Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'usb_modeswitch.1' command

$ man usb_modeswitch.1

USB_MODESWITCH(1) General Commands Manual USB_MODESWITCH(1)

NAME

usb_modeswitch - control the mode of 'multi-state' USB devices

SYNOPSIS

usb_modeswitch [-heWQDIvpVPmM23rwKdHSOBGTNALnsRiuagft] [-c filename]

DESCRIPTION

Several new USB devices have their proprietary Windows drivers onboard, most of them WWAN and WLAN dongles. When plugged in for the first time, they act like a flash storage and start installing the Windows driver from there. If the driver is installed, it makes the storage de? vice disappear and a new device, mainly composite (e.g. with modem ports), shows up.

On Linux, in most cases the drivers are available as kernel modules, such as "usbserial" or "option". However, the device initially binds to "usb-storage" by default. usb_modeswitch can then send a provided bulk message (most likely a mass storage command) to the device; this mes? sage has to be determined by analyzing the actions of the Windows driver.

In some cases, USB control commands are used for switching. These cases are handled by custom functions, and no bulk message needs to be pro? vided.

Usually, the program is distributed with a set of configurations for many known devices, which allows a fully automatic handling of a device

upon insertion, made possible by combining usb_modeswitch with the wrapper script usb_modeswitch_dispatcher which is launched by the udev daemon. This requires a Linux-flavoured system though.

Note that usb_modeswitch itself has no specific Linux dependencies.

OPTIONS

This program follows the usual GNU command line syntax, with long options starting with two dashes ('--'). A summary of options is included below.

-h --help Show summary of options.

-e --version

Print version information and exit

-v --default-vendor NUM

Vendor ID to look for (mandatory), usually given as hex number (example: 0x12d1). Each USB device is identified by a number officialy assigned to the vendor by the USB association and a number for the respective model (product ID) chosen by the vendor

-p --default-product NUM

Product ID to look for (mandatory)

-V --target-vendor NUM

Target vendor ID. When given will be searched for and detected initially for information purposes. If success checking (option -s) is active, providing target IDs (vendor/product) or target class is recommended

-j --find-mbim

Return configuration number with MBIM interface and exit.

-P --target-product NUM

Target product ID

-b --bus-num NUM

-g --device-num NUM

If bus and device number are provided, the handling of a specific device on a specific USB port is guaranteed, in contrast to using only the USB ID. This is important if there

are multiple similar devices on a system

-C --target-class NUM

Target Device Class according to the USB specification. Some devices keep their original vendor/product ID after success? ful switching. To prevent them from being treated again, the device class can be checked. For unswitched devices it is always 8 (storage class), for switched modems it is often 0xff (vendor specific). In composite modes, the class of the first interface is watched

-m --message-endpoint NUM

A specific endpoint to use for data transfers. Only for test? ing purposes; usually endpoints are determined from the de? vice attributes

-M --message-content STRING

A bulk message to send as a switching command. Provided as a hexadecimal string

-2 --message-content2 STRING

-3 --message-content3 STRING

Additional bulk messages to send as switching commands. Pro? vided as hexadecimal strings. When used with mass storage commands, setting --need-response is strongly advised to com? ply with specifications and to avoid likely errors

-w --release-delay <milliseconds>

After issuing all bulk messages, wait for the given time be? fore releasing the interface. Required for some modems on older systems (especially after an EJECT message)

-n --need-response

Obsolete. CSW is always attempted to being read after mass storage transfers. No downside

-r --response-endpoint NUM

Try to read the response to a storage command from there. Only for testing purposes; usually endpoints are determined from the device attributes

-K --std-eject

Apply the standard SCSI sequence of "Allow Medium Removal" and "Eject". Implies -n. One 'Message' can be added with -M that will be transmitted after the eject sequence. Used by many modems

-d --detach-only

Just detach the current driver. This is sufficient for some early devices to switch successfully. Otherwise this feature can be used as a 'scalpel' for special cases, like separating the driver from individual interfaces

-H --huawei-mode

Send a special control message used by older Huawei devices

-J --huawei-new-mode

Send a specific bulk message used by all newer Huawei devices

-X --huawei-alt-mode

Send an alternative bulk message to Huawei devices

-S --sierra-mode

Send a special control message used by Sierra devices

-G --gct-mode

Send a special control message used by GCT chipsets

-T --kobil-mode

Send a special control message used by Kobil devices

-N --sequans-mode

Send a special control message used by Sequans chipset

-A --mobileaction-mode

Send a special control message used by the MobileAction de? vice

-B --qisda-mode

Send a special control message used by Qisda devices

-E --quanta-mode

Send a special control message used by Quanta devices

-F --pantech-mode NUM

Send a special control message used by Pantech devices.

Value NUM will be used in control message as 'wValue'

-Z --blackberry-mode

Send a special control message used by some newer Blackberry devices

-S --option-mode

Send a special control message used by all Option devices

-O --sony-mode

Apply a special sequence used by Sony Ericsson devices. Im? plies option --check-success

-L --cisco-mode

Send a sequence of bulk messages used by Cisco devices

-R --reset-usb

Send a USB reset command to the device. Can be combined with any switching method or stand alone. It is always done as the last step of all device interactions. Few devices need it to complete the switching; apart from that it may be useful dur? ing testing

-c --config-file FILENAME

Use a specific config file. If any ID or switching options are given as command line parameters, this option is ignored. In that case all mandatory parameters have to be provided on the command line

-f --long-config STRING

Provide device details in config file syntax as a multiline string on the command line

-t --stdinput

Read the device details in config file syntax from standard input, e.g. redirected from a command pipe (multiline text)

-Q --quiet

Don't show progress or error messages

-W --verbose

Print all settings before running and show libusb debug mes? sages

-D --sysmode

Changes the behaviour of the program slightly. A success mes?
sage including the effective target device ID is put out and
a syslog notice is issued. Mainly for integration with a
wrapper script

-s --check-success <seconds>

After switching, keep checking for the result up to the given
time. If target IDs or target class were provided, their ap?
pearance indicates certain success. Otherwise the disconnec?
tion of the original device is rated as likely proof

-I --inquire

Obsolete. Formerly obtained SCSI attributes, now ignored

-i --interface NUM

Select initial USB interface (default: 0). Only for testing
purposes

-u --configuration NUM

Select USB configuration (applied after any other possible
switching actions)

-a --altsetting NUM

Select alternative USB interface setting (applied after
switching). Mainly for testing

AUTHOR

This manual page was originally written by Didier Raboud (di?
dier@raboud.com) for the Debian system. Additions made by Josua Dietze.