TOUCH BASE documentation

UPDD features, functions and issues

Revision 1.23 – 27th October 2014
www.touch-base.com\documentation\general

Double Click issues	Serial port auto detection	Right Click generation	Controller naming	Daemon task	Pen up processing
Live Annotation	Touch logging	Custom Settings	Event Selector	Touch Mouse	Touch Pad
Sound	Disabling Touch	Security feature	Virtual Devices	Cursor Utility	Data Export
Dynamic Monitor Tracking	Touch Filtering				Contact

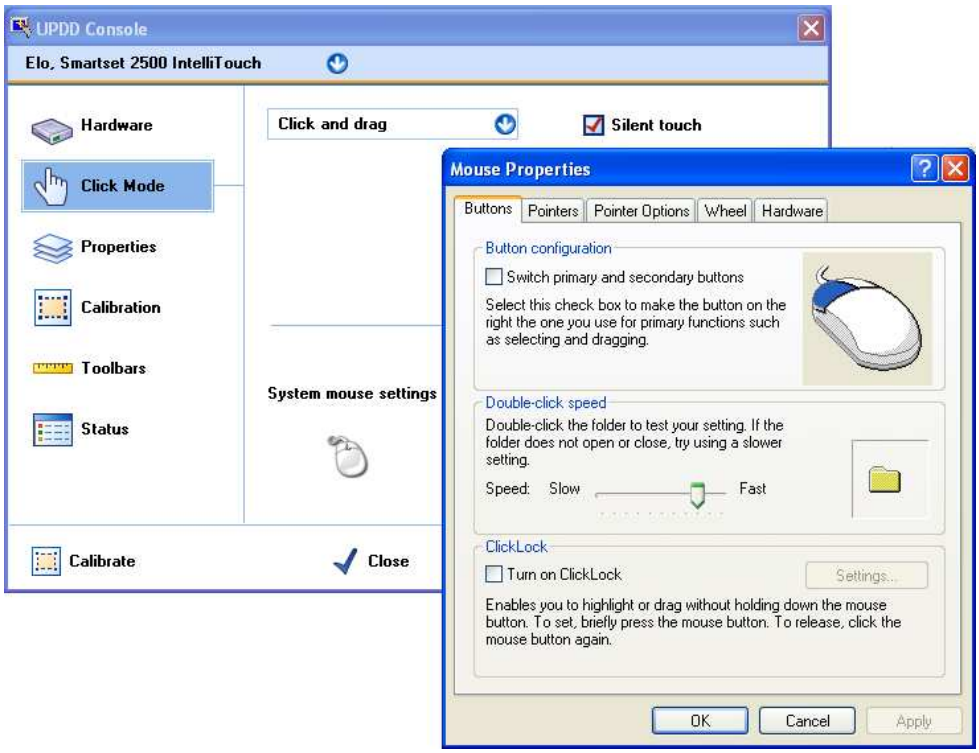
Some features and functions are referenced in the UPDD documentation that requires further explanation. Here it is!

Double Click issues

Double clicking on a touch screen relies on double click related settings set to a tolerance that allows a touch screen to be used to generate the double click. In Windows, for example, there are 3 such settings, DoubleClickSpeed, DoubleClickWidth and DoubleClickHeight. These settings establish the width of the rectangle that Windows uses to detect double-clicks. If sequential clicks are positioned within the rectangle defined by DoubleClickHeight and DoubleClickWidth, and occur within the time specified by DoubleClickSpeed they are interpreted as double-clicks. Otherwise, they are interpreted as sequential single clicks.

Windows

When UPDD is installed it sets these internal settings (for the user performing the install) to values that cater for finger double click usage. In our old version 3 driver dialog we allowed these system setting to be changed. However, our version 4 dialog calls the system's mouse dialog to change double click settings, as seen below:



Most of these mouse dialogs allow the double click speed setting to be changed and this must be a slow (ish) setting to allow for touch double clicks. Too fast a setting will prevent double clicks. The other related settings can be found in the registry at HKCU\Control Panel\Mouse. The Width and Height default settings are 4 pixels and the speed is 500 milliseconds. UPDD sets these settings to 32 and 370 respectively.

New user issue

If a new user logs in they will get the default Windows double click settings which may make it difficult to double click. With UPDD version 4.1.6, build 1156 and above, the UPDD daemon task monitors the users. When a new user logs in it automatically sets these settings.

Settings changed by 3rd party software

There have been cases of double click difficulties following the installation of mouse drivers that have set these settings to values that prevent touch double clicks requiring the setting to be reset to 'touch' values.

Disabling double click

This is not a driver issue but a system issue. If touch double click is to be disabled we suggest that the settings be set such that it is almost impossible to achieve a double click with touch. Try Height and Width = 1 and Speed = 200 – see below.

Register settings

Should you need to manually set the registry for the Double Click Settings these lines can be copied to a .reg file and double clicked to update the registry settings:

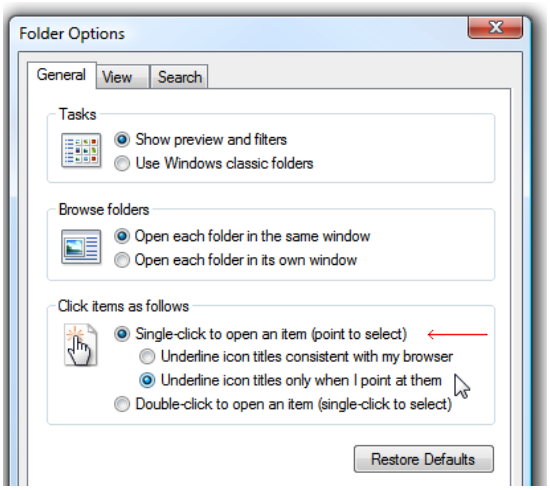
```
Windows Registry Editor Version 5.00
[HKEY_CURRENT_USER\Control Panel\Mouse]
"DoubleClickHeight"="32"
"DoubleClickSpeed"="370"
"DoubleClickWidth"="32"
```

On Windows XP the DoubleClickSpeed ranges from 200 (Fast) to 900 (Slow)

If using UPDD version 5.0.x you can use our command line processor o change these values, such as "tbutils nodevice setting dw DoubleClickHeight 64".

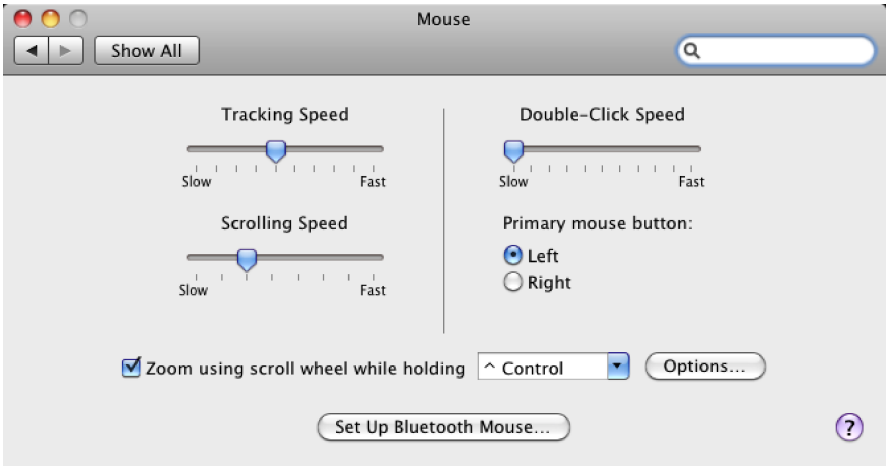
OS double click options

One potentially useful setting in Windows, when using the desktop via touch, is to invoke icons, folders and files using a single click rather than double click. This setting is located in the folder options as shown below:



Mac OS X

Under Mac the Mouse dialog is as follows:



UPDD generated double click settings

For a future release of UPDD we are considering processing our own double click setting for situations where it is still difficult to achieve a double click using system DC setting with certain touch devices such as slow infra red devices. In this case where UPDD DC setting deduce that a double click is required then two complete click sequences will be sent at the same location very quickly which will result in a double click.

Serial port auto detection

Since UPDD 4.1.6, build 1156, a serial controller can be set for auto-detection rather than set to a specific com port. This option is only available, and enabled, where a serial controller can receive and acknowledge a firmware command. Controller commands are held in UPDD macros. Typically a macro sequence used to detect a controller would look like this:

[HEX]0A0141 //Check Controller Activity
[ACK]0A0141

In this example the driver sends 0A0141 to the controller which is echoed by the controller by way of acknowledging receipt.

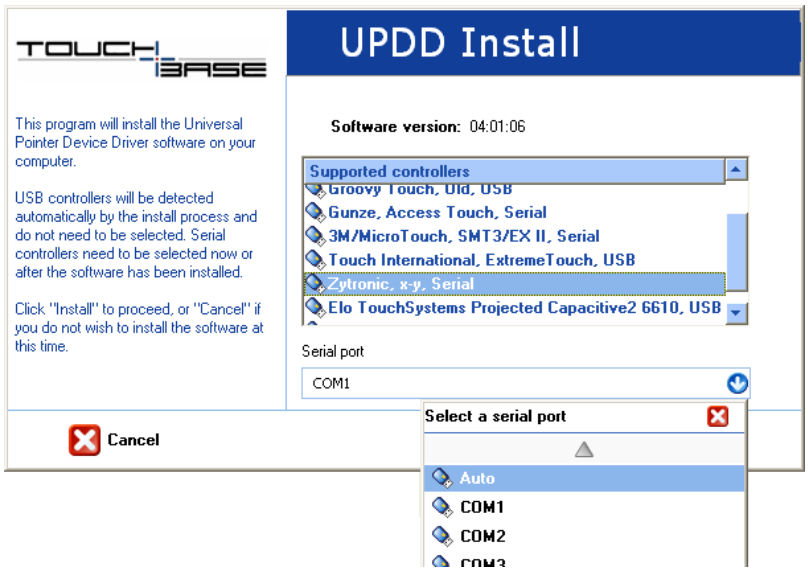
Some macros are needed for controller initialization rather than just containing commands to illicit a response but as long as the controller responds to the received commands then the macro can be used to detect the controller.

When the controller is set for Auto detect the driver issues the macro to each com port in turn looking for the response. If a response is received then the com port is set accordingly. If no response is seen from any port then the com port is set to 'None'. Each time UPDD is loaded it will automatically detect the device but on 2nd and subsequent attempts the detection will initially check the current port setting before checking thro each port in sequence.

Setting serial controller for auto detection

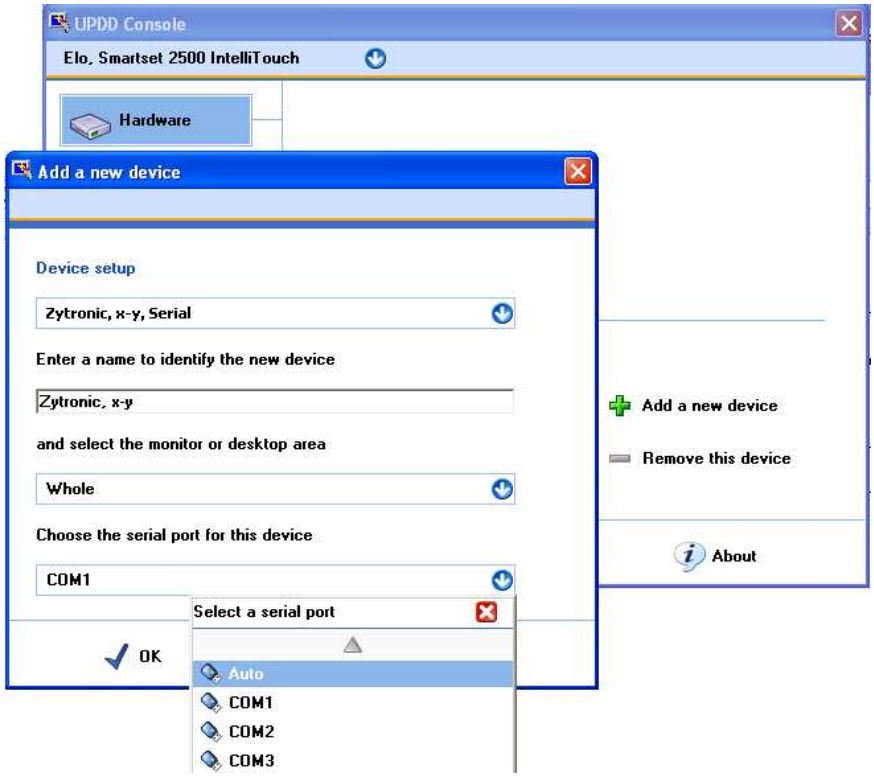
There are various ways to indicate a serial controller is to be auto-detected as follows:

Set in build	A controller is defined in our production system as auto-detected such that is the default settings in all dialogs below.
During installation	The controller is selected from the list and the com port set to Auto:



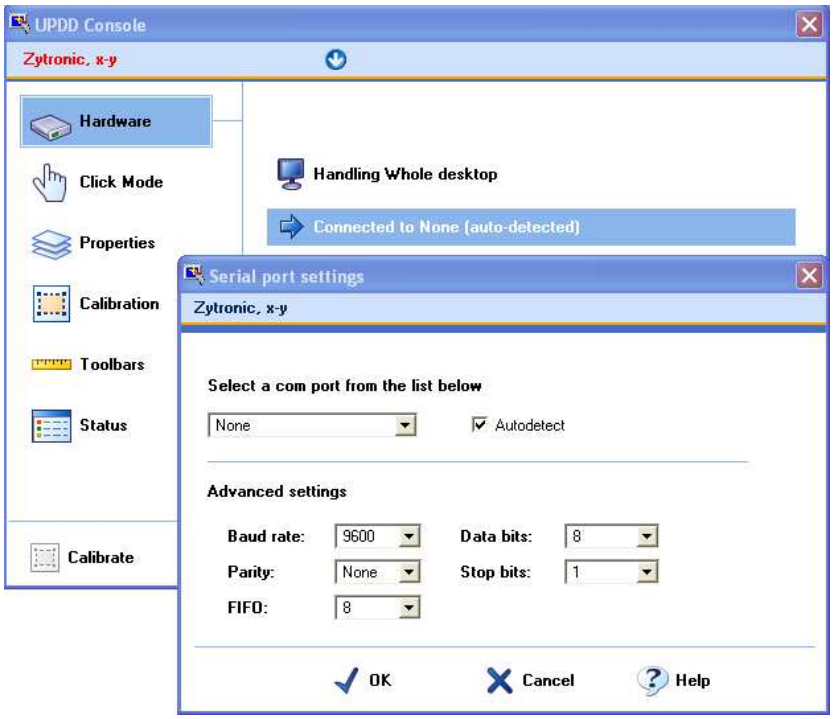
Adding a device in UPDD Console

When adding the device the com port is set to Auto:



Serial port settings

The serial port settings dialog shows the current port placement and indicates if this port has been auto-detected. In the example below the port is auto-detected but has not been found:



Right Click processing

In the majority of cases touching a touch screen or similar device will normally be the equivalent of generating a mouse left click. However, in some cases it is also desirable to be able to generate a right click via touch. With UPDD there are a number of possible methods; Right click trigger in the data stream, the Event Selector (to indicate if the next touch should generate a left or right click), using Interactive Touch mode (which caters for both left and right clicks) or using the right click mechanism built into the operating system; as described below.

Where Extended touch feature is enabled and thus utilising the OS in-built right click functions any UPDD native right click processing is disabled.

Right Click data trigger

Some devices, such as a Pen, may have a barrel switch that when pressed will alter the data packet delivered to the driver to indicate the switch is depressed. Our driver can be configured to act upon triggers in the data packet which can be set to perform certain functions, including Right click. If a device is configured in such a way and the data trigger is assigned to a right click function then pressing the switch will generate a right click.

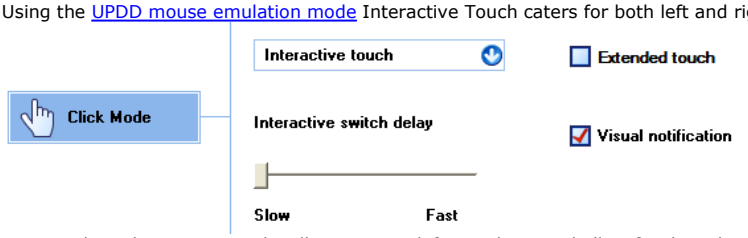
Event Selector

Since UPDD 5.1.1143 no longer available

The UPDD Event Selector, in its most simple terms is a user controlled trigger that indicates to the driver if the next touch should be a left or right click. Click the link for more details.

Interactive Touch

Using the UPDD mouse emulation mode Interactive Touch caters for both left and right clicks.



In normal touch usage a touch will generate a left pen down and allow for drag then will generate a left pen up at lift off. However, holding the stylus stationary invokes right click. The Interactive switch delay setting determines the hold delay and the visual notification setting indicates if visual feedback is shown during hold and right click, as shown. Visual feedback is Windows only until UPDD 5.1.xxxx which offers visual feedback in all OS.

Stationary Stylus
With Interactive Touch set we are looking for X/Y co-ord movement <1000 based on a Virtual desktop matrix of 65535 x 65535 which equates approx to 10 pixel square on 640 x 480, even smaller on greater resolutions.

Visual Notification settings

Starting with UPDD 5.1.1112 the visual notification feature is handled by the driver's cross platform background task, Aidaemon, and available in all desktop OS. The previous visual notification was a series of dots whereas the new one is a smooth line circle as shown.

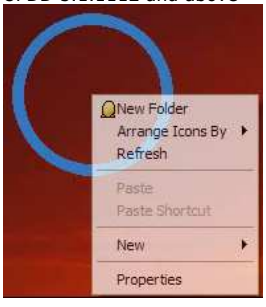
Two UPDD settings dictate the pen width and colour as follows:

The pen (line) width defaults to 9 pixels. This can be changed with the nodevice setting interactivetouchpenwidth

E.g. tbutils nodevice setting dw interactivetouchpenwidth 12

Visual notification
UPDD 4.1.10 / 5.0.2
Windows only

UPDD 5.1.1112 and above



The pen (line) colour defaults to RGB 56,120,190 (a pastel blue used in Windows 7)

This can be changed with the nodevice setting interactivetouchpencolour

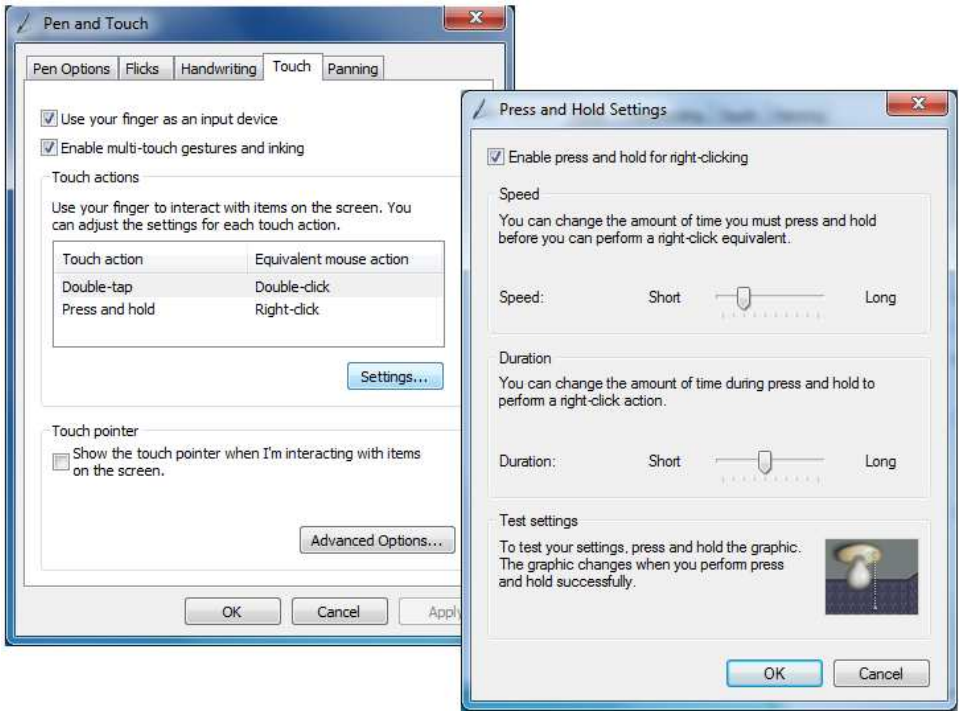
E.g. tbutils nodevice setting sz interactivetouchpencolour "56,120,0"

Operating System feature

The operating system may cater for right click generation using a pen or touch device and in some cases this can be utilised by the driver, as discussed below:

Windows

If using UPDD 4.1.8 and above in touch enabled versions of Vista or Windows 7 there is a setting in UPDD, called Extended Touch, that passes touch data to the OS such that all touch features built into the OS are available to the touch user. In the Pen and Touch system settings you will find some settings specific to touch and you can associate a right click to one of the actions as shown in this Windows 7 example:



In this example, the Touch action 'Press and Hold' has been associated with the Right-click function. During the hold phase a circle will be drawn and when complete a right click will be generated. The circle drawn on a pen device is smaller than the circle drawn on a touch device. Pen and Touch settings dictate the hold time needed to initiate a right click.

Important note:

It is our observation that for Windows gestures to work correctly, especially the hold right click feature, the touch device must deliver a stream of data at a certain rate of delivery. Cases whereby the data deliver rate is slower than that expected can cause Windows to reject the touch and not perform the gesture. This can happen in two cases:

1) Slow data rate – The controller is sending data at a rate slower than that expected.

2) Delta Mode Controller – Running in this mode of operation means that controllers do not send out co-ordinate data when there is no motion, i.e. a steady touch.

Starting with UPDD 5.1.1023 the driver can inject padding packets in the touch data posted to the OS so that the rate of transmission is achieved. This function is enabled with the following settings:

1) Slow data rate – The setting Slow Controller should be set.
e.g. tbutils setting dw "Slow Controller" 1 for device 1

2) Delta Mode Controller – The Liftoff Time Setting in the UPDD Console should be set to 0.

Important technical note: A customer using UPDD in Windows 7 extended touch mode reported that the right clicked worked everywhere except his "MFC control that is hosted in a .net application. No part of an MFC dialog that is hosted on a .NET form receives the right click. The .net form (i.e. title bar) can receive right clicks". He discovered that the issue was related to "MFC code compiled in Visual Studio 2010. Code compiled in VS2008 can receive touch right clicks fine, code compiled in VS2010 doesn't get any."

We suggested that he needed to respond to WM_TABLET_QUERYSYSTEMGESTURESTATUS as described here:
http://msdn.microsoft.com/en-us/library/bb969148(v=vs.85).aspx

He reported that "By adding the following code to my dialog class it now accepts touch screen right clicks:

```
ON_MESSAGE(WM_TABLET_QUERYSYSTEMGESTURESTATUS, OnTabletQuerySystemGestureStatus)
```

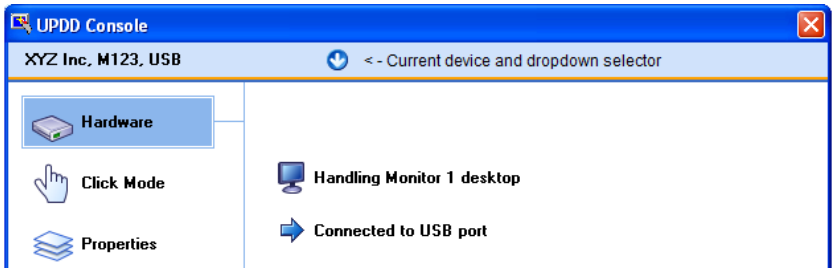
And

```
LRESULT CRightClickTestMFCDlg::OnTabletQuerySystemGestureStatus(
   WPARAM /*wParam*/, LPARAM /*lParam*/)
{
  return 0;
}"
```

Controller naming

In our controller definition database we allocate default controller names to help identify the manufacturer, model and interface, e.g XYZ Inc, M123, USB or a name as requested by the manufacturer or system integrator.

In the UPDD Console there is a Device list entry that shows the currently selected device and, if more than one device is configured on the system, offers a dropdown list to select other devices.



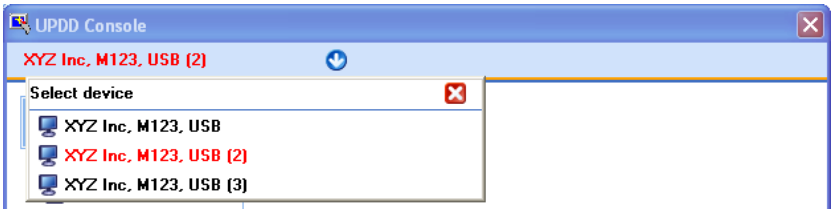
If another controller of the same type is plugged in then the driver will allocate a unique name based on the name of the controller along with a bracketed number i.e. XYZ Inc, M123, USB (2), XYZ Inc, M123, USB (3) etc

Each controller discovered on the system is allocated a unique bind key to help associate and identify UPDD controller settings with individual controllers as they are unplugged, replugged or rediscovered, such as after a reboot.

In a 3 touch monitor device with all devices connected the device list would be seen as:

XYZ Inc, M123, USB
XYZ Inc, M123, USB (2)
XYZ Inc, M123, USB (3)

Any PnP devices that are unplugged would have their name displayed in red. So unplugging (2) would result in the device list as follows:



If the controller is subsequently reconnected, then the driver will calculate the bind key to find the previous entry and settings and re associate the device with its previous settings.

However, if the device in red is deleted from the UPDD console (using the 'Remove a device' option), leaving two entries:

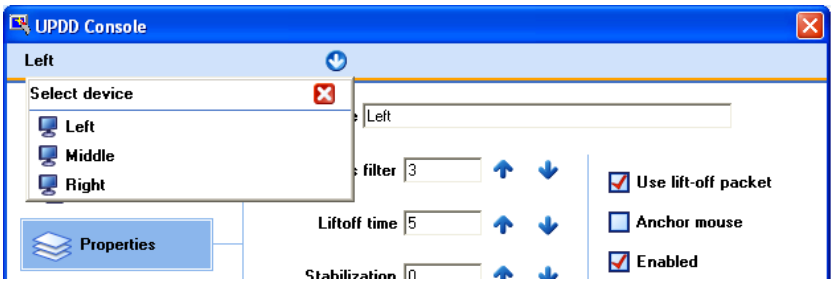
XYZ Inc, M123, USB
XYZ Inc, M123, USB (3)

then the next time the device is connected no previous settings will be found and a new device entry will be created. Given (2) is the next unique device name then the device will reappear as XYZ Inc, M123, USB (2) but will be allocated default settings which may need manually adjusting.

Our recommendation is to not delete devices if they are to be reconnected at some future date.

Controller renaming

Controller names can be manually adjusted once the device is listed in the UPDD Console device list. Simply select the device, click on Properties and change the name. In a three touch monitor system you could name the controllers to reflect their position on the desk, say Left, Middle, Right as shown below:



In this instance, plugging in a 4th controller would result in the name of the new controller being XYZ Inc, M123, USB.

Daemon task

UPDD utilises two daemon (background) processes;
TBdaemon, dedicated to Windows background functions
and
Aidaemon – that caters for cross platform functions or functions that will eventually become available across all platforms.

Aidaemon

Implements functions that are used across all platforms.

Under Mac and Linux appropriate startup methods are used.

Under Windows, in non Windows 8 systems, the daemon task is invoked at system startup by way of a registry entry in the Windows run branch HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run.

In Windows 8 aidaemon is launched as a task to satisfy the load sequence and privilege requirements of the specialised touch interface. The following commands can be used to manipulate this task;

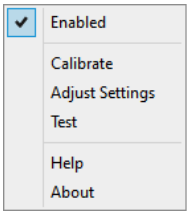
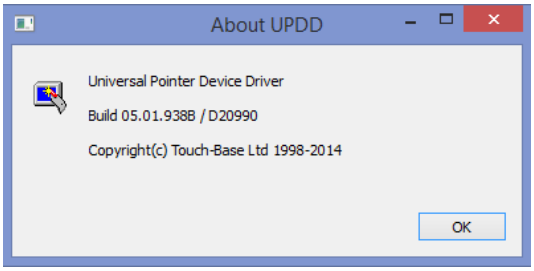
To stop the task use the command: schtasks /tn aidaemon /end

To start the task: schtasks /tn aidaemon /run

To remove the task: schtasks /tn aidaemon /delete

To install the task: schtasks /tn aidaemon /create /xml aidaemon.xml

The daemon task is responsible for the following functions:

System tray icons (since release 5.1.938)	Can be enabled or disabled in the settings file. Some builds are distributed with them disabled by default. Setting to enable (1) or disable (0) is as follows: tbutils nodevice setting dw "show systray" 1 or 0
	 Menu: Shows the UPDD System tray menu below.
System tray menu (since release 5.1.938)	 Enabled: Shows if the device is enabled at a UPDD level. Enabled state can also be switched. Devices can be enabled or disabled at the driver level irrespective of their physical state. Directly relates to the same setting in the UPDD Console, Properties page. Calibrate: Invoke calibration for a device. Adjust settings: Invoke the UPDD Console. Test: Invoke the test and drawing program for a device Event Selector: Toggle the desktop Event Selector function. Help: Load the driver's help files. About: Release information. The build id, as seen below, indicates the version (05.01.) the build no. (938) and release status 'B' within the release and the production system 'Dnnnnn' that generated the build: 
Toolbar images	Draws any images associated with Toolbar cells
Sound playback	Introduced in 4.1.10 plays the sound files associated with touch or calibration
Security feature	Introduced in 4.1.10 monitors the activity of defined programs and if inactive blocks touch. Windows only
Rotate	Informs the driver if a desktop is rotated – Linux and Mac only. Since UPDD 5.1.1130 also Windows.
Eeprom calibration data retrieval	Since UPDD 4.1.8, build id 1738 eeprom calibration data is automatically retrieved at driver startup or when USB devices are plugged in as long as UPDD eeprom calibration setting is enabled for the controller. In these cases the daemon procedure is invoked by the driver to issue the calibration eeprom retrieval function. ***Since UPDD 5.0.2 this functionality moved from Tbdaemon to Aidaemon.***
Video resolution monitoring	Introduced in 5.0.2 monitors the change of video resolution and adjusts calibration accordingly. Linux and Mac only. Windows uses different mechanism.
Mouse interface	Mac only – For Mountain Lion onwards the mouse interface only works for processes running under the logged in account. Given the driver process runs under the Root login the interface code was moved to the AIDaemon process.
Win 8.1 Touch interface	Windows 8.1 and above, UPDD version 5.1.1153 and above: Implements the user mode touch interface introduced in Win 8.x. Some Windows 8.x versions, especially, OEM system builds, reject the UPDD Virtual (software) HID interface so UPDD now supports the Win 8.1 user mode touch interface.
Evaluation 'nag' screen	Issues the nag screen on the 50th touch when using an evaluation version of the driver. Selecting the OK button resets the count.
Interactive touch click mode right click visualisation	If using Interactive click mode then visual feedback is shown during the right click countdown.

(UPDD 5.1.1112)

User double click settings	System double click settings need to be set such that double click can be achieved with a stylus. If a new user logs on these settings are set for touch usage.
Anchor mouse monitoring	Monitors the position of the cursor and resets the position after touch if the Anchor Mouse setting is enabled in the UPDD Console, Properties Page.
First touch calibration	When loading Aidaemon registers if the first touch is to invoke calibration. We would normally supply an installation of the driver that has this setting enabled by default. When the first touch is seen aidaemon invokes the calibration program and resets the first touch setting in the tbupdd.ini file. The 'first touch calibration' setting is held in the [UPDD/Paramaters] branch: To test the use of this function 1) Kill aidaemon process using the Windows Task Manager 2) Open up a command window 3) Type cd /program files/updd 4) Type tbutils nodevice setting dw "first touch calibration" 1 5) Type aidaemon.exe (to reload the daemon process) 6) Touch the screen – the calibration procedure will now run

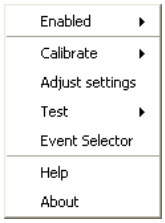
TBdaemon

No longer utilised in UPDD 5.1.1143 and above

Under Windows desktop version of the driver the software utilizes a daemon (or background) task to implement certain functions related to the driver and user interface. The program is called TBdaemon.exe and resides in the UPDD application folder. In UPDD version 3 this program was called TBsystry.exe.

The daemon task is invoked at system startup due to the 'tbdaemon' registry entry in the Windows run branch HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

The daemon task is responsible for the following functions:

System tray icons (Ver 4.1.10 and 5.0.2 only)	Can be enabled or disabled in the settings file. Some builds are distributed with them disabled by default. Setting to enable (1) or disable (0) is as follows: 4.1.8: tbcalib Device=0 "/setting:show systray =1" or 0 4.1.10: tbutils nodevice setting dw "show systray" 1 or 0
	Shows the rotate, event selector and menu icons in the system tray (if enabled – they can be disabled in the driver build if requested)
	Rotate: The rotate icon can be used to initiate desktop rotation and is only shown if the rotate interface is supported.
	Event Selector: Used to switch between left and right click generation. The 'red' area indicates the mouse click to be generated.
	Menu: Shows the UPDD System tray menu below.
System tray menu (Ver 4.1.0 and 5.0.2 only)	Enabled: Shows if the device is enabled at a UPDD level. Enabled state can also be switched. Devices can be enabled or disabled at the driver level irrespective of their physical state. Directly relates to the same setting in the UPDD Console, Properties page.
	Calibrate: Invoke calibration for a device. Adjust settings: Invoke the UPDD Console. Test: Invoke the test and drawing program for a device Event Selector: Toggle the desktop Event Selector function. Help: Load the driver's help files. About: Release information. The build id, as seen below, indicates the version (4.1.6) and release status 'R', the build no. (1221) within the release and the production system 'G' that generated the build: 
	Note: Some users require the UPDD system tray icons to be disabled and not displayed. In the UPDD settings files there is a setting called 'Show Systray' defined in the [UPDD/Parameters] section. When set to 1, system tray icons are shown. When set to 0 the Menu and Event Selector icons are not shown. The new settings are read the next time tbdaemon process is loaded. We can supply software with this setting disabled by default or you can manually update the setting (edit tbupdd.ini or run the command 'tbcalib Device=0 "/setting:show systray=0') as required and reboot or reload tbdaemon.
Rotate monitoring	For supported rotate interfaces informs the driver if a desktop is rotated and switches calibration data if a rotate style is defined (Rotate90,180,270).
User double click settings	System double click settings need to be set such that double click can be achieved with a stylus. If a new user logs on these settings are set for touch usage.
Toolbar playback	Actions associated with a toolbar when touched are initiated.
Anchor mouse monitoring	Monitors the position of the cursor and resets the position after touch if the Anchor Mouse setting is enabled in the UPDD Console, Properties Page.
Live annotation	If live annotation is enabled performs the drawing.

Extended sound	If extended sound is associated with a toolbar plays the sound.
Beep on touch	If enabled in the UPDD Console, calls the Windows beep function to produce the 'beep'.
Interactive touch click mode right click visualisation (UPDd 4.1.10 / 5.0.2 only)	If using Interactive click mode then visual feedback is shown during the right click countdown. 
Video resolution tracking	Tracks the video resolution and switches to resolution associated calibration data if it exists. E.g. If a resolution switch results in the desktop video changing position the calibration would be out. This function sees a resolution change and looks for calibration data specifically associated to the new resolution. The calibration data must be held against a calibration style in the format RxxxxXyyyy, e.g. R1024X768. If no style exist the current calibration data is used. In most cases switching video resolution does not adjust video desktop positioning so a single calibration is fine for all resolutions.
EEprom calibration data retrieval	Since UPDD 4.1.8, build id 1738 eeprom calibration data is automatically retrieved at driver startup or when USB devices are plugged in as long as UPDD eeprom calibration setting is enabled for the controller. In these cases the daemon procedure is invoked by the driver to issue the calibration eeprom retrieval function. ***Since UPDD 5.0.2 this functionality moved from TBdaemon to Aidaemon.***
First touch calibration	When loading, tbdaemon registers if the first touch is to invoke calibration. We would normally supply an installation of the driver that has this setting enabled by default. When the first touch is seen tbdaemon invokes the calibration program and resets the first touch setting in the tbupdd.ini file. The 'first touch calibration' setting is held in the [UPDD/Paramaters] branch: To test the use of this function 1) Kill tbdaemon process using the Windows Task Manager 2) Open up a command window 3) Type cd /program files/updd 4) Type tbcalib Device=0 "/setting:first touch calibration=1" 5) Type tbdaemon.exe (to reload the daemon process) 6) Touch the screen – the calibration procedure will now run
Event Selector setup	The Event Selector is implemented as a toolbar function and is defined and configured in the Event Selector definition file. The daemon process is used to import this file (create the toolbar) when the first device is added and when it is shown for the first time.

Pen up processing

A pointer device generates data packets when in use, such as touch packets whilst a stylus is in contact with a touch screen. When the device is no longer in use then obviously the data packets will stop being generated. A driver or application may need to perform a specific function at this point, such as pen up processing if the device is being used in mouse emulation in an appropriate mouse click mode.

Most touch screen devices, but not all, will indicate in the last data packet that the stylus has left the screen. These data packets are often referred to as the pen up or lift off packet.

If a device has a pen up packet defined in the UPDD controller configuration then there is a UPDD Setting called "Use Lift Off packet". This setting can be set in the UPDD settings file as appropriate or via the UPDD Console, properties page and is enabled by default.

UPDD will generate a pen up in mouse emulation mode when either:

- the 'pen up' packet is seen (and pen up processing is enabled)
- and/or when data packets cease (if Lift off time processing is enabled).

This approach caters for all methods of 'pen up' detection:

- The device generates pen up packets
- The device does not generate pen up packets
- The device generates pen up packets but due to communication error is not received or is corrupted.

To generate a pen up where data packets cease there has to be a short wait prior to generating the pen up to ensure the data has stopped rather than being a gap between the packets. In UPDD this short wait period is referred to as the Lift Off time settings, as described below.

Lift off time setting

The Lift off Time value specifies the time interval required to register a stylus lift after the last touch packet is received. Lift off time is defined in units of 20ms. This value is used to perform a pen up if the 'Use Lift off' packet is disabled otherwise Pen ups are generated as soon as the stylus leaves the pointer device display.

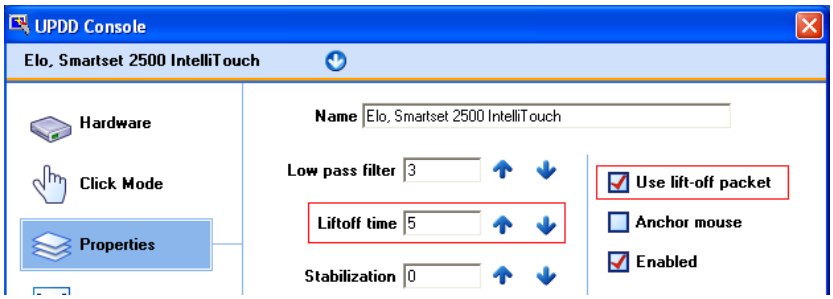
However, because this timer is triggered after each received touch packet it is important to ensure the value is greater than the time interval between data packets otherwise pen up events will constantly be generated. In default UPDD configurations this setting is often set as low as possible so that pen up is generated as quickly as possible after the last packet. However, if for any reason the packet speed is slowed down, multiple pen ups and pen downs will be generated due to the increased time gap between packets requiring the Lift off time value to be increased as appropriate.

Delta mode considerations

Delta mode refers to a pointer device mode whereby data packets with the same X and Y coordinate values (i.e. the stylus it stationary) are not sent by the device. In this mode, UPDD will generate a pen up when the stylus is held stationary. To cater for this situation a lift off time value of zero is defined to disable pen ups based on time.

Settings

The 'Lift off time' and 'Use pen up packets' settings can be set via the UPDD Console, Properties page or directly in the UPDD settings files:



If the console is not available the setting can be found in the UPDD settings files as follows:

UPDD version	Operating system	Location
UPDD 4.0.x	Windows	registry at HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TBUPDD
UPDD 4.0.x	Linux and Mac OS X	tbupdd.reg
UPDD 4.1.x	All OS	tbupdd.ini
UPDD 4.0.x	Windows CE	HKEY_LOCAL_MACHINE\Drivers\BuiltIn\TBUPDD\Parameters\{…}\1 - Where {…} is a GUID that identifies the package.

Within the files will be a branch 'Parameters'. With UPDD 4.0.x this will be followed by a GUID number (a very large number). For each controller there will be a controller number 1,2 etc. Therefore the location for a single controller system will be:

4.0.6 - CE	HKEY_LOCAL_MACHINE\Drivers\BuiltIn\TBUPDD\Parameters\{guid}\1
4.0.6 – Others	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TBUPDD\Parameters\{guid}\1
4.1.1 – All	[UPDD\parameters\1]

liftofftime=0x0000000n (n = 0 = disabled or lift of time value in units of 20ms)
use liftoff bit =0x0000000n (n = 0 = disabled, 1 = enabled)

Advance Console

With release 4.1.10 we hope to release the first version of the UPDD Advance Console.

The standard UPDD Console shows the basic driver settings and allows them to be updated. However, there are many other settings that may occasionally need to be updated and the Advance Console is being created to offer a GUI interface to these settings. Currently these settings need to either be set as required as default or updated manually.

Once this utility is available a separate web document will be created to document features and usage.

Live Annotation

With release 4.1.8 we have implemented Windows Live annotation function. This function allows a touch screen user to annotate over a live desktop. Annotation can be enabled, disabled, paused and erased. The line colour and width can also be defined.

This functionality came about from a project we undertook for a cable television company to allow a presenter to annotate over a live Windows desktop showing financial data. The presenter uses UPDD Toolbars at either side of the larger touch display (with no desktop visual representation – so not seen by the viewer) to invoke the annotate functions as required.

Annotation functions can be controlled via UPDD API calls and we have implemented two user interfaces of our own, as discussed below:

Toolbar interface

The UPDD toolbar interface implements toolbar cell actions to control all annotate functions. An example of an Annotate toolbar is described in full in the toolbar documentation.

UPDD Console interface

The UPDD Console, Extensions dialog, exposes a simple UPDD interface to the Annotate functions.

Annotate API calls

The annotate settings:

AnnotateThickness
Annotate Color
AnnotateMode

Are set via the standard api followed by TBApiApply

e.g.

```
TBApiSetSettingDWORD(0, "AnnotateThickness",3);
```

Color is a Windows COLORREF value NOT RGB; example

```
COLORREF cr = RGB(r,g,b);
TBApiSetSettingDWORD(0, "AnnotateColour", cr);
```

There's no API to clear. This can be achieved as follows (C++ example)

```
#define WM_DOCLEAR 4365
BOOL CALLBACK _ClearPicture(HWND hWnd, LPARAM lParam)
{
        char className[200];

        GetClassNameA(hWnd, className, sizeof(className));
         if(!strcmp(className, "TBUPDD_Annotate_Class"))
        {
                :PostMessage(hWnd, WM_DOCLEAR, NULL, NULL);
```

```
            }

            return true;
}

EnumWindows(_ClearPicture, NULL);
```

The values for mode are

0=pause
1=annotate
2=stop

If annotation is in a stopped state then activedw.exe must be launched to activate annotation.
Setting stop will cause activedw.exe to terminate.

Touch logging

Not available in Win CE
A primitive touch logging facility was added to UPDD 4.1.8 build 1784 to allow for the logging of touch data reported in screen co-ordinates (pixels) with the origin (0,0) at the top left. This logs first and last touches. This facility was added to allow for a customer to match touches against application layout to help track what they considered to be a 'user usage' issue rather than an application issue. If any other coordinate data is required (e.g. raw controller coordinates) please let us know.

To enable logging use the following command to define and enable the logging setting in tbupdd.ini:
UPDD 4.1.8 - **tbcalib Device=0 /settingsz:logging=Y**
UPDD 4.1.10 – **tbutils nodevice setting sz logging Y**
UPDD 5.x.x – **tbutils nodevice setting sz logging Y**
and restart the computer or restart the driver (tbupddwu process), (e.g. on windows net stop tbupddwu / net start tbupddwu)

Use the same command as above replacing **Y** with **N** to stop logging.

recording starts as soon as the driver is loaded.
reporting starts as soon as the windows desktop is fully loaded and the screen is touched (i.e. at least one touch is required on an active system to write the log).

```
C:\Program Files\UPDD>type upddlog_20100718_202531.log
2010-07-18 20:25:31 - left touch down @ x=688 y=619
2010-07-18 20:25:31 - left touch up   @ x=688 y=619
2010-07-18 20:25:31 - left touch down @ x=688 y=618
2010-07-18 20:25:31 - left touch up   @ x=674 y=438
2010-07-18 20:25:32 - left touch down @ x=475 y=316
2010-07-18 20:25:32 - left touch up   @ x=465 y=383
2010-07-18 20:25:39 - left touch down @ x=1180 y=773
2010-07-18 20:25:39 - left touch up   @ x=1179 y=771
```

The name of the log is based on the date and time the log file is created. Each entry is time stamped.

Custom settings

When the driver is installed it sets up the settings for any configured controller based on the default settings delivered as part of the driver setup or components. In some cases, as an end user, you may wish to configure a system and then capture these settings for use with subsequent installations. In Windows we have a Clone option to clone setting for use with subsequent installs but this has some limitations and is being superseded with this new custom settings file, which is called extra.ini.

For any given touch controller updd custom settings can be defined and embedded in the driver package such that when the controller is configured in the driver any settings defined in the custom settings file will be applied to the controller.

The custom settings file currently caters for the following settings:

General driver and system configuration settings

Define any dword or string scalar driver setting.
[updd]
Setting name=0x0000000a
[updd\parameters]
Setting name=0x0000000a

Controller setting
Define any dword or string scalar value associated with a controller.

[extra-controller name] – Controller name related to the subsequent settings
Setting name=Value – Setting name and value

Example
[extra-Quanta Computer, Dual, USB]
first touch calibration=0x00000001

Toolbar calibration settings

Pre-define toolbar calibration

[extra-toolbar]
ntoolbars=N (where N=no of toolbars)
and for each toolbar (1st N = 0, 2nd N = 1, 3rd N = 2 etc)
CalX0 Toolbar N=0x00000nnn
CalY0 Toolbar N=0x00000nnn
CalX1 Toolbar N=0x00000nnn
CalY1 Toolbar N=0x00000nnn

Calibration settings

Pre-define calibration data, created from the command tbcalib dump4tba

[extra-Quanta Computer, Dual, USB]
calibration styles=Normal,10,4,10,0,0,184,114,194,967,1744,121,1745,973

Other settings will be catered for as the need arises.

Once the extra.ini file is created it can be embedded in to the setup program we supply to you.

Since UPDD version 5.1.827 the setup file can be placed externally with the setup file to be used in subsequent installs.

1) On windows systems extra.ini files can be stored in the subfolder .\updd_ext relative to the location of the setup*.exe file on the installation media allowing most settings to be customised by an integrator.
2) On Linux systems the extra.ini file can be embedded in the installer (tgz) file with the path ./opt/tbupddlx/extra.ini to achieve the same.
3) For OS/X the extra.ini file can be combined with the UPDD.pkg file (extracted from updd.dmg) using the 'disk utility.app' to create a modified disk image updd.dmg.

Currently, the installation works by copying the extra.ini settings into a special section [Extra] of the main setting file, tbupdd.ini. When a device is configured manually or via the PnP manager then any settings defined in this section that relate to the new controller are applied.

Consideration

When considering the need for customized settings, please consider the following:

1) Are these settings we can define as the default settings and deliver a package that does not need cloning or extra.ini?
2) Can the settings be supplied to us in extra.ini for use in your delivered software?
3) Under Windows, can you utilize the 'old' clone method (until improvements to the extra.ini mechanism are available)
4) Are your custom settings to do with calibration? If so, there are various ways we can embed default calibration setting for you and customized settings are not necessarily the best approach. See, for example, TBcalib dump4tba command. Since UPDD version 5.1.827 this command now dumps the calibration data in a extra.ini file.

Important note: The last line of the extra.ini file must have a new line such that the last entry is a blank line

Event Selector

The UPDD driver is aware of 'Events' that can be generated by a device and all devices inherit the default event associated with the touch and release of a stylus on the device. Each Event can have two associated click modes referred to as the Primary and Secondary modes and these modes are described in great detail in the Mouse Emulation document. In standard touch screen usage the primary mode is normally associated with single left click mouse emulation and the secondary action is set to single right click mouse emulation. The current active mode is dictated by an internal setting call the Event State (Alt State) which will be inactive or active. Changing this state value will change the click mode used when the screen is next touched. We have implemented a number of methods/utilities to switch this state. The main utility used to change this state is referred to as the 'Event Selector'. This section describes how this is implemented in the various operating systems:

Windows

Extended touch consideration in Vista and Win 7

With extended touch enabled the Event Selector mechanism is disabled. Under Extended touch, using the systems HID interface we are not aware of an external method of generating a right click request. Extended touch has its own right click mechanism build into the operating system.

Linux

Currently not available.

Mac OS X

Available since UPDD 4.1.10, released 21st Oct 2010.

Event Selector Utilities

No longer available since UPDD 5.1.1143

We have implemented a number of different Event Selector utilities as discussed below. Some are OS specific as indicated:

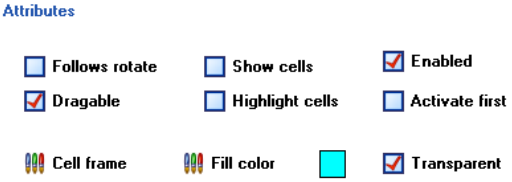
Interface	Icon Image	Description
	Primary Secondary	
System Tray *Windows Only*	 	When UPDD is installed (and if the system tray icons are enabled in the build) then an Event Selector system tray icon is shown in the system tray. This icon toggles between modes when selected. The Primary e (e.g. Left Click) is performed until the secondary mode is selected. The Secondary mode reverts to primary mode once the associated event (e.g. right click) has been performed.
Desktop Toolbar	*The toolbar Event Selector function was implemented in Windows UPDD Version 4.1.8, build 1863, released 3rd Sept 2010 and Mac OS X UPDD version 4.1.10 released 21st Oct 2010.*  	This 'floating' image is shown on the desktop when the Event Selector is enabled using one of the methods described below. This utility is implemented as a UPDD Toolbar.

Event Selector Toolbar Settings

Toolbar definition

UPDD caters for separate calibrated areas of the screen, called Toolbars, which can be used to invoke predefined functions or user written functions. In the case of the Event Selector, the toolbar is called Event Selector and defined in the UPDD Console, Toolbars.

Toolbars have a number of setting that relate to the use and visual feedback and these can be seen and adjusted via the UPDD Console, Toolbars, Change Toolbar dialog:



By default the Event Selector is Dragable which means it can be moved to a different position of the screen by holding the touch steady for a short period at

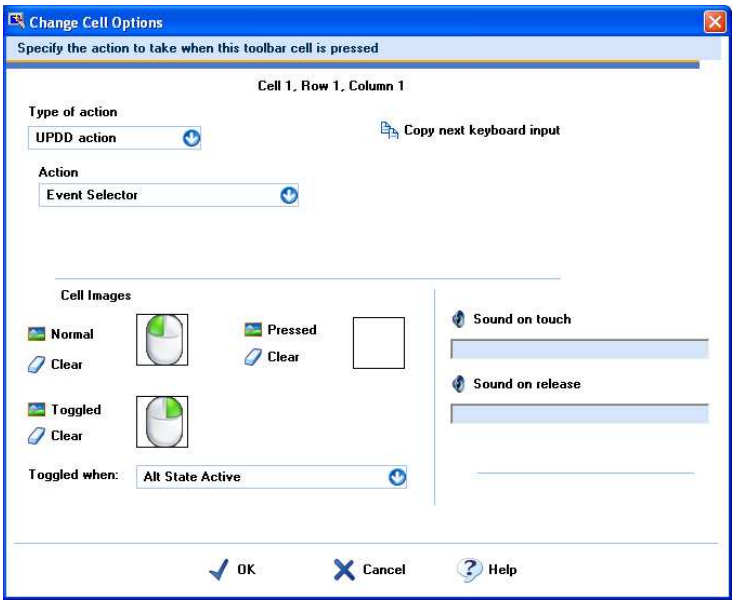
which point it enters drag mode and can be moved to a new location. This dialog can also be used to adjust the size of the toolbar if required. By default the size extends beyond the image, as represented below, so touching anywhere in this area is a toolbar touch rather than a desktop touch:

Toolbar definitions can be exported and imported and in the case of the Event Selector toolbar the default definition is distributed in a definition file called "Event Selector.ini". This file is automatically imported by the UPDD Daemon process when the first device is added and when it is shown for the first time.

Toolbar cell settings

The toolbar cell settings define the Event Selector function from the list of UPDD Actions, with two associated images, mousel.png and mouser.png. The toggled image is related to the Secondary mode state (Alt State Active). Using this dialog other images can be defined if required.

The Event Selector toolbar Change Cell Options are as follows:

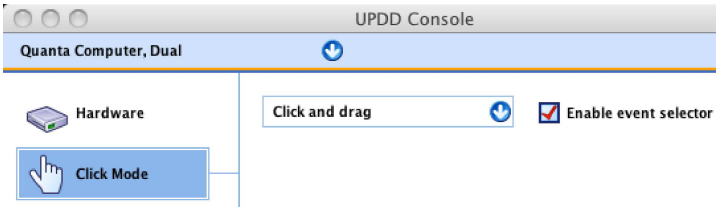


Enabling / disabling Event Selector toolbar

The Event Selector toolbars can be toggled on/off in the UPDD Console, by command line, other toolbar functions, via the UPDD API or the UPDD System tray menu (Windows). UPDD can be supplied with the Event Selector enabled by default if required. As standard the Event Selector is disabled until enabled using one of the methods below:

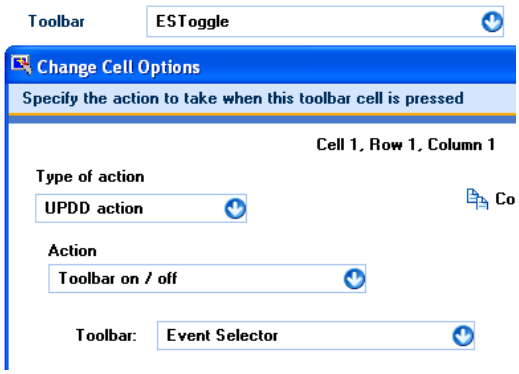
UPDD Console

In the UPDD Console (version 4.1.10), Click Mode dialog a Check box labeled "Enable event selector" can be used to enable or disable the Event Selector.



Toolbar Function

There are a number of toolbar setups that could be used in conjunction with the Event Selector. One such setup could be the use of another toolbar cell to enable or disable the Event Selector, which could, for example, be placed in a corner of the monitor without the use of any image or visual feedback (i.e. hidden). When this toolbar is touched the Event Selector will be enabled / disabled. The settings for this cell are as follows:



There are also toolbar cell Actions to set Primary (Event Selector Primary) and Secondary (Event Selector Secondary).

Command line

Using the user interface to enabled / disabled as follows:

4.1.8 / 1863	
tbcalib "/toolbaroff:Event Selector"	Disable Event Selector
tbcalib "/toolbaron:Event Selector"	Enable Event Selector
4.1.10 and above	
tbutils toolbaroff "Event Selector"	Disable Event Selector
tbutils toolbaron "Event Selector"	Enable Event Selector

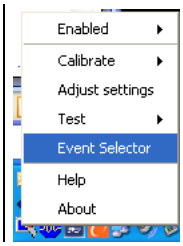
Under Windows, the previous version of the event selector was implemented as a program called tbalt.exe. We have written a 'new' tbalt.exe program that now toggles the ES toolbar on/off and is available on a per request basis should this be required for backward compatibility reasons.

API

The API TBApiEnableToolbar can be used to enable or disable the Event Selector toolbar.

UPDD System tray entry *(Windows only)*

The UPDD System tray menu has an Event Selector item that enabled / disables the event selector toolbar.



Removing Event Selector function

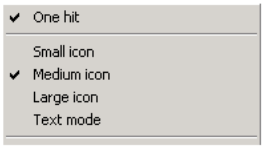
The Event Selector function can be removed by deleting the toolbar definition "Event Selector.ini" in the sub folder Toolbars under the UPDD folder.

Desktop Application *Windows only*

Implemented as a windows application (tbalt.exe) this was withdraw in favour of the Desktop Toolbar implementation above and is only available in UPDD pre 4.1.8 build 1863. This implementation has the disadvantage of an application real estate, boarder, title bar, menu, functions etc just to display an onscreen image.

This application is shown on the desktop when enabled (as default) as part of the supplied driver or from the Event Selector item of the UPDD System Tray menu and also the Program Manager

Event Selector application settings can be changed in the Event Selector menu. The menu can be viewed by clicking on the title bar mouse icon.



One hit: Indicates if the Event Selector automatically switches to the Primary setting after a single secondary use.

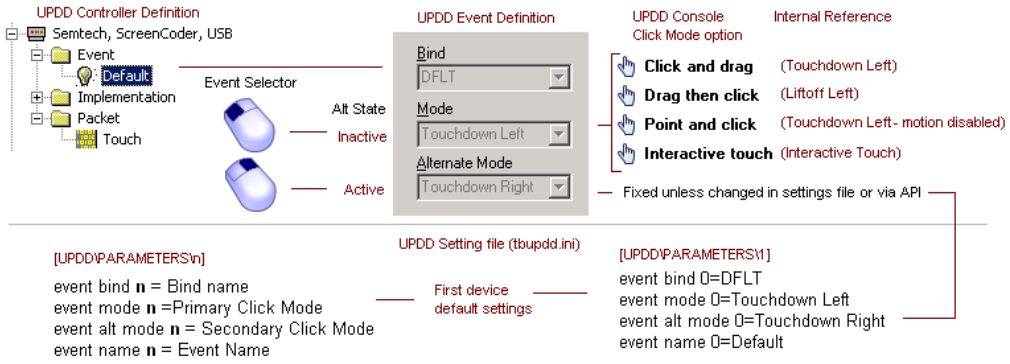
Icon: Indicates the size of the mouse icon displayed in the desktop event selector.

Text mode: Indicates that text be shown instead of a mouse icon.

Technical overview

This overview should be viewed in conjunction with the mouse emulation documentation.

A controller is defined in the UPDD Controller database. As part of the definition we specify the events that can be generated by the device. Some have more than one event, however, all touch devices have the default event defined which relates to the first and last touch. Each Event can have two click modes associated with the event, the Primary action (Mode) and Secondary action (Alternative Mode). For each device UPDD holds an internal state (Alt State) to indicate the mode to perform when the event occurs. The Event Selector ultimately modifies this internal state. This can be visually represented as below:



Event Selector API

An application can call the TBApiSetEventSelectorState(n) to switch to the primary or secondary modes as required. n = 0 = Primary or n = 1 = Secondary. When changed any active event selector utility will automatically reflect the state change.

Touch Mouse

The function has been superseded in UPDD version 5.0.2 with the Touch Pad function below.

A touch screen type device would normally function in absolute mode, that is, the system pointer or touch activation point would be directly under the point of contact. However, in some cases there is a requirement to operate a touch screen in relative mode such that the movement of the system pointer is relative to the point of contact (similar in operation to a touch pad or mouse). However, given that touch screens generate absolute co-ordinates then additional processing is required to convert the absolute touch to a relative movement. Touch Mouse implements a relative pointer device along the lines of a "trackpad" device. This feature currently only works with the system primary monitor.

Newly reinstated in UPDD 4.1.8 build 1948 this function is enabled and configured by the following settings:

UPDD setting ([UPDD\parameters\1])	**Description**
TMenable	Touch mouse enabled (1) / disabled (0) Can be set via the command: tbcalib /setting:tmenable=1 (or 0) – pre 4.1.10 tbutils setting dw tmenable 1 (or 0) – 4.1.10 and above
TMAcceleration	The speed of motion is controlled by this setting. 4 seems a good default.

By default a click is generated on first touch. Other button modes can be utilised by setting "event mode 0"=<mode>. See Mouse Emulation documentation. For example set this to "none" for no automatic click. In this case a toolbar can be used to implement the mouse buttons.

Touch Pad

The function was introduced (and only works in) UPDD version 5.0.2 and supersedes the Touch Mouse function above.

Starting with UPDD version 5.0.2 a new option is available to emulate a 'touch pad' similar to that found on laptop device using a "touch screen" input device.

When utilising this function the touch screen input device will typically not be attached to a monitor as with a normal touch screen but will still be operating in absolute mode. The absolute co-ordinate input is converted to relative motion as needed for touch pad emulation. There is additional support for multi fingered operation where the device supports it.

This feature replaces the previous "Touch Mouse" option from earlier versions. It supports the same features and more.

The following modes of operation are supported for generating button clicks

Tap — With this mode enabled a short touch of the device generates a left button down event at the current position. Allowance is made for dragging and double clicking as follows:

- Left click, a short tap.
- Drag, a short tap quickly followed by a touch and drag
- Double click, 2 short touches.

Button — With this mode enabled, for a multi touch device an area at the bottom of the pad is reserved for use as mouse button(s). One or two buttons are supported. A single button or left side of a two button configuration mimics the left mouse button. In the case of 2 buttons the right side mimics the right mouse button.

Touching the button area causes a mouse button down event for the appropriate mouse button at the current position. Releasing the touch causes a mouse button up event for the appropriate mouse button at the current position.

Note if the first touch is in a button area then a second concurrent touch is treated as stylus zero allowing dragging or drawing operations for example to be performed after and while holding the left button down.

Multi touch and gestures

When two or more concurrent touches take place the secondary touches are passed to the operating system at a position relative to the current primary touch point based on the vector between the primary and each touch and a predefined metric. This allows full support for gestures using 2 or more fingers. Single touch gestures, such as flicks, are not possible because it is difficult to distinguish between single fingered gestures and normal movement. The reliability of the gestures depends to a large extent on characteristics of the input device. UPDD for the most part is passing this data directly to the OS. When using multi-touch with the Touch Pad in button mode the touches must be outside of the dedicated button area.

Settings

The operation of this feature is defined by a number of settings to allow tailoring to a specific device. Some might usefully be adjusted to suite a specific user.

Some of the settings refer to "dwell". When a primary touch starts it is unknown if this is the start of a single touch or the start of a multi finger gesture.

The software will therefore not pass through mouse events until one of the following occurs.

1. The touch ends
2. A secondary point is touched
3. The position of the touch changes by a defined amount

The period during which events are held in this way is known as the dwell period.

The settings and their meanings are noted below. These settings are held in the UPDD settings file in the standard UPDD device specific section (one per device) and can be changed as documented here.

If using the command line utility to define the settings the 'Type' parameter shown against each setting indicates the setting type which is reflected in the tbutils command e.g. 'tbutils setting dw touchpad 1' is the command used to enable the touch pad function.

touchpad	Type: dw Default value: 0 Purpose: Enables the touchpad functionality. 0=off 1=on. All other settings are ignored if this is off.
touchpadmode	Type: sz Default value: tap Purpose: Defines the mode of operation, tap or button
touchpadtapsize	Type: dw Default value: 10 Units: Raw controller co-ordinates Purpose: Defines the amount of variance in x or y co-ordinates allow before a tap is ignored and treated as movement
touchpadtapdowntime	Type: dw Default value: 300 Units: milliseconds Purpose: Defines the maximum time before a short touch is not treated as a tap.
touchpadtapuptime	Type: dw Default value: 300 Units: milliseconds Purpose: Defines the maximum time following a tap during which a subsequent touch is considered a continuation of the previous touch
touchpadbuttons	Type: dw Default value: 1 Purpose: Defines the number of buttons in "button" mode
touchpadbuttonpct	Type: dw Default value: 15 Units: % of controller raw y range. Purpose: Defines the height of the button area at the bottom of the touch pad in "button" mode
touchpadminy	Type: dw Default value: 0 Units: raw controller co-ordinates Purpose: Defines the lower bound of the controller's output range. Only required in button mode.
touchpadmaxy	Type: dw Default value: 1024 Units: raw controller co-ordinates Purpose: Defines the upper bound of the controller's output range. Only required in button mode.
touchpadmultifactor	Type: dw Default value: 33 Units: percent. Purpose: Defines a multiplication factor to be applied to the spacing between primary and secondary touches to account for the smaller physical size of a touchpad compared to a physical screen, typically this will be set to a value based on the size difference between the monitor in use and the size of the trackpad. The default value of 33 is based on a touch device approximately one tird the size of the monitor. The intention is that for a given physical spacing on the touch device (e.g. 4 cm) the spacing of the delivered coordinates will have the same separation on the monitor.

Furthermore the following regular UPDD settings should be set to ensure that the x and y co-ordinates are reported correctly (e.g in the case of a non-standard controller orientation or wiring) and that the origin (x=0,y=0) is at the top left.

invert
invert
swapxy

Technical note.
This function is effectively utilising the device in two modes of operation, single touch mouse and multi-touch digitizer.

When used in single touch mode to move the cursor with or without the pen down the data is fed into the system as a mouse type device so as to induce cursor movement. Once more that one stylus is in use the driver switches device mode and feeds the data to the OS as a digitizer device. Unfortunately Windows requires that there is a 200ms delay between device mode switching and in some usage scenarios there is a small delay when this switching occurs. Owing to this delay the algorithm responsible for this switching reduces the mode switching to a minimum. If in the button down state of single touch mode it is not possible to transation to multi touch until the button down state ends.

Our tests have shown that the Touch pad type device works well but we are aware that there could be slight improvements but none that warrant further development at this time especially as Windows 8 has introduced a single device type that caters for single and multi-touch track-pads and the obvious future change would be to declare the UPDD Touch-pad as such a device such that no device mode switching is required.

Disabling Touch

Occasionally you may wish to, for whatever reason, disable touch for a device or disable the mouse emulation interface.

Device
The driver can be prevented from processing any touches from a device. This can be set in a number of ways:

1) Command line interface passing parameters enable or disable
2) UPDD Console, Properties, Enabled checkbox (check or uncheck)
3) Windows - UPDD System Tray, Enabled option – select to toggle setting
4) API TBApiSetSettingDWORD(passedDeviceNumber,_T("Enabled"),0); = Disabled or….. ,1); = Enabled

System Wide
The driver can be prevented from passing any touches to the 'mouse interface' of the OS. This will stop mouse emulation from all devices configured in the driver, but will, for example, still allow applications to receive touch data via the API.

1) Command line interface passing parameters pointeron or pointeroff
2) API TBApiMousePortInterfaceEnable(true); or (False);
3) TBupdd.ini setting "InitialMousePortEnabled" (false = ignores MousePortInterface setting at startup and starts up with port disabled)
   The setting can be manually edited in the file or set via API TBApiSetGlobalSettingDWORD("InitialMousePortEnabled ",0);

With UPDD 4.1.10 we can supply a driver with InitialMousePortEnabled setting = false as default so that the touch is not functioning after installed and subsequent reboots. Pre 4.1.10 this setting needs to be set manually by one of the methods described above.

Sound utilisation

Pre UPDD 4.1.10 sound (during touch or calibration) was only supported in Windows 32 bit and only available via the system speaker. With UPDD 4.1.10 we have introduced support for sound file playback during touch and calibration.

We have implemented sound playback via a cross platform utility which utilises sound playback mechanisms relevant to the OS in use:

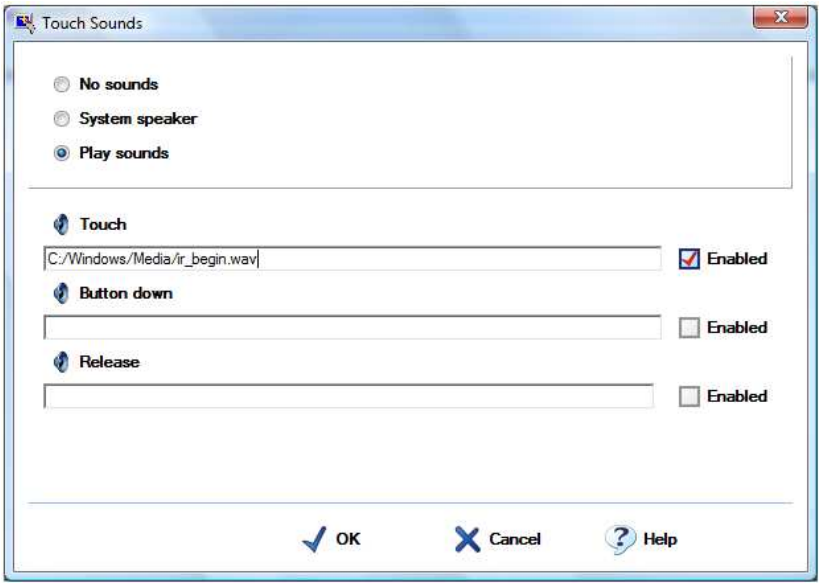
Operating system	Playback mechanism
Windows	On Microsoft Windows the underlying multimedia system is used; only WAVE format sound files are supported.
Linux / Solaris 10	On X11 the Network Audio System is used if available, otherwise all operations work silently. NAS supports WAVE and AU files. Given that the sound implementation in Linux relies on an optional component called NAS. This is not always installed by default. NAS should be available for most distributions but customers will need to refer to the installation instructions for their specific platform. As an example, here are the instructions to add NAS support for ubuntu: First launch the "Ubuntu Software Center" from the "Applications" menu. In the search box type in "nas" and press enter. The first result back should be "Network Audio System – local server". Select this and click the "Install" button. A progress bar will indicate when the installation has finished. See screenshot below: 
Mac OS X	On Macintosh QT (QuickTime) is used for sound; this means all QuickTime formats are supported.

UPDD sound configuration

Sounds can be associated with touch and calibration events and are configured as follows:

Touch
Configured in the UPDD Console, Click Mode dialog, via the Sound option button:

Enter the name of the sound file to play or select the speaker icon to invoke Windows Explorer to located the sound file:



Ensure settings or sound playback length does not result in overlapping sound. E.g. If using click mode Click and Drag there is no requirement to have sounds defined for both Touch and Button down as they occur at the same time.

No sounds Sounds are disabled

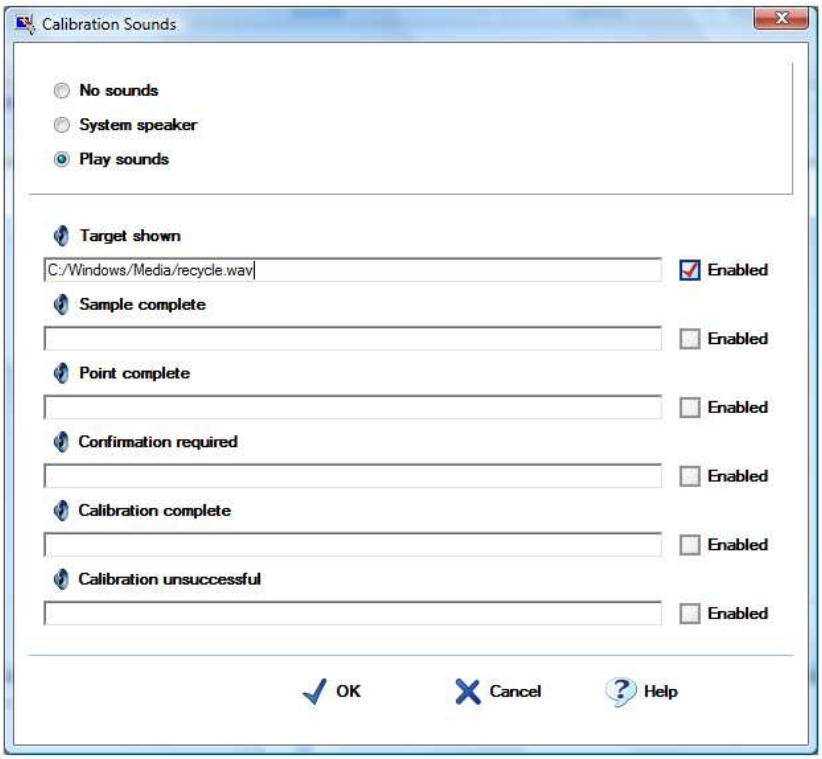
System speaker

Generates a beep on the system speaker (Win 32 only) on left or right button (pen) down
Pitch and duration are defined in the UPDD settings file for each device
Settings = sound duration=0x00000037 (default), sound pitch=0x000003E8 (default)
Note – for this to work a system speaker must be present in the system – this is not the same as a sound card.

Play sounds Plays the associated sound file, if enabled, to the audio system

Calibration

Configured in the UPDD Console, Calibration dialog, via the Sound option button



Ensure settings or sound playback length does not result in overlapping sound. Any triggers that occur close to each other could result in overlapping sounds.

No sounds Sounds are disabled

System speaker Generates a beep on the system speaker (Win 32 only) when calibration point is complete.
Pitch and duration are defined in the UPDD settings file for each device
Settings = sound duration=0x00000037 (default), sound pitch=0x000003E8 (default)
Note – for this to work a system speaker must be present in the system – this is not the

same as a sound card.

Play sounds Plays the associated sound file, if enabled, to the audio system

Touch security feature

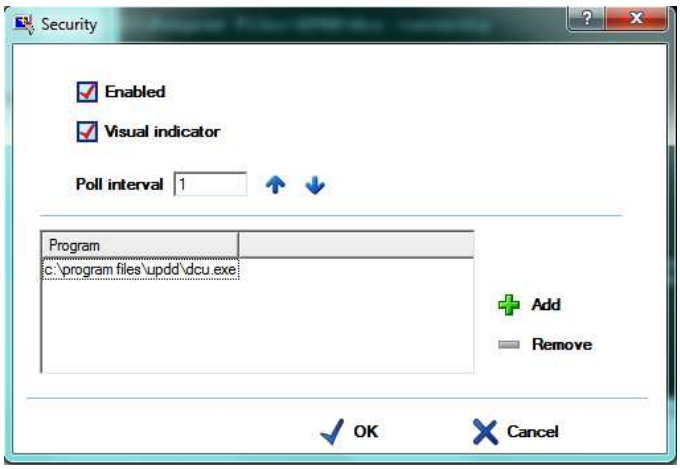
Some touch systems may be configured to run a dedicated touch application such that if the application fails then touch should be disabled to restrict access to the underlying system. This option has been implemented in UPDD 4.1.10, build 2235.

Configuring the security feature can be performed via a GUI interface or via settings in the UPDD settings file.

GUI interface

The GUI interface is invoked via the command "dcu /security" (dcu is the .exe name of the UPDD Console). This is currently Windows only but a minor change is required for this to work in other OS. Please contact us if required in non – Windows systems.

This will invoke the following security dialog:



Setting	Meaning
Enabled	Indicates if the security feature is enabled
Visual indicator	Indicates if a warning is displayed when the system is touched whilst in a blocked state
Poll interval	Poll interval is the time in seconds between each check
Program	List of programs to monitor
Add / remove	Add or remove program
OK / Cancel	Exit dialog

Settings

If following settings, in the UPDD setting file, parameters branch, are used to implement the security feature:

Setting	Meaning	Range	Example (to reflect the above GUI settings)
security.enabled	Touch security state	0 or 1	1
security.visual	Message box state	0 or 1	1
security.number.programs	Number of programs to monitor	1 to n	1
security.program.<n>	Individual program full path name	0 to n	C:\program files\updd\dcu.exe
security.poll.secs	Interval to check program process	1 to n	1
security.block	Indicates if touch to be blocked	0 or 1	0 or 1 depending on process activity

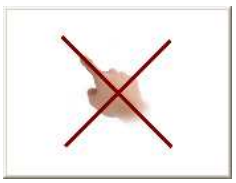
Settings can be updated using the tbutils command line interface. To replicate the above settings using tbutils would be as follows:

Tbutils setting nodevice dw security.enabled 1
Tbutils setting nodevice dw security.visual 1
Tbutils setting nodevice dw security.number.programs 1
Tbutils setting nodevice sz security.program.0 C:\program files\updd\dcu.exe
Tbutils setting nodevice dw security.poll.secs 1
Tbutils setting nodevice dw security.block 0 – needed to create setting as this is for internal use only and value will be set as required.

Checking procedure

The UPDD background task invokes the security checking if enabled. If enabled it reviews the list of processes looking for the process associated with the defined program(s). If any are missing from the process list, touches are flagged to be blocked by the driver to the UPDD API or system interfaces. If all the processes are found the touch mechanism is not blocked. The checking is continuous so if a missing process is restored the touches will automatically be unblocked so long as all defined processes are seen.

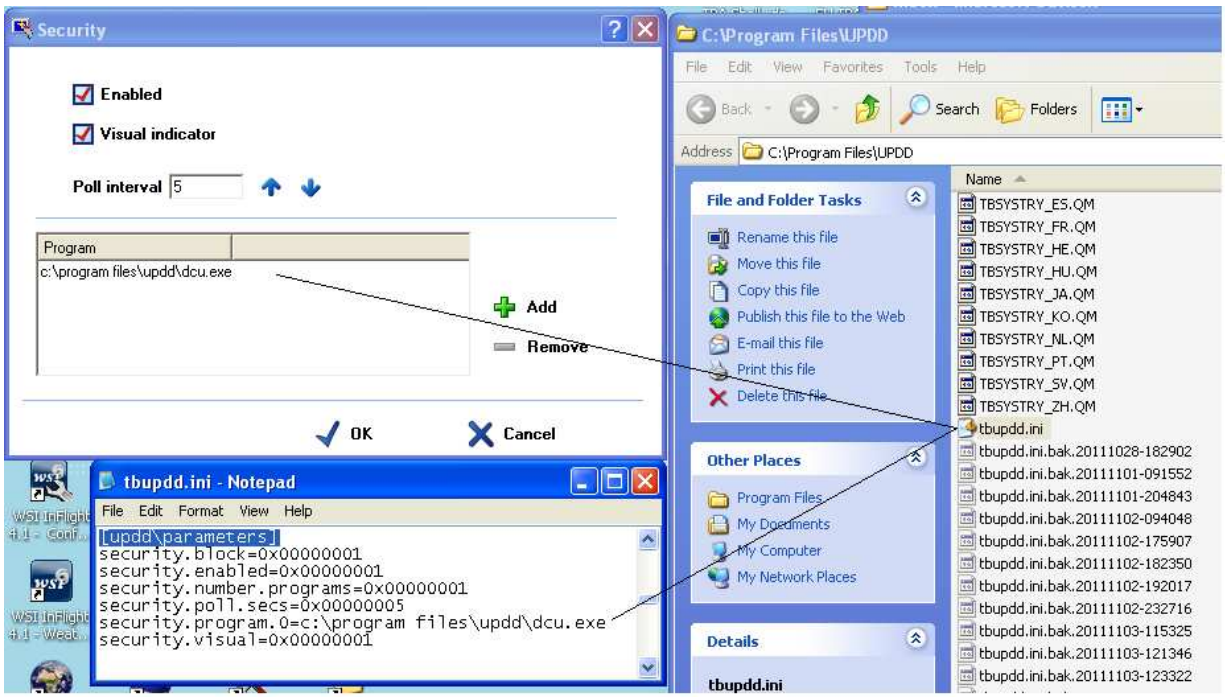
If a user touches the screen whilst the touches are blocked and the visual indicator is enabled the user sees the following dialog:



Known issues

It was reported that this feature did not work on a Korean system. Further investigation showed that the program's pathname was retaining some illegal characters

To overcome this issue please ensure the selected pathnames are in a folder that can be represented in ASCII file name, as per this example:



Virtual Devices

Starting with UPDD version 5.1.0; in addition to support for physical hardware such as USB and RS232 interfaces, virtual devices are now supported. *Note that this refers to the virtualisation of the touch data input, not the interface to mice and pointer systems that is documented elsewhere.*

This is useful, for example, where the API TBApiPostPacketBytes is to be used to support a device where the hardware is being controlled by other software. Similarly an application can be used to drive the system pointer without a physical device being used.

Data recorded with "tbutils record" can be played back to a virtual device in cases where the actual device is not available (although in most cases it is probably more useful to mark a PnP device as "virtualconnected", see below).

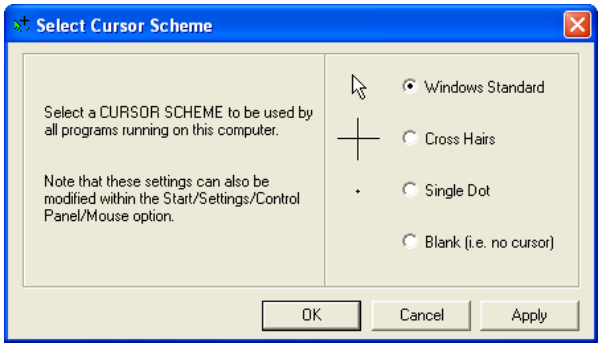
To use a virtual device you will first need a driver with support for one or more virtual devices. Then use the add device option in the UPDD console or tbutils to add an instance of such a device.

A new device setting is used in conjunction with virtual devices. "virtualconnected" will by default have a value of 1 (meaning true).
The UPDD console will show a device as connected (in black text) in this case. A program can set this value to 0 (meaning false) to have the console show disconnected (red text).

Note this option is supported for all controller types but only a value of 1 has an effect, this forces the device into a connected state allowing tbutils playback for example for a disconnected device

Cursor Utility

UPDD software consists of a Windows cursor utility, tbcursor, that can be used to change the cursor scheme:



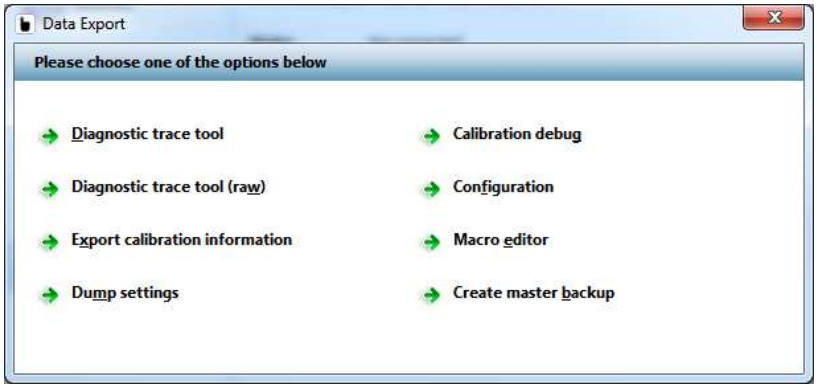
Simply select a cursor scheme from the defined set of four schemes.

Be careful if selecting blank as there is no visual feedback at point of touch.

Data Export

Starting with UPDD 5.1.1099 there is new dialog that offers a GUI interface into a number of common data capture features loosely referred to as 'Data Export'.

This dialog is invoked from the 'Dump settings' option in the UPDD Console, Status dialog as shown below:



The Data Export options 'capture' the following data:

Option	Function
Diagnostic trace tool / Diagnostic trace tool (raw)	Captures touch data to tbcalib.log as described here.
Export Calibration data	Exports calibration data to extra.ini. Can be used to embed in driver installations to set default calibration data for a given touch configuration. Can be sent to Touch-Base to embed in delivered installers or placed alongside existing installer to be used in subsequent installations: Windows – place in sub folder UPDD_EXT Mac and Linux – to follow!
Dump settings	Dumps the current driver settings to tbupdd.ini file.
Calibration debug	Shows data packet feedback during calibration
Configuration	Dumps key configuration info to upddconfig.txt
Macro editor	View and edit the controller's initialisation macro – internal/support use only.
Create master backup	Create a working copy of the settings file tbupdd.ini called tbupdd.ini.master. If this file exists UPDD does not take settings file backups but instead uses this file if tbupdd.ini becomes corrupted.

Dynamic Monitor Tracking

Starting with version 5.1.656 UPDD supports tracking dynamically attached monitors and automatically adjusting the monitor desktop assignments. This feature is implemented as an extensible framework based around monitor characteristics. Currently the only characteristics supported are Monitor dimensions. Others can be added as the need arises.

This feature is used to automatically configure touch monitors in a system as they are connected and to remove them from the driver's configuration as they are unplugged with out any user intervention.

This function is portable and supported on all platforms running daemon process Aidaemon.

Current support is for this feature is based on Monitor dimensions (video resolution). The feature also requires eeprom calibration support. With this enabled so long as monitors have different dimensions UPDD will detect insertions, position changes and removals of all monitors and configure itself accordingly.

Prerequisites:
The default settings for touch controllers must be set to "extended touch" OFF.
Default calibration data must be provided for supported controllers through one on the documented methods.
To use monitor dimension tracking a device must be precalibrated using the eeprom storage feature.

Settings used by this feature

Trackmonitors: this must be set to 1 to enable monitor tracking. This setting is not device specific.

Tbutils nodevice setting dw Trackmonitors 1

Trackmonitorsdwell: specifies a delay in seconds that must elapse after detection of a new monitor before the configuration takes place, this allows the system to "settle" avoiding multiple configuration calls in the event that the video layout changes multiple times.

Tbutils nodevice setting dw Trackmonitorsdwell 1

Eepromreadatnewmonitor: this is described in more detail in eeprom settings but should be set to 1 to track monitor dimensions. This setting is not device specific.

Tbutils nodevice setting dw eepromreadatnewdevice 1

eeprom calibration: this is described in more detail in eeprom settings but should be set to 1 to track monitor dimensions. This setting is device specific.

Tbutils device=N setting dw "eeprom calibration" 1

Example:
A system can have either an 800 x 600 and/or 1024 x 768 touch monitor connected such that they should be both correctly calibrated and associated with the correct desktop at the time the monitor is connected to the system. The daemon task will be notified when a new monitor is connected and will detect the video resolution. If it matches the video resolution associated with a touch device then the touch device is associated with the monitor with the detected resolution. Will currently only work if the monitor resolution is unique on the system.

Touch Filtering

Starting with build 1123 a new touchdown filter option is supported.

This addresses some anomalies with the original implementation that was incompatible with some event modes and the new event manager.

If the device specific option "touchdownfilter" has a nonzero value then at the start of a packet stream the specified number of packets are discarded. This is used to ignore touch data packets from controllers that can take one or two data packets to report the correct point of touch, as in this example:

01 74 09 4C 08 – first packet does not contain the correct touch data and needs to be discarded.

01 85 0D 19 08

01 A0 0D 15 08

01 AC 0D 12 08

A packet stream is considered to start when a packet is received more than (LiftOffTime +2) * 20 ms after any previously received packet.

Notes:

1) this option is not suitable for use with controllers with an irregular packet rate where the inter packet interval can exceed the time specified above, in particular delta mode controllers that do not stream data when held steady.
2) Touchdown filter discards n packets regardless of which stylus.

Since updd build 5.1.1203 the build in filters apply to all stylus. With 5.0.2 this only they only applied to the first stylus.

Averaging
Low pass filter
Stabilisation
Deglitch
Sample Rate

These filters now operate independently on each stylus.

Contact

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