Why You Probably Can Not Make ‘Custom OBD PIDs’ Work for a Recent Car, and What You Can Do About It.

Summary

Some manufacturers of recent car models use very effective methods to prevent owners from accessing information about their own cars, even though that information is present in the car network connected to the On Board Diagnostics port. This probably also helps to prevent accidental or malicious damage to the vehicle. Some manufacturers licence the information needed to access the data, to companies that sell OBD tools. This can provide convenient and inexpensive access for owners, while remaining secure if the tool includes features such as Bluetooth pairing only for a short time after pressing a physical button on the tool. This article describes what is possible, using as an example the OBDLink Android app and MX+ interface to the OBD port in a 2019 (gen5) Toyota rav4 hybrid.

OBD, PIDs?

In recent cars, there will be a port for On Board Diagnostics (OBD-II currently), including ‘legislated’ data items (Parameter IDs or PIDs) that vary according to jurisdiction. The legislated PIDs relate to air pollution, and they will be a subset of those referred to as SAE PIDs, after the standard that first defined them (*https://en.wikipedia.org/wiki/OBD-II_PIDs*).

The OBD port is usually accessible under the dash near the steering wheel. Data are accessed by plugging in a device called an OBD reader or interface, which acts like a modem between the car and a computer program (which may be a smartphone app). The OBD standards allow for various communication protocols, and evolve over time (*https://obdstation.com/obd2-protocols/*). These protocols may use data from different pins in the OBD port, and some of them allow bidirectional communication. Most interface devices use a command set named after an early manufacturer of chips for the purpose: ELM electronics. This command set has grown with new versions (*www.elmelectronics.com/wp-content/uploads/2020/05/ELM327DSL.pdf*). OBD standards use 10 ‘Modes’ (hex 01-0A); eg Mode 03 relates to Diagnostic Trouble Codes (DTCs) that are set when a malfunction causes vehicle emissions to exceed legislated thresholds. But many other Modes can exist.

The Challenge in New Cars

Over time, standards have evolved to keep pace with the increasingly sophisticated computer communications and controls in modern vehicles. Most cars now use hundreds or thousands of sensors that communicate through a common set of cables (bus) in a Controller Area Network (CAN) that includes the OBD port. Each sensor sends information into the bus as an identifier (PID) and data bits. The pattern of bits has to be translated into human-readable form through an equation that varies between sensors. Typically there are multiple networks including computer modules called Electronic Control Units (ECUs) that forward necessary information. Some (compound) PIDs give results calculated using inputs from several sensors. OBD standards allow for Headers to address up to eight ECUs, but more exist for other purposes. The wiring for legislated OBD can be used for other purposes that do not interfere. Car makers are not constrained to known ELM commands for corporate uses, which often include vehicle control, and diagnostics through the OBD port. Naturally, some technically-oriented owners would like to access some of this information about their cars!

There are several ‘standard’ CAN protocols (*https://en.wikipedia.org/wiki/CAN_bus*). CAN11 has 2047 module addresses (Headers) each of which can use multiple Mode and PID combinations for data from sensors. Toyota certainly uses Modes 21, 22 (and maybe others). Mode 21 uses hex xx=256 variant PIDs. Mode 22 allows hex xxxx=65536 variant PIDs. Mode 22 Headers can

be long (like DA18F1). It is very hard to find by chance the Header, Mode and PID for any non-legislated piece of information on the bus.

Manufacturers can implement multiple CAN protocols and/or busses in one car. For non-legislated data they can use proprietary (non-OBD) data formats (eg 11 bit CAN extended addresses, as distinct from 29 bit extended CAN addresses; see *p61 of the ELM command pdf*) and/or keys, passwords, gateway circuits or other means of obfuscation to restrict access (and to prevent accidental or malicious damage to the vehicle).

Non-critical car functions may use cheaper and slower networks, commonly LIN busses. Some of these interface with the CAN network to the OBD port, but some do not. Even if they connect to the OBD port, non-legislated messages may not follow the OBD standard. A module may only respond to requests to its physical address / Header (not the 7DF 'functional' or broadcast Header) or *vice versa*. Even for legislated PIDs, manufacturers are not obliged to allow physical addressing of requests. Message length is always 8 bytes DLC=08 in OBD, but not necessarily in other CAN. Response address is request +8h in OBD, but not necessarily in other CAN. There is so much data on a CAN bus that it is essential to know how to set up appropriate filters for responses. Even if this pattern is deduced, what equation should be used to translate bits for that PID (from sensor voltage) into human-readable results?

Most car manufacturers (OEMs) don't freely release their CAN information. Some go to great lengths to hide PIDs from car owners, particularly in the most recent models. So these PIDs have to be licensed, or discovered by 'reverse engineering' for each model. Depending on the complexity of the bus and the information sought, 'reverse engineering' can take months. People who complete the exercise typically don't freely disclose the results.

The Difficult 'Solution'

There are tools (like an OBD Y-cable and data logger) to record OBD transmissions. Some programs attempt to find (sniff) modules, modes and PIDs available on a connected CAN bus, but even those found may not be supported by a particular OBD app and/or reader. Then the equations must be deduced for translation. All of this is well into hacking nerd territory.

The Easy Solution

It is much easier to use OBD software with OEM PIDs (eg OBDLink) or more expensive OEM or licensed 'pro' tools (eg Toyota Techstream, Launch Pro, AutoEnginuity). Of course, you need to check that the tool works with the car(s) of interest, and that it provides access to all of the OEM PIDs of interest.

OCTtech say "Generic adapters ... do not support any manufacturer-specific protocols". That is probably true, but many manufacturers use 'standard' protocols with OEM-specific PIDs. The cheapest tools generally provide access only to SAE PIDs (if they work at all), but some that are only slightly more expensive provide access to many OEM PIDs.

OBDLink: Costs

After trying a cheaper (Panlong) interface with free software (Torque, Car Scanner ELM, OBD Now Terminal) without success for OEM PIDs in a 2019 (gen5) Toyota rav4 HV, I bought the OBDLink MX+. In June 2021, OBDLink LX and MX+ devices included:

- OBDLink app (= Fusion for Android which costs US$7). An iOS version is also available.
- OBDWiz program (= TouchScan for Windows US$30) but only for SAE PIDs without paid add-ons.
- No fee software and firmware updates. The software is being actively improved: check for the latest version.
- Secure (press-button) and fast Bluetooth.

MX+ only also included:

- No fee unlimited OEM-specific data add-ons (OEM-Specific Enhanced Diagnostics Support) for OBDLink (LX or Fusion equivalents = US$10 each by make and year).

The price was US$80-100 (AU$170) for the MX+ vs US$60 (AU$130) for the LX. These devices have equivalent Bluetooth capability, so unless you need access to the proprietary Ford (MS-CAN) & GM (SW-CAN) vehicle networks, or use iOs, you would need to use 2-4 OEM-specific downloads to justify the price difference. There are cheaper OBDLink-USB interfaces for those who want speed, without the convenience of wireless.

OBDLink: Features

OBDLink use their own chips (STN****; , which also support the ELM327 AT command set *https://www.scantool.net/scantool/downloads/98/stn1100-frpm.pdf*). MX+ tested as ELM v1.4b+ (so it may support some later commands). OBDLink claims advantages of speed, protocol range and firmware upgrade over ELM327. OEM-specific data add-ons provide 'OEM live parameters' for Ford, Nissan, Mazda and Toyota; but only 'OEM DTCs' for others including Holden, Honda, Hyundai etc. Check before purchase: *www.obdlink.com/wp-content/uploads/2019/01/app_support.pdf*

Those who want to access data from PIDs in recent Toyota models will find OBDLink devices to be a bargain! One limitation is that only data from the selected OEM network will show in real time on a dashboard; but most data of interest is in 'Network A' which loads by default with the SAE PIDs. OBDLink does not have a function that reads all PIDs. You need to use Home> Diagnostics> PID Values> Select PIDs, to generate and examine a list of those that interest you (in the order they are selected). This is sensible to avoid the many (thousands of) PIDs not of interest at any given time. The easiest way to identify PIDs that may be interesting is to use the ⋮ menu to 'select all' PIDs in a module.

I found the OBDLink Android app to be great (after a few set-up frustrations mentioned below, but you can work around these). In my experience, the Windows program was less impressive, and I did not try iOS. You can not use the car 'infotainment' display (unless you want to purchase a replacement head unit and/or Mirroring Kit: expensive options that may not work well with all built-in car functions). But you can use an Android pad of a size that fits neatly in the console below the car radio (in the 2019 rav4 at least). It is safest not to look at either of these places while driving!

OBDLink (MX+): Set-up

See: *https://obdsoftware.force.com/s/article/getting-started-with-obd-fusion*

Using OBDLink (v 5.17) for Android with MX+: (1) Product registration failed repeatedly (cancel out of it); (2) The 'Getting Started' method for OEM add-ons failed, but I eventually bumbled through to fetch them (no other instructions were provided, see the highlighted tip below); (3) The automatic firmware update failed (very scary) but after rebooting it was eventually fetched; (4) Sleep worked only sometimes. Pull out the device to ensure no battery drain. (5) The connection process was flaky after several restarts of the car, requiring multiple attempts or restart of the adapter, the app and/or Bluetooth. Pull the MX+ device out of the OBD port when the vehicle is turned off, and re-insert it when needed. (This remained the case under v 5.19). (6) The app was very slow on each start (or foreground after 'Exit') to reload vehicle data (but there are thousands of OEM PIDs to install). You can work around these frustrations during set-up. The result is worth the effort!

I don't know why this device is advertised not to work with hybrid vehicles, but it worked fine with my gen5 rav4 HV (Australian GX AWD). In contrast, the popular free Hybrid Assistant app did not support this Australian rav4 model (not OBDLink's fault).

First connection:

1. Plug the OBDLink device into the car ODB port.
   Green (power) and blue (bluetooth) LEDs will light.
2. Turn car Ignition On / Engine Off (key or switch position).
3. In the phone or tablet, open Android 'Settings', and enable 'Bluetooth'.
4. Press the 'Connect' button on the OBDLink device. The blue LED will blink rapidly, and allow for connection within 2 minutes.
5. Open the phone 'Bluetooth settings' menu and tap 'OBDLink device' to pair the device.
6. Tap 'OK' in the phone 'Bluetooth pairing request' dialog box.
7. Start the OBDLink app (OBDLink tells you to get it from the Google Play Store). In Settings> Preferences> Communications, tap 'Bluetooth' (it may be selected automatically).
8. Tap 'Connect' on the OBDLink app home screen.
9. Choose 'OBDLink device'
10. The app will establish a connection with the OBDLink device, and detect which OBD-II protocol your vehicle uses. Once the app establishes a connection with the OBDLink device, the 'BT' LED will display a solid blue light. If you use only one vehicle, it may later be faster to specify the appropriate protocol in Settings, rather than always use 'automatic'.
11. When you connect to any vehicle, OBDLink should allow download of any enhanced OEM add-on (for car-specific PIDs). If you got the OBDLink app from the Google Play Store (as instructed by OBDLink company), you have to be logged in to Google Play Store to obtain data add-ons for OBDLink! Those so inclined can download and install an APK (search the web) to install or update the program. I have not tested the effect on download of add-ons.
12. Then you can select from the various networks in the car, and scan for Supported PIDs (under Vehicle Editor). To eliminate some of the unsupported PIDs, you will need to rescan on each new phone / tablet while connected (eg Distance to Empty disappears under Calculated PIDS, though the rav4 trip computer has it). When you are not connected to the vehicle, all PIDs are listed.

OBDLink: Dashboard Tips

See: *https://obdsoftware.force.com/s/article/getting-started-with-obd-fusion*

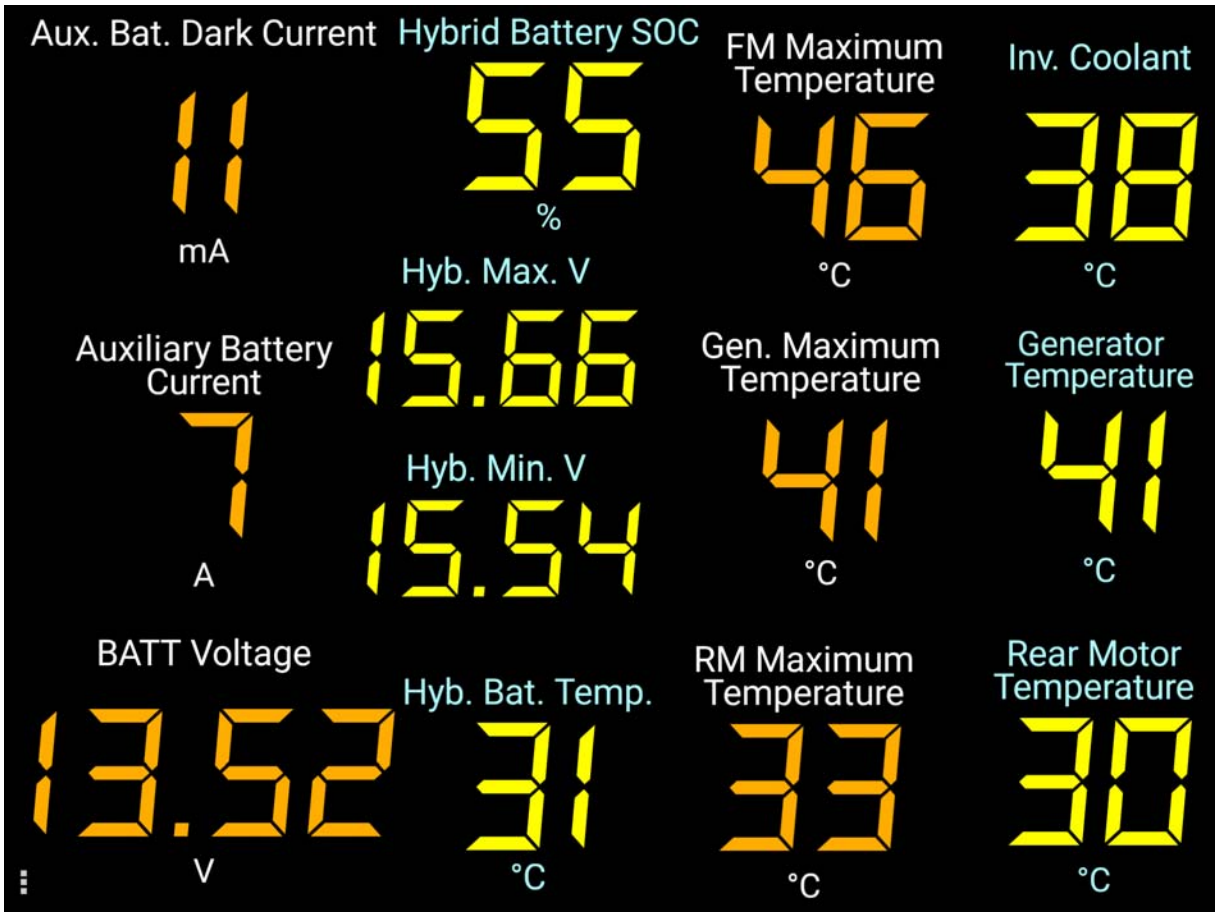
The OBDLink dashboard is very versatile for display of data from any PID in an analogue or digital gauge that makes sense to humans. You can name gauges, create custom bezels, adjust font and size, gauge size, colour, etc. Use 'Range' to set coloured arcs in analogue gauges. From gauge-style> ⋮ you can 'save template'. When you start a new gauge, you can select from stock and custom gauge templates. To change an existing gauge you can 'change display type' and/or use gauge-style> ⋮ > load template. Home> settings> ⋮ 'export settings' includes 'dashboards'. In v 5.19 there is a grid for guidance when you 'drag and move' a gauge. Unfortunately, completed gauges can not be moved or copied between dashboards; and saved templates are not in the exported settings. But you can save a template on the new device, if it is used as a gauge before export from the old device.

Gauge names auto-wrap (with some bugs) and do not permit added line breaks. Sometimes the desired effect can be achieved with spaces, or with added characters such as ~.

The screen grabs below give an indication of what is possible. Many elegant dashboard designs for OBDLink (Fusion) are posted in car-specific internet forums.


Engine Oil Pressure
136
kPa
Transaxle Oil Temp.
25
°C
1308
RPM
Fuel rate
3.56
l/hr
Start
Logging
0
km/h
LFP-T
28
°C
LFP-EMF
14.14
V
LFP-C
38
A


Aux EMF
14.1
V
Aux Temp
28
°C
LFP-C
38
A
0
km/h
Fan %
0
Inv. Coolant
30
Boost. Conv.
27
Gen. Inv.
24
FM Inv.
28
RM Inv.
24
H 2
26



A few niggles :

1. The word data is plural of datum (OBDLink should say “Data are availiable’, not “Data is available”).

2. The **degree symbol** is missing from temperature units in OBDLink v 5.17 (requiring a work-around using an overlaid empty and transparent gauge). Fixed in v 5.19!

This symbol is available in an Android mobile device keypad:

(i) Press the ‘?123’ key on the Android keypad to select the numeric keypad.
(ii) Then press ‘=\<‘ or ‘1/2’ key (depending on the Android device) to see more symbols.
(iii) The degree symbol will be available on the Android keypad.

3. For navigators, what OBDLink calls **GPS Bearing** is not a Bearing. Rather it is a Heading, Track or Course (terms commonly used as synonyms in navigation of pedestrians and cars):

- **Bearing** is the direction between any two points (usually the direction from a navigator to a nominated object; such as another vehicle, a landmark or a waypoint). Sometimes relative bearings are used (eg the angle from a ship’s long axis), but in navigation the term usually refers to compass bearings. GPS programs may give bearings to a waypoint as magnetic (compass) or true (angle from the meridian connecting N&S poles) - check!
- **Heading** is the direction the vehicle is pointed. For vehicles like aircraft and ships that may be affected by cross-currents, this can differ from (present) Track or Course, but for cars they are generally the same (except for those who like to 'drift' around corners).
- **Track** is the actual direction of vehicle travel across the surface of the earth. The term is commonly used in reference to a past track, or a path travelled. It is unambiguous to state a direction in which an object is tracking.
- **Course** is the direction in which a vessel is being steered (in shipping), or the intended path of travel to a destination (in aviation). Sometimes these are distinguished as present course vs original course. For a ship in calm weather with a steady helm, course=heading.

OBDLink: Toyota Rav4 HV-specific Tips

Many SAE PIDS (for turbocharged or diesel engines) do not apply. Some that seem to apply (eg Hybrid Charging State) do not function. Manufacturers are not obliged to use all SAE PIDs.

Of the Toyota PIDs, OBDLink (in its forum) says 'Toyota provides the list' and 'the app data used to pull the PIDs is provided directly by Toyota'. The PIDs have names but no descriptions or #IDs. (Toyota Techstream is said to give detailed PID descriptions in the data list manager, but pirated versions may not do this. It does not reveal #IDs, and live data are shown from only one ECU at a time).

Many of the Toyota PIDs are repeated across modules. Some are constant (eg Auxiliary Battery Status of Full Charge), some do not function (eg Aux Battery Capacity ..., Delta SOC), some are uninterpretable (eg DC/DC Converter Signal Voltage, Hybrid Battery Check …), and/or unitless (eg Aux Battery Integrated Thermal Load), and some give conflicting readings (eg various Temperature readings after the car has been garaged overnight). Many abbrevs are hard to guess. In some cases, several sensors may read almost the same thing (eg BATT & Battery vs Aux Battery Voltage; Engine Coolant vs Coolant Temperature; various Speeds). In other cases, the same name is used for very different data streams (eg Hybrid Battery Current). Without 'inside knowledge' you can only guess at the differences. Some things are strangely missing (eg Front Motor Temperature, Car GPS & Trip Meter). Some 'normal ranges' are apparently proprietary (eg Inverter Temperatures). Some designations vary between other Toyota documents and the PIDs (eg Generator = MG1, [Front] Motor = MG2, and Rear Motor = MGR). But many Toyota PIDs reward exploration.

Nav vehicle speed averages ~2% faster than SAE speed (probably from wheel sensors): the % varies more at low speeds. Nav speed does not seem to use GPS (it works in tunnels). It may be used in nav functions such as time to destination. Cruise control and speed alerts use the speed displayed in the MID ('combination meter' in Toyota PIDs?). This averages ~3% faster than SAE speed but the readings can be very different. Where the car has a gauge on display, it is wise to use the car gauge rather than the reading from an OBD PID.

The gen5 rav4 NiMH 'traction' battery is specified as 244.8v comprising 34 modules (each 6*1.2v=7.2v). There are Toyota PIDs for Hybrid 'Blocks' 1-11, with 3-8 being 4 modules in series and the others 2 modules in series (evident in voltages, which read about 7.7v per module). Hybrid battery voltages should be tested under load (during both charging and discharging while driving) and can be interpreted as in the table (from *https://www.youtube.com/watch?v=k0Mm8XpRVOg*).

Module? voltage spread (V)	**Remaining battery efficiency (%)**
0.2	100
0.4	75
0.7	50
0.95	25
1.2	0

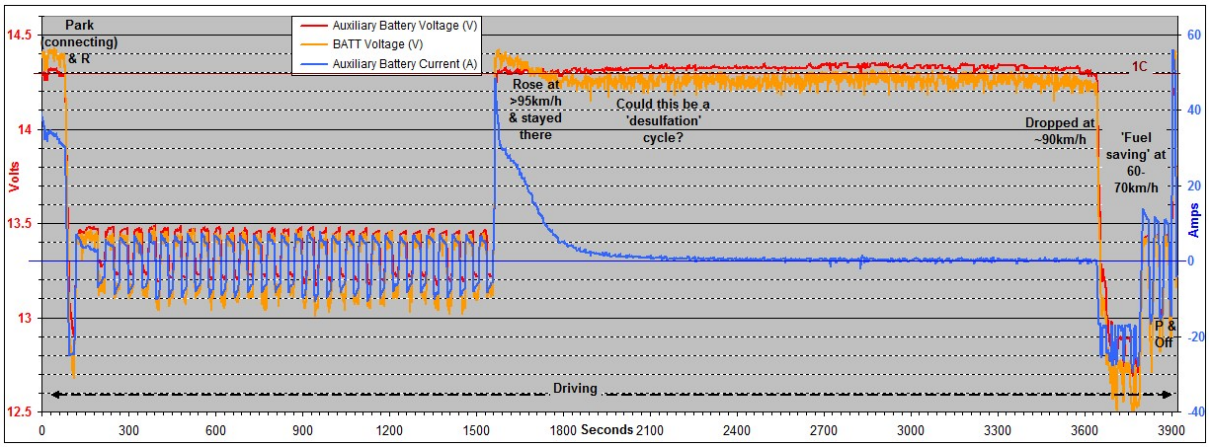
The inverter/converter is said (in the Toyota *Dismantling Manual* for rescuers) to step up to 650v, 3-phase AC for the motors; but there are Toyota PIDs for separate motor inverters.

OBDLink: Logging Tips

See: *https://obdsoftware.force.com/s/article/configuring-logging*

Automatic logging sometimes failed to start on my phone with v 5.17. But it was 100% reliable when I used an Android tablet that was not simultaneously performing other functions (like monitoring a wifi or phone network). Depending on the number of sensors and logging frequency it took ~10min for a file 'in progress' to appear in the list, to confirm that logging was working. There are useful new features in v 5.19: (i) a press-button gauge can be used to start / stop manual logging; (ii) a start/ stop logging button is on the log files page. Much better! I still need to remove / replace the MX+ device to reconnect, but that is a separate issue from logging.

For those who wish to graph from the exported (csv) log files, there is a problem that Excel sees some time values from OBDLink as text, so it graphs based on categories instead of X (time) values. This problem is reasonably common with other data sources. It requires a workaround (such as paste special> multiply x1) in Excel.



OBDLink: User-defined PIDs

See: *https://obdsoftware.force.com/s/article/user-defined-pids*

Headers, Modes and PIDs are not revealed for OEM-specific data add-ons. Therefore, I can not see how to use them in creation of used-defined PIDs (eg combine them in a formula for a calculated PID). There are unexplained settings (like 'Category') for User PIDs in OBDLink.

It appears that ScanGauge 'reverse engineers' to find some manufacturer-specific PIDs. The 'code' used in early X-gauges is known (*http://www.giffordautomotive.com/images/XGaugeCoding.pdf*). After *http://www.cleanmpg.com/community/index.php?threads/12851/page-2*, the method to 'translate' into PIDs was described more fully in the Torque forum (*https://torque-bhp.com/forums/?wpforumaction=viewtopic&t=352.0*), then widely copied (and sometimes elaborated or mistaken). However, X-gauge coding has 'evolved'. For example, what does C mean at the start of RXF? The most recent X-gauges work only with new ScanGauge firmware, and the changed methods are not freely disclosed.

I have tried but never succeeded in setting up a non-SAE User PID for a 2019 rav4, even 'translating' from X-gauges. But unless you want to create some kind of compound PID, why bother with 'User PIDs' when 'OEM PIDs' are available? Car manufacturers do not generally change PIDs every year in the same series or generation of each car model. So if PIDs for the most recent year are not yet available, it is worth trying to manually set an earlier year of the same model, and download the OEM data for that year in OBDLink.

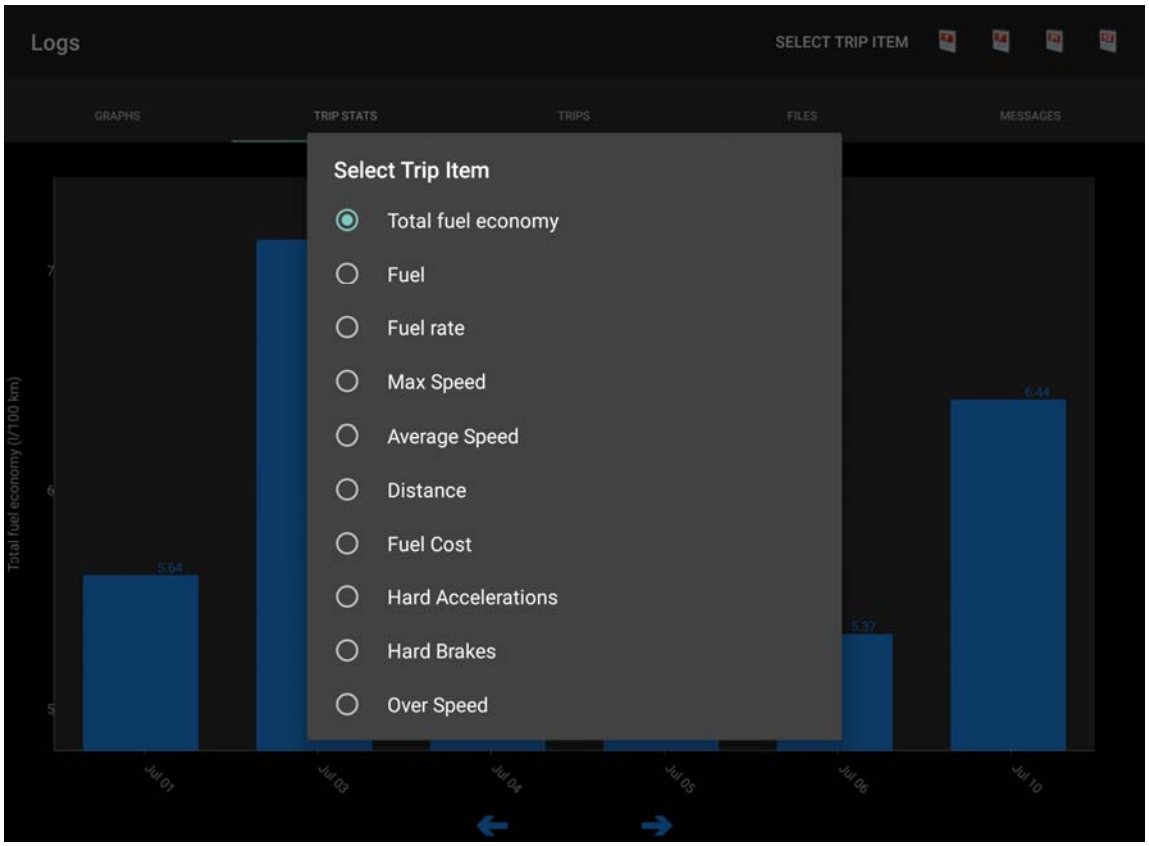
Use of an OBDLink Device with a Terminal Emulator

The OBDLink devices can be used to communicate with the car via a terminal emulator app (*https://www.glmsoftware.com/documentation/OBDNowTerminalUserGuide.pdf*) but unless you know the Toyota PID #IDs (and maybe other methods used to obfuscate the process) it is much easier to just use the OBDLink app.

For requests of data from legislated SAE PIDs via a terminal emulator, the gen 5 rav4 HV seems only to allow functional addressing (request Header 7DF but not 7E0, 7E2 or 7E6), even though results come from modules with response Headers 7E8, 7EA and 7EE.
Cheaper OBD interface devices may also work for this more limited purpose.

OBDLink: Useful Undocumented Features

Home> Logs> Trip Stats has lots of interesting information.



Home> Diagnostics> PID Values can give interesting real-time data from all selected PIDs (with min/ av/ max during the current connection or reset period).