TOUCH BASE documentation

Command line user interface

Revision 1.16 – 4th Dec 2014
www.touch-base.com\documentation\General

TButils Tbcalib Contact

TButils user interface

Starting with updd version **4.1.10** a new command line user interface utility is available called tbutils and replaces a number of functions previously located in tbcalib. For users of earlier UPDD versions see tbcalib user interface below.

The user interface program exports this interface using the following syntax:

Windows	C:\program files\updd\TButils {parameter} Note: Entering the commands from a Windows command line would be tbutils "{parameters}"
Mac OS X	5.1.x: tbutils {parameter}
Linux	export LD_LIBRARY_PATH=/opt/tbupddlx:$LD_LIBRARY_PATH /opt/tbupddlx/tbutils {parameter} Or alternatively (if updddenv script exists) /opt/tbupddlx/updddenv tbutils {parameters}
Solaris	cd /opt/tbupddso tbutils {parameter} or /opt/tbupddso/ tbutils {parameter}

Notes:

1. Output is directed to the console (stderr / stdout) allowing for scripted automation.
2. Success is indicated by rc=0, error is rc= -1.
3. When an error occurs a meaningful message is directed to the console (stderr).
4. Passed parameters are checked for correct syntax and values.
5. If running the utility outside the UPDD application folder you may need to add the UPDD path to the system library path, such that the utility can locate the ACE library, as per this Linux example,: export LD_LIBRARY_PATH=/opt/tbupddlx:$LD_LIBRARY_PATH.

Parameters	**Description**
{None}	Lists the command syntax ***and the available commands in the version you are running!***
list	Lists UPDD devices being handled / monitored by the driver. Lists for each device: Internal UPDD handle / device name / desktop segment This example shows a system with 3 touch devices >>>>>>>
	A later version of tbutils also lists the device state as below; This example shows a system with 2 devices in various states >> NOK = device not connected OK = device connected !!! = Driver interface not available (tbupddwu has been stopped)
controllers	Lists the controllers supported by the installed updd driver along with an id. The id shown relates to the controller's configuration id as held in the UPDD settings file, such as TS001, TS002 etc
adddevice	Adds a device instance for the controller with the given id as listed above. If specified the name must be quoted if it contains spaces. If not specified the standard default device naming is used. The port allows the com port to be specified for a serial device. If not specified the default defined for this controller is used. This parameter can be used in any OS but is useful under CE for adding additional touch screens to be supported by the image.
Version	Shows the driver's release information, being; Version number / build id / Production system id
[<device selector>]	Selects the UPDD device against which to perform request. Only required in multi-device environment. ***If no device is specified then device related commands will be applied to the first device listed.***

```
c:\program files\updd>tbutils list
1       BonXeon                 Monitor 1
2       Dell, ST2220T           Monitor 2
3       Quanta Computer, Dual   Monitor 3

c:\program files\updd>_
```

```
Handle  State   Device Name                    Desktop
|       |       |                              |
C:\Program Files\UPDD>tbutils list
1  NOK    Zytronic, ZXY100          Whole
2  OK     Zytronic, ZXY100(2)       Whole

C:\Program Files\UPDD>net stop tbupddwu
The tbupddwu service is stopping.
The tbupddwu service was stopped successfully.

C:\Program Files\UPDD>tbutils list
Unable to open connection to driver
1  !!!    Zytronic, ZXY100          Whole
2  !!!    Zytronic, ZXY100(2)       Whole
```

```
tbutils controllers
001 Zytronic, ZXY100, Serial
002 Zytronic, ZXY100, USB
003 Virtual Device xxxx
```

If the devices are listed as above then:

Add a Zytronic, ZXY100, Serial with a different name on 1

tbutils adddevice 1 "new serial device" COM1

Add a Zytronic, ZXY100, USB with a different name.

tbutils adddevice 2 "new usb device name"

Add a Zytronic, ZXY100, USB keeping the same name.

tbutils adddevice 2

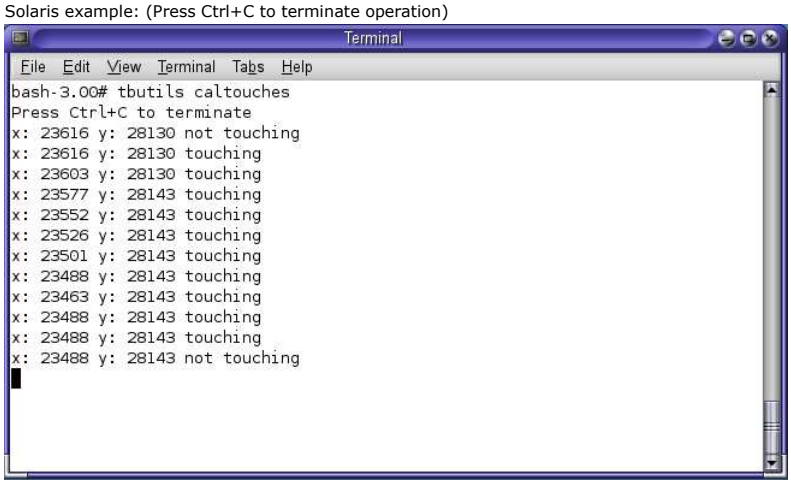
Add a Virtual Device with a different name

tbutils adddevice 3 "Dummy Device"

```
c:\program files\updd>tbutils version
04.01.10R / 2257 / D16195
c:\program files\updd>_
```

Syntax	**Description**
nodevice	The "nodevice" option allows actions on general (non-device specific) parameters [updd/parameters]
segment <name>	Perform request on the updd device associated with the updd desktop segment identifier. e.g. Tbutils segment "Monitor 2" disable - would disable the updd device associated with Monitor 2.
device <handle>	Perform request on the specified UPDD device. Handle is the device handle of the device as held by UPDD and shown in the list command above. e.g. Tbutils device 2 enable - would enable the updd device with updd handle 2 If no device selector is specified and one is required the first installed device is chosen
Connected	Perform request on the first USB connected device. E.g.tbutils connected readeeprom 0 16
global	The "global" option allows actions on system and configuration parameters [updd]. Only available with UPDD release 5.1.x and above.
UPDD specific requests	These requests relate specifically to UPDD functions
calibrate	Performs a calibration procedure appropriate to the device in question

	Cisco DMP4310	Invokes the built in calibration interface to collect calibration data. Calibration pattern defined in the UPDD settings file.
	Others	Invoke a text based top left, bottom right 2 point calibration procedure.
Pointeroff		Disable the driver mouse pointer interface (system wide – all devices)
Equivalent call	API	TBApiMousePortInterfaceEnable(false);
Pointeron		Enable the driver mouse pointer interface (system wide – all devices)
Equivalent call	API	TBApiMousePortInterfaceEnable(true);
Reload		Force the driver to re-read settings (not necessary when using this interface to change a setting)
Equivalent call	API	TBApiReloadNoApply();
toolbaroff <name>		Disable a named toolbar TBApiEnableToolbar
Equivalent call	API	
toolbaron <name>		Enable a named toolbar TBApiEnableToolbar
Equivalent call	API	
Unload		Instruct registered applications to terminate
Equivalent call	API	TBApiSendUnloadMessage
version		Returns the UPDD 3 part build number to stdout e.g.04:01:06R / 1221 / G11951. For backward compatibility this can be redirected to file version.txt if required.
dump4tba <file path>		This option is used to create default calibration data from a calibrated system. If no file path is defined the calibration data is written to file tbcalib.tba in the current folder or the file defined. Defining a path e.g. dump4tba c:\users\gary\tba.dat is useful if the current working folder is not writeable. The data is written in a format suitable for embedding in our software generation system such that the installation utilizes the default calibration data in the UPDD settings files. In this example a system has been calibrated with a 1 percent margin, 8 calibration points, 10 second timeout: Normal,1,8,10,0,0,15790,1223,15768,15642,885,1132,942,15756,12178,4929,12146,12180,4649,4959,465 The file containing the captured calibration data should be sent to Touch-Base for processing.
record <filename>		Records touch co-ordinate input to a file. Recording terminates after about 10 seconds of inactivity or using Ctrl C on the keyboard. Ensure that file name is in a folder that has write access. The file is a csv format: x, y, stylus, z, reserved-for-future-use e.g. 600,253,0,-1,0 Z is not recorded, this has to be manually added. (-1 is "no value"). Only works if the Z (pressure) axis is both supported by the device and in UPDD's configuration for the device.
playback <filename>		Used to playback previously captured touch co-ordinate data using the 'record' function above. Events are played back one every 20ms or so, so the playback might run at a slightly different speed to the recording. Periods of no input are identified by null records thus: -1,-1,-1,-1,0
rawtouches		List the raw co-ordinates values received from the controller and contact states whilst touching.
caltouches		List the calibrated co-ordinates values received from the controller and contact states whilst touching, as per this Solaris example: (Press Ctrl+C to terminate operation)



Use the following options to change arbitrary UPDD settings file entries. UPDD settings are documented in the UPDD settings file. By default the setting changes are applied to Device 1 – this equates to the settings file branch [UPDD/Paramaters/1]. To change settings in the general branch (not related to a device) use 'desktop selector' = nodevice – this equates to the settings file branch [UPDD/Parameters].

Note: Take care when updating the driver settings as setting an invalid setting could result in unpredictable behaviour or a crashed driver! You have been warned.

Some example setting changes and their use are documented here.

setting <name>	<dw>	Set the DWORD setting *name* to the hex numeric *value*

<value> Set the String setting *name* to *value*
<sz>
<name>
<value>

In the main the setting option is used to change UPDD settings but there are a few options that can change general system settings as described here:

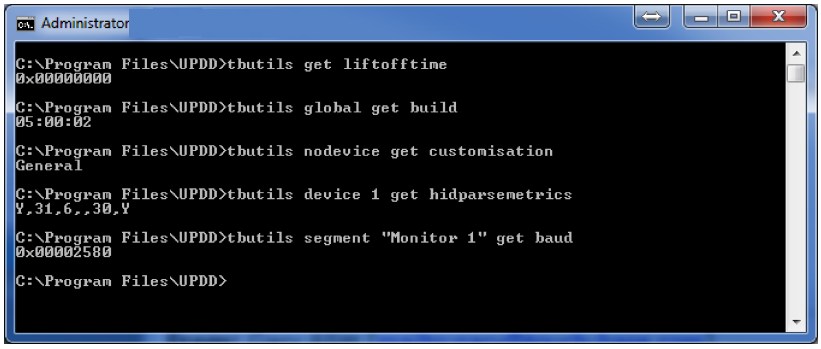
These three mouse settings allow the internal mouse settings to be adjusted:

Windows	DoubleClickHeight	Mouse double click height setting
Windows	DoubleClickwidth	Mouse double click width setting
Windows	DoubleClickSpeed	Mouse double click speed setting

Note: If the setting *name* has a space then it must be quoted, e.g. tbutils nodevice setting dw "calibration beeps" 0. Ensure the " is from 'Shift +2' keystroke. If cut and pasted, say from a Word doc, it may look correct but will not necessarily be the correct code.

get
(Since 5.1.667)

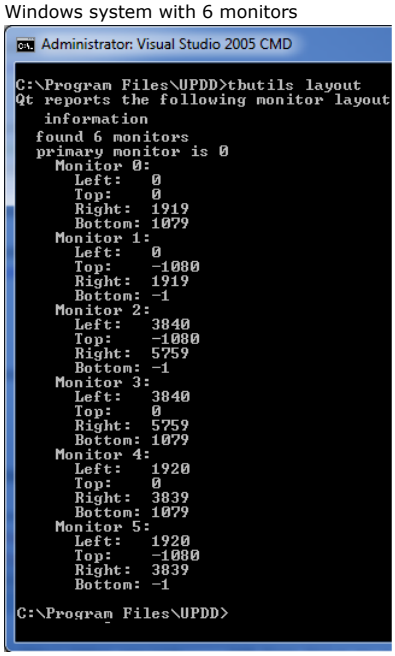
Return the value of a setting. All the usual device selectors are supported, as per these examples:



Layout

Shows system's monitor layout as seen by UPDD software. UPDD 4.1.10 and above utilises QT monitor matrix (QDesktopWidget class) in all OS to retrieve monitor layout information and for monitor handling. For successful UPDD multi-monitor operation the QT monitor properties must see either different monitor ids or co-ordinates. This option lists the monitor configuration as seen by UPDD, as per the examples below:

Windows system with 6 monitors



Linux system with 2 monitors

Note that the monitor numbers listed above are those seen by the Qt API's used by UPDD driver.

UPDD uses a numbering sequence that starts at 1, so in most cases the Monitor selected in UPDD is one more than that shown in the tbutils layout output.

Unfortunately neither of these numbering schemes match those of Windows itself as Windows does not make monitor numbers available to 3rd party applications.

When using extended touch on Windows there is no fixed link between a UPDD monitor number and a physical monitor as a touch device can be remapped by Windows to any physical device.

To simplify setup on Windows systems you should therefore allow the calibration procedure to automatically assign windows desktop association as discussed here for calibrating multiple touch devices.

Mergesettings <filename>
(since 5.1.913)

Used to import driver settings into the main settings file tbupdd.ini, in either section [updd\parameters] or [updd\parameters\N] which must be the first line in the file followed by the settings to import.

e.g. Tbutils mergesettings upddsettings.ini - will merge the settings in the file into the main updd settings file at the branch specified in the file.

[updd\parameters] <<<<<*Lower case required*
doublewordsettingexample=0x00000006
stringsettingexample=hello

Note: Currently not available in UPDD CE

USB device specific (Windows only)

"packet

Starting with UPDD version 5.1.0 the "packet disposition" setting is a controller setting that provides

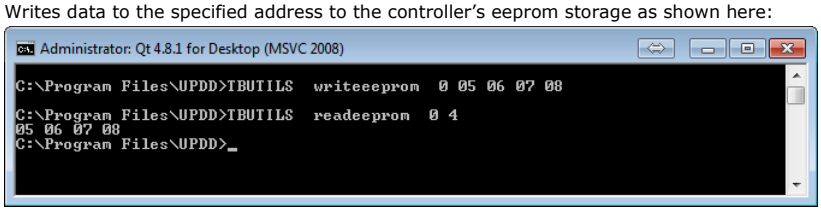
disposition"	information over how the driver will deal with "incoming data" and is described in full here.

These functions have been implemented to facilitate the removal of UPDD control of a USB device without having to uninstall UPDD. This allows for another process to access the USB device, say for maintenance purposes and then revert back to UPDD when required.

hidmode	Unloads the UPDD kernel mode driver and forces the default hid driver to install. This is an asynchronous operation. Return code value 0 indicates success, -1 indicates failure. Example tbutils hidmode tbutils waitdevice 4003 56a hidusb 60 Switches to hid mode and waits for up to 60 seconds for the specified device to be active with the hidusb driver.
upddmode	Reinstalls the UPDD kernel mode driver This is an asynchronous operation. Return code value 0 indicates success, -1 indicates failure. Example tbutils upddmode tbutils waitdevice 4003 56a tbupddsu 60 Switches to UPDD mode and waits for up to 60 seconds for the specified device to be active with the UPDD driver.
Waitdevice	Parameters: <USB Vendor id>, <USB Product id>, <hidmode / upddmode>, <timeout value - seconds>. Waits up to the defined timeout value for the device specified by the vid / pid to be controlled by the specified driver. This is a synchronous operation. Notes: A timeout value of 0 indicates no timeout. Success is indicated by a return code value of 0. Failure (including waitdevice timeout) is indicated by a return code value of -1.

Device specific requests

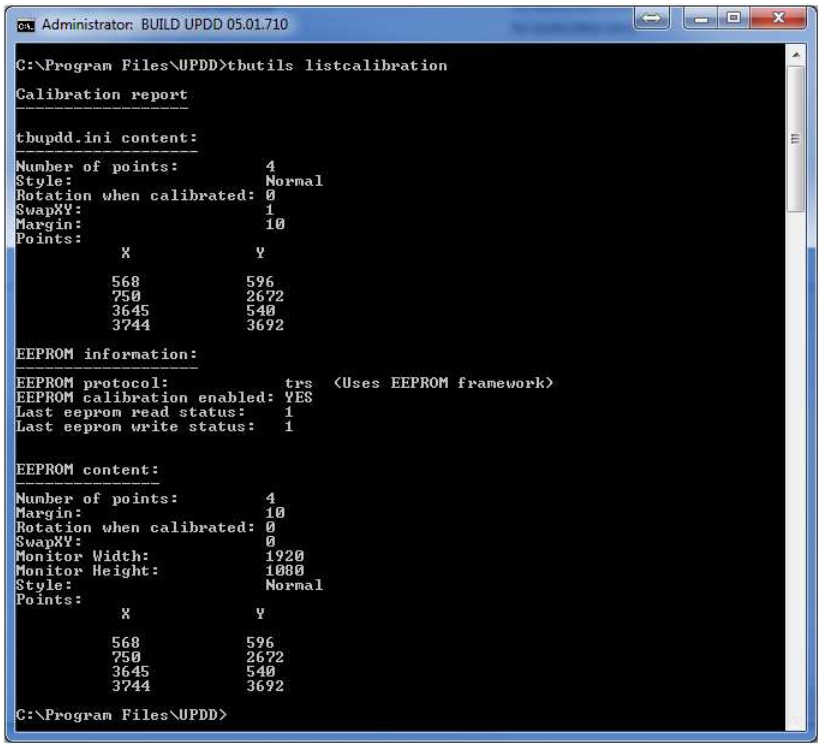
Reinit	Reinitialise the controller and re-establish a link
Equivalent API call	TBApiReinit(passedDeviceNumber);
disable	Disable the device
Equivalent API call	TBApiSetSettingDWORD(passedDeviceNumber,_T("Enabled"),0);
Enable	Enable the device
Equivalent API call	TBApiSetSettingDWORD(passedDeviceNumber,_T("Enabled"),1);
Soundoff	Turn sound off for the device
Equivalent API call	TBApiSetSettingDWORD(passedDeviceNumber,_T("Sound"),0);
Soundon	Turn sound on for the device
Equivalent API call	TBApiSetSettingDWORD(passedDeviceNumber,_T("Sound"),1);
Togglesound	Toggle sound setting for the device
Equivalent API call	DWORD dw; TBApiGetSettingDWORD(passedDeviceNumber,_T("Sound"),&dw); dw ^=1; TBApiSetSettingDWORD(passedDeviceNumber,_T("Sound"),dw); TBApiApply()
readeeprom <addr> <length>	Reads data from the specified address and of the specified length from the controller's eeprom storage and dumps this to the screen (stdout) as shown in screen shot below.
writeeeprom <addr> <data>	Writes data to the specified address to the controller's eeprom storage as shown here:



Notes regarding above eeprom commands:

1. Since 5.0.2 (Oct 13) these commands will work for controllers where the controller supports it ***and*** it is implemented in UPDD driver.
2. Use with caution: writing to eeprom does different things for different controllers and you should only use this command if you understand the consequences.
3. These commands work with controllers that support the new eeprom framework within the UPDD drivers.

listcalibration (since 5.1.710)	This dumps calibration data for the selected device. The tbupddd.ini content is always shown for the device. In addition, the EEPROM information is shown for a device with an eeprom protocol defined. The EEPROM content is shown for a device with an eeprom protocol defined which is implemented in the eeprom calibration framework, so long as there is valid data otherwise an informational message such as checksum error is given. Reading eeprom content with this command does NOT affect the values stored in tbupdd.ini including eepromreadstatus



Zytronic - The following four options are for use with the Zytronic X-Y controllers and **will not work** with the new (Apr 2010) ZY100 controller. These functions are useful in OSes (Windows CE etc) where the UPDD Console, firmware dialog is not available to make the settings.

zyavframes=n	Set number of frames for X / Y averaging in Zytronic X-Y controllers. Range 0 to 9.
zyglasstype=n	Set the glass thickness in Zytronic X-Y controllers The controller can be adjusted using this setting to operate through various overlay thicknesses. Available options are Thin, Medium and Thick. The Medium setting is the default. These settings operate on time averaging of captured data from the sensor, hence the thicker the overlay, the sensor response time is reduced due to the greater time interval of data captured. These options should be used in conjunction with the Threshold (Sensitivity) setting adjustment to obtain optimum operation when using various thicknesses of overlays. Range 0 (Thin), 1 (Medium) and 2 (Thick).
zynormalisation	Initiates a normalisation of the sensor array wire levels in Zytronic X-Y controllers.
zysensitivity=nn	Set touch sensitivity in Zytronic X-Y controllers. Range 0 to 50.

TRS - The following requests are for use with the TRS Star controllers. Most of these functions have been implemented with an equivalent UPDD API. Should you be advised how to use these functions at an API level you will need you use a TRS specific header file.

trsset	Further information available from TRS

trsget	Further information available from TRS
	The functions will retry a number of times if there is any error, which could be caused because the device does not exist.
	Some users, particularly in CE systems, might need to know if the controller is connected and may implement a heartbeat process to check a serial device exists. In this case reties and long timeout can cause an issue. At an API level we have added a setting to indicate no retries and short timeout for one specific call, Get Family Code, so it can be used to determine if the device is connected. The code to perform this check is as follows:
	`WORD val=0xff; // special option to indicate no retries and short timeout`
	`if(!TBApiGetTRSControllerOption(gSelectedDevice,TRS_OPTION_FAMILY_CODE,&val))`
	`{`
	`cerr << endl << _T("Controller not detected") << endl;`
	`return(-1);`
	`}`
	Testing for the presence of a USB controller is normally achieved with the tbapigetiproduct API.

trsrecalibrate	Further information available from TRS
trsrestart	Further information available from TRS
trsisp	Further information available from TRS
trsselftest	Further information available from TRS
readeeprom	4.1.10. For 5.0.2 and above see generic implementation of this command above Further information available from TRS
writeeeprom	4.1.10. For 5.0.2 and above see generic implementation of this command above Further information available from TRS Note: When used in WEC/ARM7 writing failed when writing large data blocks failed but worked OK when writing smaller data blocks it worked, so tbutils writeeeprom 128 00 00 A8 00 01 00 02 81 01 02 00 02 08 00 07 00 03……. Failed, but tbutils writeeeprom 128 00 00 A8 00 01 00 02 81 01 02 tbutils writeeeprom 138 00 02 08 00 07 00 03 FF FF FF worked. We are not sure what is the max size that works.

ELO - The following two requests are for use with ELO Smartset controllers.

smtwrite:nnnnnn	Set the serial number to nnnnnn on the ELO Smartset controller. Note: Changing the serial number causes UPDD to see a new device, so an additional device will be listed in the UPDD Console device list when the controller reports its serial number (this appears to be after rescan of devices, such as a replug or a reboot).
Equivalent API call	
BOOL TBAPI TBApiWriteSmartsetUSBSerialNumber(HTBDEVICE aDevice, const TCHAR* aBuffer, DWORD aSize);	
smtread	Read the serial number from the ELO Smartset controller and send to stdout. (Only use one ELO Smartset controllers, otherwise the behavior is undefined). For backward compatibility this can be redirected to file smtread.txt if required.
Equivalent API call	
BOOL TBAPI TBApiReadSmartsetUSBSerialNumber(HTBDEVICE aDevice, TCHAR* aBuffer, DWORD aSize);	
Hampshire - The following two requests are for use with Hampshire/Microchip tsharc controllers.	
tsharcwrite:n	Write the serial number n to the Hampshire TSHARC controller. (Only use one controller, otherwise the behavior is undefined).
Equivalent API call	
BOOL TBAPI TBApiWriteTSHARCUSBSerialNumber(HTBDEVICE aDevice, const TCHAR* aBuffer, DWORD aSize);	
tsharcread	Read the serial number from eeprom on the Hampshire TSHARC controller and send to stdout (Only use one controller, otherwise the behavior is undefined). For backward compatibility this can be redirected to file tsharcread.txt if required.
Equivalent API call	
BOOL TBAPI TBApiReadTSHARCUSBSerialNumber(HTBDEVICE aDevice, TCHAR* aBuffer, DWORD aSize);	

Error messages

Error messages issued by the utility are as follows:

Unable to open connection to driver	Tbutils communicates with the main driver process tbupddwu using tcp/ip on port 4141. This error means that one of the following situations exists. 1) Tbupddwu is not running 2) Port 4141 is blocked (for local traffic) 3) Tbupddwu is stalled in some way. Check that tbupddwu is running#; under Linux using the command - ps –ax | grep tbupddwu or task Manager under Windows or Activity Monitor under Mac OS X

Program execution

There may be occasions where it is appropriate or convenient to call TButils from a script or program to change UPDD settings rather than using the driver's API. However, because tbutils is a console utility expecting to be invoked from a terminal window, you may, depending on how it is invoked, briefly see a console window when the command is invoked. With each OS there are numerous ways you can invoke the program to run silently in the background and here are a few that work in Windows:

Example running TButils from a batch script
This command suppressed the terminal window:

```
start /b cmd /c ""c:\program files\updd\tbutils.exe" device 1 setting sz "logical desktop segment" "Monitor 2""
```

CreateProcess code example that suppressed the terminal window:

```
STARTUPINFO s;
 GetStartupInfo(&s);
s.dwFlags = STARTF_USESHOWWINDOW;
s.wShowWindow = SW_HIDE;
PROCESS_INFORMATION pi;
      BOOL res = ::CreateProcess(NULL,
                        pgm,NULL,NULL,FALSE,CREATE_NEW_PROCESS_GROUP,
                        NULL,aInst16Dir,&s,&pi);
```

Tbcalib user interface

In UPDD versions up to and including **4.1.8** user interface calls were held in Tbcalib. With 4.1.10 and above they were relocated to a new utility program, tbutils, as documented above.

The calibration program exports this interface using the following syntax:

Windows	TBcalib {parameter} Note: Entering the commands from a Windows command line would be tbcalib "{parameters}"
Mac OS X	/tbupddmx/tbcalib.app/Contents/MacOS/tbcalib {parameter}
Linux	/opt/tbupddlx/upddcalib {parameter} or /opt/tbupddlx/tbcalib {parameter} This command may need to be run prior to calling tbcalib: export LD_LIBRARY_PATH=/opt/tbupddlx:$LD_LIBRARY_PATH'

Notes:

1) If the parameter affected has a space then the parameter value must be quoted, e.g. tbcalib Device=0 "/setting:calibration beeps=0". ***In some cases we have seen " ignored and ' have worked! Please try ' if " cause issues.***

2) **Win7 file write issue**: Some of the user interface calls create files in the UPDD application folder and under Windows 7 this folder may not have correct write permissions to allow for files to be created. In this case you may see an error or you may not find the file (it will be remapped elsewhere). When using functions that create files ensure you have administration rights.

Please note parameters are case sensitive and must be defined as shown below.

Calibration parameters	Used with the calibration procedure
None passed	will calibrate the first active device on the system.

Device=n	perform request on the specified updd device and, if calibrating, the currently selected calibration style, default first in list. Will also calibrate any defined toolbars unless 'Toolbar=ABogusValue' is used to disable toolbar processing. Normally used by calling programs to perform a given function against a specific device, such as the UPDD Console device calibration option. N=the device handle of the device as held by UPDD. This option is used by UPDD SDK based programs tilizing the UPDD API to determine the device handle using related API calls such as TBAPIGetRelativeDevice.
Device=connected	Perform request on the first connected device.
"Segment=segment id" (UPDD ver 4.1.3 and above)	Perform request on the updd device associated with the updd desktop segment identifier and, if calibrating, the currently selected calibration style, default first in list. Will also calibrate any defined toolbars unless 'Toolbar=ABogusValue' is used to disable toolbar processing Normally used by calling programs to perform a given function against a specific device, such as the UPDD Console device calibration option. e.g. Tbcalib "Segment=Monitor 2" /disable - would disable the updd device associated with Monitor 2.

User Interface Calls	***When TBcalib is invoked with a user interface parameter only the function associated with the parameter is performed. As would be expected, calibration is not invoked.***
Device=n	See above definition.
Segment=" segment id"	See above definition.
/reinit	Reinitialise the controller and re-establish a link
Equivalent API call	TBApiReinit(passedDeviceNumber);
/reload	Force the driver to re-read settings (not necessary when using this interface to change a setting)
Equivalent API call	TBApiReloadNoApply();
/toolbaroff:toolbarname	Disable a named toolbar
Equivalent API call	TBApiEnableToolbar
/toolbaron:toolbarname	Enable a named toolbar
Equivalent API call	TBApiEnableToolbar
/toggletouch	Toggle the device enabled state
Equivalent API call	DWORD dw; TBApiGetSettingDWORD(passedDeviceNumber,_T("Enabled"),&dw); dw ^= 1; TBApiSetSettingDWORD(passedDeviceNumber,_T("Enabled"),dw);
/enable	Enable the device
Equivalent API call	TBApiSetSettingDWORD(passedDeviceNumber,_T("Enabled"),1);
/disable	Disable the device
Equivalent API call	TBApiSetSettingDWORD(passedDeviceNumber,_T("Enabled"),0);
/pointeroff	Disable the driver mouse pointer interface (system wide – all devices)
Equivalent API call	TBApiMousePortInterfaceEnable(false);
/pointeron	Enable the driver mouse pointer interface (system wide – all devices)
Equivalent API call	TBApiMousePortInterfaceEnable(true);
/soundon	Turn sound on for the device
Equivalent API call	TBApiSetSettingDWORD(passedDeviceNumber,_T("Sound"),1);
/soundoff	Turn sound on for the device
Equivalent API call	TBApiSetSettingDWORD(passedDeviceNumber,_T("Sound"),0);
/togglesound	Toggle sound setting for the device
Equivalent API call	DWORD dw; TBApiGetSettingDWORD(passedDeviceNumber,_T("Sound"),&dw); dw ^=1; TBApiSetSettingDWORD(passedDeviceNumber,_T("Sound"),dw); TBApiApply();
/screenresupdate	MAC OS X only – Requests the driver to recalculate calibration mapping based one the current screen resolution. To be used where a system is calibrated in one resolution but uses other resolutions (especially useful where applications are changing resolution)

Equivalent API call	DWORD nDevices;
	TBApiGetSettingDWORD(0, _T("Number Of Devices"), &nDevices);
	for(unsigned j = 0; j < nDevices; ++j)
	{
	int dev=0;
	dev = TBApiGetRelativeDevice(j);
	if(!dev)
	{
	continue;
	}
	else
	{
	SetupForMultiMonitor(dev,this);
	}
	}

Use the following options to change arbitrary UPDD settings file entry. UPDD settings are documented in the UPDD settings file. By default the setting changes are applied to Device 1 – this equates to the settings file branch [UPDD/Paramaters/1]. To change settings in the general branch (not related to a device) use Device=0 – this equates to the settings file branch [UPDD/Parameters].

/setting:XXX=NNN	Set the DWORD value XXX to the hex numeric value NNN
/settingsz:XXX=ZZZ	Set the string value XXX to the value ZZZ

Note: If the setting name has a space then the option must be quoted, e.g. tbcalib Device=0 "/setting:calibration beeps=0".
See note 1 above for important information!

*The following four parameters are for use with the Zytronic X-Y controllers and **will not work** with the new (Apr 2010) ZY100 controller. These functions are useful in OSes (Windows CE etc) where the UPDD Console, firmware dialog is not available to make the settings*

zysensitivity=nn	Set touch sensitivity in Zytronic X-Y controllers. Range 0 to 50.
zyavframes=n	Set number of frames for X / Y averaging in Zytronic X-Y controllers. Range 0 to 9.
zyglasstype=n	Set the glass thickness in Zytronic X-Y controllers The controller can be adjusted using this setting to operate through various overlay thicknesses. Available options are Thin, Medium and Thick. The Medium setting is the default. These settings operate on time averaging of captured data from the sensor, hence the thicker the overlay, the sensor response time is reduced due to the greater time interval of data captured. These options should be used in conjunction with the Threshold (Sensitivity) setting adjustment to obtain optimum operation when using various thicknesses of overlays. Range 0 (Thin), 1 (Medium) and 2 (Thick).
zynormalisation	Initiates a normalisation of the sensor array wire levels in Zytronic X-Y controllers.
/smtwrite:nnnnnn	Set the serial number to nnnnnn on the ELO Smartset controller. Note: Changing the serial number causes UPDD to see a new device, so an additional device will be listed in the UPDD Console device list when the controller reports its serial number (this appears to be after rescan of devices, such as a replug or a reboot).
Equivalent API call	BOOL TBAPI TBApiWriteSmartsetUSBSerialNumber(HTBDEVICE aDevice, const TCHAR* aBuffer, DWORD aSize);
/smtread (Win 7? – See note)	Read the serial number from the ELO Smartset controller and dump to the file smtread.txt (Only use one ELO Smartset controllers, otherwise the behavior is undefined).
Equivalent API call	BOOL TBAPI TBApiReadSmartsetUSBSerialNumber(HTBDEVICE aDevice, TCHAR* aBuffer, DWORD aSize);
/tsharcwrite:n	Write the serial number n to the Hampshire TSHARC controller. (Only use one controller, otherwise the behavior is undefined).
Equivalent API call	BOOL TBAPI TBApiWriteTSHARCUSBSerialNumber(HTBDEVICE aDevice, const TCHAR* aBuffer, DWORD aSize);
/tsharcread (Win 7? – See note)	Read the serial number from eeprom on the Hampshire TSHARC controller and dump to the file tsharcread.txt (Only use one controller, otherwise the behavior is undefined).
Equivalent API call	BOOL TBAPI TBApiReadTSHARCUSBSerialNumber(HTBDEVICE aDevice, TCHAR* aBuffer, DWORD aSize);
/version (Win 7? – See note)	Available in 4.1.6, (build 1221 and above), returns the UPDD version number in a text file called version.txt as a 3 part build number: e.g.04:01:06R / 1221 / G11951

Tbcalib return codes

These are the return codes from TBcalib and access to the code will be specific to the launch method used:

0	Success
4	Syntax error passing parameter
5	Failure to open API
6	Couldn't find a desktop segment ZY = specific to Zytronic X-Y controllers
7	ZY Value passed for sensitivity setting out of range
8	ZY Set sensitivity failed
9	ZY Value passed for glasstype setting out of range
10	ZY Set glasstype failed
11	ZY Value passed for average frames setting out of range
12	ZY Set averaged frames failed
13	ZY Set normalisation failed

Contact

For further information or technical assistance please email the technical support team at technical@touch-base.com.