INTERNATIONAL STANDARD

ISO 22902-6

First edition
2006-11-01

Road vehicles — Automotive multimedia interface —

Part 6: Vehicle interface requirements

Véhicules routiers — Interface multimédia pour l'automobile —

Partie 6: Exigences pour l'interface du véhicule

Reference number
ISO 22902-6:2006(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO 22902-6 was prepared by Technical Committee ISO/TC 22, *Road vehicles*, Subcommittee SC 3, *Electrical and electronic equipment*.

ISO 22902 consists of the following parts, under the general title *Road vehicles — Automotive multimedia interface*:

— *Part 1: General technical overview*

— *Part 2: Use cases*

— *Part 3: System requirements*

— *Part 4: Network protocol requirements for vehicle interface access*

— *Part 5: Common message set*

— *Part 6: Vehicle interface requirements*

— *Part 7: Physical specification*

Introduction

A **Vehicle Interface** is defined as a component that provides access to vehicle services from a compliant network. It may act as a gateway to a vehicle-manufacturer-defined network, or it may implement some or all of the vehicle services directly. The **Vehicle Services Interface** is defined as the logical collection of vehicle services implemented by all Vehicle Interfaces in the vehicle.

This document describes the services typically associated with the Vehicle Interface in a system that is compliant with the "AMI-C" specification. These services are grouped into four categories: vehicle services, power management services, Human-Machine Interface (HMI) services, and audio services.

A vehicle service is a function that controls vehicle operations such as door locking and unlocking, data related to the vehicle itself such as diagnostic information, or a system signal indicating vehicle status information such as vehicle speed or fuel level. Vehicle services must be implemented in a Vehicle Interface component.

Power management services control the states, and transition between power states, of the multimedia system, and provide for orderly startup and shutdown of system components. The policy for managing the power states and transitions is derived from carmaker-specific requirements but is implemented as a standard interface on the "AMI-C" system. Power Management Services must be implemented in a Vehicle Interface component.

HMI services provide a standard interface to displays and/or audible interfaces intended for occupants of the vehicle. The policy for displaying and/or annunciating conditions and information in the vehicle is owned by the carmaker. The HMI services are likely to be implemented in a Vehicle Interface component, however they may be partitioned elsewhere at the carmaker's discretion.

Audio services provide a method for devices in the "AMI-C" system to access audio resources. Since it is possible that more than one source may be vying for the use of a resource at a single time, the policy regarding the priority of individual services requesting access to audio services and the arbitration scheme used to ultimately grant access to one or more functions is owned by the carmaker. The audio services are likely to be implemented in a Vehicle Interface component; however, they may be partitioned elsewhere if the carmaker deems this appropriate.

Road vehicles — Automotive multimedia interface —

Part 6:
Vehicle interface requirements

1 Scope

This part of ISO 22902 defines the services provided by the Vehicle Interface. Those services are implemented in an “AMI-C” system in one or both of two ways:

1) as messages on a multimedia network, and

2) as Java Classes in the vehicle interface API (Application Programming Interface).

These are described in separate specifications.

This part of ISO 22902 has the following sections:

— **Vehicle services**. This section presents core services, the vehicle body status and control group, audio services, the powertrain status and control group, security services, and the vehicle diagnostics group.

— **Power management systems**. This section presents system state, power mode, boot sequence messages and shutdown sequence messages.

— **HMI services**. This section provides information about messages sent to the HMI manager as well as the messages that form HMI manager responses. Details about HMI manager are included in Annex A.

— **Audio services**. This section describes elements of audio services typically found in vehicles.

— **Requirements for compliance**. This section presents requirements for “AMI-C” compliance of an implementation of vehicle interface.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

Reference specifications may change. The user must check with the publishing organization and/or automakers for the current requirements.

ISO 22902-1, *Road vehicles — Automotive multimedia interface — Part 1: General technical overview*

ISO 3779, *Road vehicles — Vehicle identification number (VIN) — Content and structure*

ISO 639-1, *Codes for the representation of names of languages — Part 1: Alpha-2 code*

ISO 639-2, *Codes for the representation of names of languages — Part 2: Alpha-3 code*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 22902-1 and the following apply.

3.1
subscription types
service that allows a requestor to register for updates to a parameter without explicitly requesting each update

NOTE There are two types of subscription services, periodic and notification.

3.2
periodic subscription
subscription where the basis for a message being sent is the expiration of a timer

3.3
notification
subscription where the basis for the transmission of the message is that the parameter contained in the message has changed

3.4
service primitive reference
terms which are used in the definitions of the Vehicle Interface Services

3.5
Request
term used in the service primitive to indicate the part of the service where the value of a parameter is being queried

EXAMPLE `Request: Inquire VehicleCore Make-Model`

3.6
Reply
term used in the service primitive to indicate the part of the service where the value of a parameter is being provided

EXAMPLE `Reply:  Report VehicleCore Make-Model <Make-Modeldata>`

3.7
Inquire
term used in the service primitive as the key word for the process of querying a parameter

EXAMPLE `Request: Inquire VehicleCore Make-Model`

3.8
Report
term used in the service primitive as the key word for the process of responding with the value of a parameter

EXAMPLE `Reply:  Report VehicleCore Make-Model <Make-Modeldata>`

3.9
Command
term used in the service primitive to indicate the part of the service where control is being exercised over the value of a parameter

EXAMPLE `Command: Audio SoundChime <chimetype> [on, off]`

3.10
Update
term used in the service primitive to indicate the part of the service where a periodic subscription is being initiated

EXAMPLE `Request: Update VehicleStatus Speed`

3.11
EndUpdate
term used in the service primitive to indicate the part of the service where a periodic subscription is being cancelled

EXAMPLE `Request: EndUpdate VehicleStatus Speed`

3.12
Notify
term used in the service primitive to indicate the part of the service where a notification subscription is being initiated

EXAMPLE `Request: Notify If VehicleStatus OdometerDelta`

3.13
EndNotify
term used in the service primitive to indicate the part of the service where a notification subscription is being cancelled

EXAMPLE `Request: EndNotify VehicleStatus OdometerDelta`

4 Vehicle services

4.1 Core services

The core services must be present in every vehicle interface to comply with the specification. These services represent the basic elements needed to implement a compliant vehicle interface.

The first group of core services provides vehicle identification and configuration information to components. The configuration information is required to allow compliant components to inter-operate between different vehicles.

In the descriptions below, individual functions are identified using courier font. Parameters are enclosed in angle brackets, and alternatives are enclosed in square brackets.

4.1.1 VIN (vehicle identification number) status

This service shall provide the VIN to any node on the network that requests it. The associated messages are:

```
Request:      Inquire VehicleCore VIN
Reply:        Report VehicleCore VIN <VINdata>
```

Where `<VINdata>` is a (17 character) string, the format of the string will correspond to that specified in ISO 3779 or one of the regional variants of that standard.

4.1.2 Manufacturer, model, and model year status

This service shall return information regarding the vehicle manufacturer, model and date of manufacture.

```
Request:        Inquire VehicleCore Make-Model
Reply:          Report VehicleCore Make-Model <Make-Modeldata>
```

Where `<Make-Modeldata>` is a triplet consisting of two strings and a date. The first string (32 characters max) contains the manufacturer name, the second the model name (also 32 characters max). The date is the vehicle's model year as a four digit number. This should correspond to the model year encoded in the VIN.

4.1.3 Manufacturing date and location status

This service returns the date the vehicle was manufactured and the location.

```
Request:        Inquire VehicleStatus DateOfManufacture
Reply:          Report VehicleStatus DateOfManufacture <day> <month>
                <year> <location>
```

where <day> = (1..31), <month> = (1..12), <year> = (0..9999), and <location> is a string (max length of 32 characters). The vehicle manufacturer defines the possible values of this string.

4.1.4 Language status

The service returns the default language code to be used by the vehicle HMI.

```
Request:        Inquire VehicleStatus Language
Reply:          Report VehicleStatus Language <code>
```

where <code> is a field encapsulating the two-digit ISO 639 value for the default language. Examples of the these codes are listed in the table below:

ISO 639 Language Codes (2 byte character)					
en = English	es = Spanish	fr = French	it = Italian	de = German	ja = Japanese
zh = Chinese	ru = Russian	pt = Portuguese	ko = Korean	el = Greek	sv = Swedish
nl = Netherlands					

4.1.5 Vehicle description data status

This service shall return information about the physical description of the vehicle and systems that are present in the vehicle, such as number of doors, number of power windows, number of power seats, fuel tank capacity, cruise control, ABS, etc.

```
Request:        Inquire VehicleCore VehicleDescription
Reply:          Report VehicleCore VehicleDescription
                <VehicleDataStructure>
```

The vehicle data structure shall contain the following data:

- Display Units - This field indicates the units displayed by the vehicle HMI:
  - Displayed Units of Speed 0=Metric 1=English
  - Displayed Units of Distance 0=Metric 1=English

- Displayed Units of Volume 0=Metric 1=English
- Displayed Units of Temp 0=Metric 1=English

— Number of Doors – the number of doors in the vehicle. A door with any of the following characteristics is a candidate for inclusion in this count:

— the open/closed status can be determined

— the locked/unlocked status can be determined

— the locked/unlocked status can be changed

— Door Configuration - This field provides a door description including:

- the capability of the vehicle to report the closed or not closed status of doors (0=no capability; 1=other, 2=driver door explicit, all others combined; 3=all doors explicit).
- the capability to determine the locked vs unlocked status of doors (0=no capability; 1=other, 2=driver door explicit, all others combined; 3=all doors explicit).
- the capability to change the locked vs unlocked status of doors (0=no capability; 1=other, 2=driver door explicit, all others combined; 3=all doors explicit).

— Number and location of seats – This service shall return a data object consisting of the number of seats in the vehicle (integer) and the position of the seats in the vehicle in the form of a Boolean array:

Row 1			Row 2			Row 3			Row 4			Row 5		
L	C	R	L	C	R	L	C	R	L	C	R	L	C	R

Where L=left, C=center, R=right. A “true” indicates that the vehicle has a seat in the row and corresponding position.

— Controllable seats

```
<seatDescription> = <n> <Seat 1> <Seat 2> ... <Seat n>
```

Where each seat has the form:

```
<position> <tilt> <height> <forward> <lumbar> <number of memory
positions> <heater>
```

<position> is a value from an enumerated list of seat positions <position> = driver=1, middlefront=2, passenger=3, secondrowleft=4, secondrowmiddle=5, secondrowright=6, thirdrowleft=7, thirdrowmiddle=8, thirdrowright=9, fourthrowleft=10, fourthrowmiddle=11, fourthrowright=12, fifthrowleft=13, fifthrowmiddle=14, fifthrowright=15.

<tilt> <height> <forward> <lumbar> are each values from an enumerated list that includes 0=no control or position reported, 1=control but no position, 2=no control but position reported, 3=both control and position reported

<number of memory positions> is an integer indicating the number of possible memory presets supported by the seat.

<heater> is a Boolean indicating that the seat is equipped with a seat heater that can be controlled via the network.

— Number / location of windows - this service returns a window description including:

  — The number of windows. Windows that can be either controlled, or whose position can be known (or both) are candidates for inclusion in this count.

  — A parameter that describes the configuration of controllable windows 0=none; 1=other, 2=coupe (driver and passenger windows); 3=sedan (four controllable windows); 4=wagon (4 plus rear window).

  — A parameter that indicates if the vehicle is capable of reporting no information about position, windows are open, windows are closed, or specific window positions.

Windshield Washer Liquid Capacity – indicates the capacity of the front, rear and headlamp washer liquid tanks in liters.

Wheels – an integer indicating the number of wheels on the vehicle.

— Mirrors (number, type) - This service returns a data object with the form:

```
<mirrorDescription> = <n> <mirror 1> ... <mirror n>
```

where each mirror has the form:

```
<position> <l limit> <r limit> <u limit> <d limit> <surface>
```

<position> is a value from an enumerated list of mirror locations including Driver, Center and Passenger.

<l limit> and <u limit> are negative values indicating the maximum movement in the left or up directions.

<r limit> and <d limit> are positive values indicating the maximum movement in the right or down directions.

NOTE the normal or unrotated position of the mirror is 0 in the left and right direction and 0 in the up and down position.

<surface> is a Boolean value indicating the presence of a photo chromic surface.

— Interior lights capability - a value from an enumerated list including no control/no reporting, control/no reporting, no control/reporting, reporting and control.

— Exterior lights capability - a value from an enumerated list including no control/no reporting, control/no reporting, no control/reporting, reporting and control.

— Engine description - This service returns a data object of the form:

<engineDescription> = < Engine Identification> < Engine type> <Number of cylinders> <displacement> <rated power>

where:

  <Engine Identification> a string representing the manufacturer's name for the engine.

  <Engine type> (SI, Diesel, Electric, Hybrid, Other)

  <Number of cylinders> = 0 to 16

  <Displacement> = Cubic Centimeters

  <Rated power> = kilowatts

— Cruise Control - the vehicle’s cruise control type. Supported types are none, standard, adaptive.

— Number of Gears – indicates the number of gears in the vehicle's transmission.

— Sunroofs – indicates the number of sunroofs in the vehicle.

— Transmission Type - the transmission type in the vehicle. Supported types are: manual, automatic or semi-automatic.

— ABS - the vehicle's antilock brake system type, where <type> = none, two wheel, four wheel.

— Traction Control – [present, not present]

— Variable Suspension – [present, not present]

— Vehicle AntiTheft System - indicates the presence (or lack) of a system to prevent a vehicle from being stolen (Immobilizer or restart inhibit systems).

— Content Theft System - indicates the presence (or lack) of a system to deter a thief from stealing possessions from inside a vehicle or entering the vehicle to attempt to steal it (Warning Siren or Burglar Alarm systems).

— Door Lock Type – indicates if the vehicle has manual or electrically actuated door locks.

— Convertible Top Type – none, manual soft top, automatic soft top or hard top.

— Fuel Capacity - the vehicle fuel capacity. Includes parameters for tank and capacity where <tank> = (first, second), and <capacity> is the individual tank's capacity in liters.

— Fuel Type – an enumerated list indicating the type (or types) of fuel used in the vehicle. Supported types are: unleaded gasoline, leaded gasoline, electric, diesel, natural gas, and methanol.

— Vehicle Dimensions – indicates the vehicle length (m), width (m), height (m) and weight (kg).

— Drive Type – indicates front, rear, 4-wheel or all wheel drive type.

— Steering Wheel Location – Indicates right or left hand drive.

— System Voltage – allows indication of 12, 24 or 42 volts systems (a single vehicle may have more than one type indicated).

— Tire Monitoring System – indicates in the vehicle has a tire inflation monitoring system (no absolute pressure available), an actual tire pressure monitoring system, or no monitoring system installed.

— OBD code regional variant - [U.S., Europe, Japan] - This field shall indicate the regional variant of the OBD codes used by the vehicle.

— Implemented warning lights—binary vector indicating the warning lights implemented in the vehicle.

```
For each field, the data structure may return a value of "no
                information". This means that the vehicle services
                interface has no information about the corresponding
                characteristics of the vehicle and also that it does
                not implement any services that use that feature.
```

4.1.6 AMI-C version / release status

This service returns the data describing the version of the ISO 22902 specification that is implemented by this system.

```
Request:        Inquire VehicleCore AMI-C version
Reply:          Report VehicleCore AMI-C version <versiondata>
```

Where <versiondata> consists of:

majorRelease INTEGER (2..31)

minorRelease INTEGER (0..31)

majorVersion INTEGER (0..31)

minorVersion INTEGER (0..31)

4.1.7 Configured services status

Describe the services that the vehicle interface supports, returning a Vehicle Interface Service structure with each of the supported services described.

```
Request:        Inquire VehicleCore Services
Reply:          Report VehicleCore Services <Vehicle Interface
                Service Structure>
```

The Vehicle Interface Service Structure is a Boolean vector with one entry for each vehicle service described in this specification., including the core services. The entry is TRUE if the corresponding service is implemented in the vehicle interface and FALSE otherwise. The Vehicle service structure may be used to determine if the vehicle interface will support the set of services required for a particular application.

4.2 Vehicle status and control group

This group includes services that return information on the current state of the vehicle, such as vehicle speed, odometer reading, etc. This information is provided in response to a request, so each service consists of a request, reply pair.

4.2.1 Vehicle speed status

Vehicle speed is returned by this service. The value is returned in kilometers per hour.

```
Request:        Inquire VehicleStatus Speed
Reply:          Report VehicleStatus Speed <Speed>
```

where <Speed> is a number from 0 to 4095, with each bit equal to 0.1 kph.

Vehicle speed may also be requested as a periodic service (or subscription).

```
Request:        Update VehicleStatus Speed
Reply:          Report VehicleStatus Speed <Speed>
Request:        EndUpdate VehicleStatus Speed
```

The response is the same message as the response to a single vehicle speed request, but it is repeated at a rate of one message every 100 ms.

4.2.2 Vehicle location status

Vehicle location is included in the vehicle interface, in the expectation that it may be implemented on the vehicle side of the interface rather than the

ISO 22902 side. However, it might also be included in the navigation interface specification.

```
Request:    Inquire Vehicle Location Coordinates
Reply:      Report Vehicle Location Coordinates <year>, <month>,
            <day>, <hours>, <minutes>, <secs>, <latitudeZone>,
            <latitudeDegree>, <latitudeMinute>,
            <latitudeSubMinute>, <longitudeZone>,
            <longitudeDegree>, <longitudeMinute>,
            <longitudeSubMinute>, <velocity>, <direction>
```

Where:

<year>=INTEGER(0..255)

<month>=INTEGER(1..12)

<day>=INTEGER(1..31)

<hours>=INTEGER(0..23)

<minutes>=INTEGER(0..59)

<secs>=INTEGER(0..59)

<latitudeZone>=INTEGER(0..1)

<latitudeDegree>=INTEGER(0..90)

<latitudeMinute>=INTEGER(0..59)

<latitudeSubMinute>=INTEGER(0..9999)

<longitudeZone>=INTEGER(2..3)

<longitudeDegree>=INTEGER(0..179)

<longitudeMinute>=INTEGER(0..59)

<longitudeSubMinute>=INTEGER(0..9999)

<velocity>=INTEGER(0..255)

<direction>=INTEGER(0..359)

Vehicle location information may also be requested as a periodic service (or subscription).

```
Request:    Update Vehicle Location Coordinates <interval>
Reply:      Report Vehicle Location Coordinates <year>, <month>,
            <day>, <hours>, <minutes>, <secs>, <latitudeZone>,
            <latitudeDegree>, <latitudeMinute>,
            <latitudeSubMinute>, <longitudeZone>,
            <longitudeDegree>, <longitudeMinute>,
            <longitudeSubMinute>, <velocity>, <direction>
Request:    EndUpdate Vehicle Location Coordinates
```

The response is the same message as the response to a single Vehicle location request; however it is repeated at the rate of one message every 1000 ms.

4.2.3 Odometer reading status

This service returns the current odometer reading in kilometers.

```
Request:      Inquire VehicleStatus Odometer
Reply:        Report VehicleStatus Odometer <value>
```

where <value> is an integer from 0 to 999,999.

It is also possible to request notification[1)] on a change of this parameter.

```
Request:      Notify If VehicleStatus OdometerDelta
Reply:        Report VehicleStatus Odometer <value>
Request:      EndNotify VehicleStatus OdometerDelta
```

4.2.4 High resolution distance accumulator status

This is a higher accuracy distance indicator.

```
Request:      Inquire VehicleStatus DifferentialOdometer
Reply:        Report VehicleStatus DifferentialOdometer <value>
```

where <value> is a number from 0 to 9999. The resolution of this value is 0.1 km per bit.

The high-resolution odometer may also be requested as a periodic service (or subscription).

```
Request:      Update VehicleStatus DifferentialOdometer
Reply:        Report VehicleStatus DifferentialOdometer <Value>
Request:      EndUpdate VehicleStatus DifferentialOdometer
```

The response is the same message as the response to a single high-resolution odometer request, but it is repeated at the rate of one message every 250 ms.

4.2.5 Fuel level status

This service returns the fuel tank level in liters.

```
Request:      Inquire VehicleStatus FuelLevel <tank>
Reply:        Report VehicleStatus FuelLevel <tank> <value>
```

The <tank> parameter is for vehicles that have more than one fuel tank. The return parameter <value> is a number from 0 to 4095. The resolution is 0.1l per bit.

4.2.6 Battery charge status

This service returns battery charge as a percentage of full charge.

```
Request:      Inquire VehicleStatus BatteryCharge <battery>
Reply:        Report VehicleStatus BatteryCharge <battery> <value>
```

The <battery> parameter is for vehicles that have more than one battery, and <value> is a number from 0 to 255, representing the state of charge of the battery (0=0%, 255=100%)

---

1) A request for *Notification* indicates that the requester requires that a message be transmitted only when the value of the parameter changes. This allows data monitors to be 'notified' when the parameter changes as opposed to having to poll the data owner to determine when a change occurs.

4.2.7 Airbag status

This service allows an application to discover whether a given airbag is installed and if so, deployed.

```
Request:      Inquire VehicleStatus AirBags <seatposition>
Reply:        Report VehicleStatus AirBags <seatposition>
              <installed_airbags><airbag_status>
```

<seatposition> = driver=1, middlefront=2, passenger=3, secondrowleft=4, secondrowmiddle=5, secondrowright=6, thirdrowleft=7, thirdrowmiddle=8, thirdrowright=9, fourthrowleft=10, fourthrowmiddle=11, fourthrowright=12, fifthrowleft=13, fifthrowmiddle=14, fifthrowright=15.

<installed_airbags> and <airbag_status> are bit strings of the form:

```
<installed_airbags>
      bit0     Front Airbag  1=Present 0=Not Present
      bit1     Side Airbag   1=Present 0=Not Present
      bit2     Head Airbag   1=Present 0=Not Present

<airbag_status>
      bit0     Front Airbag  1=Deployed   0=Not Deployed
      bit1     Side Airbag   1=Deployed   0=Not Deployed
      bit2     Head Airbag   1=Deployed   0=Not Deployed
```

It is also possible to request notification on a change of this parameter. A device requesting notification will be provided a message on the change of status of any airbag associated with any seat.

```
Request: Notify if VehicleStatus AirBagsDelta
Reply:   Report VehicleStatus AirBags <seatposition>
         <installed_airbags><airbag_status>
Request: EndNotify VehicleStatus AirBagsDelta
```

4.2.8 Seat belt status

```
Request:      Inquire VehicleStatus SeatBelt <position>
Reply:        Report VehicleStatus SeatBelt <position> <SBstate>
```

Where <SBstate> = [latched, unlatched]

It is also possible to request notification on a change of this parameter:

```
Request:      Notify if VehicleStatus SeatBeltDelta
Reply:        Report VehicleStatus SeatBelt <position> <SBstate>
Request:      EndNotify VehicleStatus SeatBeltDelta
```

4.2.9 Brake applied status

This service returns whether or not a particular brake is applied.

```
Request:      Inquire VehicleStatus BrakeApplied <wheel>
Reply:        Report VehicleStatus BrakeApplied <wheel> [applied,
              not applied]
```

Where <wheel> = [any, front left, front right, rear left, rear right]

4.2.10 Brake status

This service returns the value of the brake on-off switch. This indicates whether the (main) braking system is engaged.

```
Request:      Inquire VehicleStatus BrakeOnOff
Reply:        Report VehicleStatus BrakeOnOff <state>
```

Where <state> = [Off, On]

It is also possible to request notification on a change of this parameter:

```
Request:      Notify if VehicleStatus BrakeOnOffDelta
Reply:        Report VehicleStatus BrakeOnOff <state>
Request:      EndNotify VehicleStatus BrakeOnOffDelta
```

4.2.11 Hand Brake status

This service returns the value of the hand brake (also known as the park brake). This indicates whether the hand brake is engaged.

```
Request:      Inquire VehicleStatus HandBrake
Reply:        Report VehicleStatus HandBrake <state>
```

Where <state> = [not engaged, engaged]

It is also possible to request notification on a change of this parameter:

```
Request:      Notify if VehicleStatus HandBrakeDelta
Reply:        Report VehicleStatus HandBrake <state>
Request:      EndNotify VehicleStatus HandBrakeDelta
```

4.2.12 Traction control status

This service returns the traction control status. A client can use the vehicle data message (reference section) to determine whether traction control is present.

```
Request:      Inquire VehicleStatus TractionControl
Reply:        Report VehicleStatus TractionControl <TCstate>
```

<TCstate> = [off, on, engaged]

It is also possible to request notification on a change of this parameter:

```
Request:      Notify if VehicleStatus TractionControlDelta
Reply:        Report VehicleStatus TractionControl <TCstate>
Request:      EndNotify VehicleStatus TractionControlDelta
```

4.2.13 Anti-lock brake status

This service returns the ABS status (off, on, engaged).

```
Request:      Inquire VehicleStatus ABS
Reply:        Report VehicleStatus ABS <ABSstate>
```

Where <ABSstate> = [off, on, engaged]

It is also possible to request notification on a change of this parameter:

```
Request:      Notify if VehicleStatus ABSDelta
Reply:        Report VehicleStatus ABS <ABSstate>
Request:      EndNotify VehicleStatus ABSDelta
```

4.2.14 Variable suspension status

This service returns the setting for those vehicles that have variable suspension control.

```
Request:      Inquire VehicleStatus SuspensionControl
Reply:        Report VehicleStatus SuspensionControl <state>
```

Where <state> = [off, soft, normal, sport]

It is also possible to request notification on a change of this parameter:

```
Request:      Notify if VehicleStatus SuspensionControlDelta
Reply:        Report VehicleStatus SuspensionControl <state>
Request:      EndNotify VehicleStatus SuspensionControlDelta
```

4.2.15 Cruise control status

This service returns the Cruise Control status (off, on, engaged).

```
Request:      Inquire VehicleStatus CruiseControl
Reply:        Report VehicleStatus CruiseControl <state>
```

Where <state> = [off, on, engaged]

It is also possible to request notification on a change of this parameter:

```
Request:      Notify if VehicleStatus CruiseControlDelta
Reply:        Report VehicleStatus CruiseControl <state>
Request:      EndNotify VehicleStatus CruiseControlDelta
```

Where <value> = a number from 0 to 4095 with a resolution of 0.1l per bit representing the setpoint speed in kilometers/per hour.

4.2.16 Set point speed status

```
Request:      Inquire VehicleStatus CruiseControlSetpoint
Reply:        Report VehicleStatus CruiseControlSetpoint <value>
```

Where <value> = a number from 0 to 4095 with a resolution of 0.1l per bit representing the setpoint speed in kilometers/per hour.

4.2.17 Ignition key status

This service returns the position of the ignition key.

```
Request:      Inquire VehicleStatus IgnitionStatus
Reply:        Report VehicleStatus IgnitionStatus <state>
```

Where <state> = [off-no key in, off-key in, accessory, run, start]

It is also possible to request notification on a change of this parameter:

```
Request:      Notify if VehicleStatus IgnitionStatusDelta
Reply:        Report VehicleStatus IgnitionStatus <state>
```

NOTE This information should not be used in place of the Power Mode and System State information to perform device power moding. It may be used, for example, as an input to the decision about which features to enable. It may not be used, however by a device in making the determination to go to sleep.

4.2.18 Ignition lock status

This service returns the status of the ignition key lock.

```
Request:      Inquire VehicleStatus IgnitionLock
Reply:        Report VehicleStatus IgnitionLock <state>
```

Where <state> = [unlocked, locked due to no key-in, locked due to anti-theft measures]

It is also possible to request notification on a change of this parameter:

```
Request:      Notify if VehicleStatus IgnitionLockDelta
Reply:        Report VehicleStatus IgnitionLock <state>
Request:      EndNotify VehicleStatus IgnitionLockDelta
```

4.2.19 Warning light status

This service returns the status of the vehicle warning lights as a Binary vector. The following warning lights are supported, although not all will be present in each vehicle: oil pressure, oil level, change oil soon, check engine, service vehicle, air bag malfunction, vehicle stability control, traction control, ABS, battery warning, brakes, parking brakes, door ajar, tire pressure, tire inflation, seat belt, exhaust temperature, power steering, low fuel, exterior light on, traction Control on/off, four wheel drive on/off, cruise control on/off, hazard lights on/off. The vehicle description service indicates whether a given warning light is present, and must be consulted to determine the meaning of the status returned by this service. If a given warning light is not present in the vehicle, the vehicle interface shall return a constant value of "off" for the corresponding entry in the vector.

```
Request:      Inquire VehicleStatus WarningLights
Reply:        Report VehicleStatus WarningLights
              <warningLightVector>
```

Each entry in the vector can have one of two values: off or on.

It is also possible to request notification on a change of this parameter:

```
Request:      Notify if VehicleStatus WarningLightDelta
Reply:        VehicleStatus WarningLightOn <warningLightVector>
Request:      EndNotify VehicleStatus WarningLightDelta
```

4.2.20 Chime status and control

This service returns the active chime from the vehicle, or sounds a particular chime in the vehicle. Characteristics of the chime (pitch, duration, etc.) are determined by the vehicle implementation and may vary between different vehicles. Where the vehicle is capable of multiple "chime" sounds, a specific one shall be picked to implement this service at the discretion of the automaker.

```
Request:      Inquire VehicleStatus Chime <chimetype>
Reply:        Report VehicleStatus Chime <chimetype> [on, off]
Command:      Audio SoundChime <chimetype> [on, off]
```

Where <chimetype> includes: key-in-ignition minder, check gages, security warning, headlamp warning, activation confirmation, seatbelt warning, park brake warning, and turn signal minder.

4.2.21 Door lock status

The parameter <doorNumber> is a value from an enumerated list defining the position of the door in the vehicle. The syntax and values for the parameter of this message set representation are given in the AMI-C 2002, AMI-C Common Message Set.

```
Request:      Inquire VehicleStatus DoorLockStatus <doorNumber>
Reply:        Report VehicleStatus DoorLockStatus <doorNumber>
              [locked, unlocked]
```

where <doornumber> is hood=1, driver=2, passenger=3, rearleft=4, rearright=5, rear=6, driver's sliding=7, passenger's sliding=8.

It is also possible to request notification on a change of this parameter:

```
Request:      Notify if VehicleStatus DoorLockStatusDelta
Reply:        Report VehicleStatus DoorLockStatus <doorNumber>
              [locked, unlocked]
Request:      EndNotify VehicleStatus DoorLockStatusDelta
```

4.2.22 Door lock control

The commands in this category are to unlock all doors, and to unlock the driver's door.

```
Command:      Vehicle UnlockAllDoors
Command:      Vehicle LockAllDoors
Command:      Vehicle UnlockDriverDoor
Command:      Vehicle LockDriverDoor
```

4.2.23 Door/trunk status

This function returns the value of the "door ajar" warning signal.

NOTE The door number can also be used to designate the trunk and the hood.

```
Request:      Inquire VehicleStatus DoorStatus <doorNumber>
Reply:        Report VehicleStatus DoorStatus <doorNumber>
              [closed, ajar]
```

where <doorNumber> is a value from the following list: hood=1, driver=2, passenger=3, rear left=4, rear right=5, rear=6, driver's sliding=7, passenger's sliding=8.

It is also possible to request notification on a change of this parameter:

```
Request:      Notify if VehicleStatus WindowStatusDelta
Reply:        Report VehicleStatus WindowStatus <windowNumber>
              <position>
Request:      EndNotify VehicleStatus WindowStatusDelta
```

4.2.24 Sliding door control

This command allows either the driver or passenger's sliding doors to be opened or closed.

```
Command:      Vehicle SlidingDoorControl <doorloc><state>
```

where <doorloc> is 0=driver sliding door, 1=passenger sliding door and <state> = [open, close].

4.2.25 Open trunk command

This command causes the trunk to be opened.

```
Command:      Vehicle TrunkOpen
```

4.2.26 Open fuel filler door command

This command causes the fuel filler door to be opened.

```
Command:      Vehicle FuelFillerDoorOpen
```

4.2.27 Window closure status

This service returns the state of the designated window.

```
Request:      Inquire VehicleStatus WindowCloseStatus
              <windowNumber>
Reply:        Report VehicleStatus WindowCloseStatus
              <windowNumber> [not_closed, closed]
```

where <windownumber> = driver=1, passenger=2, rearpassengerleft=3, rearpassengerright=4, rear=5.

4.2.28 Window position status

This service returns the position of the designated window, in counts. The meaning of the counts returned is specific to a given class of vehicle. It does not correspond to an absolute position across all vehicles.

```
Request:      Inquire VehicleStatus WindowStatus <windowNumber>
Reply:        Report VehicleStatus WindowStatus <windowNumber>
              <position>
```

where <windownumber> = driver=1, passenger=2, rearpassengerleft=3, rearpassengerright=4, rear=5; <position> is a number from 0 to 255 (0=completely open and 255=completely closed).

It is also possible to request notification on a change of this parameter:

```
Request:      Notify if VehicleStatus WindowStatusDelta
Reply:        Report VehicleStatus WindowStatus <windowNumber>
              <position>
Request:      EndNotify VehicleStatus WindowStatusDelta
```

4.2.29 Window position control

Window control may be supported in one of two forms. This form commands the window to open or close to a specified position. The position parameter is a count between 0 and 255. The meaning of the count on a given vehicle is not specified. Some vehicles may only support setting the position full open or full closed. If this is the case on a given vehicle, the vehicle interface must return an error if an attempt is made to set the window to an unsupported (intermediate) position.

```
Command:      Vehicle WindowMove <windowNumber> <position>
where <windownumber> = driver=1, passenger=2, rearpassengerleft=3,
              rearpassengerright=4, rear=5; <position> is a number
              from 0 to 255 (0=completely open and 255=completely
              closed).
```

4.2.30 Window motion control

Window control may be supported in one of two forms. This form simply sends a command to the window to move down (or up) for a fixed period of time. The period is unspecified. In general, multiple commands may be required to fully open/close a window using this method.

```
Command:      Vehicle WindowDown <windowNumber>
Command:      Vehicle WindowUp <windowNumber>
```

where <windownumber> = driver=1, passenger=2, rearpassengerleft=3, rearpassengerright=4, rear=5.

4.2.31 Sun/moon roof status

This service returns the state of a specific Sun/Moon roof.

```
Request:      Inquire VehicleStatus SunRoofStatus <sunroof>
Reply:        Report VehicleStatus SunRoofStatus <sunroof><value>
```

where <sunroof> is an integer from 1 to 7 (sunroof 1 is the closest to the front of the vehicle).

It is also possible to request notification on a change of this parameter:

```
Request:      Notify if VehicleStatus SunRoofStatusDelta
Reply:        Report VehicleStatus SunRoofStatus <value>
Request:      EndNotify VehicleStatus SunRoofStatusDelta
```

4.2.32 Sun/moon roof control

```
Command:      Vehicle RoofClose <sunroof>
Command:      Vehicle RoofOpen <sunroof>
```

where <sunroof> is an integer from 1 to 7 (sunroof 1 is the closest to the front of the vehicle).

4.2.33 Interior lights status and control

Interior lights' services control all interior lights as a group. Some vehicles may only support values 0 and 255 (off or on) for the interior lights. In this case, a message that tries to set the lights to an intermediate level should return an error.

```
Request:      Inquire VehicleStatus Interiorlights
Reply:        Report VehicleStatus Interiorlights <ILlevel>
Command:      Vehicle SetInteriorLights <ILlevel>
```

where <ILlevel> = 0..255 (255=100% interior light intensity).

4.2.34 Courtesy light switch status

This service returns the status of the switch governing the behavior of the vehicle's courtesy lights.

```
Request:      Inquire VehicleStatus CourtesylightSwitch
Reply:        Report VehicleStatus Courtesylights
              <CtsySwitchStatus>
```

where <CtsySwitchStatus> = is an enumerated variable with the three states listed below:

1 = this position turns the courtesy lights on,

2 = this position turns the courtesy lights on only if a door is open,

3 = this position never turns the courtesy lights even if a door is opened.

It is also possible to request notification on a change of this parameter:

```
Request:      Notify if VehicleStatus CourtesylightSwitchDelta
Reply:        Report VehicleStatus Courtesylights
              <CtsySwitchStatus>
Request:      EndNotify VehicleStatus CourtesylightSwitchDelta
```

4.2.35 Map lights status and control

This service returns the status of the map lights (driver's, passenger's and rear).

```
Request:      Inquire VehicleStatus Maplights
Reply:        Report VehicleStatus Maplights <driver>
              <passenger><rearleft><rearright>
Command:      Vehicle Maplights <driver>
              <passenger><rearleft><rearright>
```

where <driver>, <passenger> and <rearleft> and <rearright> each may either be [on, off].

4.2.36 Dashboard illumination status and control

This service returns the percentage value of interior dimming (percent of full bright).

```
Request:      Inquire VehicleStatus DimmingLev
Reply:        Report VehicleStatus Dimming Level <level>
Command:      Command VehicleStatus Dimming Level <level>
```

where <level> = 0..255 (255=100% dimming level)

4.2.37 Head lights status and control

```
Request:      Inquire VehicleStatus Headlights
Reply:        Report VehicleStatus Headlights <HLstatus>
Command:      Vehicle HeadlightsOn
Command:      Vehicle HeadlightsOff
Command:      Vehicle HeadlightsHighBeamOn
Command:      Vehicle HeadlightsHighBeamOff
Command:      Vehicle HeadlightsFlashOn
```

```
Command:        Vehicle HeadlightsFlashOff
Command:        Vehicle HighBeamFlashOff
Command:        Vehicle HighBeamFlashOff
```

where <HLstatus> = [off, lowbeam, highbeam, flash, highbeam flash].

It is also possible to request notification on a change of this parameter:

```
Request:        Notify if VehicleStatus HeadlightsDelta
Reply:          Report Report VehicleStatus Headlights <HLstatus>
Request:        EndNotify VehicleStatus HeadlightsDelta
```

4.2.38 Headlight tilt status

This service returns information regarding the current tilt of the headlights.

```
Request:        Inquire VehicleStatus HeadlightTilt
Reply:          Report VehicleStatus HeadlightTilt <Tilt_level>
```

where <Tilt_level> = 0..255 (0=maximum down tilt, 255=maximum up tilt).

4.2.39 Fog lamps status and control

```
Request:        Inquire VehicleStatus FogLamps
Reply:          Report VehicleStatus FogLamps <FrontFLstatus>
                <RearFLstatus>
```

It is also possible to request notification on a change of this parameter:

```
Request:        Notify if VehicleStatus FogLampsDelta
Reply:          Report VehicleStatus FogLamps <FrontFLstatus>
                <RearFLstatus>
Request:        EndNotify VehicleStatus FogLampsDelta
```

where <FrontFLstatus> and <RearFLstatus> = [off, on].

```
Command:        Vehicle FrontFLOn
Command:        Vehicle FrontFLOff
Command:        Vehicle RearFLOn
Command:        Vehicle RearFLOff
```

4.2.40 Parking lights status and control

```
Request:        Inquire VehicleStatus Parkinglights
Reply:          Report VehicleStatus Parkinglights <PLstate>
Command:        Vehicle ParkingLightsOn <PLstate>
```

where <PLstate> = [off, left side only on, right side only on, all on].

It is also possible to request notification on a change of this parameter:

```
Request:        Notify if VehicleStatus ParkinglightsDelta
Reply:          Report VehicleStatus Parkinglights <PLstate>
Request:        EndNotify VehicleStatus ParkinglightsDelta
```

4.2.41 Turn signal status

```
Request:      VehicleStatus TurnSignal
Reply:        Report VehicleStatus TurnSignal <TSstatus>
```

where <TSstatus> = [off, left, right].

It is also possible to request notification on a change of this parameter:

```
Request:      Notify if VehicleStatus TurnSignalDelta
Reply:        Report VehicleStatus TurnSignal <TSstatus>
Request:      EndNotify VehicleStatus TurnSignalDelta
```

4.2.42 Hazard signal status

```
Request:      Inquire VehicleStatus HazardSignal
Reply:        Report VehicleStatus HazardSignal<HSstatus>
Command:      Command Vehicle HazardSignal<HSstatus>
```

where <HSstatus> = [off, on].

It is also possible to request notification on a change of this parameter:

```
Request:      Notify if VehicleStatus HazardSignalDelta
Reply:        Report Report VehicleStatus HazardSignal<HSstatus>
Request:      EndNotify VehicleStatus HazardSignalDelta
```

4.2.43 Emergency light status

This service returns the state, or sets the state of a vehicle's emergency roof lights.

```
Request:      Inquire VehicleStatus EmergencyLights
Reply:        Report VehicleStatus EmergencyLights [on, off]
Command:      Command Vehicle EmergencyLights [on, off]
```

4.2.44 Mirror status

This section includes a request to get the current mirror position for each specific mirror, a command that allows setting a specific mirror position, as well as moving an amount relative to the current position. The mirror position is given in terms of pan and tilt counts. However, applications should not assume that a given set of pan and tilt counts corresponds to an absolute mirror position, relative to the driver in all vehicles. Applications should not make assumptions regarding the resolution with which a given mirror will carry out an angle change.

```
Request:      Inquire VehicleStatus MirrorPosition <mirror>
Reply:        Report VehicleStatus MirrorPosition <mirror> <pan>
              <tilt>
Command:      Command VehicleStatus MirrorPosition <mirror>
              [relative_motion, absolute_motion] <pan> <tilt>
```

where <mirror> = [driver, center, passenger] , <pan> is a number from –127 to 128 (-127 is maximum pan to the left, 128 is the maximum pan to the right), <tilt> > is a number from –127 to 128 (-127 is maximum tilt down, 128 is the maximum tilt up).

4.2.45 Mirror control

A single command of this service results in mirror motion for a certain period of time. Multiple commands have to be issued to move the mirror smoothly. When commands to move a single object to different direction arrive from different applications or devices, the receiving application should arbitrate these commands with its own policy.

```
Command:        Vehicle MirrorPanLeft <mirror>
Command:        Vehicle MirrorPanRight <mirror>
Command:        Vehicle MirrorTiltUp <mirror>
Command:        Vehicle MirrorTiltDown <mirror>
```

where <mirror> = [driver, center, passenger].

4.2.46 Mirror fold control

This service commands the driver and passenger outside rearview mirrors to fold or unfold.

```
Command:        Vehicle FoldMirror [fold, unfold]
```

4.2.47 Seat position status

This service returns the position of the requested seat. The position parameters are specified as a fraction of range of travel in each direction to allow applications to be written, which are independent of the dimensions of particular vehicles.

```
Request:        Inquire VehicleStatus SeatPosition <seat>
Reply:          Report VehicleStatus SeatPosition <seat> <tilt>
                <headrest> <seatfront> <seatback> <forward/back>
                <up/down> <lumbar>
```

where

<seat> = (1 = driver, 2 = middleFront, 3 = passenger, 4 = secondRowLeft, 5 = secondRowMiddle, 6 = secondRowRight, 7 = thirdRowLeft, 8 = thirdRowMiddle, 9 = thirdRowRight, 10 = fourthRowLeft, 11 = fourthRowMiddle, 12 = fourthRowRight, 13 = fifthRowLeft, 14 = fifthRowMiddle, 15 = fifthRowRight);

<tilt> = -127..128, -127 =vertical, 128 =full tilt;

<headrest> = -127..128, -127 =full down, 128 =full up;

<seatfront> = -127..128, -127 =full down, 128 =full up;

<seatback> = -127..128, -127 =full down, 128 =full up;

<forward/back> = -127..128, -127= forward, 128 =back;

<up/down> = -127..128, -127 =down, 128 =up;

<lumbar> = -127..128, -127 =soft, 128 =firm.

4.2.48 Seat occupied status

This service returns the occupied status of the requested seat.

```
Request:        Inquire VehicleStatus SeatOccupied <seat>
Reply:          Report VehicleStatus SeatOccupied <seat> [true,
                false]
```

It is also possible to request notification on a change of this parameter:

```
Request:        Notify if VehicleStatus SeatOccupiedDelta
Reply:          Report VehicleStatus SeatOccupied <seat> [true,
                false]
Request:        EndNotify VehicleStatus SeatOccupiedDelta
```

4.2.49 Seat adjust control

This service adjusts the position of the designated seat. See the "Body Status" group for more information.

```
Command:        VehicleStatus SeatPosition <seat> <tilt> <headrest>
                <seatfront> <seatback> <forward/back> <up/down>
                <lumbar>
```

where

<seat> = (1 = driver, 2 = middleFront, 3 = passenger, 4 = secondRowLeft, 5 = secondRowMiddle, 6 = secondRowRight, 7 = thirdRowLeft, 8 = thirdRowMiddle, 9 = thirdRowRight, 10 = fourthRowLeft, 11 = fourthRowMiddle, 12 = fourthRowRight, 13 = fifthRowLeft, 14 = fifthRowMiddle, 15 = fifthRowRight);

<tilt> = -127..128, -128 =vertical, 128 =full tilt;

<headrest> = -127..128, -127 =full down, 128 =full up;

<seatfront> = -127..128, -127 =full down, 128 =full up;

<seatback> = -127..128, -127 =full down, 128 =full up;

<forward/back> = -127..128, -127= forward, 128 =back;

<up/down> = -127..128, -127 =down, 128 =up;

<lumbar> = -127..128, -127 =soft, 128 =firm.

4.2.50 Antenna status and control

This service returns the antenna position, up or down.

```
Request:        Inquire VehicleStatus Antenna
Reply:          Report VehicleStatus Antenna [up, down]
```

It is also possible to request notification on a change of this parameter:

```
Request:        Notify if VehicleStatus AntennaDelta
Reply:          Report VehicleStatus Antenna [up, down]
Request:        EndNotify VehicleStatus AntennaDelta
```

This function raises and lowers the antenna.

```
Command:        Vehicle AntennaUp
Command:        Vehicle AntennaDown
```

4.2.51 Tire inflation status

This service returns the tire inflation status for the vehicle.

```
Request:        Inquire VehicleStatus Antenna
Reply:          Report VehicleStatus Antenna [up, down]
```

where <TIstatus> = [low, normal, high].

It is also possible to request notification on a change of this parameter:

```
Request:        Notify if VehicleStatus AntennaDelta
Reply:          Report VehicleStatus Antenna [up, down]
Request:        EndNotify VehicleStatus AntennaDelta
```

4.2.52 Tire pressure status

This service returns the tire pressure from the requested tire.

```
Request:        Inquire VehicleStatus TirePressure <tireNumber>
Reply:          Report VehicleStatus TirePressure <tireNumber>
                <TPressure>
```

where <TPressure> = a number from 0 to 255 with a resolution of 4 kPa per bit (0 to 1020 kPa).

4.2.53 Horn status and control

This service indicates that the status of the horn, or sounds the horn.

```
Inquire:        Inquire VehicleStatus Horn
Reply:          Report VehicleStatus Horn [on, off]
Command:        Vehicle HornOn
Command:        Vehicle HornOff
Command:        Vehicle HornPulse
```

It is also possible to request notification on a change of this parameter:

```
Request:        Notify if VehicleStatus HornDelta
Reply:          Report VehicleStatus Horn [on, off]
Request:        EndNotify VehicleStatus HornDelta
```

4.2.54 Security alert status and control

The effect of this command is defined by the automaker, and may not be appropriate for all applications in which the horn service is used.

```
Inquire:        Inquire VehicleStatus SecurityAlert
Reply:          Report VehicleStatus SecurityAlert <status>
Command:        Command Vehicle TriggerSecurityAlert
Command:        Command Vehicle EndSecurityAlert
```

4.2.55 Noise level status

This service returns the value of the noise level sensor.

```
Request:      Inquire VehicleStatus NoiseLevel
Reply:        Report VehicleStatus NoiseLevel <level>
```

where <level> = 0..255.

Noise Level may also be requested as a periodic service (or subscription).

```
Request:      Update VehicleStatus NoiseLevel
Reply:        Report VehicleStatus NoiseLevel <level>
Request:      EndUpdate VehicleStatus NoiseLevel
```

The response is the same message as the response to a single Noise Level request, but it is repeated at the rate of one message every 1000 ms.

4.2.56 Obstacle distance status

This service returns the distance to an obstacle as measured by a parking assist or obstacle detection system.

```
Request:      Inquire VehicleStatus ObstacleDist
Reply:        Report VehicleStatus ObstacleDist
              <front_sensor_status> <front_distance>
              <rear_sensor_status> <rear_distance>
```

where <front_sensor_status> and <rear_sensor_status> = [not equipped, off, on]; <front_distance> and <rear_distance> = 0..4096 mm.

4.2.57 Wiper system status

This service returns the system state of the various wipers in the vehicle.

```
Request:      Inquire VehicleStatus Wipers
Reply:        Report VehicleStatus Wipers
              <windshield_wiperstatus><rear_wiperstatus><headlamp_
              wiperstatus>
Command:      Command VehicleStatus Wipers
              <windshield_wiperstatus><rear_wiperstatus><headlamp_
              wiperstatus>
```

where <wiperstatus> = [not equipped, off, intermittent, low, high, maintenance] (the maintenance state refers to a wiper position that is accessible for replacing or cleaning the wiper blades, etc).

It is also possible to request notification on a change of this parameter:

```
Request:      Notify if VehicleStatus WipersDelta
Reply:        Report VehicleStatus Wipers <wiperstatus>
Request:      EndNotify VehicleStatus WipersDelta
```

4.2.58 Wiper speed status and control

This service returns the speed that the various wipers are moving.

```
Request:        Inquire VehicleStatus WiperSpeed
Reply:          Report VehicleStatus WiperSpeed
                <windshield_wiperspeed> <rear_wiperspeed>
                <headlamp_wiperspeed>
Command:        Command VehicleStatus WiperSpeed
                <windshield_wiperspeed> <rear_wiperspeed>
                <headlamp_wiperspeed>
```

4.2.59 Washing liquid level status

This service returns the level of the liquid in various washer systems (windshield, back window, headlamps) as a percentage of full.

```
Request:        Inquire VehicleStatus WasherLiquidLevel <position>
Reply:          Report VehicleStatus WasherLiquidLevel <position>
                <level>
```

where <position> = [windshield, rear_window, headlight] and <level> = 0..255 (0= empty, 255=full).

4.2.60 Steering lock status

This service returns the status of the steering lock.

```
Request:        Inquire VehicleStatus SteeringLock
Reply:          Report VehicleStatus SteeringLock [unlocked, locked]
```

4.2.61 Steering column status and control

This service returns the position of the steering column. It may also be used to set the steering column to a specific position, or simply command motion.

```
Request:        Inquire VehicleStatus SteeringColPos
Reply:          Report VehicleStatus SteeringColPos <tilt pos>
                <telescope pos>
Command:        Set  Vehicle SteeringColPos <tilt pos> <telescope
                pos>
```

where <tilt pos> =0..255 (0=maximum down tilt, 255=maximum up tilt) and <telescope pos> = 0..255 (0=maximum telescope in, 255=maximum telescope out).

It is also possible to simply command steering column motion, without providing a specific location.

```
Command:        Vehicle SteeringColMove <tilt_telescope motion>
```

where

<tilt_telescope motion> = [tilt_up, tilt_down, telescope_in, telescope_out].

4.2.62 Interior temperature status

This service returns the vehicle interior temperature in Celsius.

```
Request:      Inquire VehicleStatus InteriorTemperature <zone>
Reply:        Report VehicleStatus InteriorTemperature <zone>
              <temp>
```

where <zone> = [driver, passenger, rear_left, rear_right] and <temp> is an integer from –40..150 degrees C.

4.2.63 Exterior temperature status

This service returns the exterior temperature in Celsius.

```
Request:      Inquire VehicleStatus ExteriorTemperature
Reply:        Report VehicleStatus ExteriorTemperature <temp>
```

where <temp> is an integer from –40..150 degrees C.

4.2.64 HVAC fan speed status and control

This service returns the speed of the HVAC blower fan as a percentage of full on. There is also a command to set a specific blower level.

```
Inquire:      Inquire VehicleStatus FanSpeed <zone>
Reply:        Report VehicleStatus FanSpeed <zone> <speed>
Command:      Command Vehicle FanSpeed <zone> <speed>
```

where <zone> = [driver, passenger, rear_left, rear_right] and <speed> is a number from 0 to 255 (0=off, 255=full speed).

4.2.65 HVAC fan mode status and control

This service returns the mode of the HVAC blower fan, or alternately commands a change in the mode.

```
Inquire:      Inquire VehicleStatus FanMode <zone>
Reply:        Report VehicleStatus FanMode <zone> <mode>
Command:      Command Vehicle FanMode <zone> <mode>
```

where <zone> = [driver, passenger, rear_left, rear_right] and <mode> is one of the following states: off, defog, defog and lower vent, lower vent, panel vent, panel and lower vent.

4.2.66 HVAC mix door position status and control

This service returns the relative position of the device that determines the temperature of the air exiting the HVAC ducts. It is also possible to command a change.

```
Inquire:      Inquire VehicleStatus MixDoor <zone>
Reply:        Report VehicleStatus MixDoor <zone> <position>
Command:      Command Vehicle MixDoor <position>
```

where <zone> = [driver, passenger, rear_left, rear_right] and <position> is 0..255 (255=100% or full hot).

4.2.67 Rear window defrost status and control

This service returns the binary state (on, off) of the Rear Window defog/Defrost. It is also possible to command a change.

```
Inquire:      Inquire VehicleStatus RearDefrost
Reply:        Report VehicleStatus RearDefrost <on, off>
Command:      Command Vehicle RearDefrost <on, off>
```

4.2.68 External mirror defrost status and control

This service returns the binary state (on, off) of the External Mirror Defog/Defrost. It is also possible to command a change.

```
Inquire:      Inquire VehicleStatus ExtMirrorDefrost
Reply:        Report VehicleStatus ExtMirrorDefrost <on, off>
Command:      Command Vehicle ExtMirrorDefrost <on, off>
```

4.2.69 Air conditioning status and control

This service returns the binary state (on, off) of the Air Conditioning Compressor (or appropriate technology) for creating cold air for the passenger compartment. It is also possible to command a change.

```
Inquire:      Inquire VehicleStatus ACstate
Reply:        Report VehicleStatus ACstate [on, off]
Command:      Command Vehicle ACstate [on, off]
```

4.2.70 Automatic HVAC control set temperature status and control

This service returns, or alternately commands, the temperature that the automatic HVAC is using as a set point.

```
Inquire:      Inquire VehicleStatus HVACSetTemp <zone>
Reply:        Report VehicleStatus HVACSetTemp <zone> <set_temp>
Command:      Command Vehicle HVACSetTemp <zone> <set_temp>
```

where <zone> = [driver, passenger, rear_left, rear_right] and <set_temp> is an integer representing the temperature in the range of 15 to 40 degrees C.

4.2.71 Seat heater/cooler status and control

This service returns, or alternately commands, the binary status of the seat heaters and/or coolers.

```
Inquire:      Inquire VehicleStatus SeatHVAC <position>
Reply:        Report VehicleStatus SeatHVAC <position> <state>
Command:      Command Vehicle SeatHVAC <position> <state>
```

where <position> = driver=1, middlefront=2, passenger=3, secondrowleft=4, secondrowmiddle=5, secondrowright=6, thirdrowleft=7, thirdrowmiddle=8, thirdrowright=9, fourthrowleft=10, fourthrowmiddle=11, fourthrowright=12, fifthrowleft=13, fifthrowmiddle=14, fifthrowright=15 and

<state>=off, heater on, cooler on.

4.2.72 Steering wheel heater status and control

This service returns, or alternately commands, the binary status of the steering wheel heater.

```
Inquire:        Inquire VehicleStatus SteeringWhlHeater
Reply:          Report VehicleStatus SteeringWhlHeater [on,off]
Command:        Command Vehicle SteeringWhlHeater [on,off]
```

4.2.73 Sun sensor status

This service returns the value of the vehicle sun intensity sensor.

```
Request:        Inquire VehicleStatus SunSensor
Reply:          Report VehicleStatus SunSensor <value>
```

where <value> is an integer from 0 to 16 measured as W/m2.

4.2.74 Rain sensor status

This service returns the measured value of rain intensity as a percentage of measurable intensity.

```
Request:        Inquire VehicleStatus RainSensor
Reply:          Report VehicleStatus RainSensor <intensity>
```

where <intensity> = 0..255 (255=100 percent).

Rain Sensor may also be requested as a periodic service (or subscription).

```
Request:        Update VehicleStatus RainSensor
Reply:          Report VehicleStatus RainSensor <intensity>
Request:        EndUpdate VehicleStatus RainSensor
```

The response is the same message as the response to a single Rain Sensor request, but it is repeated at the rate of one message every 1000 ms.

4.2.75 Service due status

This service returns information about when dealer service is next required for the vehicle. The returned fields indicate the odometer reading or date that service is due. It also includes an indication about the nature of the type of service due (routine or for detected malfunctions). For a more thorough description of the required service procedures, use the Service Description service.

```
Request:        Inquire VehicleStatus ServiceDue
Reply:          Report Vehicle VehicleStatus ServiceDue
                <month><day><year><odometer><type>
```

where:

<month> is an integer between 0 and 12 (0 indicates that the date is invalid);

<day> is an integer between 0 and 31 (0 indicates that the date is invalid);

<year> is and integer between 0 and 255 (0 indicates that the date is invalid);

NOTE The actual year that service is required is obtained by adding an offset of 2000.

<odometer> is the value in km when service is next due (0 indicates the odometer value is invalid);

<type> is [routine, malfunctions detected].

4.2.76 Service description status

This service returns a string describing the next dealer service required for the vehicle. The length of the string is limited to 1024 characters.

```
Request:      Inquire VehicleStatus ServiceDesc
Reply:        Report Vehicle VehicleStatus ServiceDesc <string>
```

4.2.77 Engine start disable status

This service returns the value of the engine disable mechanism, if one exists on the vehicle. If none exist, the return so indicates.

```
Request:      Inquire VehicleStatus EngineDisableService
Reply:        Report VehicleStatus EngineDisableService <EDStatus>
```

where <EDStatus> = [notimplemented, disabled, enabled].

It is also possible to request notification on a change of this parameter:

```
Request:      Notify if VehicleStatus EngineDisableServiceDelta
Reply:        Report VehicleStatus EngineDisableService <EDStatus>
Request:      EndNotify VehicleStatus EngineDisableServiceDelta
```

4.2.78 Disable engine start control

This service disables the engine start. There is no effect if the engine is running. The start enable status request should normally be issued before this command to determine if the feature is supported.

```
Command:      EngineStartDisable
Command:      EngineStartEnable
```

4.2.79 Performance status

This service returns the current performance setting.

```
Request:      Inquire VehicleStatus PerformanceStatus
Reply:        Report VehicleStatus PerformanceStatus <mode>
Command:      Vehicle SetPerformanceStatus <mode>
```

Where <mode> = normal, sport , or economy.

It is also possible to request notification on a change of this parameter:

```
Request:      Notify if VehicleStatus PerformanceStatusDelta
Reply:        Report VehicleStatus PerformanceStatus <mode>
Request:      EndNotify VehicleStatus PerformanceStatusDelta
```

4.2.80 Remote start control

This function allows the engine to be started remotely.

```
Command:        Vehicle StartEngine
```

4.2.81 Engine running status

This service returns true if the engine has been started and is running.

```
Request:        Inquire VehicleStatus EngineRunning
Reply:          Report VehicleStatus EngineRunning [true,false]
```

It is also possible to request notification on a change of this parameter:

```
Request:        Notify if VehicleStatus EngineRunningDelta
Reply:          Report Report VehicleStatus EngineRunning
                [true,false]
Request:        EndNotify VehicleStatus EngineRunningDelta
```

4.2.82 Engine speed status

This service returns the engine speed in RPM.

```
Request:        Inquire VehicleStatus EngineSpeed
Reply:          Report VehicleStatus EngineSpeed <rpm>
```

where <rpm> is an integer from 0 to 8000, with a resolution of 1 rpm per bit.

Engine speed may also be requested as a periodic service (or subscription).

```
Request:        Update VehicleStatus EngineSpeed
Reply:          Report VehicleStatus EngineSpeed <rpm>
Request:        EndUpdate VehicleStatus EngineSpeed
```

The response is the same message as the response to a single engine speed request, but it is repeated at a rate of one message every 100 ms.

4.2.83 Wheel speed status

This service returns the speed of each wheel in kph.

```
Request:        Inquire VehicleStatus WheelSpeed
Reply:          Report VehicleStatus WheelSpeed <LFWheelSpeed>
                <RFWheelSpeed> <LRWheelSpeed> <RRWheelSpeed>
```

where each <WheelSpeed> = 0..16535 and 1 bit = 0.01 kph.

Wheel speed may also be requested as a periodic service (or subscription).

```
Request:        Update VehicleStatus WheelSpeed
Reply:          Report Report VehicleStatus WheelSpeed
                <LFWheelSpeed> <RFWheelSpeed> <LRWheelSpeed>
                <RRWheelSpeed>
Request:        EndUpdate Vehicle WheelSpeed
```

The response is the same message as the response to a single wheel speed request, but it is repeated at a rate of one message every 250 ms.

4.2.84 Wheel rotations status

This service returns a number of wheel rotations as well as a timestamp. The timestamp is the value of the timer at the last update of the Wheel Rotation message.

```
Request:    Inquire VehicleStatus WheelRotations <wheel>
Reply:      Report VehicleStatus WheelRotations
            <wheel><rotations><timestamp>
```

where <rotations> = 0..16535 rotations and <timestamp> is 0..16535 ms.

Wheel rotations may also be requested as a periodic service (or subscription).

```
Request:    Update VehicleStatus WheelRotations
Reply:      Report VehicleStatus WheelRotations
            <wheel><rotations><timestamp>
Request:    EndUpdate VehicleStatus WheelRotations
```

The response is the same message as the response to a single wheel rotations request, but it is repeated at a rate of one message every 250 ms.

4.2.85 Engine coolant temperature status

This service returns the engine coolant temperature in degrees Celsius.

```
Request:    Inquire VehicleStatus EngineCoolantTemp
Reply:      Report VehicleStatus EngineCoolantTemp <temperature>
```

where <temperature> is an integer from –40 to 214 degrees C, with a resolution of 1 degree C per bit.

Engine coolant temperature may also be requested as a periodic service (or subscription).

```
Request:    Update VehicleStatus EngineCoolantTemp
Reply:      Report VehicleStatus EngineCoolantTemp <temperature>
Request:    EndUpdate VehicleStatus EngineCoolantTemp
```

The response is the same message as the response to a single engine coolant temperature request, but it is repeated at a rate of one message every 1000 ms.

4.2.86 Engine coolant pressure status

This service returns information regarding the engine coolant pressure.

```
Request:    Inquire VehicleStatus EngCoolantPressure
Reply:      Report VehicleStatus EngCoolantPressure
            <pressureLev>
```

where <pressureLev> = 0..255, (255=100% of the full measurable pressure).

4.2.87 Engine coolant level status

This service returns information regarding the engine coolant level.

```
Request:    Inquire VehicleStatus EngCoolantLevel
Reply:      Report VehicleStatus EngCoolantLevel <fluidLev>
```

where <fluidLev> = 0..255, (255=100% of full capacity).

Engine coolant level may also be requested as a periodic service (or subscription).

```
Request:      Update VehicleStatus EngCoolantLevel
Reply:        Report VehicleStatus EngCoolantLevel <fluidLev>
Request:      EndUpdate VehicleStatus EngCoolantLevel
```

The response is the same as the response to a single engine coolant level request, but is repeated at the rate of one message every 1000 ms.

4.2.88 Engine oil pressure status

This service returns information regarding the value of the oil pressure sensor.

```
Request:      Inquire VehicleStatus EngineOilPressure
Reply:        Report VehicleStatus EngineOilPressure <OPstatus>
```

where <OPstatus> = [low, normal].

It is also possible to request notification on a change of this parameter:

```
Request:      Notify if VehicleStatus EngineOilPressureDelta
Reply:        Report VehicleStatus EngineOilPressure <OPstatus>
Request:      EndNotify VehicleStatus EngineOilPressureDelta
```

4.2.89 Engine oil temperature status

This service returns information regarding the value of the oil temperature sensor.

```
Request:      Inquire VehicleStatus Engine OilTemperature
Reply:        Report VehicleStatus Engine OilTemperatureure
              <OTstatus>
```

where <OTstatus> = temp in 0.1 deg C.

Engine Oil Temperature may also be requested as a periodic service (or subscription).

```
Request:      Update VehicleStatus EngineOilTemperature
Reply:        Report VehicleStatus EngineOilTemperatureure
              <OTstatus>
Request:      EndUpdate VehicleStatus EngineOilTemperature
```

The response is the same message as the response to a single Engine Oil Temperature request, but it is repeated at the rate of one message every 1000 ms.

4.2.90 Engine oil level status

This service returns information regarding the value of the oil level sensor.

```
Request:      Inquire VehicleStatus EngineOilLevel
Reply:        Report VehicleStatus EngineOilLevel <OLstatus>
```

where <OLstatus> = 0..255, (255=100% oil level).

Engine oil level may also be requested as a periodic service (or subscription).

```
Request:      Update VehicleStatus EngineOilLevel
Reply:        Report VehicleStatus EngineOilLevel <OLstatus>
Request:      EndUpdate VehicleStatus EngineOilLevel
```

The response is the same message as the response to a single engine oil level request, but it is repeated at the rate of one message every 1000 ms.

4.2.91 Current gear status

This service returns the current gear.

```
Request:      Inquire VehicleStatus TransmissionCurrGear
Reply:        Report VehicleStatus TransmissionCurrGear <gear>
```

where <gear> = 0..31 (0=reverse, 31=neutral).

It is also possible to request notification on a change of this parameter:

```
Request:      Notify if VehicleStatus TransmissionCurrGearDelta
Reply:        Report VehicleStatus TransmissionCurrGear <gear>
Request:      EndNotify VehicleStatus TransmissionCurrGearDelta
```

4.2.92 PRNDL position status

This service returns information regarding the PRNDL position for automatic transmission and the current gear for manual. The transmission type and number of gears should be returned as part of the vehicle description data structure.

```
Request:      Inquire VehicleStatus PRNDL
Reply:        Report VehicleStatus PRNDL <state>
```

It is also possible to request notification on a change of this parameter:

```
Request:      Notify if VehicleStatus PRNDLDelta
Reply:        Report VehicleStatus PRNDL <state>
Request:      EndNotify VehicleStatus PRNDLDelta
```

4.2.93 Engine off time status

This service returns the amount of time that has elapsed since the last time the vehicle's engine was running. If the engine is running when the request for this information is processed, the vehicle interface will return an error.

```
Request:      Inquire VehicleStatus EngOffTime
Reply:        Report VehicleStatus EngOffTime <time>
```

where <time> = 0..65,535 minutes.

4.2.94 Brake fluid level status

This service returns information regarding the brake fluid level.

```
Request:      Inquire VehicleStatus BarkeFluidLevel
Reply:        Report VehicleStatus BarkeFluidLevel <fluidLev>
```

where <fluidLev> = [low, normal).

4.2.95 Reverse gear lights status and control

This service is to get or set the status of the lights that are activated when the vehicle is in a reverse gear.

```
Request:        Inquire VehicleStatus ReverseGearLightStatus
Reply:          Report VehicleStatus ReverseGearLightStatus [off,on}
Command:        Vehicle ReverseGearLightStatus [off,on}
```

4.2.96 Rear door inside handle disable status and control

This service is to get or set the status of the inside rear door handles to open the rear doors. This function is sometimes referred to as 'Child Locks' because it is intended to prevent young children from opening the rear doors of a vehicle.

```
Request:        Inquire VehicleStatus RearDoorInsideHandleDisable
Reply:          Report VehicleStatus RearDoorInsideHandleDisable
                [off,on}
Command:        Vehicle RearDoorInsideHandleDisable [off,on}
```

4.3 Security services

This group of services includes authentication protocols, encryption services (if required at the vehicle interface), access protocols, and non-repudiation protocols. There are four areas in which security must be provided:

— Anti-theft: The vehicle interface provides secure identification of the vehicle so that an installed device will not work in any other vehicle. (see use case SECU 1 – Disablement of stolen ISO 22902 components).

— Device authentication: The vehicle interface supports authentication protocols that support restriction of devices on the ISO 22902 compliant network to automaker approved devices. This authentication is required only at the automaker's discretion.

— Content protection: Suppliers of copyright material, such as downloaded video, can protect their content against unauthorized duplication. The AV/C Digital Interface Command Set for Secure Bus System specification, which is based on the DTCP protocol, supports content protection on 1394b.

— Driver / user authentication and personal information protection. This includes support for any authentication protocols used by external service providers if the information from those providers needs to pass through the vehicle gateway.

Within the ISO 22902 Specification, a secure VIN number only supports anti-theft protection for components at this time.

The protocol uses a private key / public key mechanism; the components request a secure VIN number, which the vehicle interface encodes together with other information, such as the current date / time using the vehicle's private key. The component then uses the vehicle's public key to decode the VIN number and checks it against the original VIN. It also checks the decoded date / time against the actual date / time. This is necessary to protect against reuse of a previously generated response from the vehicle.

Both vehicle public key and the VIN number must be programmed into the component when it is installed. (The component must never use a public key for the vehicle that was broadcast on the network.)

```
Request:        Inquire Security SecureVIN
Reply:          Report Security SecureVIN <VINCodeSequence>
```

The second form of authentication is the inverse of the first. In the anti-theft case, the vehicle authenticates itself to the component. In the component restriction case, the component must authenticate itself to the

vehicle. In this case, however, the result of the transaction should be a "session key" that the component uses to enable further transactions with the system.

Two authentication protocols in wide use are Kerberos, which is a symmetric (single) key algorithm, and the authentication protocol in DTCP, which is an asymmetric (public / private) key algorithm based on Diffie-Hellman key exchange.

In either case, there is an authentication server, which contains a list of the keys (public keys) of all approved devices. The authentication server is presumably, though not necessarily, implemented in the vehicle interface.

The message set for component authentication will depend on the protocol selected. The following example is based on one of the public key cryptography standards, this one based on the Diffie-Hellman algorithm (PKCS-3)[2]:

```
Request:        Inquire Security VehiclePublicKey
                <componentRegistrationNumber>
Reply:          Security VehiclePublicKey <KeyValue>
```

The component does not send its public key, since it is known to the vehicle gateway (if this is a registered device). After this exchange, the vehicle interface and the component use each other's public key values to generate a session key, using the Diffie-Hellman algorithm.

The secure VIN described above could also be made part of this exchange (by encoding it with the component's registered public key), but the two are kept separate so automakers can implement anti-theft, as described in SECU 1 Disablement of stolen AMI-C components without registering every protected component. (Some automakers will not wish to implement device authentication as described herein, but all must implement an anti-theft component for devices that require it.)

Content protection is provided on IEEE 1394b by the DTCP protocol. This is a proprietary protocol, which is available (in complete form) from the DTLA (Digital Transmission Licensing Administrator). Even though this protocol is proprietary, ISO 22902 may have to support it in order to allow video and audio data protected by the protocol to be used on 1394. MOST will also support DTCP, but it is expected that there will be modifications to the protocol for the MOST implementation.

Authentication requires a password, but this might be a surrogate for a mechanical authentication system, such as a smart card or a coded ignition key. If the device that generates the password (such as the smart card reader) is on the network, a protocol such as Kerberos should be used which does not require transmitting passwords across the network. Thus, the password is not included in the command below, assuming it will be transmitted implicitly.

```
Request: Inquire Security Authentication <userIdentification>
Reply:          Report Security Authentication <SessionKey>
```

where <SessionKey> has a specific value indicating that authentication has failed.

NOTE Session keys should expire after a fixed time. In Kerberos, the default lifetime of a session key is 8 hours, but in a vehicle, a more natural lifetime is from the time it is issued until key off. In this scenario, components would have to re-authenticate each time the ignition is turned on. This lifetime may be required for the anti-theft case, and is a natural choice for user authentication as well. (A service provider can, of course, require a user to re-authenticate more frequently, for example, on a per transaction basis.)

The fourth area, user authentication and personal information protection, requires both authentication mechanisms and authorization mechanisms. The number and nature of authorization levels to be implemented is a policy question that is left to the automakers. At this point, a placeholder message is defined within, by which the automaker can implement an authorization policy.

---

2) http://www.rssecurity.com/rsalabs/pkcs

```
Request:        Inquire Security Authorization <service>
                <SessionKey>
Reply:          Report Security Authorization [succeed, fail]
```

If the authorization succeeds, the requested service is enabled. The requestor's identification should be that used in a previous authentication, as described below.

4.4 Vehicle diagnostics group

This set of services includes all of those that relate to vehicle diagnostics, i.e., to components and systems on the vehicle side of the vehicle interface. It does not include diagnostics of the ISO 22902 system.

The diagnostics services can be divided into obtaining OBD codes, which are standardized by regulation, and manufacturer specific diagnostics. Two approaches are provided for the latter; one in which the manufacturer specific diagnostics are requested with a single command that is implemented by the vehicle interface, and one in which existing automaker diagnostic commands are encapsulated and sent through the vehicle interface to the appropriate component on the automaker's side. The latter approach allows existing test equipment to be used on the Multimedia bus by simply providing the appropriate encapsulation facility in the form of a bus adapter.

4.4.1 On-board diagnostic (OBD) codes status

This service returns OBD II codes. The specific syntax of an OBD code sequence is defined in ISO 22902-5, AMI-C Common Message Set.

The interpretation of the OBD code sequence depends on the regional variant of the OBD codes that is used by the vehicle (U.S., European). The client must examine the vehicle data structure to determine which variant of OBD codes are returned in the current vehicle.

```
Request:        Inquire VehicleDiagnostics OBDCodes
Reply:          Report VehicleDiagnostics OBDCodes <codeSequence>
```

4.4.2 Run diagnostics control

This command causes vehicle diagnostics to be run, if the vehicle supports this service. The conditions under which diagnostics can be run are vehicle specific. When the diagnostics execution is complete, the vehicle interface shall return a diagnostics reply to the node that issued the RunDiagnostics command. The data can then be obtained by the automaker diagnostics request described in the next section.

```
Command:        VehicleDiagnostics RunDiagnostics
Reply:          Report VehicleDiagnostics DiagnosticsComplete
```

4.4.3 Automaker specific diagnostics status

Each vehicle manufacturer may choose to make automaker specific diagnostic codes, which are available on the ISO 22902 compliant network. This message provides a way to transmit those codes. The format of the returned data is vehicle specific and is not specified by ISO 22902. Return of this information may or may not require the RunDiagnostics command described in the previous section to be executed in order to get current data.

```
Request:        Inquire VehicleDiagnostics OEMDiagnostics
Reply:          Report VehicleDiagnostics OEMDiagnostics <data>
```

4.5 AMI-C diagnostics

This set of messages relates to diagnostics for the ISO 22902 compliant system itself, as opposed to vehicle diagnostics, which are described in section 4.4 Vehicle diagnostics group.

```
Request:      Inquire VehicleCore SystemDiagnostics
Reply:        Report VehicleCore SystemDiagnostics
              <diagnosticCodeSequence>
Request:      Inquire VehicleCore NetworkDiagnostics <network>
Reply:        Report VehicleCore NetworkDiagnostics <network>
              <NWdiagnosticCodeSeq>
```

where <network> = [1394, MOST, BlueTooth].

In addition to messages which request diagnostic data from the vehicle interface, the following message causes the network from which it is sent to be logically decoupled from the vehicle interface to aid in isolating faults. In this mode, the only message that can be received is Connect Network, and no message can be sent from the vehicle interface. The second message undoes the effect of the first.

```
Command: VehicleCore DisconnectNetwork
Command: VehicleCore ConnectNetwork
```

5 Power management services

5.1 Power state

```
Request:      Inquire VehicleCore MultiMediaSystem_State
Reply:        ReportVehicleCore MultiMediaSystem_State <state>
```

where <state> = sleep, active, boot and shutdown.

It is also possible to request notification on a change of this parameter:

```
Request:      Notify if VehicleCore MultiMediaSystem_StateDelta
Reply:        Report VehicleCore MultiMediaSystem_State <state>
Request:      EndNotify VehicleCore MultiMediaSystem_StateDelta
```

5.2 System power mode

The functional power status of the vehicle is returned by this service.

```
Request:      Inquire VehicleCore PowerMode
Reply:        Report VehicleCore PowerMode <Mode>
```

It is also possible to request notification on a change of this parameter:

```
Request:      Notify if VehicleCore PowerModeDelta
Reply:        VehicleCore Status PowerModeDelta <newMode>
Request:      EndNotify VehicleCore PowerModeDelta
```

The parameters <Mode> and <newMode> can have the values off, active, low power or ulta low power. Low power mode is also known as sleep mode.

The parameters <Mode> and <newMode> can have the values off, active, low power or ulta low power. Low power mode is also known as sleep mode.

5.3 Boot sequence messages

The vehicle interface controls the power moding of the ISO 22902 network, and is responsible for booting the system. The boot sequence messages described here are issued after the network has reset and at the point that the vehicle interface determines that system boot is complete.

The boot sequence messages include the broadcast of the current date and time by the vehicle interface (see section 0, Command: VehicleCore EnterSleepMode Time / date), so that components can synchronize to the system time. This information also allows components the option to take appropriate action if the system boot does not occur within a specified period (reference the component anti-theft protocol as an example for the use of this facility).

```
Broadcast: VehicleCore InitiateBoot
Broadcast: VehicleCore EnterActiveState
```

In addition to the boot messages, there is a message to initiate wake up. This message is required by the use cases that call for a device to wake up the system (e.g., remote status inquiry).

```
Command: VehicleCore InitiateSystemWakeUp
```

5.4 Shutdown sequence messages

Corresponding to the wake-up message, there is a message for the system to enter sleep mode. As with the initiate wake up message, this is sent from a component to the vehicle interface, and may be used to perform remote shutdown of the system.

```
Broadcast:     VehicleCore InitiateShutdown
```

In addition to authenticating the device that sends the message, the vehicle interface may require other conditions to hold before acting on this message (vehicle not moving, ignition off, etc.).

```
Command:       VehicleCore EnterSleepMode
```

5.5 Time / date

These functions return and set the time and date. The time zone messages use a positive offset from GMT to indicate the time zone. The parameters to the time service are in local time, according to this offset. The clock implemented by this service shall have a resolution of one second.

```
Request:       Inquire VehicleCore Time
Reply:         Report VehicleCore Time <hours> <minutes> <seconds>
```

It is also possible to request notification on a change of this parameter:

```
Request:       Notify if VehicleCore TimeDelta
Reply:         Report VehicleCore Time <hours> <minutes> <seconds>
Request:       EndNotify VehicleCore TimeDelta
Request:       Inquire VehicleCore Date
Reply:         Report VehicleCore Date <year> <month> <day>
```

It is also possible to request notification on a change of this parameter:

```
Request:       Notify if VehicleCore DateDelta
Reply:         Report VehicleCore Date <year> <month> <day>
Request:       EndNotify VehicleCore DateDelta
```

```
Request:      Inquire VehicleCore TimeZone
Reply:        Report    VehicleCore TimeZone <offset>
```

It is also possible to request notification on a change of this parameter:

```
Request:      Notify if VehicleCore TimeZoneDelta
Reply:           Report    VehicleCore TimeZone <offset>
Request:      EndNotify VehicleCore TimeZoneDelta

Command:      VehicleCore SetTime <hours><minutes><seconds>
Command:      VehicleCore SetDate <year> <month> <day>
Command:      VehicleCore SetTimeZone <offset>
```

6 HMI services

6.1 AmicHMIRequest

This service allows an application to request the HMI Manager to present to the user audio or visual information using the vehicle audio and other HMI interfaces.

The HMI Manager uses the following messages to respond to the request: AmicIsHMIResourceAvailable, AmicHMIReleaseAudio, AmicHMISetupAudio, AmicHMIRequest, AmicHmiSend. See the Schema document for a detailed description of the message content.

An application or embedded device sends a request to the HMI Manager indicating that it needs to present some audio or visual information to the driver. It may include acquiring driver input. The message payload may optionally contain a VUIML string. The AmicHMIRequest XML message is also used when the device has finished with its audio need. It sends this message with the audio-off parameter.

A simple example where no XML is sent – only an audio channel is requested. The XML string is an optional parameter.

```
AmicHMIRequest(Source ID, audio – on, no pageContent,”Telephone ringing”,10
seconds’)
```

If automaker’s buttons are required, then an XML (VUIML) string must be contained in the parameter list describing the interaction and control information.

```
Request:      request HMI Manager for audio/visual presentation
              <SourceID> <AudioProperty> <VideoProperty>
              <PresentationProperties> <VUIMLContent>
Reply:        AmicHmiSend <RequestorID> <UserAction>
              <ResponseString> <ResponseString> <ParialResponse>
```

The possible values of the parameters are:

<SourceID > = [integer];

<AudioProperty > = [see Schema document] audio properties associated with the request;

<VideoProperty > = [see Schema document] video properties associated with the request;

<PresentationProperties > = [see Schema document] properties relating to priority and urgency;

<VUIMLContent> = [see Schema document] content to be presented to the driver.

6.2 AmicHmiSend

This XML Message is sent by the HMI Manager to an application that needs audio or other HMI interaction. It may be a response to the AmicHMIRequest Message.

After service, application or device sends a request to the HMI Service to present some audio or visual information to the driver, the HMI Service will respond with the user's request using the AMICHMISend message. It may also use this to initiate an interaction with an application.

```
Reply:        AmicHmiSend <RequestorID> <UserAction>
              <ResponseString> <ResponseString> <ParialResponse>
```

The possible values of the parameters are:

<RequestorID > = [integer] ID of the requestor;

<UserAction > = [String] User selection;

<ResponseString > = [String] – data provided by the user;

<ResponseString > = [String] – data provided by the user;

<ParialResponse > = [true, false] – user interrupted.

6.3 AmicHMISetupAudio

Once the HMI Manager decides to allow the device to gain access to the audio system, the HMI Manager sends the AmicHMISetupAudio message to the Audio Service telling it to setup a connection with requesting Audio source. The HMI Manager tells the Audio Service(maybe Network Bandwidth Manager) which audio sinks to connect the requesting audio source.

```
Request:      AmicHmiSetupAudio <SourceID> <AudioSink>
              <AudioProperties>
```

The possible values of the parameters are:

<SourceID> = [integer] ID of the requestor;

<AudioSink> = [see Schema for details] speakers that consume the audio;

<AudioProperties> = [see Schema for details].

6.4 AmicIsHMIResourceAvailable

After service, application or device sends a request to the HMI Manager to present some audio or visual information to the driver, the HMI Service may decide to not allow access to that resource. The HMI manager uses the AmicHMIResourceNotAvail message to inform the requestor that it cannot have the resource. If the requestor asked for both audio and visual resources, and one of them is not available, neither resource will be granted and the AmicHMIResourceNotAvail message will be sent to the requestor.

The HMI Manager decides when to send the AmicHMIResourceNotAvail message. It uses the expiration and priority information and resource availability to decide whether to respond promptly.

```
Request:      AmicIsHMIResourceAvailable <ResourceNotAvailable>
              <DoNotUseInternalHMI>
```

The possible values of the parameters are:

<ResourceNotAvailable > = [boolean] requested resource not available, either audio or a display;

<DoNotUseInternalHMI > = [boolean].

6.5 AmicHMIReleaseAudio

This XML message informs the audio service to release the audio connected to the passed in source.

```
Request:        AmicHMIReleaseAudio <SourceID>
```

The possible values of the parameters are:

<SourceID> = [integer] ID of the audio source.

7 Audio services

This section specifies the ISO 22902 audio services interface. The interface is independent of whether the audio devices providing the services are on the ISO 22902 side of the vehicle interface or on the vehicle side. It is also independent of whether the audio data is transmitted over an analog or a digital link. The purpose of this interface is to allow clients on the ISO 22902 side (devices or software applications) to access the audio services in a vehicle independent manner.

The interface description does not specify the policy by which audio resources are arbitrated between competing clients, nor the algorithms used to arbitrate audio resources.

7.1 Audio configuration

All vehicles providing audio services shall implement this service. The service provides the configuration of audio services, including volume control, mute capability, fade and balance capability.

```
Request:        Get Audio AmplifierConfiguration
Reply:          Audio AmplifierConfiguration
                <volume> <volumeLevels> <fade> <fadeRange> <balance>
                <balanceRange>
                <mute>
```

where <volume>, <fade> and <balance> have the values unsupported, status or status_control; <volumeLevels> gives the volume range as (low,high);<fadeRange> gives the fade range as (rear, front);<balanceRange> gives the range as left,right)and <mute> is a boolean that is TRUE if the amplifier has mute capability.

7.2 Audio volume state

This request returns both the on / off state for the mute function, and the current level setting for the volume control. This request returns an error if the <volume> parameter in the configuration response is UNSUPPORTED.

```
Request:        Get Audio VolumeState
Reply:          Audio VolumeState <muteOnOff> <level>
```

7.3 Mute / unmute

This service sets the mute function on or off. This request returns an error if the <mute> parameter in the configuration response is FALSE.

```
Command:        Audio Mute
Command:        Audio UnMute
```

This service sets the mute function on or off. This request returns an error if the <mute> parameter in the configuration response is FALSE.

```
Command:        Audio Mute
Command:        Audio UnMute
```

7.4 Set volume

This service sets the audio system volume level. This request returns an error if the <volume> parameter in the configuration response is UNSUPPORTED or STATUS.

```
Command:        Audio SetVolume <level>
```

where <level> = 0..255 (255=100% full volume).

7.5 Audio fade state

This service returns the current settings for fade (front / rear) of the audio output. The request returns an error if the <fade> parameter in the configuration response is UNSUPPORTED.

```
Request:        Get Audio FadeState
Reply:          Audio FadeState <fade>
```

where <fade> = 0 .. 31 (rear .. front).

7.6 Audio balance state

This service returns the current settings for balance (left / right) of the audio output. The request returns an error if the <fbalance> parameter in the configuration response is UNSUPPORTED.

```
Request:        Get Audio BalanceState
Reply:          Audio BalanceState <balance>
```

where <balance> = -15 .. 15 (left .. right).

7.7 Set fade

This service sets the fade and balance settings for the audio system. This request returns an error if the <fade_balabce> parameter in the configuration response is UNSUPPORTED or STATUS.

```
Command:        Audio SetFade <fade>
```

where <fade> = 0 .. 31 (rear .. front).

7.8 Set balance

This service sets the fade and balance settings for the audio system. This request returns an error if the <fade_balabce> parameter in the configuration response is UNSUPPORTED or STATUS.

```
Command:        Audio SetBalance <balance>
```

where <balance> = -15 .. 15 (left .. right).

7.9 Equalizer configuration

All vehicles implementing audio services shall provide this service, whether or not an equalizer is present. This service gets the equalizer including the number of bands and the frequency range for each. If no equalizer is present, the number of bands is returned as zero.

```
Request:        Get Audio EqualizerConfiguration
Reply:          Audio EqualizerState <nrBands>
                <frequencyRangeSequence>
```

where <nrBands> is an integer that is zero if no equalizer is present and <frequencyRangeSequence> is a sequence of pairs of the form <low,high> measured in Hz.

7.10 Equalizer state

This service gets the equalizer settings for each of the frequency divisions. If the number of bands returned by the configuration request above is zero, this request returns a "not present" error. Otherwise the number of levels returned is equal to the number of bands returned by the configuration request.

```
Request:        Get Audio EqualizerState
Reply:          Audio EqualizerState <levelSequence>
```

7.11 Equalizer

This service sets the equalizer setting for each of the frequency divisions. If the number of bands returned by the configuration request above is zero, this request returns a "not present" error. Otherwise the number of levels must be equal to the number of bands returned by the configuration request.

```
Command:        Audio SetEqualizer <levelSequence>
```

7.12 Audio channel configuration

This service shall be provided by all vehicles that implement audio services. It defines the audio channels that are available, both the total number and the types of channels. There are a fixed set of types identified by intended functions. The actual characteristics of each channel type are vehicle specific. PHONE is intended for voice, with a possibly restricted volume range. WARNING is intended for driver warning messages. It may support only a fixed volume level. TUNER and CD are intended to be two classes of music channel. NORMAL is whatever the vehicle manufacturer defines as the default audio characteristics. It may be the same as one of the other types or it may be unique. Type NORMAL shall be provided. All other types are optional.

```
Request:        Get Audio ChannelConfiguration
Reply:          Audio ChannelConfiguration <nrChanels> <types>
```

where <types> - Sequence of <type>

and <type> = [normal, phone, warning, tuner, cd].

7.13 Allocate audio channel

This service allocates an audio channel of the specified type. Type is a designation of the intended use of the channel that can be used by the amplifier or receiver to set the characteristics of the output, such as volume, mono/stereo, etc. The specific characteristics associated with a given type depend on the vehicle.

```
Request:        Audio GetAudioChannel <type>
Reply:          Audio AudioChannel <channel number>
Command:        Audio ReleaseAudioChannel <channel number>
```

where <type> = [normal, phone, warning, tuner, cd].