OBD Now Terminal

OBD Now Terminal for Android devices is similar to programs such as Hyper Terminal or Tera Term for Windows computers. The major difference is OBD Now Terminal is already preconfigured to connect to any ELM327 or compatible OBDII Bluetooth scan tool. The user's only requirement is to select the Bluetooth scan tool they wish to connect to.

Once connected, the user can issue any ELM327 AT or ST (Scantool.net's alternative AT command set) command or hexadecimal OBDII command, by typing the command and tapping the Send key on the keyboard. The app will immediately respond with a response as can be seen in the screenshots. Multi line responses will automatically be formatted into each separate response, one per line.

The app also includes a help manual which explains some of the basic AT commands. The help manual is a pdf file viewable via Adobe Reader, Google's QuickOffice, Amazon Kindle or any other Android designed pdf viewer from the Help menu item. Please refer to the paragraphs at the end of the help file for links to comprehensive information about the ELM327 chip (and compatibles) and more detailed information on the OBDII specification.

Disclaimer:

This app is not a typical OBDII app which interprets the responses from your ECU(s) in human readable format. This app is designed for OBDII developers and or ELM327 enthusiasts who wish to observe the raw data responses from the ECU(s) of their test vehicles or ELM327 compatible simulators. OBD Now Terminal makes no attempt to interpret the responses returned from the ECU(s) as it assumes the user is already familiar with the responses and knows how to interpret the data in the responses. For those users who are new to OBDII and wish to learn more, we would suggest checking the links at the end of our help manual and our basic tutorial below.

Basic Tutorial

To confirm that you have a connection always first issue the ATZ command which will reset the ELM or ELM compatible chip to its default settings. If connected properly, OBD Now Terminal will reply with a response like the following

ELM327 v1.3

>

The actual text of the response will vary according the version of the chip in your scan tool.

To distinguish between commands and responses, responses from the scan tool are always in ***Bold Italic***, whereas commands display as normal text.

The “>” character on the line after the response indicates that the scan tool has finished responding and is ready for additional commands to be entered. This is known as the ELM prompt. Once you are more familiar with the responses you may wish to turn off the display of the prompt to fit more information onto the screen. It is controlled as a preference in the Settings menu. You may also notice that by default each command sent to the scan tool is echoed back on a separate line as part of the response. This is controlled by the command ATE0 which will turn off the echo and ATE1 which will return it to the default on setting.

Typically, an OBD scan tool user, using OBD Now Terminal would first send a Service $01 request to establish a list of supported pids for the vehicle.

Before you can issue any of the 01xx commands, you should first establish the protocol of your vehicle. The protocol, if known, can be set manually with the ATSP x command, where x is a number 1 through 9 or the letters A, B and C. The letters only apply to the Elm 327 1.2 version and above.

However first you need to establish what the protocol of the vehicle is, if you don’t already know it. The simplest way to establish the protocol is to use the command ATSP0. This number (0) is a special case, which will automatically search all protocols and only return when the correct protocol for the particular vehicle being tested has been established. It is used in conjunction with hexadecimal command 0100.

Tip: Hexadecimal commands can only contain the letters A-F and digits 0-9. Please do not confuse the number 0 with the letter O.

The following is the list of commands as discussed above that will display a list of supported pids in the range 0x01 through 0x20 for all ECU(s) of your vehicle.

ATZ

ATE0

ATSP0

0100

The following OBD Now Terminal screenshot shows the commands send and responses received from the scan tool.

A noticeable delay (3-5 secs) can be observed after typing 0100. This is caused by the scan tool doing an automatic protocol search of all the available protocols until it finds the correct protocol. Once found, it then automatically outputs the responses from the ECU(s).

The example in the screen shot is displaying the output from 3 different ECUs for pid values 0x01 through 0x20 for each ECU.

Figure 1 – OBD Now Terminal - Initialization on a Nexus 7 of a CAN vehicle.

We can now establish what the protocol is, using either of the following ELM AT commands ATDP or ATDPN.

ATDP describes the protocol in text form and ATDPN describes the protocol by number. The A in the A6 response indicates Automatic as we originally requested ATSP0 to automatically searching all protocols. 6 indicates the protocol number which is the number for ISO 15765-4 (CAN 11/500). For more information on protocols please refer to the links listed at the end of this document.

Now that we know the protocol we can avoid the use of ATSP0, we only need to specify ATSP6 before we issue the 0100 command. In fact, if we don't change vehicles and connect with another protocol, most ELM scan tools will remember the last used protocol, so even the ATSP command can be skipped.

We can then reduce the number of commands to the following

ATZ

ATE0

0100

If, however you swap your scan tool to another vehicle using a different protocol, you would have to use the original procedure to establish the correct protocol of the new vehicle.

Figure 2 - Shorter connection sequence - Nexus 7

What if my vehicle returns a NO DATA responses for the 0100 command?

If your vehicle returns a NO DATA response when you issue the command 0100 as described above, then your vehicle is simply not OBDII compliant and your scan tool and our software or any other OBDII compliant software is of no use with that vehicle. A vehicle can't be made compliant if it has not been built as an OBDII compliant vehicle. Please note that the presence of the 16 pin OBDII connector in the vehicle is meaningless as those or similar connectors were in use for factory scan tools long before the OBDII specification was established.

If your response returns "?" then you have most likely made a typo. A "?" response is returned when the ELM chip doesn't understand the command it has been sent.

There is the possibility that some cheap clone scan tools will return NO DATA, when they should return a list of pids, but it is our experience that this is very rare. The easiest way to establish if your scan tool is faulty is to check if it will connect to a known OBDII compliant vehicle. Test enough vehicles and it should become evident if your scan tool is faulty.

Logging

OBD Now Terminal can log every command (and typo..) and Ecu response for later study. The log files are stored in a folder called OBD Now Terminal. If you have also installed OBD Now, you will also find a folder called OBD Now containing OBD Now's diagnostic log files, so doesn't confuse the folder names. The log files use the same naming format as the OBD Now's diagnostic files. Each file name is created with date and time in the file name. The format is "Logddmmyyyy hrmmss.txt". A new file is created for each connection. Each file is automatically closed when you tap the Disconnect button or when you exit the app.

You will need a file manager to locate these files. We use and recommend AirDroid by Sand Studio, but any Android file manager will do. When using a file manager look for the folder name in the following format /storage/emulated/0/OBD Now Terminal. This is a folder that Android deems as External Storage. The zero in the folder name can be any number and is related to the number of users of device.

The log files can be viewed on your device with any HTML browser app, which most devices provide by default. If wish to view them on your computer, note that they will open in Window's Notepad, but the formatting will be incorrect. If you wish to view them on your desktop, copy the file to a convenient folder and then just drag the file into an open tab of any internet browser.

Logging can be turn on/off via the Settings menu.

OBDII Compliance Notes

OBDII compliant first started in the US in 1996 for passenger vehicles. It has been adopted by various countries since then on different dates. See the following link for the year when various countries regulated for OBDII compliance. http://elmelectronics.com/obdhelp.html

It has been observed that some vehicle manufacturers do turn off OBDII compliance in vehicles that are normally OBDII compliant, in countries that don't regulate for OBDII compliance.

How to build a list of supported Pids

Before you can do much with a scan tool, you need to establish a list of supported pids for the vehicle you are connected to. There is not much point sending an OBDII command to an ECU if that pid is not supported, for all you will get in return is a NO DATA response. The following tutorial explains how to build a list of supported pids.

For example, assume that the 0100 command returned the following

41 00 BF BF B9 93

The 4 bytes returned **"BF BF B9 93"** would then need to be translated from a master pid list table into their meaningful parameter ids and descriptors. Our app *OBD Now* makes this tedious and error prone calculation simple by our Display Supported Pids test.

The ISO – 15031-5 or SAE J1979 standard supports approximately 162 pids. However, if you wish to manually calculate the list of supported pids, the following tutorial takes you through the process so that you may understand what the automatic process does.

Typically, most users only ever enter 0100 as per the previous example, but to get a complete list of pids that a vehicle supports, it is often necessary to issue the following extra commands 0120, 0140, 0160, 0180 and 01A0. The question then arises how does one know if the last commands are required? To understand we need to study the results of the first command 0100.

As previously mentioned the return values are always 4 bytes. The information contained within these bytes is contained within each bit of the byte; therefore, we need to look at the binary representation of each byte.

Using the return value from the example **BF BF B9 93** we get the following results for each byte, using a scientific calculator, such as the Windows calculator. Or if you want to stay with your device, download from Google Play, TechCalc at the following link
https://play.google.com/store/apps/details?id=com.roamingsquirrel.android.calculator

Hex	Binary
BF	1011 1111
BF	1011 1111
B9	1011 1001
93	1001 0011

The above table breaks down each of the 4 bytes into their binary values – the binary values have been pasted from the Window's calculator – see Figure 3. Each byte is represented by 8 bits, so in total we have 32 bits. Each of those 32 bits is either 1 or 0. If the value is 1, then the bit is turned on and if 0 then it is considered turned off. If a bit is turned on then that bit position represents a single pid as being present. So, all we need to do now is to calculate which bits are turned on and we therefore have our list of pids.

Looking at the first line (which contains bits 0-8), reading from left to right, we can see all bits are turned on except for the second bit. Therefore, the pids that are supported are pid 0x01, 0x03 – 0x08. Note there is no entry for Pid 0x02.

The second byte (which contains bits 9-16) just happens to have the same value as the first byte and you can see the missing bit is 0xA (10 in decimal). The remaining bytes (17-24 and 25-32) show different patterns, but the principle is the same, if a bit is turned on, it represents a pid, if it is not on, then there is no pid at that position.

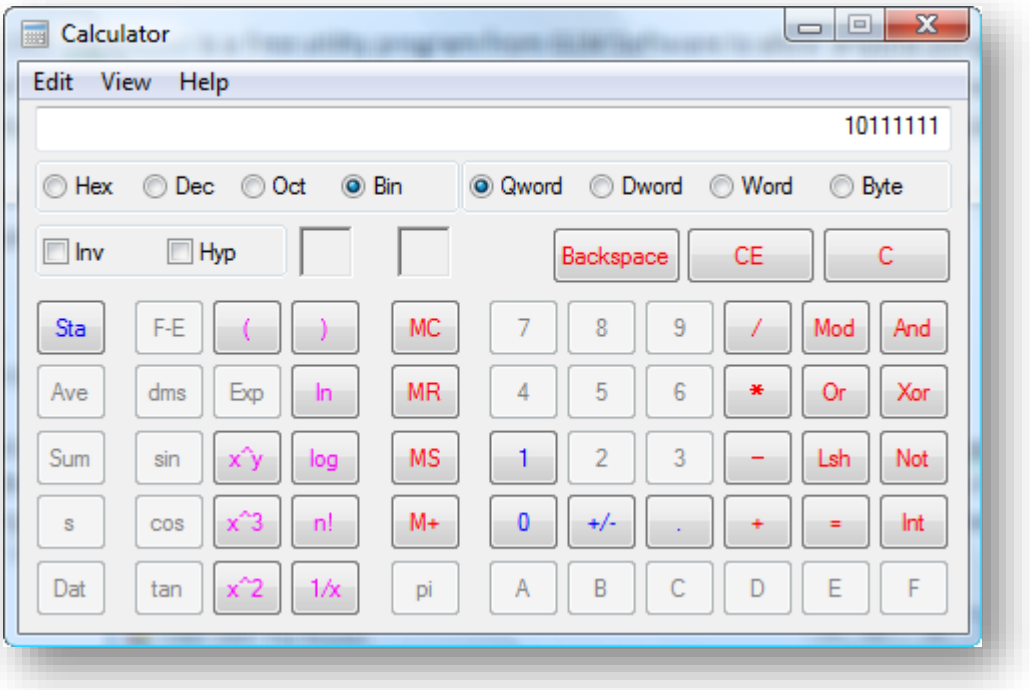


Figure 3 – Example conversion of 0xBF to binary

So we can summarise that this vehicle supports the following pids 0x01, 0x03 – 0x08 from the 1st byte, 0x09, 0x0B – 0x10 from the 2nd byte, 0x11, 0x13-0x15 and 0x18 from the 3rd byte and 0x19, 0x1C and 0x1F - 0x20 from the 4th byte.

The last bit, bit 32 or 0x20 is a special bit. It does not actually represent a pid, even if it is on. If it is on, it tells us that this ECU supports more pids in the next 32-bit range. If it is set, then we know we also need to issue 0120 to learn which pids are supported in the 0x21 – 0x40 range. If it is off, then we know that this vehicle does not support any pids in the higher ranges. This bit is normally off for the early model protocols, but will usually be on for late model protocols such as CAN as the later protocols usually support more pids.

If bit 32 is on, then we must issue 0120 and in our example, we get back the following

0120	Request
41 20 80 07 E0 19	Response

We use exactly the same procedure as we used for the response to the 0100 command. The only difference now is that we are dealing with bits 0x21 – 0x40 as compared to the 0100 command where we were dealing with bits 0x01 – 0x20.

Hex	Binary
80	1000 0000
07	0000 0111
E0	1110 0000
19	0001 1001

Again, the last bit 0x40, is an indicator bit, if it is on, we know we then need to issue 0140. If it is off, we are done and our list of pids is complete. In the above example, it is on, so we do have to issue 0140.

0140
41 40 FE D0 00 00

I'll leave this one as an exercise, if you wish to use the Window's calculator to polish your hexadecimal skills

More than one ECU

Just when you thought this OBD stuff was getting easier, I need to point out that many vehicles respond to the above commands with multiple responses. The reason for the multiple responses is that more than one ECU is responding. The extra ECU is usually the transmission ECU. The responses can look very similar, so working out the list of supported ECU's suddenly gets more complicated.

The following is a typical response to a 0100 command from a vehicle with more than one ECU.

0100
41 00 BE 1B 30 13
41 00 88 18 00 10
41 00 00 08 00 10

So how do we know, which response is from which ECU? The answer from the above information is that we don't. There is just no way of telling, other than it is usual to have more supported pids from the engine ECU than the transmission ECU.

The only way to really tell is to issue the Elm command ATH1, which will turn header information on. When you issue an ATZ (reset) command to an Elm chip, by default headers are turned *off.* Therefore, to obtain header information you must issue ATH1, to turn headers off again issue ATH0. With headers on we get additional information. The example below shows the responses from three 3 ECUs

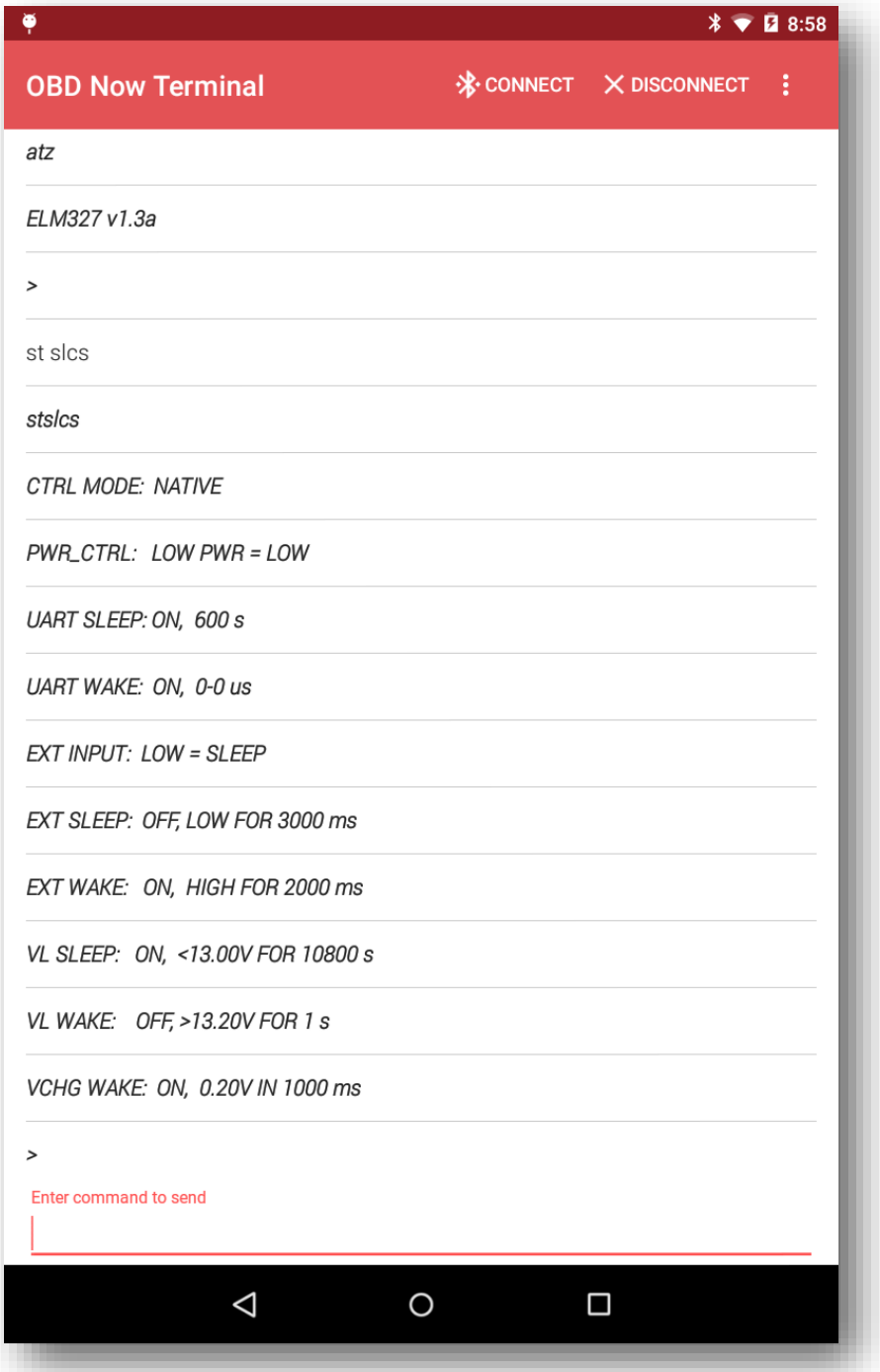
Figure 4 - 3 ECUs responding with Headers off and then on.

For this vehicle (actually a simulator) the ECUs are 7E8, 7E9 and 7EA.

How to check an OBDLink scan tool's sleep mode

Checking the sleep mode of an OBDLink scan tool is a common requirement often requested by the support staff on the Scantool.net forums. Some apps interfere with this setting, which prevents the OBD Link from going to sleep and thus potentially flattening the battery of the vehicle.

OBD Now Terminal makes this extremely easy to check as the next screen shot demonstrates.



Links to the data sheets for the 3 versions of the ELM327 chip from ELM Electronics.

ELM327 v1.3a

http://www.elmelectronics.com/DSheets/ELM327DSF.pdf

ELM327 v1.4b

http://www.elmelectronics.com/DSheets/ELM327DSH.pdf

ELM327 v2.1

http://www.elmelectronics.com/DSheets/ELM327DS.pdf

Links to the data sheet from Scantool.net, the manufacturers of the popular OBDLink MX Bluetooth scan tool.

STN1100 Family Reference and Programming Manual

General information on OBDII Pid information from Wikipedia.

http://en.wikipedia.org/wiki/OBD-II_PIDs

The SAE OBDII standard SAE 1979, also available as ISO 15031-5.

SAE 1979

http://standards.sae.org/j1979_201408/

ISO 15031-5

http://infostore.saiglobal.com/store/Details.aspx?productID=1814156

GLM Software
Surrey Hills, Victoria, Australia
Support: support@glmsoftware.com