OVMS

Open Vehicle Monitoring System

www.openvehicles.com

OVMS Protocol
Guide v2.1.3 (18th February 2013)

History

v2.1.3	18th February 2013	Add support for owner sync
v2.1.2	20th December 2012	Add support for command #6 (charge alert)
v2.1.1	16th December 2012	Add support for historical records
v1.5.1	5th November 2012	Conversion to new format

Change Log

v2.1.3

- Add support for owner sync

v2.1.2

- Add command #6 (charge alert)

v2.1.1

- Add “H” Historical Data Update message
- Add command 31 (Request historical data summary)
- Add command 32 (Request historical data records)

Welcome

The OVMS (Open Vehicle Monitoring System) team is a group of enthusiasts who are developing a means to remotely communicate with our cars, and are having fun while doing it.

The OVMS module is a low-cost hardware device that you install in your car simply by installing a SIM card, connecting the module to your car’s Diagnostic port connector, and positioning a cellular antenna. Once connected, the OVMS module enables remote control and monitoring of your car.

WELCOME 3

THE OVMS PROTOCOL 6

TERMS 6
ON STARTUP 6
ENCRYPTION PROTECTION SCHEME 0X30 6
AUTO PROVISIONING 8
Auto Provisioning Protection Scheme 0x30 8
BACKWARDS COMPATIBILITY 10
CAR <-> SERVER <-> APP 10
Ping message 0x41 "A" 11
Ping Acknowledgement message 0x61 "a" 11
Command message 0x43 "C" 11
Command response 0x63 "c" 11
Car Environment message 0x44 "D" 12
Paranoid-mode encrypted message 0x45 "E" 13
Car firmware message 0x46 "F" 14
Server firmware message 0x66 "f" 14
Car group subscription message 0x47 "G" 15
Car group update message 0x67 "g" 15
Historical Data update message 0x48 "H" 16
Push notification message 0x50 "P" 17
Push notification subscription 0x70 "p" 17
Server -> Server Record message 0x52 "R" 18
Server -> Server Message Replication message 0x72 "r" 18
Car state message 0x53 "S" 19
Car update time message 0x53 "T" 20
Car location message 0x4C "L" 20
Car Capabilities message 0x56 “V” 20
Car TPMS message 0x57 "W" 21
Peer connection message 0x5A "Z" 22
COMMANDS AND EXPECTED RESPONSES 23
1 - Request feature list 23
2 - Set feature 23
3 - Request parameter list 24

4 - Set parameter 24
5 - Reboot 25
6 – Charge Alert.. 25
10 - Set Charge Mode.. 25
11 - Start Charge 25
12 - Stop Charge.. 25
15 - Set Charge Current.. 26
16 - Set Charge Mode and Current.. 26
17 - Set Charge Timer Mode and Start Time 26
18 - Wakeup car.. 26
19 - Wakeup temperature subsystem 26
20 - Lock Car 27
21 - Activate Valet Mode.. 27
22 - Unlock Car.. 27
23 - Deactivate Value Mode... 27
24 - Home Link... 28
30 - Request GPRS utilization data 28
31 - Request historical data summary 29
32 - Request historical data records 29
40 - Send SMS.. 30
41 - Send MMI/USSD Codes 30
49 - Send raw AT Command 30

The OVMS Protocol

Terms

- Server: the OVMS server
- Car: the OVMS car module
- App: a mobile app talking to the OVMS server

On startup

Caller sends welcome message to the server:

- For cars:

  MP-C <protection scheme> <car token> <car digest> <car id>

- For apps:

  MP-A <protection scheme> <app token> <app digest><car id>

- For servers:

  MP-S <protection scheme> <server token> <server digest><car id>

Server responds welcome message to the caller:

MP-S <protection scheme> <server token> <server digest>

Encryption Protection Scheme 0x30

This scheme is based on using shared secrets, hmac digest for authentication and encryption key negotiation, with RC4 stream cipher and base64 encoding.

Upon startup, both the server and callers generate random tokens (encoded as textual characters). Each party then hmac-md5s the token with the shared secret to create a digest, base64 encoded.

Note that the server will only issue it's welcome message after receiving and validating the caller's welcome message.

Validation of the welcome message is performed by:

1. Checking the received token to ensure that it is different from its own token, and aborting the connection if the same.

2. Hmac-md5s the received token with the shared secret and comparing the result to the received digest.

3. I the digest match, then the partner had authenticated itself (proven it knows the shared secret, or has listened to a previous conversation).

4. If the digests don't match, then abort the connection as the partner doesn't agree with the shared secret.

Once the partner has been authenticated:

1. Create a new hmac-md5 based on the client and server tokens concatenated, with the shared secret.

2. Use the new hmac digest as the key for symmetric rc4 encryption.

3. Generate, and discard, 1024 bytes of cipher text.

From this point on, messages are rc4 encrypted and base64 encoded before transmission. Lines are terminated with CR+LF.

Auto Provisioning

A caller can perform auto-provisioning at any time (authenticated or not). However, only one auto-provision can be performed for each connection.

Auto-Provisioning relies on two secrets known both to the server and client. The first is usually the VIN of the vehicle and the second is usually the ICCID of the SIM card in the vehicle module. The reason these two are chosen is that they can be auto-determined by the vehicle module, but also clearly seen by the user (for entry into the server).

The mechanism works by the client module first determining its VIN and ICCID secrets, then connecting to the server and sending a AP-C message to the server proving its VIN. The server will then lookup the auto-provisioning record, and reply with that to the client (via a AP-S or AP-X message).

The auto-provisioning record itself is platform dependent, but will typically be an ordered space separated list of parameter values. For OVMS hardware, and OVMS.X PIC firmware, these are merely parameters #0, #1, #2, etc, to be stored in the car module.

Auto Provisioning Protection Scheme 0x30

- For cars:

  AP-C <protection scheme> <apkey>

- For servers:

  AP-S <protection scheme> <server token> <server digest> <provisioning>

  AP-X

When the provisioning record is created at the server, a random token is generated (encoded as textual characters). The server then hmac-md5s this token with the shared secret (usually the ICCID known by the server) to create a digest, base64 encoded. Using this hmac digest, the server generates and discards 1024 bytes of cipher text. The server then rc4 encrypts and base64 encodes the provisioning information and stores its token, digest and encoded provision record ready for the client to request.

The car knows its VIN and ICCID. With this information, it makes a connection to the OVMS service on the server, and provides the VIN as the <apkey> in a AP-C message to the server.

The server will ensure that it only responds to one AP-C message for any one connection. Once responded, all subsequent AP-C requests will always be replied with a AP-X message.

Upon receiving the AP-C message, the server looks up any provisioning records it has for the given <apkey>. If it has none, it replies with an AP-X message.

If the server does find a matching provisioning record, it replies with an AP-S message sending the previously saved server token, digest and encoded provisioning record to the car.

If the car receives an AP-X message, it knows that auto-provisioning was not successful.

If the car receives an AP-S message, it can first validate the server authenticity. By producing its own hmac-md5 of the server token and secret ICCID, the car can validate the server-provided digest is as expected. If this validation step does not succeed, the car should abort the auto-provisioning.

If acceptable, the car can decrypt the provisioning record by first generating and discarding 1024 bytes of cipher text, then decoding the provisioning record provided by the server.

Backwards Compatibility

Typically, comma-separated lists are used to transmit parameter. Applications, Servers and the Car firmware should in general ignore extra parameters not expected. In this way, the protocol messages can be extended by adding extra parameters, without breaking old Apps/Cars that don't expect the new parameters.

Similarly, unrecognized messages should due ignored. Unrecognised commands in the "C" (command) message should be responded to with a generic "unrecognized" response (in the "c" (command response) messages).

Car <-> Server <-> App

After discarding CR+LF line termination, and base64 decoding, the following protocol messages are defined.

```
<message> ::= <magic> <version> <space> <protmsg>

<magic> ::= MP-
<version> ::= 1 byte version number - this protocol is 0x30
<space> ::= ' ' (ascii 0x20)

<protmsg> ::= <servertocar> | <cartoserver> | <servertoapp> | <apptoserver>
<servertocar> ::- "S" <payload>
<cartoserver> ::= "C" <payload>
<servertoapp> ::= "s" <payload>
<apptoserver> ::= "c" <payload>

<payload> ::= <code> <data>
<code> ::= 1 byte instruction code
<data> ::= N bytes data (dependent on instruction code)
```

Ping message 0x41 "A"

This message may be sent by any party, to test the link. The expected response is a 0x61 ping acknowledgement. There is no expected payload to this message, an any given can be discarded.

Ping Acknowledgement message 0x61 "a"

This message is sent in response to a 0x41 ping message. There is no expected payload to this message, an any given can be discarded.

Command message 0x43 "C"

This message is sent <apptoserver> then <servertocar> and carries a command to be executed on the car. The message would normally be paranoid-encrypted.

<data> is a comma-separated list of:

- command (a command code 0..65535)
- parameters (dependent on the command code)

For further information on command codes and parameters, see the command section below.

Command response 0x63 "c"

This message is sent <cartoserver> then <servertoapp> and carries the response to a command executed on the car. The message would normally be paranoid-encrypted.

<data> is a comma-separated list of:

- command (a command code 0..65535)
- result (0=ok, 1=failed, 2=unsupported, 3=unimplemented)
- parameters (dependent on the command code and result)

For result=0, the parameters depend on the command being responded to (see the command section below for further information).

For result=1, the parameter is a textual string describing the fault.

For result=2 or 3, the parameter is not used.

Car Environment message 0x44 "D"

This message is sent <cartoserver> "C", or <servertoapp> "s", and transmits the environment settings of the vehicle.

<data> is comma-separated list of:

- Door state #1
  bit0 = Left Door (open=1/closed=0)
  bit1 = Right Door (open=1/closed=0)
  bit2 = Charge port (open=1/closed=0)
  bit3 = Pilot present (true=1/false=0) (always 1 on my 2.5)
  bit4 = Charging (true=1/false=0)
  bit5 = always 1
  bit6 = Hand brake applied (true=1/false=0)
  bit7 = Car ON (true=1/false=0)
- Door state #2
  bit3 = Car Locked (locked=1/unlocked=0)
  bit4 = Valet Mode (active=1/inactive=0)
  bit6 = Bonnet (open=1/closed=0)
  bit7 = Trunk (open=1/closed=0)
- Lock/Unlock state
  4 = car is locked
  5 = car is unlocked
- Temperature of the PEM (celcius)
- Temperature of the Motor (celcius)
- Temperature of the Battery (celcius)
- Car trip meter (in 1/10th of a distance unit)
- Car odometer (in 1/10th of a distance unit)
- Car speed (in distance units per hour)
- Car parking timer (0 for not parked, or number of seconds car parked for)
- Ambient Temperature (in Celcius)
- Door state #3
  bit1 = cooling pump (on=1/off=0)
- Stale PEM,Motor,Battery temps indicator (-1=none, 0=stale, >0 ok)
- Stale ambient temp indicator (-1=none, 0=stale, >0 ok)
- Vehicle 12V line voltage
- Door State #4
  bit2 = alarm sounds (on=1/off=0)

Paranoid-mode encrypted message 0x45 "E"

This message is sent for any of the four message <protmsg> types, and represents an encrypted transmission that the server should just relay (or is relaying) without being able to interpret it. The encryption is based on a shared secret, between the car and the apps, to which the server is not privy.

<data> is:

- <paranoidtoken> l <paranoidcode>
- <paranoidtoken> ::= "T" <ptoken>
- <paranoidcode> ::= "M" <code> <data>

In the case of <paranoidtoken>, the <ptoken> is a random token that represent the encryption key. It can only be sent <cartoserver> or <servertoapp>. Upon receiving this token, the server discards all previously stored paranoid messages, sends it on to all connected apps, and then stores the token. Every time an app connects, the server also sends this token to the app.

In the case of <paranoidcode>, the <code> is a sub-message code, and can be any of the codes listed in this document (except for "A", "a" and "E"). The <data> is the corresponding encrypted payload message. The encryption is performed on the <data> by:

1. Create a new hmac-md5 based on the <ptoken>, with the shared secret.
2. Use the new hmac digest as the key for symmetric rc4 encryption.
3. Generate, and discard, 1024 bytes of cipher text.

and the data is base64 encoded. Upon receiving a paranoid message from the car, the server forwards it on the all connected apps, and then stores the message. Every time an app connects, the server sends all such stored messages. Upon receiving a paranoid message from an app, if the car is connected, the server merely forwards it on to the car, otherwise discarding it.

Car firmware message 0x46 "F"

This message is sent <cartoserver> "C", or <servertoapp> "s", and transmits the firmware versions of the vehicle.

<data> is comma-separated list of:

- Car firmware version
- Car VIN
- GSM signal level
- Write-enabled firmware (0=read-only, 1=write-enabled)
- Car type (TR=Tesla Roadster, others may follow)
- GSM lock

Server firmware message 0x66 "f"

This message is sent <servertocar> "S", or <servertoapp> "s", and transmits the firmware versions of the server.

<data> is comma-separated list of:

- Server firmware version

Car group subscription message 0x47 "G"

This message is sent <apptoserver> "A", and requests subscription to the specified group.

<data> is comma-separated list of:

- Group name

Car group update message 0x67 "g"

This message is sent <cartoserver> "C", or <servertoapp> "s", and transmits a group location message for the vehicle.

<data> is comma-separated list of:

- Vehicle ID (only <servertoapp>, not sent <cartoserver>)
- Group name
- Car SOC
- Car Speed
- Car direction
- Car altitude
- Car GPS lock (0=nogps, 1=goodgps)
- Stale GPS indicator (-1=none, 0=stale, >0 ok)
- Car latitude
- Car longitude

Historical Data update message 0x48 "H"

This message is sent <cartoserver> "C, and transmits a historical data message for storage on the server.

<data> is comma-separated list of:

- type (unique storage class identification type)
- recordnumber (integer record number)
- lifetime (in seconds)
- data (a blob of data to be dealt with as the application requires)

The lifetime is specified in seconds, and indicates to the server the minimum time the vehicle expects the server to retain the historical data for. Consideration should be made as to server storage and bandwidth requirements.

The type is composed of <vehicletype> - <class> - <property>

<Vehicletype> is the usual vehicle type, or "*" to indicate generic storage suitable for all vehicles.

<Class> is one of:

- PWR (power)
- ENG (engine)
- TRX (transmission)
- CHS (chassis)
- BDY (body)
- ELC (electrics)
- SAF (safety)
- SEC (security)
- CMF (comfort)
- ENT (entertainment)
- COM (communications)
- X** (unclassified and experimental, with ** replaced with 2 digits code)

<Property> is a property code, which the vehicle decides.

The server will timestamp the incoming historical records, and will set an expiry date of timestamp + <lifetime> seconds. The server will endeavor to retain the records for that time period, but may expend data earlier if necessary.

Push notification message 0x50 "P"

This message is sent <cartoserver> "C", or <servertoapp> "s". When used by the car, it requests the server to send a textual push notification alert message to all apps registered for this car. The <data> is 1 byte alert type followed by N bytes of textual message. The server will use this message to send the notification to any connected apps, and can also send via external mobile frameworks for unconnected apps.

Push notification subscription 0x70 "p"

This message is sent <apptoserver> A". It is used by app to register for push notifications, and is normally at the start of a connection. The <data> is made up of:

<appid>,<pushtype>,<pushkeytype>{,<vehicleid>,<netpass>,<pushkeyvalue>}

The server will verify the credentials for each vehicle, and store the required notification information.

Server -> Server Record message 0x52 "R"

This message is sent <servertoserver> "S", and transmits an update to synchronized database table records.

Sub-type RV (Vehicle record):

<data> is comma-separated list of:

- Vehicleid
- Owner
- Carpass
- v_server
- deleted
- changed

Sub-type RO (Owner record):

<data> is comma-separated list of:

- Ownerid
- OwnerName
- OwnerMail
- PasswordHash
- OwnerStatus
- deleted
- changed

Server -> Server Message Replication message 0x72 "r"

This message is sent <servertoserver> "S", and replicates a message for a particular car.

<data> is comma-separated list of:

- vehicleid
- message code
- message data

Car state message 0x53 "S"

This message is sent <cartoserver> "C", or <servertoapp> "s", and transmits the last known status of the vehicle.

<data> is comma-separated list of:

- SOC
- Units ("M" for miles, "K" for kilometers)
- Line voltage
- Charge current (amps)
- Charge state (charging, topoff, done, prepare, heating, stopped)
- Charge mode (standard, storage, range, performance)
- Ideal range
- Estimated range
- Charge limit (amps)
- Charge duration (minutes)
- Charger B4 byte (tba)
- Charge KWH consumed
- Charge sub-state
- Charge state (as a number)
- Charge mode (as a number)
- Charge Timer mode (0=onplugin, 1=timer)
- Charge Timer start time
- Charge timer stale (-1=none, 0=stale, >0 ok)

Car update time message 0x53 "T"

This message is sent <servertoapp> "s", and transmits the last known update time of the vehicle.

<data> is the number of seconds since the car last sent an update message

Car location message 0x4C "L"

This message is sent <cartoserver> "C" and transmits the last known location of the vehicle.

<data> is comma-separated list of:

- Latitude
- Longitude
- Car direction
- Car altitude
- Car GPS lock (0=nogps, 1=goodgps)
- Stale GPS indicator (-1=none, 0=stale, >0 ok)

Car Capabilities message 0x56 “V”

This message is sent <cartoserver> “C”, or <servertoapp> “s”, and transmits the vehicle capabilities. It was introduced with v2 of the protocol.

<data is comma-separated list of vehicle capabilities of the form:

- C<cmd> indicates vehicle support command <cmd>
- C<cmdL>-<cmdH> indicates vehicle will support all commands in the specified range

Car TPMS message 0x57 "W"

This message is sent <cartoserver> "C", or <servertoapp> "s", and transmits the last known TPMS values of the vehicle.

<data> is comma-separated list of:

- front-right wheel pressure (psi)
- front-right wheel temperature (celcius)
- rear-right wheel pressure (psi)
- rear-right wheel temperature (celcius)
- front-left wheel pressure (psi)
- front-left wheel temperature (celcius)
- rear-left wheel pressure (psi)
- rear-left wheel temperature (celcius)
- Stale TPMS indicator (-1=none, 0=stale, >0 ok)

Peer connection message 0x5A "Z"

This message is sent <servertocar> or <servertoapp> to indicate the connection status of the peer (car for <servertoapp>, app for <servertocar>). It indicates how many peers are currently connected. It is suggested that the car should use this to immediately report on, and to increase the report frequency of, status - in the case that one or more Apps are connected and watching the car.

<data> is:

- Number of peers connected, expressed as a decimal string

Commands and Expected Responses

For message types "C" and "c", the following commands and responses are expected to be supported:

1 - Request feature list

Command parameters are unused and ignored by the car.

Response is a sequence of individual messages with each message containing the following parameters:

- feature number
- maximum number of features
- feature value

Registered features are:

- 0: Digital SPEEDO (experimental)
- 8: Location STREAM mode (consumes more bandwidth)
- 9: Minimum SOC
- 15: CAN bus can write-enabled

Note that features 0 through 7 are 'volatile' and will be lost (reset to zero value) if the power is lost to the car module, or module is reprogrammed. These features are considered extremely experimental and potentially dangerous.

Features 8 through 15 are 'permanent' and will be stored as parameters 23 through 31. These features are considered more stable, but optional.

2 - Set feature

Command parameters are:

- feature number to set
- value to set

Response parameters are unused, and will merely indicate the success or not of the result.

3 - Request parameter list

Command parameters are unused and ignored by the car.

Response is a sequence of individual messages with each message containing the following parameters:

- parameter number
- maximum number of parameters
- parameter value

Registered parameters are:

- 0: Registered telephone number
- 1: Registration Password
- 2: Miles / Kilometer flag
- 3: Notification method list
- 4: Server IP
- 5: GPRS APN
- 6: GPRS User
- 7: GPRS Password
- 8: Vehicle ID
- 9: Network Password
- 10: Paranoid Password

Note that some parameters (24 through 31) are tied directly to the features system (for permanent features) and are thus not directly maintained by the parameter system or shown by this command.

4 - Set parameter

Command parameters are:

- parameter number to set
- value to set

Response parameters are unused, and will merely indicate the success or not of the result.

5 - Reboot

Command parameters are unused and ignored by the car.

Response parameters are unused, and will merely indicate the success or not of the result. Shortly after sending the response, the module will reboot.

6 – Charge Alert

Command parameters are unused and ignored by the car.

Response parameters are unused, and will merely indicate the success or not of the result. Shortly after sending the response, the module will issue a charge alert.

10 - Set Charge Mode

Command parameters are:

- mode (0=standard, 1=storage,3=range,4=performance)

Response parameters are unused, and will merely indicate the success or not of the result.

11 - Start Charge

Command parameters are unused and ignored by the car.

Response parameters are unused, and will merely indicate the success or not of the result.

12 - Stop Charge

Command parameters are unused and ignored by the car.

Response parameters are unused, and will merely indicate the success or not of the result.

15 - Set Charge Current

Command parameters are:

- current (specified in Amps)

Response parameters are unused, and will merely indicate the success or not of the result.

16 - Set Charge Mode and Current

Command parameters are:

- mode (0=standard, 1=storage,3=range,4=performance)
- current (specified in Amps)

Response parameters are unused, and will merely indicate the success or not of the result.

17 - Set Charge Timer Mode and Start Time

Command parameters are:

- timermode (0=plugin, 1=timer)
- start time (0x059F for midnight GMT, 0x003B for 1am GMT, etc)

Response parameters are unused, and will merely indicate the success or not of the result.

18 - Wakeup car

Command parameters are unused and ignored by the car.

Response parameters are unused, and will merely indicate the success or not of the result.

19 - Wakeup temperature subsystem

Command parameters are unused and ignored by the car.

Response parameters are unused, and will merely indicate the success or not of the result.

20 - Lock Car

Command parameters are:

- pin (the car pin to use for locking)

Response parameters are unused, and will merely indicate the success or not of the result.

N.B. unlock/lock does not affect the immobilizer+alarm (when fitted)

21 - Activate Valet Mode

Command parameters are:

- pin (the car pin to activate valet mode)

Response parameters are unused, and will merely indicate the success or not of the result.

22 - Unlock Car

Command parameters are:

- pin (the car pin to use for unlocking)

Response parameters are unused, and will merely indicate the success or not of the result.

N.B. unlock/lock does not affect the immobilizer+alarm (when fitted)

23 - Deactivate Value Mode

Command parameters are:

- pin (the car pin to use for deactivating value mode)

Response parameters are unused, and will merely indicate the success or not of the result.

24 - Home Link

Command parameters are:

- button (home link button 0, 1 or 2)

Response parameters are unused, and will merely indicate the success or not of the result.

30 - Request GPRS utilization data

Command parameters are unused and ignored by the car.

Response is a sequence of individual messages with each message containing the following parameters:

- record number
- maximum number of records
- date
- car received bytes
- car transmitted bytes
- apps received bytes
- apps transmitted bytes

Note that this request is handled by the server, not the car, so must not be sent in paranoid mode. The response (from the server) will also not be sent in paranoid mode.

N.B. Dates (and GPRS utilization data) are in UTC.

31 - Request historical data summary

Command parameters are unused and ignored by the car.

Response is a sequence of individual messages with each message containing the following parameters:

- type number
- maximum number of types
- type value
- number of unique records (per type)
- total number of records (per type)
- storage usage (in bytes, per type)
- oldest data timestamp (per type)
- newest data timestamp (per type)

N.B. Timestamps are in UTC.

32 - Request historical data records

Command parameters are:

- type (the record type to retrieve)

Response is a sequence of individual messages with each message containing the following parameters:

- response record number
- maximum number of response records
- data record type
- data record timestamp
- data record number
- data record value

40 - Send SMS

Command parameters are:

- number (telephone number to send sms to)
- message (sms message to be sent)

Response parameters are unused, and will merely indicate the success or not of the submission (not delivery) of the SMS.

41 - Send MMI/USSD Codes

Command parameters are:

- USSD_CODE (the ussd code to send)

Response parameters are unused, and will merely indicate the success or not of the submission (not delivery) of the request.

49 - Send raw AT Command

Command parameters are:

- at (the AT command to send - including the AT prefix)

Response parameters are unused, and will merely indicate the success or not of the submission (not delivery) of the request.