

Cyber Vehicle Overhaul – Modification tutorial:

A tutorial for TweakDB car modifications using CVO

**THIS TUTORIAL IS LONG
GO TO THE MARKED SECTIONS FOR DETAILS ON EACH PART**

Requirements:

Cyber Engine Tweaks – 1.16.0 or higher

Cyber Vehicle Overhaul – 1.2 or higher

tweakdb.str - <https://www.cyberpunk.net/en/modding-support> (Metadata download)

Permissions:

(Not 100% finished but gives some details, for anything else please send me a message):

You can modify only the tweak files or create your own new ones.

These can be uploaded to be used with CVO (Must add CVO as requirement), however you cannot re-upload the CVO loader, CVO main functions or any other functions/templates as your own.

You can simply use CVO loader and simply exchange my tweaks with yours, but you must link CVO to do with it.

There is a possibility I will also release an empty CVO loader for other people to get their own tweaks to setup properly. Good Luck with car modding 😊

Contents

Requirements:.....	1
Permissions:	1
How to find default values:.....	3
Template structure:	4
VERY IMPORTANT:	5
Corresponding data:	5
How to tune the cars:	5
Cloning of Vehicle Settings:	5
Setting of new data:.....	6
How to setup engine power:	6
How to setup the gearbox:	6
How to setup tire friction:.....	8
How to setup braking power:	8
How to setup suspension:.....	9
Basic suspension setup:	10
How to setup the Drive-Model-Data:	10
How to setup a drivetrain (AWD/RWD/FWD) conversion:.....	11
Additional notes:.....	13

How to add your template to CVO:

Simply add the given template, or your own tweak and format it accordingly:

must be .lua format and follow the same **.new function system** as all other CVO tweaks
preferably ending with _tweak (since I may add this to another version to search only for _tweak.lua endings)

How to find default values:

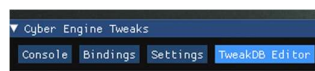
This is the most annoying part. And the most time consuming.

I am yet to find simple and efficient way to dump all the values for all the TweakDB entries.
(Remember to go to 'Advanced' tab in CET's TweakDB settings and follow the instructions to load the tweakdb.str otherwise every TweakDB entry will be named 'TDBID:number:number' and not actual names)

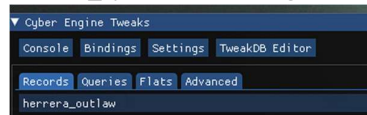
How to find details for each vehicle using CET:

(in my opinion, this is the best option for cars, since even if we would to have access to tweak details, many cars use default values and we can view these easily in CET)

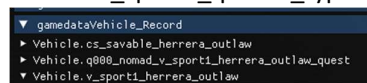
1. Go into TweakDB in CET



2. Go into records tab and type in your desired vehicle in this format: 'make_model' or 'model_specification' (e.g. Herrera_outlaw or type66_nomad (javelina))



3. In gamedataVehicleRecord find your desired vehicle, it should be in naming scheme: 'v_typeNumber_make_model_specification' (e.g. Vehicle.v_sport1_herrera_outlaw or Vehicle.v_sport2_quadra_type66_nomad)



4. Scroll to the bottom of the record and find each record you wish to tweak, these will include the name of the records. These records could include:

vehDriveModelData:

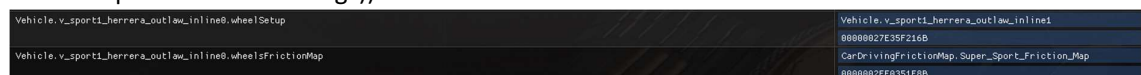


vehEngineData:

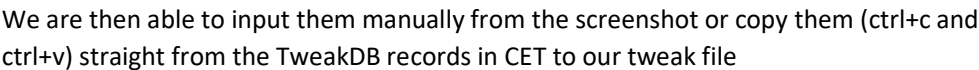


vehWheelDimensionsSetup (not overly useful)

In DriveModelData you can also find wheelSetup (suspension data) and wheelsFrictionMap (tire friction map for each vehicle type (I would advise not changing this unless you really went deep into the car settings))



- Example use (Engine Data for Original Herrera Outlaw):



To start off with, the template contains all the basic variables for the vehicle tweaking. This also applies to all of my tweaks in CVO.

The car you will be setting to should be defined in the 'vehicleName' variable
The naming scheme can be found in the gamedataVehicleRecord table of TweakDB

This is an example on how this can be done:

There are 6 main types of variables:
Engine Power – values for the engine capability

Brakes – value for braking power (including the drive-model-data breaking friction factor)

Drive Model Data – Car's drive details (includes things like car weight and air resistance)

Tire Friction – Settings for front and rear tire friction modifiers

Suspension setup – Suspension setup and its corresponding changes

Gearbox – Settings for each gear and gearbox table that is set into engine data

Put the corresponding variables in each one of these variable types, it will make everyone's life easier.

VERY IMPORTANT:

Make sure that if you are cloning Vehicle data to use the correct corresponding Variable Load functions:

```
VehicleBaseFunctions.cloneVehicleSettings(fullVehicleName, ShouldCloneVehicleEngineData, ShouldCloneVehicleDriveModelData, ShouldCloneVehicleDataPackage, ShouldCloneVehicleWheelDimensionsSetup, ShouldCloneVehicleWheelSetup)

-- Clone Variable Load
VehicleEngineData = fullVehicleName..".full_enginedata"
VehicleDriveModelData = fullVehicleName..".veh_drive_modeldata"
VehicleDataPackage = fullVehicleName..".veh_data_package"
VehicleWheelDimensionsSetup = fullVehicleName..".veh_wheel_dimensions_setup"
VehicleWheelSetup = fullVehicleName..".veh_wheel_setup"
FrontWheelSuspension = VehicleWheelSetup..".full_veh_wheel_setup_front"
RearWheelSuspension = VehicleWheelSetup..".full_veh_wheel_setup_back"

-- Basic Variable Load
VehicleEngineData = TweakDB:GetFlat(fullVehicleName..".veh_enginedata")
VehicleDriveModelData = TweakDB:GetFlat(fullVehicleName..".veh_drive_modeldata")
VehicleDataPackage = TweakDB:GetFlat(fullVehicleName..".veh_data_package")
VehicleWheelDimensionsSetup = TweakDB:GetFlat(fullVehicleName..".veh_wheel_dimensions_setup")
VehicleWheelSetup = TweakDB:GetFlat(VehicleDriveModelData..".wheel_setup")
FrontWheelSuspension = TweakDB:GetFlat(VehicleWheelSetup..".front_preset")
RearWheelSuspension = TweakDB:GetFlat(VehicleWheelSetup..".back_preset")
```

This is from our example template, where only EngineData was cloned (Gear data is always cloned)

If you cloned:

For cloned data: uncomment (remove --) corresponding cloned variable load and comment corresponding (add --) basic variable load

Corresponding data:

ShouldCloneVehicleEngineData – VehicleEngineData

ShouldCloneVehicleDriveModelData - VehicleDriveModelData

ShouldCloneVehicleDataPackage - VehicleDataPackage

ShouldCloneVehicleWheelDimensionsSetup - VehicleWheelDimensionsSetup

ShouldCloneVehicleWheelSetup – VehicleWheelSetup, FrontWheelSuspension, RearWheelSuspension

How to tune the cars:

This part of the tutorial will include information on how to tweak the car to your desired capability, each part will have corresponding variables and functions to set them to the TweakDB table for each car.

Remember, every time you change anything in the tweaks and reload the mod in CET, make sure to also reload the save file, otherwise you will not see any changes.

Cloning of Vehicle Settings:

Each car will have its own corresponding engine, gearbox (set in engine), suspension, drive-model-data and wheel (and drivetrain) setup.

Sometimes cars have Default type of data set, if you do not want to be setting this for all the cars that uses the same data set you will have to clone the settings using the clone function.

For this you need to be cloning the records, here is the function:

```
'VehicleBaseFunctions.cloneVehicleSettings(fullVehicleName, ShouldCloneVehicleEngineData, ShouldCloneVehicleDriveModelData, ShouldCloneVehicleDataPackage, ShouldCloneVehicleWheelDimensionsSetup, ShouldCloneVehicleWheelSetup)'
```

It takes variables for cloning of 5 different vehicle sectors

CloneVehicleEngineData – This clones the Vehicle's Engine Data

CloneVehicleDriveModelData – This clones the Vehicle Drive Model Data

CloneVehicleDataPackage – This clones the Vehicle Data Package (Car information, doesn't contain handling)

CloneVehicleWheelDimensions – This clones the wheel dimensions (I suggest not touch these much)

CloneVehicleWheelSetup – This clones the suspension data

Setting of new data:

Each variable has to be set to the TweakDB table through SetFlat command

Example of how we do this is:

```
TweakDB:SetFlat(VehicleEngineData.."engineMaxTorque", MaxTorque)
```

Where 'VehicleEngineData' is the Variable to point to engine data

".engineMaxTorque" is the flat name we want to set to

'MaxTorque' is the variable we are setting

Can also set manually without variable pointer but this is insanely inefficient:

E.g. `TweakDB:SetFlat("Vehicle.VehicleEngineData_4_Sport_AWD.engineMaxTorque", 1500)`

This would set the torque of 4_sport_awd engine to 1500nm (insane value)

How to setup engine power:

Each car has a corresponding engine, sometimes these engines are default so don't forget to clone them.

Engine Data contains:

Min/Max RPM – Rev limiter range

MaxTorque – (multiplier for curvset(lut) tables (curvesets are not tweakable in tweakDB))

ResistanceTorque – Acts like flywheel weight (very odd setting)

WheelsResistanceRatio – I suggest not touching this

Gears – Gearbox Table (sets each gear data that this engine will use)

To setup the car for High Horsepower and High Torque, I would highly advise also tweaking the tire/suspension data.

These are very self-explanatory, the more power the faster the car will accelerate but it will also wheel spin if you give it too much for the tire friction capability.

Min/Max RPM do not affect gearbox directly, so if you change them make sure to change the gearbox also

ResistanceTorque/WheelsResistanceRatio are better left alone.

How to setup the gearbox:

Usually I'd advise keeping the Min/Max RPM of the gearbox and engine data, since it takes forever to make better and the audio will be off.

What can you change in each gear:

Min/MaxEngineRPM – Corresponds to engine RPM

Min/MaxSpeed – The minimum and maximum speed achievable in the specific gear

TorqueMultiplier – Torque speed at the wheel (the more it has the faster the car will accelerate)
(should be under 1.0x e.g. 0.25 for 7th gear and 0.60 for 3rd gear etc.)

Vehicle.VehicleEngineData_4_Sport_inline2	
Vehicle.VehicleEngineData_4_Sport_inline2.maxEngineRPM	6400.000000
Vehicle.VehicleEngineData_4_Sport_inline2.maxSpeed	21.000000
Vehicle.VehicleEngineData_4_Sport_inline2.minEngineRPM	3300.000000
Vehicle.VehicleEngineData_4_Sport_inline2.minSpeed	13.000000
Vehicle.VehicleEngineData_4_Sport_inline2.torqueMultiplier	0.750000

Speed settings:

Each gear for the gearbox has its own Min and Max speed

These will essentially be the Minimum speed (at set Min RPM) and Maximum speed (at set Max RPM) that the car is going to be driving at.

As you can see in the example gearbox speeds are not tied to MPH in the game. For example, 70 units here, should roughly correspond to 320mph. (a factor of more or less 4.55x vs MPH in game)
(The game states MPH in vehicles, while the speeds are closer to KPH but are actually neither vs distance to locations calculated by the game's waypoint system :D (KPH is off by like 10/20%))

Gear adding to Engine:

Use the template and copy the gear setting section and gearsToUse table

The gearsToUse table should look like this:

```
gearsToUse = {
fullVehicleName.."inline_gear0",
fullVehicleName.."inline_gear1",
fullVehicleName.."inline_gear2",
fullVehicleName.."inline_gear3",
fullVehicleName.."inline_gear4",
fullVehicleName.."inline_gear5",
fullVehicleName.."inline_gear6"
}
```

Cars can have a maximum of 7 gears and 1 reverse (0-reverse, 1,2,3,4,5,6,7)

fullVehicleName.."inline_gearX" – Where X is the gear number is how you add a new one.

(Make sure to add the comma(,) for each new gear you add and to end the gear table without a comma)

Example gear:

```
--setting '1' cuz there is gear 0
gearsTorqueMult[1] = 1.20
gearsMinSpeed[1] = 0*FinalDriveMultToSet
gearsMaxSpeed[1] = 9*FinalDriveMultToSet
gearsMinRPM[1] = 800
gearsMaxRPM[1] = 4400
```

How to set the values to each gear:

e.g. gearsTorqueMult[X] = 1.00 – Where X is the gear number, and 1.0 is the torque at the gear (yes it should be lower at higher gears)

These two functions clone and set gears respectively,
what you will put into the table entries it will set into the engine here:

```
--Gear Settings
--Gear Cloning
GearsData = TweakDB:GetFlat(VehicleEngineData.."gears")
for index=0,AmountOfGears do
    TweakDB:CloneRecord(fullVehicleName.."inline_gear"..index, GearsData[index])
end

--Gear Settings
TweakDB:SetFlat(VehicleEngineData.."gears", gearsToUse)
for index=1,AmountOfGears do
    VehicleBaseFunctions.setGearData(fullVehicleName.."inline_gear"..index, gearsTorqueMult[index], gearsMinSpeed[index], gearsMaxSpeed[index], gearsMinRPM[index], gearsMaxRPM[index])
end
```

How to setup tire friction:

The data for this in the suspension setup:
'gamedataWheelDrivingPreset_Record'

This is very simple, but takes a while to tweak.

Explanation:

Each car has a tire friction-map (in DriveModelData) that I've shown earlier, these variables in each car's settings are just multipliers, so they will react the same on specific surface as on the tire friction-map, but will also multiply accordingly

FrictionLateralMult – Multiplier for friction of the forward/backwards action of the tire (accelerating/decelerating)

FrictionLongitudinal – Multiplier for friction of sideways action of the tire (turning/drift/sliding)

Set to what you feel drives best, I advise not going above 1.35 or the cars will start feel like karts

There are also (I do not use them, these work kinda like real life counterparts so look them up):

tireFrictionCoef – Seems to be a mult like(coefficient) scalar for the tire friction factor

tireRollingResistanceCoef – Seems to be mult (coefficient) for rolling resistance

tireLateralSlipEffectsMul – Lateral 'slip' effects mult (essentially mult for slide factor)

tireLongitudinalSlipEffectsMul – Longitudinal 'slip' effects mult (essentially mult for slide factor)

How to setup braking power:

Braking power is divided into two sections:

BrakingFrictionFactor – Multiplier for CAR(tire+air+car it seems?) friction for braking

Do not set this one past 2.0 (1.75 is max advisable) it will make the car nose dive too much.

BrakingTorque – front/back braking torque

The true braking power of your brakes.

To make the car understeer while braking - ratio front to back: 1:1.5 or 1:2

To make the car brake straight - ratio front to back: 1:1 or 1.2:1

To make the car understeer (like real life) – ratio front to back: 1.5:1 or 2:1

Play around with these and make sure to not overdo it on braking torque, if tire friction too low it can make the wheels lock up too easily and slide more than brake.

Advice:

Braking torque of real life racecars is like around 4000nm+ (4500/5000 also applies somewhat)

Braking torque of real life sport street cars is like around 2500/3000nm (sometimes more)

Braking torque of real life normal cars is like around the 1500/2500nm

How to setup suspension:

This is probably the hardest thing to setup.

Most of handling is done by suspension setup, but some of it is also in DriveModelData.

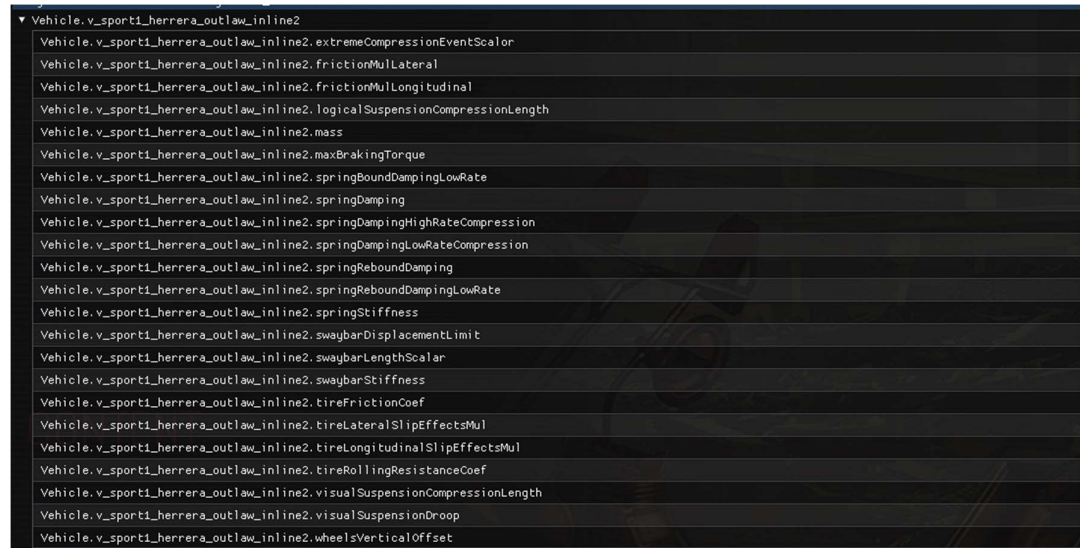
Suspension setups from real life are for the most part similarly efficient to here,

look for tutorials how to setup suspension for your favorite setup

(good example would be: <https://speed.academy/how-to-fix-understeer-or-oversteer-suspension-handling-setup-tuning/>)

What can you tweak:

'gamedataWheelDrivingPreset_Record' will contain all the suspension data for the car



Variables for each part:

ExtremeCompressionEvenScalar – This is a weird setting, seems to multiply some part of the maximum compression and decompression for the suspension (Only use it if the car seems to have too big suspension travel (seems to help slightly in weird situations?))

FrictionMulLateral/Longitudinal – Read Tire Setup

LogicalSuspensionCompressionLength – Suspension compression length used by gameplay calculations (seems separate from visual compression)

Mass – This seems like tire or suspension bound mass? (this can be used to make the car feel more... heavy to drive)

maxBrakingTorque – Braking Power

SpringDamping, SpringBound, SpringReboundDamping,

SpringDampingHighRate/LowRateCompression and springReboundDampingLowRate – These values are very hard to tweak and I suggest leaving them to the very very end.

I would also highly advise following some real life tutorial, and testing if sticks

SpringStiffness – Suspension stiffness

SwaybarDisplacementLimit – Seems to be a value for how much movement the swaybar has before it starts reacting

SwaybarLengthScalar – Limited knowledge on this one, I think it corresponds to how much the car will rebound on swaybar action

SwaybarStiffness – The stiffness of the swaybar (important for oversteer/understeer)

TireCoeff/SlipEffect – Look tire setup

VisualSuspensionCompressionLength – The visual system similar to the logical one earlier (suspension travel distance (kinda))
VisualSuspensionDroop – Partially visual suspension height (seems to not affect driving until large values are used)
WheelsVerticalOffset – Real suspension height offset (both visual and gameplay)

Basic suspension setup:

Once again look for tutorials for suspension setup, the ones I found easiest is:
<https://speed.academy/how-to-fix-understeer-or-oversteer-suspension-handling-setup-tuning/>
for oversteer and understeer specifically but applies all around

I'm not amazing at tuning car suspension, I only know a bit, there is a LOT of tutorials on the net for real life/simulators, for the most part it should also apply here, I will only provide you basic setups which you can work off of. Remember suspension is hard to get perfect, and I suggest mostly leaving cars as they are, unless you want to build a simulator out of this game :D.

Basic setup for understeer:

Tire mass: 20/25 (based on 1500kg car)
Swaybar Stiffness: 20/10 – 1.8:1 or 2:1 ratios
Spring Stiffness: 20:15 – 2:1.5 or 1:1.15 sometimes 1.15:1 also works well

Basic setup for oversteer:

Mass: 35+ (based on 1500kg car)
Swaybar Stiffness: 10/10 – 1:1 or 1:2 ratios (or 0:0 (disables sway bar like javelina))
Spring settings: 23/20 – 1.15:1 or 1.5:1 etc

How to setup the Drive-Model-Data:

Drive Model Data contains basic car data, from stuff like mass of the car, to maximum turning angle, turning speeds (how fast can wheels turn to max angle), turning roll multiplier, air resistance and so on.

Most useful variables:

airResistanceFactor – Air resistance of the vehicle (how much its slow down at higher speed)
chassis_mass and total_mass – Weight of the vehicle
brakingFrictionFactor – Look brakes setup
maxWheelTurnDeg – Maximum turning angle of the wheels
wheelTurnMaxAddPerSecond – maximum steering angle added per second
wheelTurnSubAddPerSecond – maximum steering angle reduced per second
turningRollFactor – scalar for suspension travel in longitudinal direction (how much the car rolls)
bikeMaxTilt – Maximum bike tilt
wheelsFrictionMap – Tire friction map (different for different car types)
forwardWeightTransferFactor – Mult/Ratio of car weight front/back

There are lots more variables, each relatively easy to understand if you know how cars work on the deep end, but you can always research every part.

How to setup a drivetrain (AWD/RWD/FWD) conversion:

This is relatively easy to do, but the problem comes from cars feeling like they are still FWD even if you do swap it.

So, you must either tweak suspension and the DriveModelData to fit the new vehicle type, or you can simply clone wheelSetup from another vehicle, and stick it in full to the vehicle you are swapping the drivetrain into.

The hard way (full swap to new drivetrain (in this example FWD to RWD)):

So how do we swap to a different drive layout first of all:

1. Get your current Vehicle's wheelSetup from DriveModelData (template should contain VehicleWheelSetup with flat already set)



2. Now we need to change the 4 wheel variables for wheel drive capability,
LB – Left Back
LF – Left Front
RB – Right Back
RF – Right Front



3. We can change these values in two ways, by simply getting the flat(variable) from another vehicle or setting it directly through a hash value or string name

Example of setting a car to RWD from villefort cortes police car

```
TweakDB:SetFlat(VehicleWheelSetup..".LB",  
TweakDB:GetFlat("Vehicle.VehicleDriveModelData_Cortes_Police_inline0.LB"))  
TweakDB:SetFlat(VehicleWheelSetup..".LF",  
TweakDB:GetFlat("Vehicle.VehicleDriveModelData_Cortes_Police_inline0.LF"))  
TweakDB:SetFlat(VehicleWheelSetup..".RB",  
TweakDB:GetFlat("Vehicle.VehicleDriveModelData_Cortes_Police_inline0.RB"))  
TweakDB:SetFlat(VehicleWheelSetup..".RF",  
TweakDB:GetFlat("Vehicle.VehicleDriveModelData_Cortes_Police_inline0.RF"))
```

Example of setting a car to RWD through hash values:

```
TweakDB:SetFlat(VehicleWheelSetup..".LB", 0x00000032325E4F38ull)  
TweakDB:SetFlat(VehicleWheelSetup..".LF", 0x0000002E7998A248ull)  
TweakDB:SetFlat(VehicleWheelSetup..".RB", 0x0000003245597FAEull)  
TweakDB:SetFlat(VehicleWheelSetup..".RF", 0x0000002E0E9F92DEull)
```

Example of setting a car to RWD through named flats:

```
TweakDB:SetFlat(VehicleWheelSetup..".LB",  
"Vehicle.VehicleWheelDrivingSetup_4_Default_inline3")
```

```

TweakDB:SetFlat(VehicleWheelSetup..".LF",
"Vehicle.VehicleWheelDrivingSetup_4_RWD_inline0")
TweakDB:SetFlat(VehicleWheelSetup..".RB",
"Vehicle.VehicleWheelDrivingSetup_4_Default_inline2")
TweakDB:SetFlat(VehicleWheelSetup..".RF",
"Vehicle.VehicleWheelDrivingSetup_4_RWD_inline1")

```

4. The car is now new drive layout, now you need to change suspension accordingly (go to suspension setup part for more details)

To make it easier, you could possibly clone the entire wheelSetup from another car you like that has the drivetrain you would want, you could do this easily through simply setting the wheelSetup from another car into the current vehicle, and then proceeding with the 'cloneVehicleDetails' function.

This is how you would do this:

Before VehicleBaseFunctions.cloneVehicleSettings(), we would add these two lines:

```

PreCloneVehicleDriveModelData = TweakDB:GetFlat(fullVehicleName..".vehDriveModelData")
TweakDB:SetFlat(PreCloneVehicleDriveModelData..".wheelSetup", TweakDB:GetFlat("Vehicle.VehicleDriveModelData_Cortes_Police.wheelSetup"))
TweakDB:GetFlat("Vehicle.VehicleDriveModelData_Cortes_Police.wheelSetup"))

```

And then we would send a VehicleBaseFunctions.cloneVehicleSettings()

With all variables as needed previously, plus ShouldCloneVehicleWheelSetup being set to true.

Like so:

```

PreCloneVehicleDriveModelData = TweakDB:GetFlat(fullVehicleName..".vehDriveModelData")
TweakDB:SetFlat(PreCloneVehicleDriveModelData..".wheelSetup", TweakDB:GetFlat("Vehicle.VehicleDriveModelData_Cortes_Police.wheelSetup"))
-- Cloning Variables: (fullVehicleNameToUse, ShouldCloneVehicleEngineData, ShouldCloneVehicleDriveModelData, ShouldCloneVehicleDataPackage,
VehicleBaseFunctions.cloneVehicleSettings(fullVehicleName, true, false, false, false, true)

```

Additional notes:

I plan on giving everyone the ability to add their own 'version' of a vehicle or new vehicles entirely, but this still has to be implemented into CVO, so once I get around to adding it I will most likely update this tutorial :)

I hope this comes in handy... it took me good couple hours to write up so I hope it helps a couple people with tweaks