RFC 0034 Vehicle Interface

Confidential, Draft
<Document Number Here>

56 Pages

Abstract

In a vehicle, several devices own status information, which is of common interest. Status changes have to be observed by clients without the knowledge of details about the status owner or any other services that care about obtaining the status from the different devices. In other cases, a client needs to change the status of a vehicle object. Security aspect have to be considered since a status may have a safety critical character.

The topology of a vehicle is not known at the design time of a status client. Further a client needs to obtain properties of a status and also it's owner.

0 Document Information

0.1 Table of Contents

All Page Within This Box

0.2 Status

This document specifies **...** for the Open Services Gateway Initiative, and requests discussion and suggestions for improvements. Distribution of this document is unlimited within OSGi.

0.3 Acknowledgement

0.4 Terminology and Document Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interpreted as described in 0.

`Source code is shown in this typeface.`

All Page Within This Box

0.5 Revision History

The last named individual in this history is currently responsible for this document.

Revision	Date	Comments
Initial	Sep 12 2001	Christof Menzenbach, Siemens VDO christof.menzenbach@de3.vdogrp.de
First Revision	Oct 01 2001	Christof Menzenbach, Siemens VDO christof.menzenbach@de3.vdogrp.de; Vehicle status list added.

1 Introduction

A vehicle has a lot of components which own status information. Those statuses are typically accessible via in vehicle networks like CAN or MOST. Examples for a status are "mirror horizontal tilt", “vendor”, “production date” or "external Lights on". Status information has to be obtained or controlled by clients. The client of a status is not interested in the services that provide a status, but has usually a high interest in the object that owns a status. The owner of a status is typically a "real-world" in vehicle object and can not be mapped to bundles or services.

Properties of a status owner are e.g. "position in Vehicle".

2 Motivation and Rationale

Real world components of a vehicle are very likely vendor specific. Therefore it is very difficult to model those components and their status information as they appear for a specific vehicle. Also the vehicle configuration differs extremely between vehicles of different types and vendors. Anyhow it is required to get access to vehicle objects in order to build value adding services on top.
Nevertheless it is possible to agree on a set of status objects which can be shared upon several vehicles and vehicle vendors. A client should have capabilities to obtain instances of components that will own a status of a defined type. Further it is required that the client of a status is able to identify the position of a status owning component.
During a discussion in the VEG meeting (29/30 Aug. 2001) it became clear that the requirement are not limited to the vehicle domain.

3 Technical Discussion

3.1 Boundaries

Figure 1 gives an overview of vehicle use cases and the subsystems, which are responsible to realize those use cases. It becomes obvious, that most of the use cases will be realized by the wiring subsystem. This includes the wiring API as well as supporting and administrative services. Nevertheless, it seems to be wise to encapsulate the features offered by wiring. This document is focused on the use case introspect
status, which includes:

- availability of status instances with a given type
- control supported
- position of the status owner
- textural description of status and status owner
- upper/lower limit of status
- description of states

Set- and get Status are also realized by utilizing the wiring API. The Vehicle API defines a Status object in order to exchange the status between the actors.
The status owner is not shown as actor since it does not directly communicate with the vehicle respectively the wiring subsystem. The status owner is an external logical component, which has no impact on the modeling of the aforementioned subsystems. Any references to the status owner are either modeled in the status namespace or in the location assigned to the status.
Open issues:

- How to realize the "notify set status unsupported" use case is not clear yet.
- Notify permission denied is not jet supported by Vehicle API.

All Page Within This Box

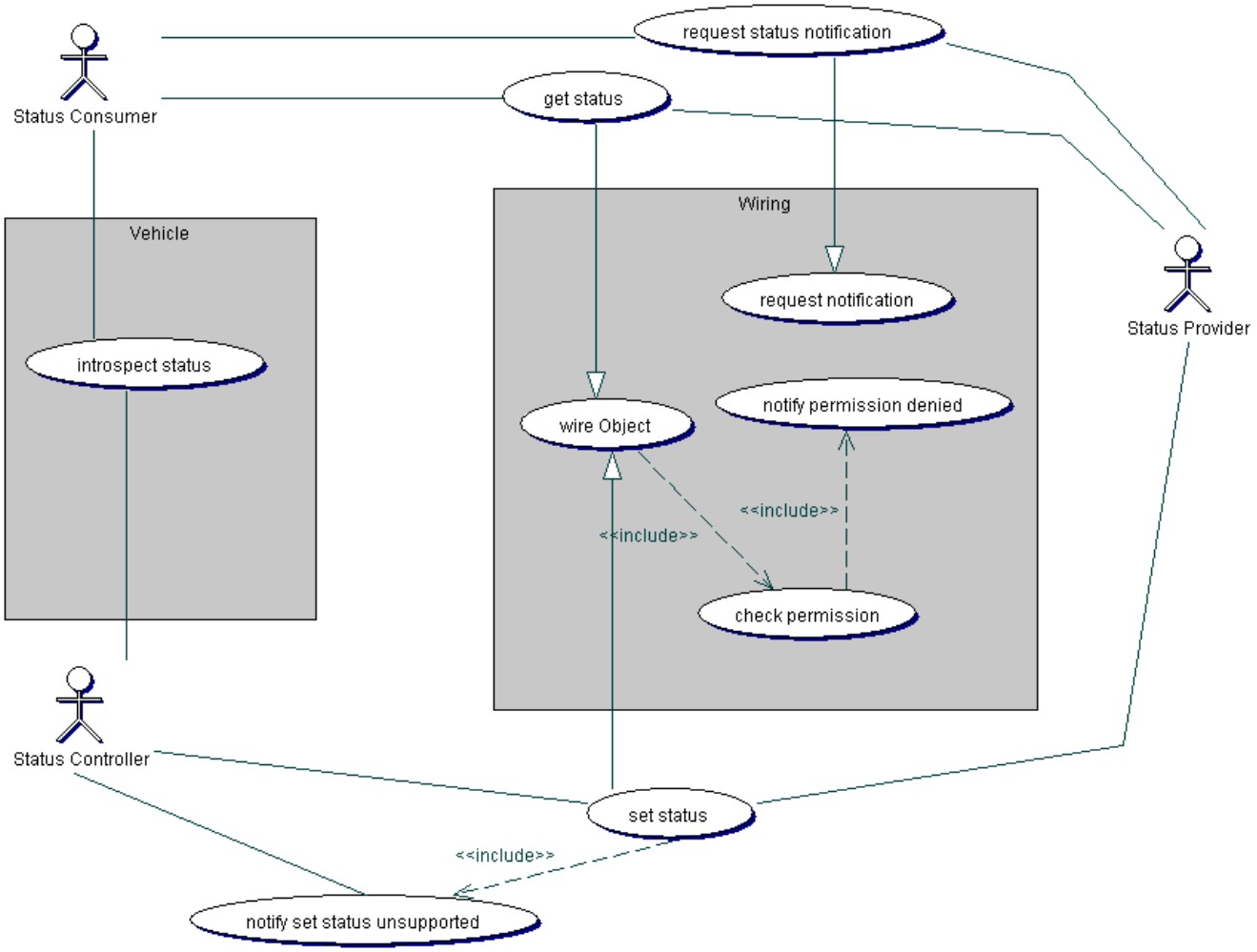


Figure 1: Subsystem Boundaries

3.2 Class Diagram

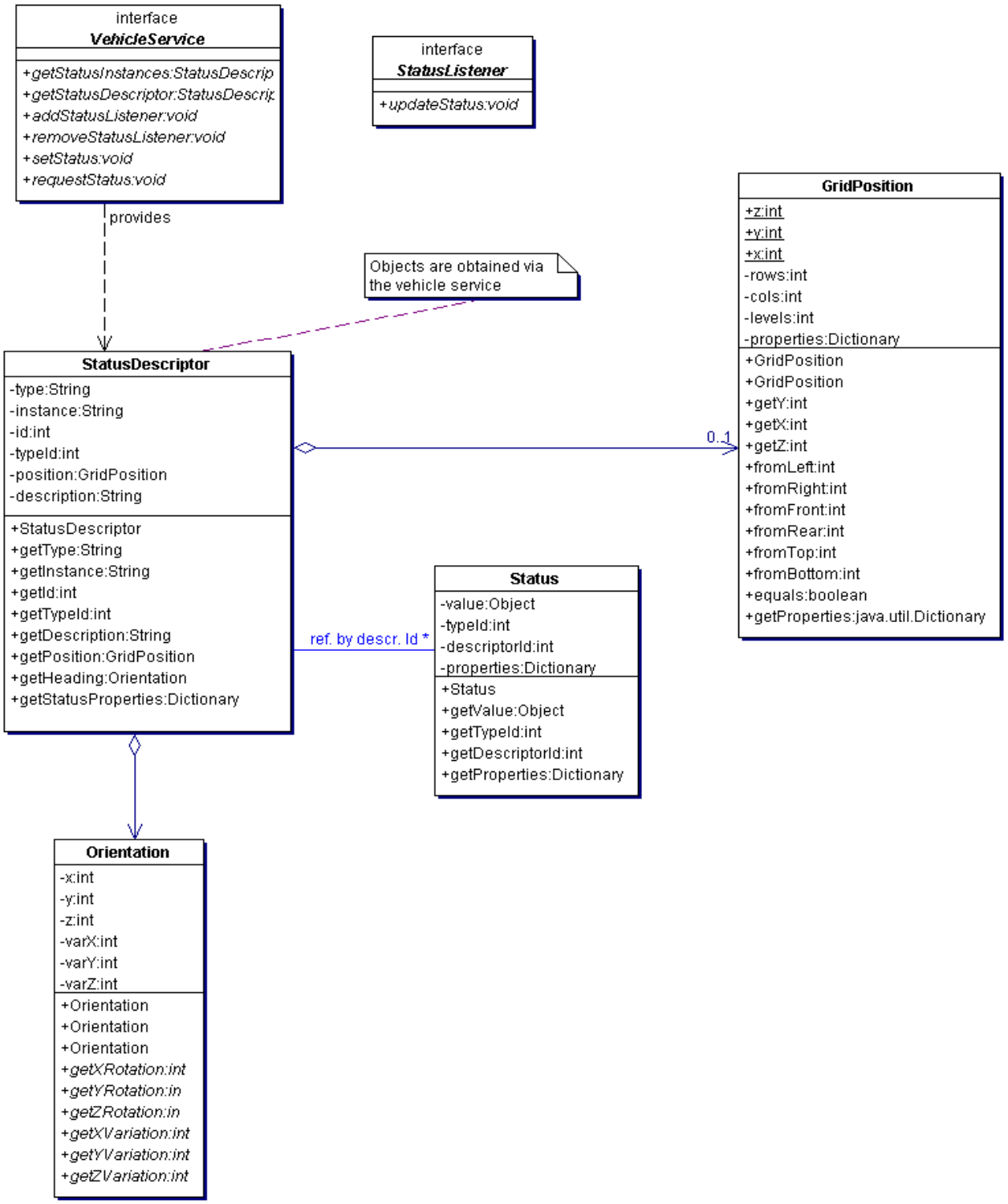
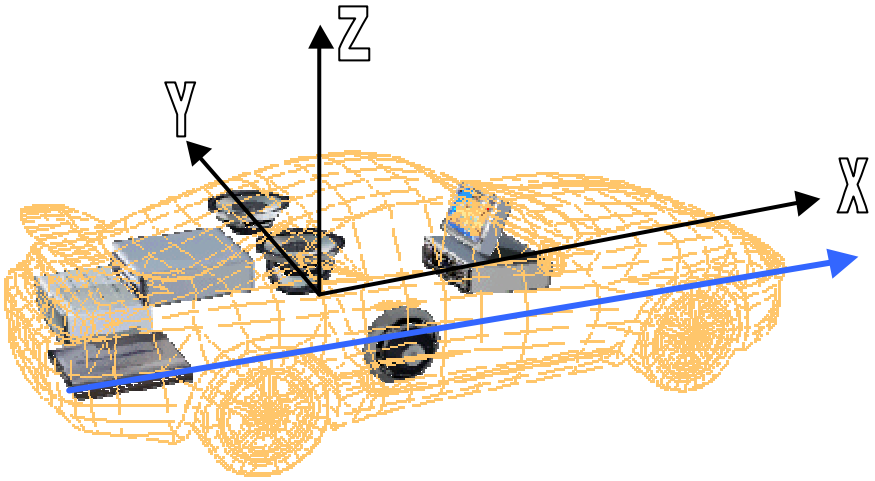


Figure 2: Vehicle API Class Diagram

The VehicleService is used to obtain vehicle components, which own status of a known type. The type is identified by a name. A status can be either static like vendor, date of production or dynamic like door lock and fuel level. A status can be either read/write or read only.
A client which wants to obtain a status value or to control the status of a vehicle component will query StatusDescriptions. The StatusDescription offers information about the status and also the owning component. A query for “vehicle.mirror.horizontalTilt” may return 3 instances of StatusDescriptions. Each representing an individual mirror in conjunction with the horizontal tilt status. Examples for the 3 instances are leftOutboard, inside and rightOutboard. Together with the type one status of the mirrors can be identified as “vehicle.mirror.horizontalTilt.leftOutboard”.
The client requires further information about the position of a status owning vehicle component. The position is described by the grid class. The grid describes a cell in a rectangle or cuboid in order to specify the location. The grid is not supposed to put the entire vehicle in it. The grid is useful to get relative positions in limited context. The client requires further information about the position of a status owning vehicle component. The position is described by the GridPosition class. The grid position describes a cell in a rectangle or cuboid in order to specify the location. The grid is not supposed to put the entire vehicle in it. The grid is useful to get relative positions in limited context. Seats are probably located in a 3 * 2 grid. The cylinder of the engine are probably located in a 2*3, 1*4 or 2*2 grid. Both grids have no relation and do not describe the absolute position within the vehicle. The GridPosition has methods to get the distance of a component from the left, right, front, etc. Other properties like vehicle area, inside or outside are kept in the properties attribute of the Grid.
The heading of the status owning component is described as the rotation around axis in a cartesian system. The following picture shows how the system is located in the vehicle space.



X represents the role Y the yaw and Z the pitch of the each component in relation to the vehicle (blue line).

The Status class contains an id of the descriptor in order to allow an efficient filtering if the client expects more than one status via the same listener method. The id is also required to reference the descriptor that holds important descriptions. The value of the status is usually a Measurement or a State. Both are defined by the measurement API. The properties attribute contains status properties like warning level, notification reason, etc. The vehicle API provides a StatusListener in order to receive status updates. The status can be changed by setStatus. An update of a status can be requested by requestStatus. The status will be notified asynchronously as it is done in the case of normal update notification.

Since no class is vehicle dependant, the package and the VehicleService interface should be renamed.

All Page Within This Box

3.3 Javadoc

org.osgi.service.vehicle

Interface VehicleService

public interface **VehicleService**

Offers configuration of components, which own vehicle related status information. The service provides descriptions of statuses and the vehicle components that own a status of interest. The VehicleService gives a view on the availability of status instances with a given type. An instance of a status belongs to a real-world component inside the vehicle.

Method Summary

void	**addStatusListener**(StatusListener l, org.osgi.service.vehicle.Dictionary properties)
StatusDescriptor	**getStatusDescriptor**(int descriptorId) Returns the StatusDescriptor, which matches the specified descriptorId.
StatusDescriptor[]	**getStatusInstances**(java.lang.String type) Provide StatusDescriptions, which will give further information about the status and the vehicle component that has a status of the given type.
void	**removeStatusListener**(StatusListener l)
void	**requestStatus**(StatusDescriptor dscr) Request the update of the specified status.
void	**setStatus**(Status status)

Method Detail

getStatusInstances

public StatusDescriptor[] **getStatusInstances**(java.lang.String type)

Provide StatusDescriptions, which will give further information about the status and the vehicle component that has a status of the given type. The StatusDescription includes ID's in order to query statuses and to filter incoming statuses.

Parameters:

type - the type of the status that should be obtained of controlled

Returns:

an array of status description that matches the status type.

getStatusDescriptor

public StatusDescriptor **getStatusDescriptor**(int descriptorId)

Returns the StatusDescriptor, which matches the specified descriptorId.

All Page Within This Box

Parameters:

`descriptiorId` - the id of the requested StatusDescriptor

Returns:

the status descriptor, which matches with the specified id

---

addStatusListener

```
public void addStatusListener(StatusListener l,
                              org.osgi.service.vehicle.Dictionary properties)
```

---

removeStatusListener

```
public void removeStatusListener(StatusListener l)
```

---

setStatus

```
public void setStatus(Status status)
```

---

requestStatus

```
public void requestStatus(StatusDescriptor dscr)
```

Request the update of the specified status. The status is delivered asynchronously by the StatusListener.

All Page Within This Box

org.osgi.service.vehicle

Class StatusDescriptor

```
java.lang.Object
  |
  +--org.osgi.service.vehicle.StatusDescriptor
```

public class **StatusDescriptor**
extends java.lang.Object
The StatusDescriptor provides further information about a Status and a real world component that owns a status.

Constructor Summary

StatusDescriptor(java.lang.String type, java.lang.String instance, int id, int typeId, GridPosition pos, Orientation heading, java.lang.String description, java.util.Dictionary properties) Creates a new StatusDescriptor instance.	

Method Summary

java.lang.String	**GetDescription**() Return a textural description of the status and the component that owns the status.
Orientation	**getHeading**() Returns the heading of the status owner.
int	**getId**() Returns the Id of this descriptor.
java.lang.String	**getInstance**() Returns the name of the status instance in respect to the status namespace.
GridPosition	**getPosition**() Returns the position of the status owner.
java.util.Dictionary	**GetStatusProperties**() Returns a Dictionary with properties of the status.
java.lang.String	**getType**() Returns the name of the status type in respect to the status namespace.
int	**getTypeId**() Returns the Id of the status type.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

All Page Within This Box

Constructor Detail

StatusDescriptor

```
public StatusDescriptor(java.lang.String type,
                        java.lang.String instance,
                        int id,
                        int typeId,
                        GridPosition pos,
                        Orientation heading,
                        java.lang.String description,
                        java.util.Dictionary properties)
```

Creates a new StatusDescriptor instance.

Parameters:

`type` - typename of the status, which is described by this StatusDescriptor
`instance` - instancename of the status, which is described by this StatusDescriptor
`id` - unique id of this StstusDescriptor
`typeId` - id of the status type
`position` - reference to a position, which describes the position of the status owner
`heading` - heading of the status owner
`description` - textural description of the status and status owner
`properties` - status properties

Method Detail

getType

```
public java.lang.String getType()
```

Returns the name of the status type in respect to the status namespace.

---

getInstance

```
public java.lang.String getInstance()
```

Returns the name of the status instance in respect to the status namespace.

All Page Within This Box

getId

`public int **getId**()`

Returns the Id of this descriptor.

---

getTypeId

`public int **getTypeId**()`

Returns the Id of the status type.

---

getDescription

`public java.lang.String **getDescription**()`

Return a textural description of the status and the component that owns the status.

---

getPosition

`public GridPosition **getPosition**()`

Returns the position of the status owner.

---

getHeading

`public Orientation **getHeading**()`

Returns the heading of the status owner.

Returns:

the horizontal orientation of the status owner in degree.

---

getStatusProperties

`public java.util.Dictionary **getStatusProperties**()`

Returns a Dictionary with properties of the status. Status properties can be: - warning level - value limits - state descriptions

All Page Within This Box

org.osgi.service.vehicle

Class Status

```
java.lang.Object
  |
  +--org.osgi.service.vehicle.Status
```

---

public class **Status**
extends java.lang.Object
A status of a component to be obtained or controlled.

Constructor Summary

Status(java.lang.Object value, int typeId, int descriptorId, java.util.Dictionary properties) Creates a status instance.	

Method Summary

int	**getDescriptorId**() Returns the unique id of the status descriptor.
java.util.Dictionary	**getProperties**() Returns a Dictionary with properties of the status.
int	**getTypeId**() Returns the id of the status type.
java.lang.Object	**getValue**() Returns the value of the status.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Status

```
public Status(java.lang.Object value,
              int typeId,
              int descriptorId,
              java.util.Dictionary properties)
```

Creates a status instance.

All Page Within This Box

Method Detail

getValue

```
public java.lang.Object getValue()
```

Returns the value of the status. The type of the value object depends on the status type. Values are usually expressed by the types Measurement and State as defined in org.osgi.util.measurement.

Returns:

the value of this status as an Object.

---

getTypeId

```
public int getTypeId()
```

Returns the id of the status type. The typeId is useful for filtering.

---

getDescriptorId

```
public int getDescriptorId()
```

Returns the unique id of the status descriptor. The descriptorId is useful for filtering and can be used to get further information of the status.

---

getProperties

```
public java.util.Dictionary getProperties()
```

Returns a Dictionary with properties of the status. Status properties are: - the type name of the status (e.g. seat position, door lock) - the instance name of the object where the status belongs to (status owner). - warning level - notification reason

---

org.osgi.service.vehicle

Class GridPosition

```
java.lang.Object
  |
  +--org.osgi.service.vehicle.GridPosition
```

---

public class **GridPosition**
extends java.lang.Object

The Grid class describes the layout of components in a quarder, without defining the exact position of the component within a grid cell. Position in a 3D grid. - x from left to right. Left = 0. - y from front to rear. Front = 0. - z from bottom to top. Bottom = 0.

Field Summary

static int	**x** The column of this position. x is counted from left to right.
static int	**y** The row of this position. y is counted from front to rear.
static int	**z** The level of this Position. z is counted from bottom to top.

Constructor Summary

GridPosition(int x, int y, int cols, int rows, java.util.Dictionary properties) Create a 2D position.	
GridPosition(int x, int y, int z, int cols, int rows, int levels, java.util.Dictionary properties) Create a 3D position.	

Method Summary

boolean	**equals**(GridPosition position) Compares two positions.
int	**fromBottom**() Returns the level of the position from the bottom.
int	**fromFront**() Returns the row of the position from the front.
int	**fromLeft**() Returns the column of the position from the left.
int	**fromRear**() Returns the row of the position from the rear.
int	**fromRight**() Returns the column of the position from the right.

All Page Within This Box

int	**fromTop**() Returns the level of the position from the top.
java.util.Dictionary	**getProperties**() Return additional position properties.
int	**getX**() Return the position in the left/right axis.
int	**getY**() Return the position in the front/rear axis.
int	**getZ**() Return the position in the bottom/top axis.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

z

```
public static final int z
```

The level of this Position. z is counted from bottom to top.

---

y

```
public static final int y
```

The row of this position. y is counted from front to rear.

---

x

```
public static final int x
```

The column of this position. x is counted from left to right.

Constructor Detail

GridPosition

```
public GridPosition(int x,
                    int y,
                    int cols,
                    int rows,
```

```
                    java.util.Dictionary properties)
```

Create a 2D position.

Parameters:

`x` - column of this Position
`y` - row of this Position
`cols` - number of columns
`rows` - number of rows
`properties` - additional position properties

GridPosition

```
public GridPosition(int x,
                    int y,
                    int z,
                    int cols,
                    int rows,
                    int levels,
                    java.util.Dictionary properties)
```

Create a 3D position.

Parameters:

`x` - column of this Position
`y` - row of this Position
`z` - level of this Position
`cols` - number of columns
`rows` - number of rows
`levels` - number of levels
`properties` - additional position properties

Method Detail

getY

```
public int getY()
```

Return the position in the front/rear axis. Front = 0.

Returns:

the row of this position; front = 0

---

getX

```
public int getX()
```

Return the position in the left/right axis. Left = 0.

Returns:

the column of this position; left = 0

---

getZ

```
public int getZ()
```

Return the position in the bottom/top axis. Bottom = 0.

Returns:

the level of this position; bottom = 0

---

fromLeft

```
public int fromLeft()
```

Returns the column of the position from the left. The most left position in the grid returns 0.

Returns:

the column of this Position from the left

---

fromRight

```
public int fromRight()
```

Returns the column of the position from the right. The most right position in the grid returns 0.

Returns:

the column of this Position from the right

---

fromFront

```
public int fromFront()
```

Returns the row of the position from the front. The first row in the front returns 0.

Returns:

the row of this Position from the front

---

fromRear

```
public int fromRear()
```

Returns the row of the position from the rear. The last row in the rear returns 0.

Returns:

the row of this Position from the rear

---

fromTop

```
public int fromTop()
```

Returns the level of the position from the top. The highest level returns 0.
Returns:
the level of this Position from the top

---

fromBottom

```
public int fromBottom()
```

Returns the level of the position from the bottom. The lowest level returns 0.
Returns:
the level of this Position from the bottom

---

equals

```
public boolean equals(GridPosition position)
```

Compares two positions.
Returns:
true if this position is equal to the specified position.

---

getProperties

```
public java.util.Dictionary getProperties()
```

Return additional position properties.

---

org.osgi.service.vehicle

Class Orientation

```
java.lang.Object
  |
  +--org.osgi.service.vehicle.Orientation
```

public class **Orientation**
extends java.lang.Object
The orientation of a real world object.

Constructor Summary

Orientation() Creates a new (0,0,0) Orientation instance.	
Orientation(int xRotation, int yRotation, int zRotation) Creates a new Orientation instance with the specified rotation.	
Orientation(int xRotation, int yRotation, int zRotation, int xVariation, int yVariation, int zVariation) Creates a new Orientation instance with the specified rotation and the specified variation.	

Method Summary

int	**getXRotation**() Rotation around the X axis.
int	**getXVariation**() x rotation variation in degrees. no variation = 0 full variable = 360 The variation is symmetric to the orientation.
int	**getYRotation**() Rotation around the Y axis.
int	**getYVariation**() y rotation variation in degrees. no variation = 0 full variable = 360 The variation is symmetric to the orientation.
int	**getZRotation**() Rotation around the Z axis.
int	**getZVariation**() z rotation variation in degrees. no variation = 0 full variable = 360 The variation is symmetric to the orientation.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

All Page Within This Box

Constructor Detail

Orientation

```
public Orientation()
```

Creates a new (0,0,0) Orientation instance.

Orientation

```
public Orientation(int xRotation,
                   int yRotation,
                   int zRotation)
```

Creates a new Orientation instance with the specified rotation. The Variation is (0, 0, 0).

Orientation

```
public Orientation(int xRotation,
                   int yRotation,
                   int zRotation,
                   int xVariation,
                   int yVariation,
                   int zVariation)
```

Creates a new Orientation instance with the specified rotation and the specified variation.

Method Detail

getXRotation

```
public int getXRotation()
```

Rotation around the X axis. Value in degrees.

Returns:

the orientation in degree

---

getYRotation

```
public int getYRotation()
```

All Page Within This Box

Rotation around the Y axis. Value in degrees.

Returns:

the orientation in degree

---

getZRotation

```
public int getZRotation()
```

Rotation around the Z axis. Value in degrees.

Returns:

the orientation in degree

---

getXVariation

```
public int getXVariation()
```

x rotation variation in degrees. no variation = 0 full variable = 360 The variation is symetric to the orientation.

Returns:

the variation of the x rotation in degree

---

getYVariation

```
public int getYVariation()
```

y rotation variation in degrees. no variation = 0 full variable = 360 The variation is symetric to the orientation.

Returns:

the variation of the y rotation in degree

---

getZVariation

```
public int getZVariation()
```

z rotation variation in degrees. no variation = 0 full variable = 360 The variation is symetric to the ord of Document**eturns:**

the variation of the z rotation in degree

---

org.osgi.service.vehicle

Interface StatusListener

public interface **StatusListener**

Method Summary

void	**updateStatus**(Status status) Invoced when a status has changed.

Method Detail

updateStatus

```
public void updateStatus(Status status)
```

Invoced when a status has changed.

3.4 Vehicle status types

3.4.1 vehicle.chassisNumber

The vehicle chassis number.

Type	Multiple Instances	Control supported
String	No	No

Properties	
WarningLevel	No

3.4.2 vehicle.manufacturer

Vehicle manufacturer string.

Type	Multiple Instances	Control supported
String	No	No

Properties	
WarningLevel	No

3.4.3 vehicle.model

Internal model name.

Type	Multiple Instances	Control supported
String	No	No

Properties	
WarningLevel	No

3.4.4 vehicle.constructionYear

Year and month of construction.

Type	Multiple Instances	Control supported
String	No	No

Properties	
WarningLevel	No

3.4.5 vehicle.productionLocation

Location of construction.

Type	Multiple Instances	Control supported
String	No	No

Properties	
WarningLevel	No

3.4.6 vehicle.descriptionData

Vehicle description

Type	Multiple Instances	Control supported
String	No	No

Properties	
WarningLevel	No

3.4.7 vehicle.ignition.key

State of the ignition key.

Type	Multiple Instances	Control supported
State	No	No

Properties	
WarningLevel	No

Table 1: Ignition key states

State	Value
Key absent	0x00
Key present	0x01

All Page Within This Box

3.4.8 vehicle.ignition.electricity

State of the ignition electricity.

Type	Multiple Instances	Control supported
State	No	No

Properties	
WarningLevel	No

Table 2: Ignition electricity states

State	Value
Electricity off	0x00
Electricity on	0x01

3.4.9 vehicle.ignition.lock

Ignition lock state

Type	Multiple Instances	Control supported
State	No	Yes

Properties	
WarningLevel	No

Table 3: Lock states

All Page Within This Box

State	Value
Unlocked	0x00
Locked by electronic burglar protection	0x01
Locked by removed ignition key	0x02

3.4.10 vehicle.ignition.engine

State of the ignition.

Type	Multiple Instances	Control supported
State	No	No

Properties	
WarningLevel	No

Table 4: Ignition engine states

State	Value
Engine off	0x00
Engine on	0x01

3.4.11 vehicle.ignition.engineOffTime

Ignition off time.

Type	Multiple Instances	Control supported
String	No	No

Properties	
WarningLevel	No

3.4.12 vehicle.ignition.start

State of the ignition.

Type	Multiple Instances	Control supported
State	No	No

Properties	
WarningLevel	No

Table 5: Ignition starter states

State	Value

All Page Within This Box

Starter inactive	0x00
Starter active	0x01

3.4.13 vehicle.modelCategory

Category of the model.

Type	Multiple Instances	Control supported
String	No	No

Properties	
WarningLevel	No

3.4.14 vehicle.engineDescription

Power, number of cylinders.

Type	Multiple Instances	Control supported
String	No	No

Properties	
WarningLevel	No

3.4.15 vehicle.door.lock

Lock status for each door.

Type	Multiple Instances	Control supported
State	Yes	Yes

Properties	
WarningLevel	No

Table 6: Door lock states

State	Value
Door unlocked	0x00
Door locked	0x01

3.4.16 vehicle.door.open

Opening state of each door.

Type	Multiple Instances	Control supported
State	Yes	No

Properties	
WarningLevel	Yes

Table 7: Door open states

State	Value
Door closed	0x00
Door opened	0x01
Door ajar	0x02

3.4.17 vehicle.window

Opening state of each window.

Type	Multiple Instances	Control supported
State	Yes	Yes

Properties	
WarningLevel	No

Table 8: Window states

State	Value
Window closed	0
...	
Window full open	100

All Page Within This Box

3.4.18 vehicle.sunRoof

Opening state of the sun roof.

Type	Multiple Instances	Control supported
State	No	Yes

Properties	
WarningLevel	No

Table 9: Sunroof states

State	Value
Sunroof closed	0
...	
Sunroof full open	100
Tilted	0x100

3.4.19 vehicle.mirror.horizontalTilt

Horizontal movement of all mirrors.

Type	Multiple Instances	Control supported
State	Yes	Yes

Properties	
WarningLevel	No

Table 10: Horizontal tilt states

State	Value
Leftmost	0
...	
Rightmost	100

All Page Within This Box

3.4.20 vehicle.mirror.verticalTilt

Vertical movement of all mirrors.

Type	Multiple Instances	Control supported
State	Yes	Yes

Properties	
WarningLevel	No

Table 11: Horizontal tilt states

State	Value
Uppermost	0
...	
Lowermost	100

3.4.21 vehicle.seat.slide

Slide movement of all seats (forward, backward).

Type	Multiple Instances	Control supported
Measurement	Yes	Yes

Properties	
WarningLevel	No

Table 12: Slide position states

State	Value
Foremost	0
...	
Rearmost	100

All Page Within This Box

3.4.22 vehicle.seat.lift

Up/down movement of all seats.

Type	**Multiple Instances**	Control supported
Measurement	Yes	Yes

Properties	
WarningLevel	No

Table 13: Lift position states

State	Value
Uppermost	0
...	
Lowermost	100

3.4.23 vehicle.seat.tilt

Type	**Multiple Instances**	Control supported
Measurement	Yes	Yes

Properties	
WarningLevel	No

Table 14: Tilt position states

State	Value
Foremost	0
...	
Rearmost	100

All Page Within This Box

3.4.24 vehicle.seat.back

Type	Multiple Instances	Control supported
Measurement	Yes	Yes

Properties	
WarningLevel	No

Table 15: Back position states

State	Value
Uppermost	0
...	
Lowermost	100

3.4.25 vehicle.seat.headRestraint

Type	Multiple Instances	Control supported
Measurement	Yes	Yes

Properties	
WarningLevel	No

Table 16: Head restraint position states

State	Value
Uppermost	0
...	
Lowermost	100

3.4.26 vehicle.seat.bottomCushion

Type	Multiple Instances	Control supported
Measurement	Yes	Yes

Properties	
WarningLevel	No

Table 17: Cushion states

State	Value
Soft	0
...	
Hard	100

3.4.27 vehicle.seat.lowerBackCushion

Type	Multiple Instances	Control supported
Measurement	Yes	Yes

Properties	
WarningLevel	No

States see Table 17: Cushion states.

3.4.28 vehicle.seat.upperBackCushion

Type	Multiple Instances	Control supported
Measurement	Yes	Yes

Properties	
WarningLevel	No

States see Table 17: Cushion states.

3.4.29 vehicle.seat.sideCushion

Type	Multiple Instances	Control supported
Measurement	Yes	Yes

Properties	
WarningLevel	No

States see Table 17: Cushion states.

3.4.30 vehicle.antenna

Status of the antenna (inside, outside).

Type	Multiple Instances	Control supported
State	No	No

Properties	
WarningLevel	No

Table 18: Antenna states

State	Value
Inside	0x00
Outside	0x01

All Page Within This Box

3.4.31 vehicle.tire.pressure

Pressure of all tires.

Type	Multiple Instances	Control supported
Measurement	Yes	No

Properties	
WarningLevel	Yes

Measurement unit is bar (pressure unit `bar`).

3.4.32 vehicle.seatbelt

Status of all seatbelts (fastened, unfastened).

Type	Multiple Instances	Control supported
State	Yes	No

Properties	
WarningLevel	Yes

Table 19: Seatbelt state

State	Value
unfastened	0x00
fastened	0x01

3.4.33 vehicle.horn

Status of the horn.

Type	Multiple Instances	Control supported
State	No	Yes

Properties	
WarningLevel	No

Table 20: Horn states

State	Value
Inactive	0x00
Active	0x01

All Page Within This Box

3.4.34 vehicle.wiper.mode

Status of all wipers.

Type	Multiple Instances	Control supported
State	Yes	No

Properties	
WarningLevel	No

Table 21: Wiper modes

State	Value
Off	0
Manuel	1
Normal	2
Fast	3
High	4

3.4.35 vehicle.wiper.interval

Wiper interval speed.

Type	Multiple Instances	Control supported
State	Yes	No

Properties	
WarningLevel	No

Table 22: Wiper states

State	Value
Off	0
…	
Full Speed	100

All Page Within This Box

3.4.36 intTemperature

Internal temperature.

Type	Multiple Instances	Control supported
Measurement	No	No

Properties	
WarningLevel	No

Measurement unit is K (temperature unit `kelvin`).

3.4.37 vehicle.lights.headlight

Status of all Headlights.

Type	Multiple Instances	Control supported
State	Yes	Yes

Properties	
WarningLevel	No

Table 23: Headlight states

State	Value
Off	0x00
On	0x01
Low beam	0x11
High beam	0x21

3.4.38 vehicle.lights.foglight

Status of all foglights.

Type	Multiple Instances	Control supported
State	Yes	Yes

Properties	
WarningLevel	No

Table 24: Light states

State	Value
Off	0x00
On	0x01

All Page Within This Box

3.4.39 vehicle.lights.indicator

Status of all indicator lights.

Type	Multiple Instances	Control supported
State	Yes	Yes

Properties	
WarningLevel	No

Table 254a: Indicator states

State	Value
Left turn signal	0x01
Right turn signal	0x02
Reverse light	0x04
Braking lights	0x08
Roof emergency lighting	0x10

3.4.40 vehicle.illumination.internal

Status of all internal illumination.

Type	Multiple Instances	Control supported
State	Yes	Yes

Properties	
WarningLevel	No

Table 26: Internal illumination

State	Value
Car illumination switched on when doors are open	0x01
Car illumination on permanently	0x02
Car illumination currently on	0x04
Driver's local illumination on	0x08
Co-driver's local illumination on	0x10
Rear local illumination on	0x20

3.4.41 vehicle.illumination.dashboard

Status of the dashboard illumination.

All Page Within This Box

Type	Multiple Instances	Control supported
State	No	Yes

Properties	
WarningLevel	No

Table 27: Dashboard illumination

State	Value
Off	0
…	
Full brightness	100

3.4.42 vehicle.washingLiquidLevel

Level of washing liquid.

Type	Multiple Instances	Control supported
Measurement	Yes	No

Properties	
WarningLevel	No

Measurement unit is ltr (volume unit liter).

3.4.43 vehicle.interiorNoise

Internal noise level.

Type	Multiple Instances	Control supported
Measurement	No	No

Properties	
WarningLevel	No

Measurement unit is dB (volume unit dB).

3.4.44 vehicle.window.shutter

Status of all window shutters.

Type	Multiple Instances	Control supported
State	Yes	No

Properties	
WarningLevel	No

All Page Within This Box

Table 25: Shutter states

State	Value
Opaque, closed	0
...	
Transparent, open	100

3.4.45 vehicle.brake.fluidLevel

Level of brake fluid.

Type	**Multiple Instances**	Control supported
State	No	No

Properties	
WarningLevel	Yes

Table 28: Fluid level states

State	Value
Empty	0
...	
Full	100

3.4.46 vehicle.handbrake

Type	**Multiple Instances**	Control supported
State	No	No

Properties	
WarningLevel	Yes

Table 29: Brake states

State	Value
Released	0x00
Brake	0x01

All Page Within This Box

3.4.47 vehicle.steeringWheel.tilt

Type	Multiple Instances	Control supported
Measurement	No	Yes

Properties	
WarningLevel	No

Table 30: Steering wheel tilt states

State	Value
Uppermost	0
...	
Lowermost	100

3.4.48 vehicle.steeringWheel.lock

Type	Multiple Instances	Control supported
State	No	Yes

Properties	
WarningLevel	No

Table 31: Lock states

State	Value
Unlocked	0x00
Locked	0x01

3.4.49 vehicle.seat.Occupation

Type	Multiple Instances	Control supported
State	Yes	No

Properties	
WarningLevel	No

Table 30: Occupation states

State	Value
Free	0x00
Occupied	0x01

All Page Within This Box

3.4.50 vehicle.engine.speed

Engine speed in rpm.

Type	Multiple Instances	Control supported
Measurement	No	No

Properties	
WarningLevel	No

3.4.51 vehicle.engine.coolant.temperature

Type	Multiple Instances	Control supported
Measurement	No	No

Properties	
WarningLevel	No

Measurement unit is K (temperature unit `kelvin`).

3.4.52 vehicle.engine.coolant.pressure

Type	Multiple Instances	Control supported
Measurement	No	No

Properties	
WarningLevel	No

Measurement unit is bar (pressure unit `bar`).

3.4.53 vehicle.engine.coolant.level

Type	Multiple Instances	Control supported
Measurement	No	No

Properties	
WarningLevel	No

Table 32: Fluid level states

State	Value
Empty	0
...	
Full	100

All Page Within This Box

3.4.54 vehicle.engine.oil.temperature

Type	Multiple Instances	Control supported
Measurement	No	No

Properties	
WarningLevel	No

Measurement unit is K (temperature unit `kelvin`).

3.4.55 vehicle.engine.oil.pressure

Type	Multiple Instances	Control supported
Measurement	No	No

Properties	
WarningLevel	No

Measurement unit is bar (pressure unit `bar`).

3.4.56 vehicle.engine.oil.level

Type	Multiple Instances	Control supported
Measurement	No	No

Properties	
WarningLevel	No

See Table 32: Fluid level states.

3.4.57 vehicle.gear.oil.temperature

Type	Multiple Instances	Control supported
Measurement	No	No

Properties	
WarningLevel	No

Measurement unit is K (temperature unit `kelvin`).

3.4.58 vehicle.gear.oil.pressure

Type	Multiple Instances	Control supported
Measurement	No	No

Properties

All Page Within This Box

WarningLevel	No

Measurement unit is bar (pressure unit `bar`).

3.4.59 vehicle.gear.oil.level

Type	Multiple Instances	Control supported
Measurement	No	No

Properties	
WarningLevel	No

See Table 32: Fluid level states.

3.4.60 vehicle.gear.current

The current gear.

Type	Multiple Instances	Control supported
State	No	No

Properties	
WarningLevel	No
Type	Switch, automatic, stepless

Table 32: Gear states

State	Value
N	0
Gear 1	1
...	
Gear 15	15
R	17

3.4.61 vehicle.gear.selected

Type	Multiple Instances	Control supported
State	No	No

Properties	
WarningLevel	No

Type	Switch, automatic, stepless

Table 33: Gear states

State	Value
Switch N	0
Gear 1	1
...	
Gear 15	15
Reverse	17
N	0x100
Limit 1	0x101
...	
Limit 15	0x10F
D	0x110
R	0x111
P	0x112

3.4.62 vehicle.gear.settings

Variable gear settings.

Type	**Multiple Instances**	Control supported
State	No	No

Properties	
WarningLevel	No
Type	Switch, automatic, stepless

Table 34: Gear states

All Page Within This Box

State	Value
Economic	0x01
Sportive	0x02
NoAutomaticShifting	0x04
LowerGearsAutomaticShifting	0x08
KickdownActive	0x10
KickdownLimited	0x20
FirstGearStart	0x40

3.4.63 vehicle.fuelConsumption

Current fuel consumption.

Type	**Multiple Instances**	s
Measurement	No	No

Properties	
WarningLevel	No

3.4.64 vehicle.wheelSpeed

Type	**Multiple Instances**	Control supported
Measurement	No	No

Properties	
WarningLevel	No

Measurement unit is kmh (speed unit `km/h`).

3.4.65 vehicle.odometer.trip

Type	**Multiple Instances**	Control supported
Measurement	Yes	No

Properties	
WarningLevel	No

Measurement unit is km (length unit `kilometer`).

3.4.66 vehicle.odometer.total

Type	**Multiple Instances**	Control supported
Measurement	Yes	No

All Page Within This Box

Properties	
WarningLevel	No

Measurement unit is km (length unit `kilometer`).

3.4.67 vehicle.fuelLevel

Type	Multiple Instances	Control supported
Measurement	No	No

Properties	
WarningLevel	No

Measurement unit is ltr (volume unit `liter`).

3.4.68 vehicle.battery.load

Type	Multiple Instances	Control supported
Measurement	No	No

Properties	
WarningLevel	No

Measurement unit is Ah (charge unit Ah).

3.4.69 vehicle.battery.voltage

Type	Multiple Instances	Control supported
Measurement	No	No

Properties	
WarningLevel	No

Measurement unit is V (`Volt`).

3.4.70 vehicle.airbag

Airbag state

Type	Multiple Instances	Control supported
State	Yes	No

Properties	
WarningLevel	No

Table 35: Airbag states

State	Value
Enabled	0x01
Released	0x02

All Page Within This Box

3.4.71 vehicle.tractionControl

Traction control state

Type	Multiple Instances	Control supported
State	No	No
Properties		
WarningLevel	No	

Table 36: Traction control states

State	Value
Enabled	0x01
Active	0x02

3.4.72 vehicle.brake.antilock

antilock brake state

Type	Multiple Instances	Control supported
State	Yes	No
Properties		
WarningLevel	No	

Table 37: Antilock brake system states

State	Value
Enabled	0x01
Active	0x02

3.4.73 vehicle.cruiseControl.status

Cruise control settings

Type	Multiple Instances	Control supported
State	No	No
Properties		
WarningLevel	No	

Table 34b: cruise control status

All Page Within This Box

State	Value
Reserved	0x01
Active	0x02
Reference speed valid	0x04
Speed up mode	0x08
Speed down mode	0x10

3.4.74 vehicle.cruiseControl.referenceSpeed

Cruise control reference speed.

Type	**Multiple Instances**	Control supported
Measurement	No	No

Properties	
WarningLevel	No

Measurement unit is kmh (speed unit `km/h`).

3.4.75 vehicle.rainSensor

Type	**Multiple Instances**	Control supported
State	No	No

Properties	
WarningLevel	No

Table 38: Rain intensity states (this requires a more precise definition)

State	Value
Dry	0
...	
Max	100

3.4.76 outsideTemperature

Type	**Multiple Instances**	Control supported
Measurement	No	No

Properties	
WarningLevel	Yes

Measurement unit is K (temperature unit `kelvin`).

3.4.77 airPressure

Type	Multiple Instances	Control supported
Measurement	No	No

Properties	
WarningLevel	No

Measurement unit is bar (pressure unit `bar`).

3.4.78 sunIntensity

Type	Multiple Instances	Control supported
Measurement	No	No

Properties	
WarningLevel	No

Measurement unit is lx (illuminance unit `lux`).

3.4.79 vehicle.driverIdentification

Type	Multiple Instances	Control supported
State	No	No

Properties	
WarningLevel	No

Table 39: Driver ID

State	Value
Driver ID	0x00
	…
	0xFE
Unknown	0xFF

3.4.80 vehicle.crashDetector

Type	Multiple Instances	Control supported
State	No	No

Properties	
WarningLevel	No

All Page Within This Box

Table 40: Crash states

State	Value
No Crash	0x00
Crash	0x01

3.4.81 vehicle.distance

Type	Multiple Instances	Control supported
Measurement	Yes	No

Properties	
WarningLevel	No

Measurement unit is km (length unit `kilometer`).

3.4.82 vehicle.maintenance.date

Date of the next vehicle service.

Type	Multiple Instances	Control supported
String	No	No

Properties	
WarningLevel	No

3.4.83 vehicle.maintenance.description

Service description.

Type	Multiple Instances	Control supported
String	No	No

Properties	
WarningLevel	No

3.4.84 vehicle.maintenance.mileage

Type	Multiple Instances	Control supported
Measurement	No	No

Properties	
WarningLevel	No

Measurement unit is km (length unit `kilometer`).

All Page Within This Box

3.4.85 vehicle.centralLocking

State of the central lock.

Type	Multiple Instances	Control supported
State	No	No
Properties		
WarningLevel	No	

Table 41: Central locking status

State	Value
Locked	0x00
Unlocked	0x01

3.4.86 vehicle.cabriolet.category

Indicates the category of cabriolet top.

Type	Multiple Instances	Control supported
State	No	No
Properties		
WarningLevel	No	

Table 42: Cabriolet top category

State	Value
Not available	0x00
Soft top	0x01
Hard/vario top	0x02

3.4.87 vehicle.cabriolet.state

Indicates the category of cabriolet top.

Type	Multiple Instances	Control supported
State	No	Yes
Properties		
WarningLevel	No	

Table 43: Cabriolet top state

All Page Within This Box

State	Value
Error	-1
Closed/mounted	0
…	…
Fully open/unmounted	100

4 Security Considerations

Security issues are covered by wiring.

5 Document Support

5.1 References

Bradner, S., Key words for use in RFCs to Indicate Requirement Levels, RFC2119, March 1997.

5.2 Author's Address

Name	Christof Menzenbach
Company	Siemens VDO
Address	Philipstr. 1; D-35576 Wetzlar
Voice	+49 6441 370 591
e-mail	christof.menzenbach@de3.vdogrp.de

Name:	Vladimir Beliavski
Company:	BMW Group
Address:	Max-Diamand-Str. 13, 80788 München
Voice:	+49 89 382-42680
e-mail:	Vladimir.Beliavski@bmw.de

5.3 Acronyms and Abbreviations

5.4 End of Document

All Page Within This Box