Subject: Re: WinAPI UNICODE question
Posted by cbpporter on Tue, 18 Nov 2008 11:41:57 GMT
View Forum Message <> Reply to Message

Another possible solution: introduce a flag and depending on it use either WinAPI types or their equivalent types.

```
#idndef flagNOPS
typedef WCHAR UWCHAR;
#else
typedef wchar_t UWCHAR;
#endif
```

The different name is needed to differentiate between the types if the flag is not set. All functions with platform dependent parameters or return values will use the Uxxxx variant, which will default to the WinAPI version by default, but if someone want the other version, they simply add the flag. I still have about 3 other solutions for that problem, but these two are the prettiest.

Except the issues of type names, there are two more:
1. Some macro's defined in windows.h are used which will no longer be available at that point. This can easily be solved with introducing either a copy of that macro with a different name (so that later you can include windows.h if needed or even better, by using an inline function, i.e. replacing HIWORD with HighWord inline function.

2. The use of the operators that convert from Point to POINT must be replaces with the uses of a function with exactly the same body as the operator, but which is an inline function that takes a Point parameter rather than an operator. I've done this in my code base already and only accrued in a few places, like in Draw.

And that's about it. With types not taken from windows.h, not using the macros from there and the issue with U++ geometric types, one can easily remove the windows reference and add it only to really low level compilation units.

I'm pretty much done with code and preparing to tackle Draw, I'll still hand around Core a little more to test and make sure that everything is OK. I also went over the packages from Bazaar (not the latest from SVN, the ones from 2008.1) and there are no issues here either. I was really expecting the docking functionality to at least not compile, but I'm glad to say I was wrong.