iPCU.scpt

This is a basic-sh script that takes a tab-delimited file and turns it into configuration profiles for iOS devices. This has been tested up through the iPhone Configuration Utility 3.4 and OS X Server 10.7.2.

The reason it uses the iPCU is convenience. By scripting the iPhone Configuration Utility to create the file, the syntax and format will be correct, and the configuration profiles will be available within the copy of the iPCU on the machine the script is run on. Obviously, you will need a copy of the iPhone Configuration Utility to run, or even compile this script. Since this is AppleScript, you'll need the Mac OS version of the iPhone Configuration Utility, and a Mac running at least Mac OS X 10.6. The iPCU is available at http://support.apple.com/kb/DL1465

The script is highly commented, and should be fairly simple to follow provided you know something about AppleScript. If you don't, well, it's going to be difficult, and as this is a readme and not a book on AppleScript, I'm not going to explain all the details on that level. There are a ton of ways to learn AppleScript, just google the word, and you'll be well on your way.

The one thing I will explain are TIDs, or Text Item Delimiters, because they're important to the script and somewhat misunderstood. TIDS are how AppleScript keeps track of text items. Normally, the TID is "" or the null character. So, let's take a sentence like:

John is snarky.

Were you to use that in a two-line script like:

```
set theText to "John is snarky"

set theTextItems to every text item of theText
```

You'd get two values:

theText would be a string: "John is snarky"

theTextItems would be a list: {"J", "o", "h", "n", " ", "i", "s", " ", "s", "n", "a", "r", "k", "y"}

Because AppleScript uses "" as the TID by default, we get every item in that sentence. We don't get the return, or end of line (EOL) character, that's not considered a text item.

So, let's take a tab-delimited line like "jwelch\tJohn Welch" (\t is the tab character. When you compile that, you'd see "jwelch   John Welch"

If we leave the default TIDs, getting those two elements out, while not hard, is tedious. If you go with every word, you get: {"jwelch", "John", "Welch"}

If we use every paragraph we get: {"jwelch   John Welch"}

If we use every text item, we get: {"j", "w", "e", "l", "c", "h", " ", "J", "o", "h", "n", " ", "W", "e", "l", "c", "h"} So you can use every word I suppose, but what if someone includes a middle initial? Then you have more parsing work to do. It's kind of a pain.

But what if we change the TID to be the tab character? It's fairly easy, adds three lines to the script:

```
set saveTID to AppleScript's text item delimiters

set AppleScript's text item delimiters to "	"
```

And

```
set AppleScript's text item delimiters to saveTID
```

Pretty simple. We save the current TID to a variable, saveTID. We set the TID to the tab character, \t which compiles out to a blank space. The uncompiled line would be: **set** AppleScript's text item delimiters **to** "\t". Then when we're done, we set the TID back to the default.

What's this do for us? Well, let's look at our original source and see what every text item gets us now: {"jwelch", "John Welch"}

Well, look at that. A list with all the contents of each field as a separate item in the list. What happens if we add an initial with a period? Say: "jwelch	John C. Welch"?

We get: {"jwelch", "John C. Welch"} Again, both fields as their own list items. Now it's just item 1 and item 2. As long as the source file is consistent, you have no additional work to do. That's why I use TIDs in this script, because it really does make things MUCH easier.