




















Operating systems [Windows](#) [Win CE](#) [Mac OS X](#) [Linux](#) [Solaris](#)  
 Development languages [Java](#)




The UPDD API allows user mode applications to interface directly with the driver and/or the pointer devices handled by UPDD. It is assumed that the reader is familiar with the various functions and parameters of TBUPDD, since that information is not duplicated here.

The UPDD API is supported on all platforms supported by UPDD (and is in fact used by UPDD's own programs, such as the UPDD Console, test program, calibration tool etc). On each platform the interface is similar to that used natively by that platform itself. For example on Windows systems a DLL based interface, on mac os/x a dylib interface and so on. In all cases the C calling convention is used. This means that the API can be utilised from any programming language that can execute native OS API calls. Examples are provided for C++ as this is the language on which UPDD is built but these should be easily adapted to other languages. If using another language you will probably already know how to make library calls, but in the case of any doubt that you should refer to documentation for your software generation tool. Whilst we can advise in general terms full support of all possible programming tools is outside the scope of normal support. For example Java developers will need to use the Java Native Interface (JNI) as per the example shown below.

Depending on which operating system and client language is used the user has a choice of linking to the API statically or dynamically. For example on Windows both static and dynamic libraries are available, however Visual Basic only supports dynamic linking.

Depending on the OS in use the following files implement the UPDD API:

Operating System	Files	Description			
	TBapi.h	function declarations			
	Tbapi.lib	lib file used for dynamic linking to TBAPI.DLL			
	Tbapi.dll	Dynamically linked library (supplied with the driver)			
	Tbapi.bas	contains DLL function declarations, constants and data types and helper functions for VB			
	Download	Version	Build (from)		
		32bit	4.1.6	Date	
			13 <sup>th</sup> July 2010		
		32bit	4.1.8	Pre 2030	22 <sup>th</sup> Aug 2010
		32bit	4.1.8	2030	24 <sup>th</sup> Mar 2011
		32bit	4.1.10		10 <sup>th</sup> Jan 2011
	TBapi.h	function declarations			
	TBApi.lib	4.0.6	statically linked implementation for a given target processor		
	TBbundle.h	4.0.6	contains bundle-specific data		
	TBapi.dll	4.1.10	Dynamically linked library (supplied with the driver)		
	Download	Version	Build (from)	Date	
		X86	CE6	4.1.10	22 <sup>nd</sup> Mar 2011
		ARM	CE6	4.1.10	01 <sup>st</sup> Apr 2011
		X86	CE6	4.0.6	13 <sup>th</sup> Nov 2008
	ARM	CE6	4.0.6	12 <sup>th</sup> July 2010	
	X86	CE5	4.0.6	13 <sup>th</sup> Nov 2008	
	ARM	CE5	4.0.6	13 <sup>th</sup> Nov 2008	
	TBapi.h	Function declarations			
	nonwindows.h	declarations of Windows specific constructs for non Windows targets			
	nonwininternals.h	declarations of Windows specific constructs for non Windows targets			
	libTBApi.a	statically linked implementation for a given target processor			
		<i>When linking to libTBApi.a you must also link in /usr/local/lib/libACE.dylib</i>			
	Download	Version	Build (from)	Date	
	Intel	32bit	4.1.1	1 <sup>st</sup> May 2009	
	PPC	32bit	4.1.1	1 <sup>st</sup> May 2009	
	Intel	32bit	4.1.10	15 <sup>th</sup> Sept 2010	
	TBapi.h	function declarations			
	nonwindows.h	declarations of Windows specific constructs for non Windows targets			
	nonwininternals.h	declarations of Windows specific constructs for non Windows targets			
	libTBApi.so	shared library for a given target processor			
	Download	Version	Build (from)	Date	
	X86	32bit	4.1.1	1 <sup>st</sup> May 2009	
	X86	32bit	4.1.8	16 <sup>th</sup> Aug 2010	

		X86	64bit	4.1.8	7 <sup>th</sup> Sept 2010
		TBapi.h		function declarations	
		nonwindows.h		declarations of Windows specific constructs for non Windows targets	
		nonwininternals.		declarations of Windows specific constructs for non Windows targets	
		libTBApi.so		shared library for a given target processor – <i>supplied with the driver</i>	
		Please note the libTBApi.so is supplied with the driver and not in the download file			
		X86	32bit	4.1.10	1 <sup>st</sup> Feb 2011

### Development Languages specifics



Java applications will need to use the [Java Native Interface](#) (JNI). This can link using our standard Windows DLL or shared library in other unix based OS.

Usage of our API within a Java application requires a JNI implementation, an API declaration and usage within the Java application as per this example declaring and using the TBApiInit function call:

```
JNI Implementation      JNIEXPORT void JNICALL Java_TBApiClass_TBApiInit
                        (JNIEnv * a, jobject b, jbyte c)
                        TBApiInit(c);
```

```
Declaration             public static TBApiClass api;
```

```
Usage                   api = new TBApiClass();
                        api.TBApiInit((byte)1);
```

### Contact

For further information or technical assistance please email the technical support team at [technical@touch-base.com](mailto:technical@touch-base.com).