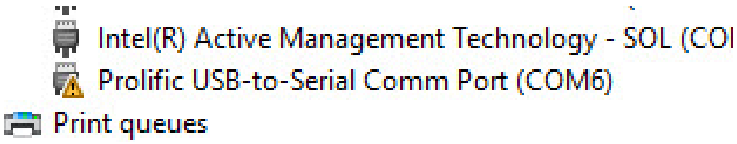
The Myth of USB-to-Serial Port Adapters and Bad Drivers in Windows

- the sad, ugly trick to get rid of "bad drivers”…

If you’ve seen error messages like: “*Unable to open port…”* when dealing with USB-to-serial/USB-to-radio-plug adapters to program a radio, interface with a packet TNC, repeater controller or other serial device – you’re probably thinking “*bad driver*” or “*screw Microsoft*.” Neither are usually the case.



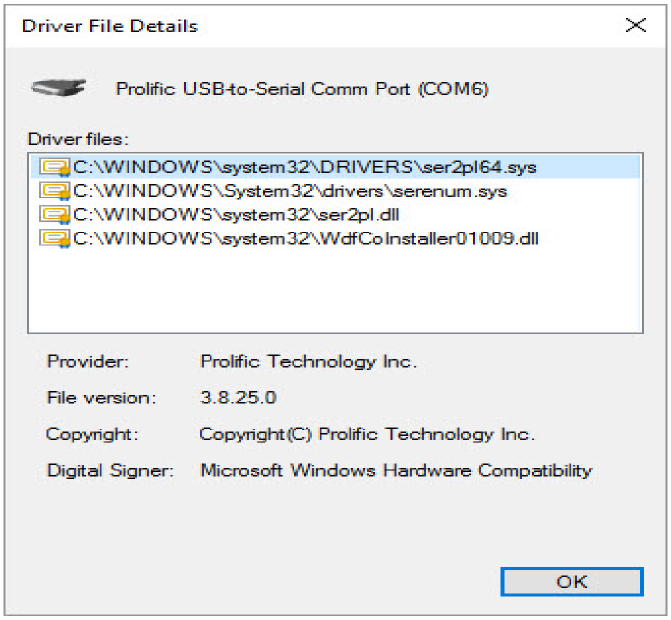
Intel(R) Active Management Technology - SOL (COI
Prolific USB-to-Serial Comm Port (COM6)
Print queues

Windows Device Manager flagging a “bad device/driver”

Of course you could have picked the wrong port designation, or some background process has blocked access to the driver or something is stuck in a retry loop, but you’re sure that’s not the case.

Next you consult ‘Google’ for an answer and get 5700 results most of which tell you to uninstall and re-install the ‘driver’, update a ‘driver’, get a different ‘driver’, etc. I’m going to venture a guess that most of those 5700 results are dead wrong, and maybe this can be result #5701 that actually helps!?

In most cases there is no device-specific driver. Most serial port devices use Windows’ common *serial.sys* and *serenum.sys* 'drivers'. A couple of USB-adapter makers (FTDI for one) DO provide their own actual device drivers but that’s not typical, common – especially for the “offshore” products that use one of the common chips or clone thereof.



Driver File Details

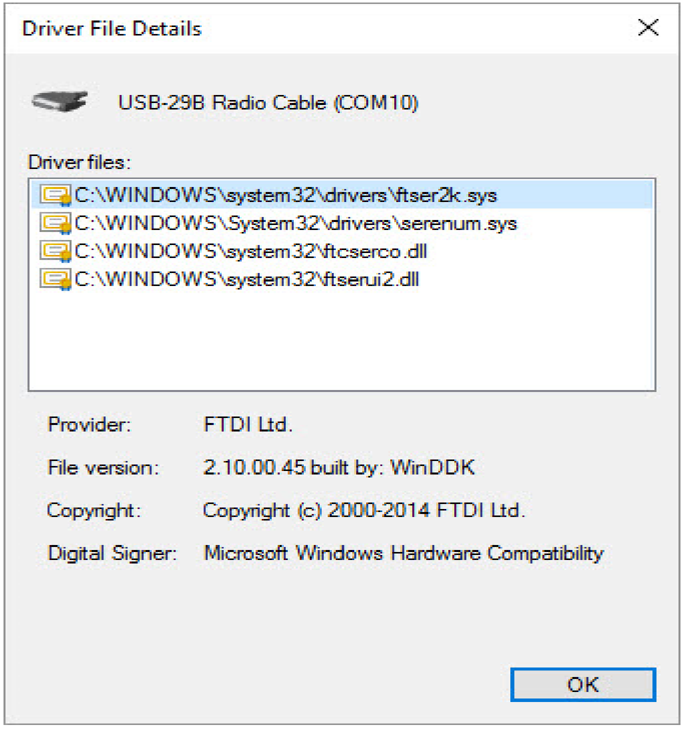
Prolific USB-to-Serial Comm Port (COM6)

Driver files:

C:\WINDOWS\system32\DRIVERS\ser2pl64.sys
C:\WINDOWS\System32\drivers\serenum.sys
C:\WINDOWS\system32\ser2pl.dll
C:\WINDOWS\system32\WdfCoInstaller01009.dll

Provider: Prolific Technology Inc.
File version: 3.8.25.0
Copyright: Copyright(C) Prolific Technology Inc.
Digital Signer: Microsoft Windows Hardware Compatibility

Prolific Devices Use Standard Microsoft OS Drivers



Driver File Details

USB-29B Radio Cable (COM10)

Driver files:

C:\WINDOWS\system32\drivers\ftser2k.sys
C:\WINDOWS\System32\drivers\serenum.sys
C:\WINDOWS\system32\ftcserco.dll
C:\WINDOWS\system32\ftserui2.dll

Provider: FTDI Ltd.

File version: 2.10.00.45 built by: WinDDK

Copyright: Copyright (c) 2000-2014 FTDI Ltd.

Digital Signer: Microsoft Windows Hardware Compatibility

The RT Systems and BrainBox Adapters Use the FTDI Chipset and Drivers

Most of the problems arise when multiple/different but similar USB device IDs appear to the system’s Device Manager, but they require different internal parameters to function. What makes one device different from another in Windows is the USB device ID and the parameters associated with it. When a port is selected to be used the serial 'driver' looks for the USB ID in the registry and grabs a block of hex that tells the driver how to talk to that specific USB device.

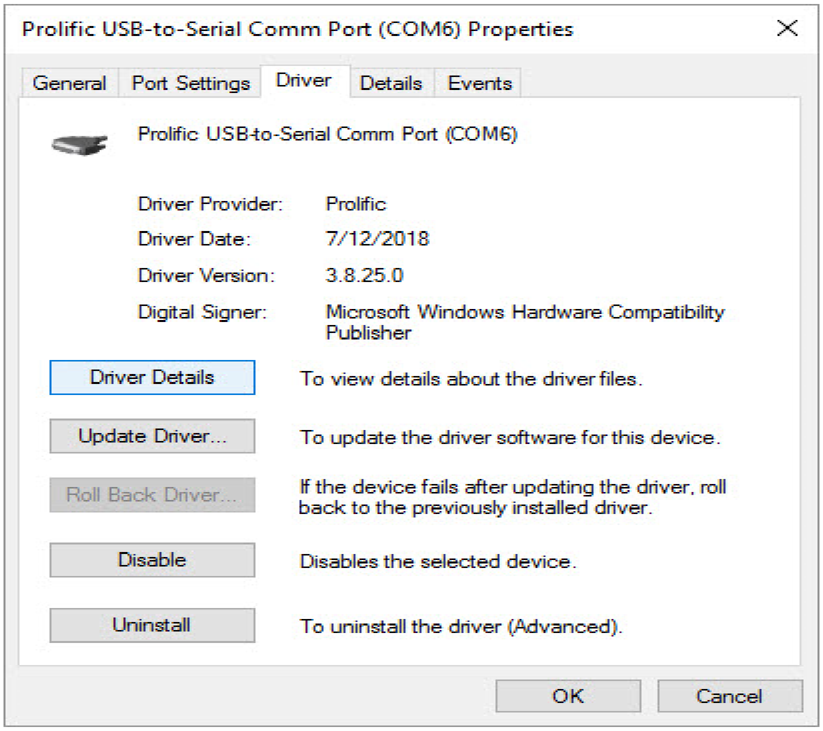
The bad device/bad driver problem is that the same USB device ID had been used for a couple of different chips/fobs but the hex configuration bits are different - so unless/until you clear the device ID from the registry and install "the right driver" (bits) for that specific fob, you'll always end up with this problem.

In most cases, uninstalling/'deleting' the "bad driver" and re-installing another does NOT work - first because that uninstall process does NOT (and should not try to) rip out the native Windows files, and second because it does not delete the USB Dev ID and bits from the Registry. Thinking you’ve removed a “bad driver”, installing a “different version” only adds another same/similar device ID with different parameters, but Device Manager cannot tell them apart. *Bad installer/coder problem*.

The solution in this latter case is to get the device ID from Device Manager, search the Windows Registry for any/all instances and DELETE those references – THEN install the “*new driver*” that might contain the proper operating parameters for that device/chip implementation. This is obviously an *ugly and tedious* solution. The device/’driver’ maker/install COULD have kept track of the information it placed in the Registry and provided a simple code routine to actually remove it – but not. *Bad installer/coder problem*.

Ready to dive-in?

In Device Manager, right-click and select Properties to get to the device. Select Driver Details:



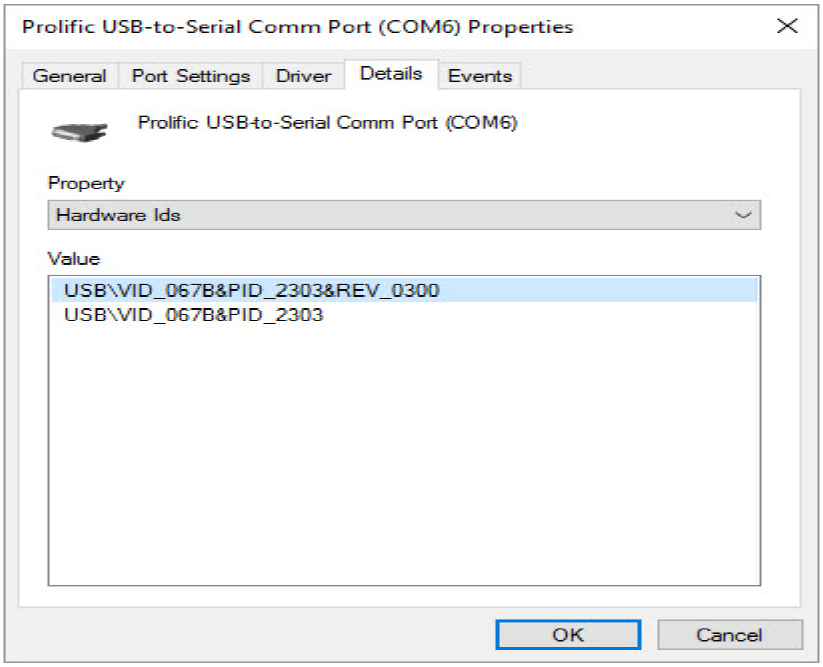
Prolific USB-to-Serial Comm Port (COM6) Properties

Prolific USB-to-Serial Comm Port (COM6)

Driver Provider: Prolific
Driver Date: 7/12/2018
Driver Version: 3.8.25.0
Digital Signer: Microsoft Windows Hardware Compatibility Publisher

Driver Details — To view details about the driver files.
Update Driver... — To update the driver software for this device.
Roll Back Driver... — If the device fails after updating the driver, roll back to the previously installed driver.
Disable — Disables the selected device.
Uninstall — To uninstall the driver (Advanced).

Select **Hardware Ids** under **Property**, right-click on the ID then select **Copy**:



Prolific USB-to-Serial Comm Port (COM6) Properties

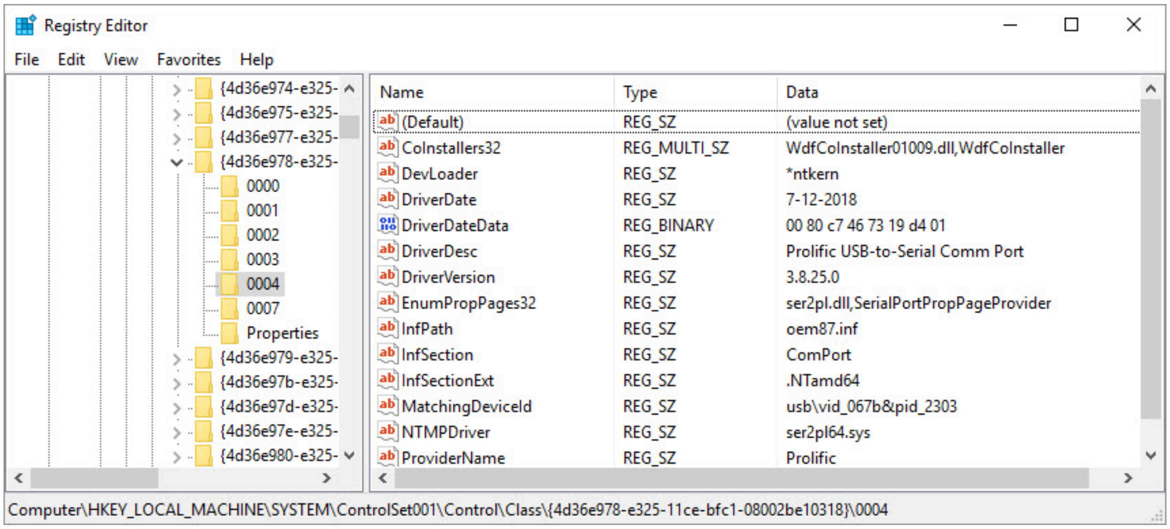
Prolific USB-to-Serial Comm Port (COM6)

Property: Hardware Ids

Value:

```
USB\VID_067B&PID_2303&REV_0300
USB\VID_067B&PID_2303
```

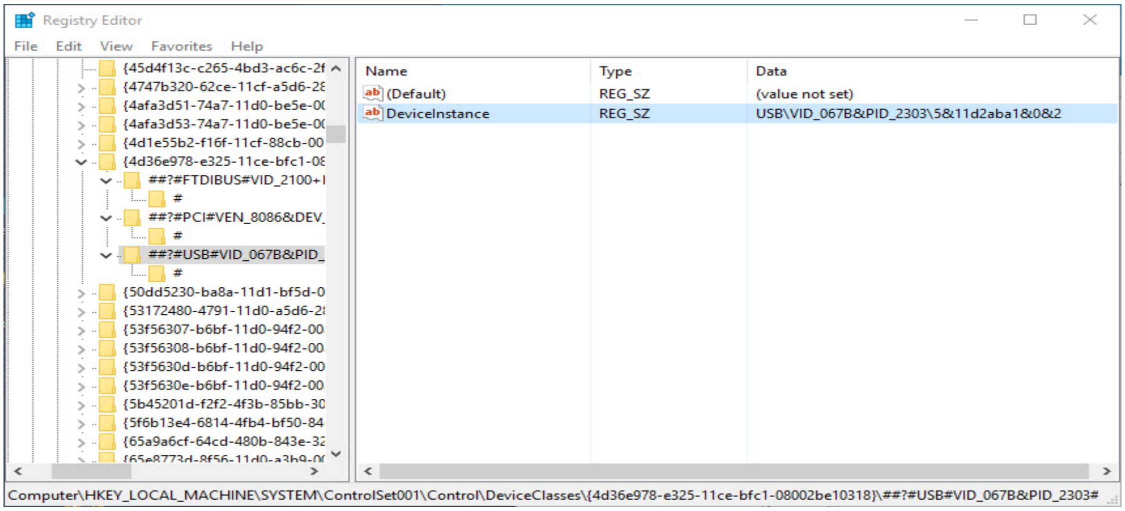
Open Regedit, then Edit, Find, paste in the Device ID copied from above:



Name	Type	Data
(Default)	REG_SZ	(value not set)
CoInstallers32	REG_MULTI_SZ	WdfCoInstaller01009.dll,WdfCoInstaller
DevLoader	REG_SZ	*ntkern
DriverDate	REG_SZ	7-12-2018
DriverDateData	REG_BINARY	00 80 c7 46 73 19 d4 01
DriverDesc	REG_SZ	Prolific USB-to-Serial Comm Port
DriverVersion	REG_SZ	3.8.25.0
EnumPropPages32	REG_SZ	ser2pl.dll,SerialPortPropPageProvider
InfPath	REG_SZ	oem87.inf
InfSection	REG_SZ	ComPort
InfSectionExt	REG_SZ	.NTamd64
MatchingDeviceId	REG_SZ	usb\vid_067b&pid_2303
NTMPDriver	REG_SZ	ser2pl64.sys
ProviderName	REG_SZ	Prolific

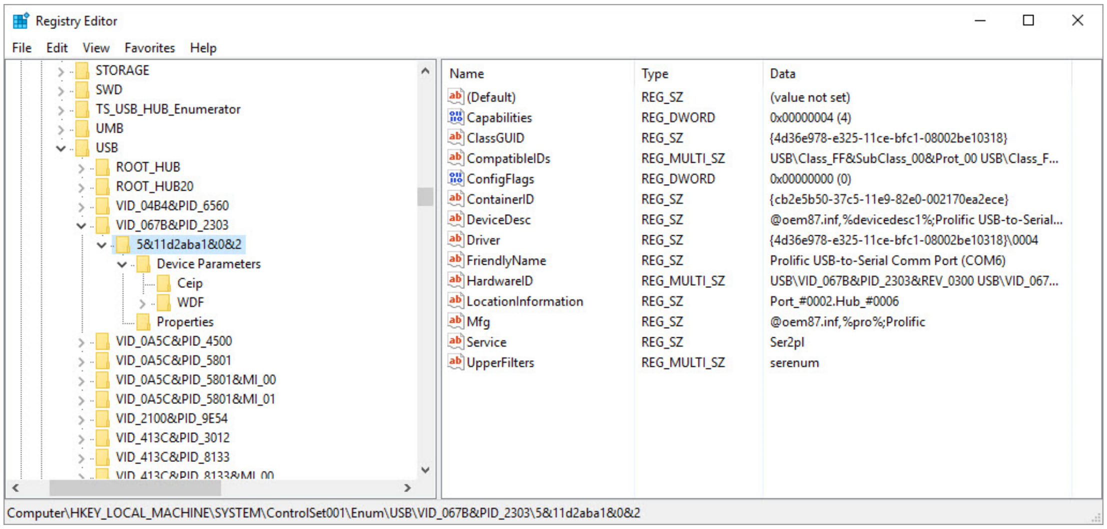
Computer\HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Class\{4d36e978-e325-11ce-bfc1-08002be10318}\0004

Delete the reg-key for the device – in this case the **0004**. Press **F3** to keep searching for and then delete subsequent references until they’re all gone.


Registry Editor
File
Edit
View
Favorites
Help
{45d4f13c-c265-4bd3-ac6c-2f
{4747b320-62ce-11cf-a5d6-28
{4afa3d51-74a7-11d0-be5e-00
{4afa3d53-74a7-11d0-be5e-00
{4d1e55b2-f16f-11cf-88cb-00
{4d36e978-e325-11ce-bfc1-08
##?#FTDIBUS#VID_2100+I
#
##?#PCI#VEN_8086&DEV_
#
##?#USB#VID_067B&PID_
#
{50dd5230-ba8a-11d1-bf5d-0
{53172480-4791-11d0-a5d6-28
{53f56307-b6bf-11d0-94f2-00
{53f56308-b6bf-11d0-94f2-00
{53f5630d-b6bf-11d0-94f2-00
{53f5630e-b6bf-11d0-94f2-00
{5b45201d-f2f2-4f3b-85bb-30
{5f6b13e4-6814-4fb4-bf50-84
{65a9a6cf-64cd-480b-843e-32
{65e8773d-8f56-11d0-a3b9-00

Name	Type	Data
(Default)	REG_SZ	(value not set)
DeviceInstance	REG_SZ	USB\VID_067B&PID_2303\5&11d2aba1&0&2

Computer\HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\DeviceClasses\{4d36e978-e325-11ce-bfc1-08002be10318}\##?#USB#VID_067B&PID_2303#



Name	Type	Data
(Default)	REG_SZ	(value not set)
Capabilities	REG_DWORD	0x00000004 (4)
ClassGUID	REG_SZ	{4d36e978-e325-11ce-bfc1-08002be10318}
CompatibleIDs	REG_MULTI_SZ	USB\Class_FF&SubClass_00&Prot_00 USB\Class_F...
ConfigFlags	REG_DWORD	0x00000000 (0)
ContainerID	REG_SZ	{cb2e5b50-37c5-11e9-82e0-002170ea2ece}
DeviceDesc	REG_SZ	@oem87.inf,%devicedesc1%;Prolific USB-to-Serial...
Driver	REG_SZ	{4d36e978-e325-11ce-bfc1-08002be10318}\0004
FriendlyName	REG_SZ	Prolific USB-to-Serial Comm Port (COM6)
HardwareID	REG_MULTI_SZ	USB\VID_067B&PID_2303&REV_0300 USB\VID_067...
LocationInformation	REG_SZ	Port_#0002.Hub_#0006
Mfg	REG_SZ	@oem87.inf,%pro%;Prolific
Service	REG_SZ	Ser2pl
UpperFilters	REG_MULTI_SZ	serenum

Computer\HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Enum\USB\VID_067B&PID_2303\5&11d2aba1&0&2

“But wait!!!!! This will break Windows!!!”

Relax – this will NOT “break Windows” !

Yes, you might goof and delete the wrong references – but Windows Plug-and-Play, WMI, Device Manager processes are pretty robust and can/will “re-install” a valid device. If you get the wrong one, look again, search more specifically and review what the search in the Registry produces, delete…

You may also have to find and delete the .inf or other files related to the first bad install – and that’s a matter of digging into the installer’s files, find filenames and dates to search for. Not pretty. Not Microsoft’s fault.

I’ve had to do this numerous times to reconcile this or that “cheap ***” cable, the included ‘driver’, the “updated driver” from this or ‘expert’ XYZ cheap radio support sites, etc. This is not specifically part of the debacle related to one of the chips getting cloned and a vendor locking out or bricking a cloned/stolen device design – it’s simply a *bad installer/coder problem* because the designers were too cheap to be deliberate and specific about device IDs, differentiating devices, and assuming Microsoft’s processes would deal with their inadequacies, and couldn’t be bothered with cleaning up after themselves.

As far as who uses the “bad devices”/”bad drivers” in their products – they’re also too cheap to use a decent product and install/de-install process – this goes for most any generic, no-name Amazon, eBay, or computer-store USB-serial implementation who won’t tell you what chip is in the plastic pod or USB connector.

Summary

Sad, ugly, tedious, time-consuming, but not impossible. You can fix this…

FWIW – if you need a USB-to-serial adapter that supports the RS-232 standard fully, most/all devices that use the FTDI chipset do so and will pass industry-standard RS-232 protocol and hardware tests. One product I am familiar with as my go-to USB-serial device is: http://www.brainboxes.com/usb-to-serial - and the FTDI driver install/uninstall process actual does clean-up after itself.