” (like in Tesla, assuming no privacy issues associated), “pin”, “keyfob”, “Bluetooth”, “NFC”, “fingerprint” “camera” and “voice”