Gesture API

UPDD V6 supports a gesture module which delivers information about detected on screen gestures via the UPDD API.

For Windows and Mac the gesture API function is part of the Commander utility that can perform system actions associated with a gesture before the gesture is posted into the OS to invoke any gesture related functionality on the desktop or specific application.

Under Linux a Gesture module is supplied specifically to post gesture data to the driver for onward dispatch to UPDD Client applications.

Download and installation

The MacOS and Windows the Commander software needs to be part of the driver installation.

For Linux, download and install the gesture module from here. Copy and run the updd gesture application program from the UPDD installation folder, usually this is /opt/updd.

Once it is installed then the API can be used to receive gesture information.

If using the gesture API interface with the 'old' gesture software and you only require the API interface set the mode of operation to API only.

Invoke at system startup

It may be desirable to have the UPDD gesture / commander software running at all times. In this case you need to implement a start up function to invoke the software at system startup or at user login, as per these suggestions:

OS	Start up method
MacOS	This should be the default state as Ccommander is set as a launch agent by default.
Windows	Commander can be set to launch after driver install. If not, enable the 'Run at startup' setting in the Commander settings dialog.
Linux	You need to use an appropriate application startup method to execute the following

command as the logged in user at login time:

```
/opt/updd/upddenv "/opt/updd/UPDD Gestures" &
```

Once installed a Commander / gesture system tray / menu bar item icon will be seen to invoke various functions.

API functions

The method for receiving gesture information via the UPDD API is the same as for other data types, although a couple of additional steps are required in setting up the software to build.

Mac builds distributed since 23rd Oct 2019, Windows builds since Oct 2020.

```
//Add the following 2 lines before inclusion of the upddapi.h

#define UPDD_GESTURES

#include "TBTouchGestureConstants.h"

#include "upddapi.h"

//Ensure that the files TBTouchGestureConstants.h and gestures_event.inl are both in the include path.

//Having done this simply use the constant _ReadDataTypeGestureEvent in an appropriate call to TBApiRegisterDataCallback

e.g.

TBApiRegisterEvent(0, (unsigned long)this, _EventTypeGesture, MyGestureCallback);

//Data specific to the gesture will be reported in the GestureEvent section of the PointerData structure to the registered callback.

//e.g.
```

```
void TBCALL
MyGestureCallback(unsigned long /*context*/,
_PointerEvent* ev)

{
if (ev->pe.gestureEvent.type == kTBGestureTap &&
cv->pe.gestureEvent.touchCount == 2)
{
cout << "Saw a two finger tap" << endl;
}
}
```

Linux, Windows and Mac builds distributed pre 23rd Oct 2019

```
//Add the following 2 lines before inclusion of the
upddapi.h
#define UPDD_GESTURES
#include "TBGestureAnalysisTypes.h"
#include "upddapi.h"

//Ensure that the files TBGestureAnalysisTypes.h and
gestures_event.inl are both in the include path.
//Having done this simply use the constant _ReadDataTypeGestureEvent in an
appropriate call to TBApiRegisterDataCallback

e.g.

TBApiRegisterEvent(0, (unsigned long)this,
_EventTypeGesture, MyGestureCallback);

//Data specific to the gesture will be reported in
the GestureEvent section of the PointerData structure
to the registered callback.
```

```
//e.g.

void TBCALL
MyGestureCallback(unsigned long /*context*/,
_PointerEvent* ev)


{
if (ev->pe.gestureEvent.type ==
kTBGestureTwoFingerTap)
{
cout << "Saw a two finger tap" << endl;
}
}
```

The gesture/commander software must be running to utilise the gestures API.

Gestures_Event.inl

This is an optional part of the API.

It is only used if you define the pre-processor macro UPDD_GESTURES

It provides additional information about gesture events.

Events are described here

https://support.touch-base.com/Documentation/50293/Callback-Events

the file gestures_event.inl has detailed information about the returned information for gestures.

Disabling / enabling gestures

Should you have any reason to temporarily disable gestures, such as when running you own touch screen calibration program under Mac, we have introduced a setting for enabling / disabling the gesture functionality and can be set as follows:

```
TBApiSetSettingFromInt(0, "gesture enabled", 0); // disable

TBApiSetSettingFromInt(0, "gesture enabled", 1); // enable
```

Alternatively it can be set from the command line interface like so:

```
upddutils nodevice set "gesture enabled" 0
```