Manage Your Computer Tweakomatically

One of the most popular pieces of software Microsoft has ever released is TweakUI, easily the most impressive item in the Windows PowerToys. To quote from Microsoft.com, TweakUI " … gives you access to system settings that are not exposed in the Windows XP default user interface, including mouse settings, Explorer settings, taskbar settings, and more." Obviously this is the sort of thing people want: since the first version of the PowerToys was released for Windows 95, hundreds of thousands of people have downloaded and used TweakUI. In fact, if *you* haven't downloaded it yet, we highly recommend that you do. We'll wait.

Nice weather we've been having, huh? How 'bout those Yankees?

Oh, good, you're back. Now that you've checked out TweakUI, you can see what a great little utility it is. And as loyal and devoted Microsoft employees, you would think that the Scripting Guys would be the first to applaud the TweakUI team for their efforts; after all, they created something that is incredibly popular and incredibly useful. Furthermore, they didn't do that to make money or to sell more software, but simply to help address customer needs and desires. What's not to like about a piece of software, and a team, like that?

But if you thought the Scripting Guys would applaud the TweakUI team, you thought wrong. As it turns out, the Scripting Guys are shallow and petty. It bothers us that more people have downloaded TweakUI than have downloaded the original WMI Scriptomatic. (And don't even get us started on the fact that more people play Age of Mythology than write scripts using the ADSI Scriptomatic.) We admit it: we're jealous, and we vowed to do anything we could to topple TweakUI as Microsoft.com's most popular download.

(Well, ok, we didn't really vow to do *anything* in order to accomplish this coup. Instead, we vowed to do whatever didn't really require a lot of time and effort on our part.)

The problem we faced, though, was this: how can we outdo a piece of software as slick and as useful as TweakUI? How can we find a way to provide even *more* of what customers need and desire? How can we justify playing Age of Mythology all day when we're supposed to be working? We pondered this dilemma long and hard when suddenly it hit us. There's only one way to outdo a great piece of software like TweakUI: the Tweakomatic. (Yes, it's so obvious you're probably wondering why *you* didn't think of it.)

As it turns out, even though TweakUI is an incredibly cool piece of software, it has one limitation that we could capitalize on and use against it: Kryptonite. No, wait; that's another project we're working on. Um, just forget we ever said anything about Kryptonite, ok? Especially if some reporter from *The Daily Planet* starts nosing around.

At any rate, TweakUI lets you configure Windows to look and act the way you want it to, *provided* you carry out those operations on the local computer. But what if you want to configure Windows on a remote computer? Can't do it with TweakUI. What if you want to configure Windows on a *bunch* on computers? Can't do it with TweakUI, at least not without installing the software on all those machines and then manually configuring all the desired settings.

That, of course, left a hole big enough to drive a Tweakomatic through. The Tweakomatic is a nifty new utility that writes scripts that allow you to retrieve and/or configure Windows and Internet Explorer settings. So what, you might ask. Well, think about it. Because these are WMI scripts, they can be used to configure settings *on remote computers*. Need to change something on a remote machine? Then haul out the Tweakomatic. And think about *this*: because the Tweakomatic writes scripts, you could do something like run these scripts as logon or logoff scripts. In other words, you could quickly, easily, and *automatically* configure settings on any or

all the computers in your organization. The Tweakomatic is undoubtedly the single best thing Microsoft has released since Windows itself!

Well, OK. But it *is* better than Microsoft Bob.

How does the Tweakomatic work?

Work? Who said anything about the Tweakomatic actually *working*?

No, we're just kidding: of *course* the Tweakomatic works. And the reason it works is because it takes full advantage of the "template approach" that's built right into WMI's registry provider. (As you might have guessed, the Tweakomatic uses WMI to read from and write to the registry.) Let's take a look at one of these templates:

```
HKEY_CURRENT_USER = &H80000001
strComputer = "."
Set objReg = GetObject("winmgmts:\\" & strComputer & _
    "\root\default:StdRegProv")
strKeyPath = "XXXXXXXXXX"
objReg.CreateKey HKEY_CURRENT_USER, strKeyPath
ValueName = "XXXXXXXXXX"
strValue = "XXXXXXXXXX"
objReg.SetStringValue HKEY_CURRENT_USER, strKeyPath, ValueName, strValue
```

In the preceding script, the XXXXXXXXXX's represent items that will vary from script-to-script. Depending on what you're changing in the registry, you'll always have to specify a different registry path (Control Panel\Desktop), a different registry key (ScreenSaverIsSecure), and a different value (1 to enable password protection on the screen saver, 0 to disable it). Everything else is boilerplate: all scripts that change a REG_SZ value in the HKEY_CURRENT_USER portion of the registry (like the script above does) will follow this exact same pattern. There is a similar pattern for REG_WORD values and for scripts that read from the registry as opposed to write to it. (And, of course, you can easily swap HKEY_LOCAL_MACHINE in for HKEY_CURRENT_USER as needed.)

So how does the Tweakomatic take advantage of that? Well, the Tweakomatic database (Tweakomatic.mdb) includes fields (and values) similar to these:

Field Name	Variable Name in Script	Value
RegKey	strKeyPath	Control Panel\Desktop
RegValue	ValueName	ScreenSaverIsSecure
DefaultValue	strValue	1

When you click an item in the Tweakomatic (such as **Password-protect the screen saver**), the Tweakomatic queries the database and retrieves the values for fields such as those listed above. The Tweakomatic then uses the appropriate template to create a finished script, substituting the values retrieved from the database for the XXXXXXXXXX's in the template:

```
HKEY_CURRENT_USER = &H80000001
strComputer = "."
Set objReg = GetObject("winmgmts:\\" & strComputer & _
     "\root\default:StdRegProv")
strKeyPath = "Control Panel\Desktop"
objReg.CreateKey HKEY_CURRENT_USER, strKeyPath
ValueName = "ScreenSaverIsSecure"
strValue = "1"
objReg.SetStringValue HKEY_CURRENT_USER, strKeyPath, ValueName, strValue
```

Of *course* it's easy. If it was hard, do you think the Scripting Guys would bother doing it?

I thought we weren't supposed to change settings in the registry?

As you probably know, Microsoft has a sort of love-hate relationship with the registry. The registry is *the* configuration database for Windows and Windows applications, and many options can only be set by manually changing a value in the registry. For example, if you've ever read a Microsoft Knowledge Base article, you've likely seen a sentence similar to this:

"To correct this problem, change the following value in the registry."

Now that's fine, except that this sentence is invariably followed by a disclaimer similar to this one:

> **Warning**
> Don't ever change a value in the registry. Ever. We know we just told you to do that, but would you jump off a cliff if we told you to? Don't ever change a value in the registry. Don't even say the word registry. We know a guy once who said the word registry, and three days later he was hit by a bus. True story. As a matter of fact, you shouldn't even have a registry on your computer. If you suspect that you do have a registry on your computer, please call us and a trained professional will be dispatched to your office to remove the registry immediately. If you accidentally touch the registry, wash your hands with soap and water and call a doctor. Do not swallow the registry or get it in your eyes!

Now, to be honest, some of those fears are a bit exaggerated, and the disclaimer is there largely for legal reasons (remember, this is the day and age when you can order hot coffee in a restaurant, and then sue the restaurant when the coffee they give you turns out to be, well, hot). If you do it correctly, changing the registry is perfectly harmless. At the same time, however, it's true that there *are* certain values in the registry that should never be changed; in fact, changing them can pretty much wipe your computer out, once and for all. It's like working on the bomb squad: if you snip the right wire, the bomb is defused and everything is fine. But if you snip the wrong one—Boom! You just created Microsoft Bob!

Um, not that we're saying Microsoft Bob was a bomb or anything ….

So is it possible to snip the wrong registry wire using the Tweakomatic? We don't think so; we've tried to steer clear of registry values that, if changed to an incorrect setting, could cause a fluctuation in the space-time continuum and destroy the universe as we know it. (Something that looks bad on your annual performance review, as we Scripting Guys know from painful experience. "Well, you guys did the Scriptomatic, and that was good. But you also destroyed the known universe, and that wasn't so good.") As far as we know, the Tweakomatic only configures "safe" registry settings. What does that mean? Well, take the registry value ScrnSave.exe, which allows you to specify the path to the screen saver file. Suppose you make a mistake, and enter the path to a non-existent screen saver. Will that cause your computer to die a quick and painful death? Fortunately, no; it just means that you won't have a screen saver. Change the value to an actual screen saver, and you'll have a screen saver again. A bit of a nuisance, perhaps, but hardly fatal.

On the other hand, we know that the last thing you want to give a customer who orders hot coffee is hot coffee. Therefore, *officially* the Tweakomatic should be used only to read values from the registry, never to write values to the registry. Unofficially, well, who are we to tell you what to do?

Does the Tweakomatic do everything TweakUI does?

Next question, please.

Ok, fine, we admit it: the Tweakomatic does *not* do everything TweakUI does. Is that because the Scripting Guys are a bunch of lazy good-for-nothings who couldn't take their eyes off the TV long enough to add a couple more features to the Tweakomatic? In the words of Homer Simpson, "Oh, no, no, no, no, no. Well, yes."

Actually, although we *are* a bunch of lazy good-for-nothings, we had a legitimate reason for leaving out some of the items that can be configured with TweakUI. Our goal here was not just to write scripts for you, but to write scripts that you could easily modify. For example, here, again, is a Tweakomatic script that password protects the screen saver:

```
HKEY_CURRENT_USER = &H80000001
strComputer = "."
Set objReg = GetObject("winmgmts:\\" & strComputer & _
    "\root\default:StdRegProv")
strKeyPath = "Control Panel\Desktop"
objReg.CreateKey HKEY_CURRENT_USER, strKeyPath
ValueName = "ScreenSaverIsSecure"
strValue = "1"
objReg.SetStringValue HKEY_CURRENT_USER, strKeyPath, ValueName, strValue
```

Pretty handy, huh? However, suppose for some reason you wanted to *disable* password protection on the screen saver. How can you do that?

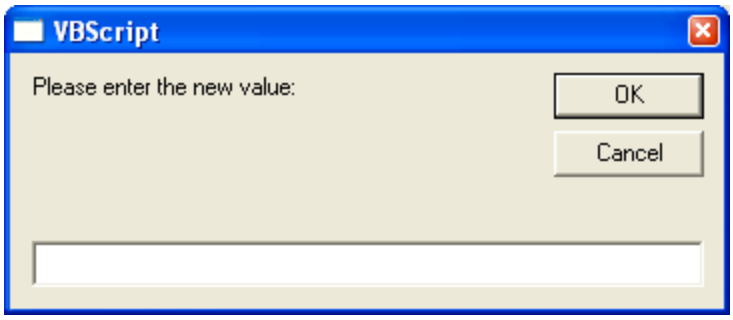
Well, right there on the Tweakomatic screen you'll see a spot labeled **Task description**. And in this spot you'll see help text similar to this:

> Specify whether a password is required to unlock the screen saver and return to the desktop. To require a password to unlock the screen saver, set this value to 1. Otherwise, set this value to 0.

If you read that text, you'll see that setting the value of ScreenSaverIsSecure to 0 will disable password protection. In other words, make the following change to the Tweakomatic script, and you'll be able to turn password protection off:

```
HKEY_CURRENT_USER = &H80000001
strComputer = "."
Set objReg = GetObject("winmgmts:\\" & strComputer & _
    "\root\default:StdRegProv")
strKeyPath = "Control Panel\Desktop"
objReg.CreateKey HKEY_CURRENT_USER, strKeyPath
ValueName = "ScreenSaverIsSecure"
strValue = "0"
objReg.SetStringValue HKEY_CURRENT_USER, strKeyPath, ValueName, strValue
```

In fact, to make this even easier, there's a button in the Tweakomatic labeled **Change Value**. Click this button, and a dialog box like this will appear:



Enter a new value (like 0), click **OK**, and the Tweakomatic will rewrite the script for you. At no additional charge!

So what does this have to do with the fact that TweakUI can configure some things that the Tweakomatic can't? Well, TweakUI configures some settings that *could* be modified using a script, but would require millions of pages of accompanying documentation to explain all the possible changes and how to make them. We decided to limit Tweakomatic to settings that can *easily* be modified (most Tweakomatic settings are of the on-off variety: 1 to turn the setting on, 0 to turn it off). This makes the Tweakomatic easier to use, and decreases the possibility of you entering an incorrect value and destroying the known universe.

So that means that TweakUI *is* better than the Tweakomatic, doesn't it? Well, maybe not better so much as different. Besides, the Tweakomatic lets you configure a ton of settings that you *can't* configure using TweakUI. Want to change the default folder view in Windows Explorer? Can't do it with TweakUI. Want to configure Internet Explorer so that it doesn't pop up a message box any time it finishes downloading a file? Can't do it with TweakUI. Want to have your clothes washed, pressed, and neatly put away? Can't do it with TweakUI.

Well, ok, you can't do the latter with the Tweakomatic, either, although we're looking into it for a future release.

Now, we're not saying that this makes the Tweakomatic a better utility than TweakUI, and we're not saying that this makes the Tweakomatic a lot more fun than Age of Mythology. We're just saying that the Tweakomatic lets you determine whether Search should, by default, search tape drives.

Whoa, search tape drives, *by default* !?! When we put it that way, we guess we *are* saying that the Tweakomatic is more fun than Age of Mythology. By *default* no less!

Shouldn't I use Group Policy instead of the Tweakomatic?

Originally our plan was two-fold. After we humbled TweakUI, we would set our sights on Group Policy. As it turns out, however, those Group Policy guys are mean and nasty, and we were scared to death of them. Because of that, we made a conscious decision to keep a safe distance between the Tweakomatic and Group Policy. Although a setting or two might have slipped through the cracks, we limited the Tweakomatic to settings not covered by Group Policy. (Yes, technically you can manage *any* registry setting using Group Policy. But that would involve writing a custom .ADM file, and we decided it'd be just as easy—and just as effective—to manage these settings using scripts instead.)

But what if there *is* overlap, isn't it better to use Group Policy? Yes. However, we realize that: 1) Not everyone uses Group Policy; 2) Not everyone uses Active Directory; and, 3) Sometimes you might want to make a simple change to a handful of machines, and Group Policy seems like overkill. If so, then go ahead and use the Tweakomatic. (Just don't tell the Group Policy guys.)

What platforms does the Tweakomatic support?

The Tweakomatic is really designed for Windows XP, although many (most?) of the settings also apply to Windows 2003 and to Windows 2000. However, at the moment we've done very little testing on either of those platforms; we've done even *less* testing on Windows NT 4.0 or Windows 98. It's highly unlikely that anything bad would happen to your computer if you ran this on something other that Windows XP, but we don't want anyone saying, "Hey, you didn't tell us that this was really designed for XP." Therefore, we're telling you: this was really designed for XP.

So what *do* I need in order to run the Tweakomatic?

That's a good question, and we don't really know the answer. As intimated above, we know that the Tweakomatic runs just dandy on Windows XP and Windows 2003. It also seems to run quite nicely on Windows 2000. Keep in mind, however, that we tested it on a Windows 2000 computer with Internet Explorer 6.0 and with Service Pack 4. What if you try it on Windows 2000 with no service packs installed? Beats us, though we wouldn't be surprised if it *didn't* work. If you want to run this on Windows 2000, it's a good idea to have the latest version of IE and the latest service pack installed. But, then again, that's a pretty good idea regardless of whether or not you're going to run the Tweakomatic.

Will the Tweakomatic run on Windows NT 4.0? Probably, as long as you have a recent version of IE and provided you have WMI installed. The same thing is true of Windows 98. Remember, though, that many of the settings the Tweakomatic changes simply don't exist in Windows NT 4.0 or Windows 98. That means that while you'll be able to *write* scripts on those platforms, you won't necessarily be able to *run* scripts on those platforms. (Well, actually, the scripts will run, they just won't do anything.) All things considered, in might be a good idea to try the Tweakomatic out in a test situation before turning it loose on NT 4.0 or Windows 98 computers.

It's a good idea to have Windows installed on your computer before you try running the Tweakomatic; however, you shouldn't have to install Microsoft Office as well (even though the Tweakomatic relies on an Access database). As long as you have ODBC and as long as you have a Microsoft Access Driver installed, you should be fine.

Oh: and you will need to run the Tweakomatic in 1024x768 screen resolution. We know that isn't always possible and isn't always convenient, so, when we get a chance, we'll create a version that can run in 800x600 mode.

The Tweakomatic is fully guaranteed and supported, right?

Hey, let's be serious: this thing is called the *Tweakomatic*. Can you imagine this scenario:

"Microsoft Product Support. How may I help you?"

"Uh, yes, I seem to be having some problems with my Tweakomatic."

In other words, no, the Tweakomatic is not fully guaranteed and supported; in fact, it isn't even *partially* guaranteed and supported. Here's the official disclaimer:

> This software is not supported under any Microsoft standard support program or service. The software is provided AS IS without warranty of any kind. Microsoft further disclaims all implied warranties including, without limitation, any implied warranties of

> merchantability or of fitness for a particular purpose. The entire risk arising out of the use or performance of the software and documentation remains with you. In no event shall Microsoft, its authors, or anyone else involved in the creation, production, or delivery of the software be liable for any damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or other pecuniary loss) arising out of the use of or inability to use the software or documentation, even if Microsoft has been advised of the possibility of such damages.

On the other hand, if you have problems with or questions about the Tweakomatic, write to us at scripter@microsoft.com. We can't make any promises, but we'll try to help you if we can.

So I guess that the Tweakomatic isn't extensible, either, right?

You want to know the sordid truth about the TweakUI team? Well, actually, we don't know anything at all about the TweakUI team (although we sure hope they have a sense of humor). So in order to make our point, we'll make something up instead:

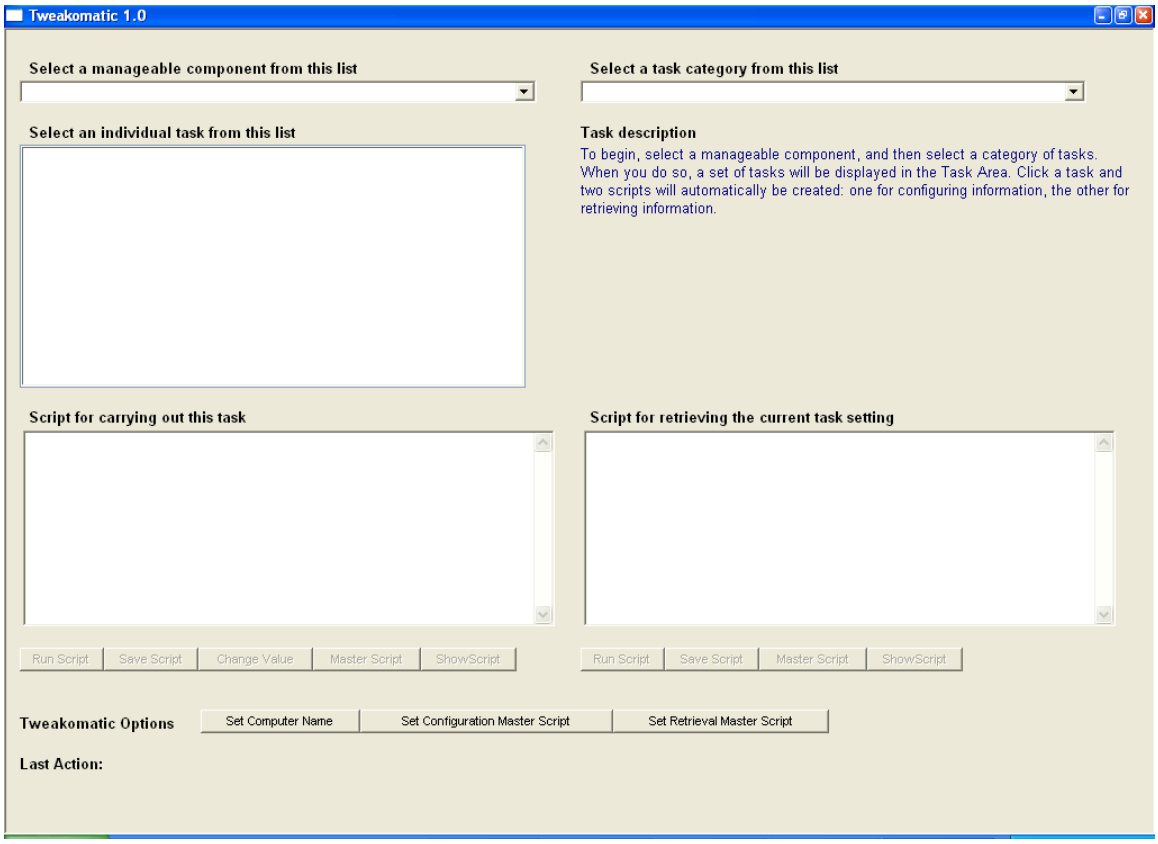
The truth is, the TweakUI team has the cushiest job on the planet. These guys spend all their time at the baccarat tables in Monte Carlo. Every couple years or so, Microsoft releases a new version of Windows, and these guys drain the last of their martinis, stumble out of the casino, hammer out a new version of TweakUI, and then head back to the gaming tables.

How can they get away with this? Well, they hit upon the perfect formula: find something that works, and then just keep doing that same thing over and over again. And we Scripting Guys want in on that action. Do we want to spend our days trying to think up clever new utilities to give to the world? Heck no; we want to think up *one* clever new utility, play some baccarat and then, just when they're about to fire us, cobble together a new version, post it to the Web, and call it good.

Therefore, we intend to periodically release new Tweakomatic Management Packs. (Cool name, huh?) Whenever we get bored, we'll investigate the registry settings for something, package them up, and then post them to the Script Center. You'll be able to grab one of those Management Packs, run some sort of installer program to add them to your Tweakomatic database, and then be able to manage even *more* things using the Tweakomatic. First on the list: An IIS Management Pack. (Yes, an odd choice, but we had to gather this information for another project, so ….) Up after that: Microsoft Office.

How do I use the Tweakomatic?

The Tweakomatic consists of two files: Tweakomatic.mdb (a database containing all the relevant information), and Tweakomatic.hta, the front-end to that database. To use the Tweakomatic, make sure the two files are in the same folder (and that this folder is on the local computer; HTAs don't work well across the network), and double-click Tweakomatic.hta. You should see something that looks like this:

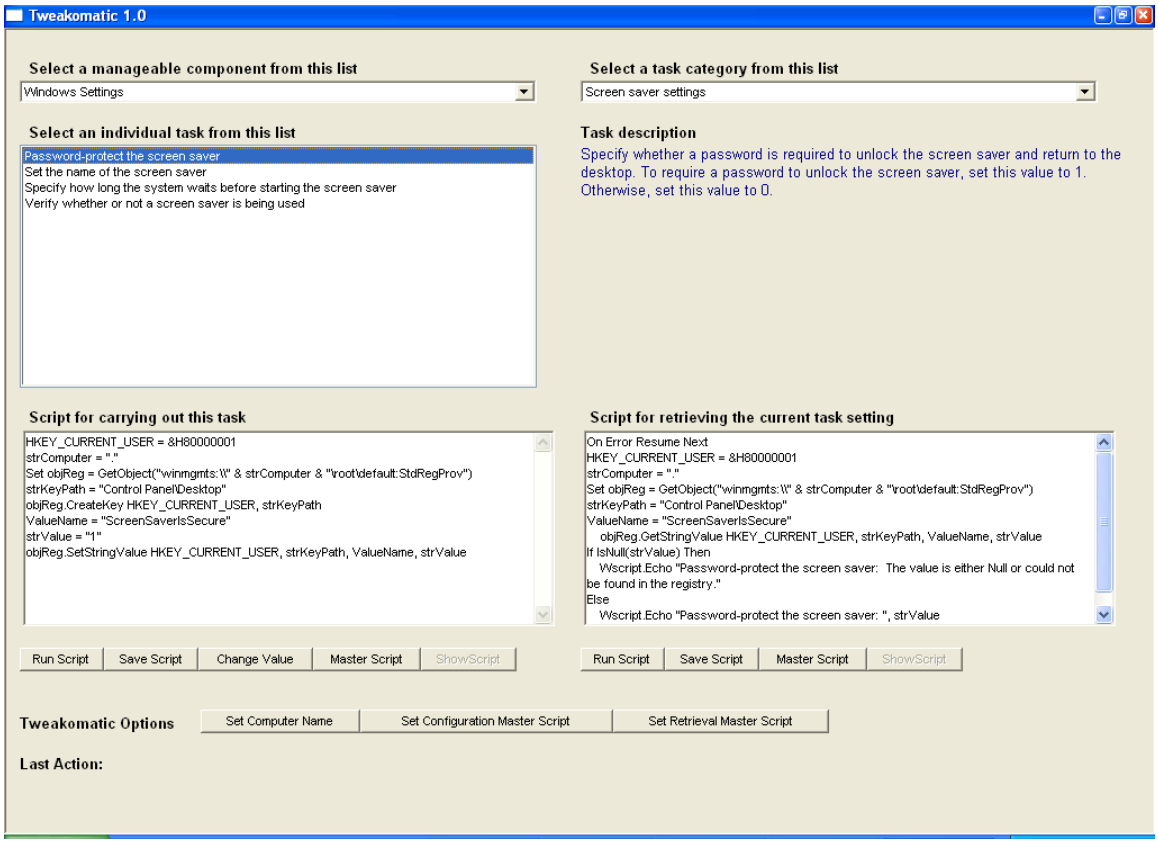


So far so good. Now, click the **Select a manageable component from this list** dropdown, and select a manageable component. In the initial release you have three options: **Internet Explorer Security Settings**; **Internet Explorer Settings**; and **Windows Settings**.

After you select a component, pick something from the **Select a task category from this list** dropdown. When you do *that,* a list of tasks will appear in the **Select an individual task from this list** listbox. (Yes, we know: it *does* sound like a lot of work, doesn't it?) Click one of the tasks, and three things will happen:

- A script will appear in the spot labeled **Script for carrying out this task**. As the name implies, this script can be used to configure the registry setting in question.
- A script will appear in the spot labeled **Script for retrieving the current task setting**. If you run this script as-is, it will return the registry value as currently configured on the local computer.
- Information will appear in the spot labeled **Task description** describing what the task actually entails, and what the values actually mean. (For example, "If this value is 1, then it means whatever we're talking about is enabled. If the value is 0, then whatever we're talking about is disabled.")

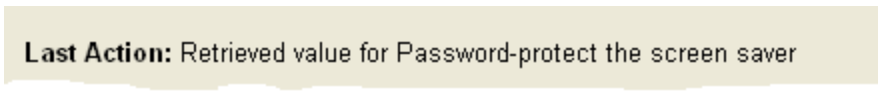
Your screen should look something like this:



This is actually pretty straightforward, and maybe we should have just stopped there and called it good. We decided to add a few additional capabilities, however, and while we think these are useful, it might not be so obvious what they are. With that in mind, here's a quick rundown of what the various Tweakomatic buttons do:

Run Script button

Ok, this one *is* pretty obvious: it runs the script. The only thing to keep in mind here is that there are two scripts (one for configuring info, one for retrieving info), and thus two Run buttons. Don't get confused and click the button for configuring info (the one on the left) rather than the one for retrieving info. To help you out a little, there is a status box at the bottom of the screen (labeled **Last Action**) that will constantly remind you what you just did. (Yes, it *is* just like being married, isn't it?)

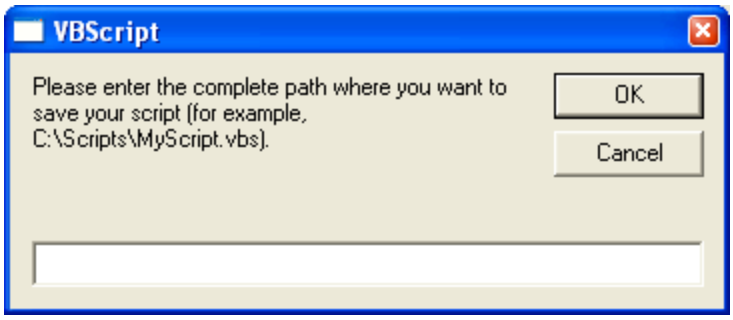


One thing we should note is that when you click the **Run** button the script just runs; we don't pop up a dialog box that asks, "Are you sure you want to run this script?" If you'd like to have that as a sort of safety device, remember, the Tweakomatic is just an HTA: open up Tweakomatic.hta in Notepad, and make whatever changes you wish.

Save Script button

Well, OK, this one is pretty obvious, too. Click it, type in a name, and whatever script is onscreen at the time will get saved. Again, the only thing to remember is that you have two different scripts, and thus two different **Save** buttons.

Oh, fine. You'll be a bit disappointed, but here's what the dialog box looks like:



Change Value button

Here's some good news: there's only one **Change Value** button! The **Change Value** button (which we introduced earlier) is found only in the configuration part of the Tweakomatic. What does it do? Well, the Tweakomatic will write a default script for you that, say, enables X. Thus the script might set the value of X to 1. But suppose you'd really like to *disable* X, meaning you have to set the value to 0. Well, you can either directly edit the script code, or you can click **Change Value**, type in the new value (0), and let the Tweakomatic make the change for you. (Yes, you *would* have to be pretty lazy to go this route, wouldn't you?)

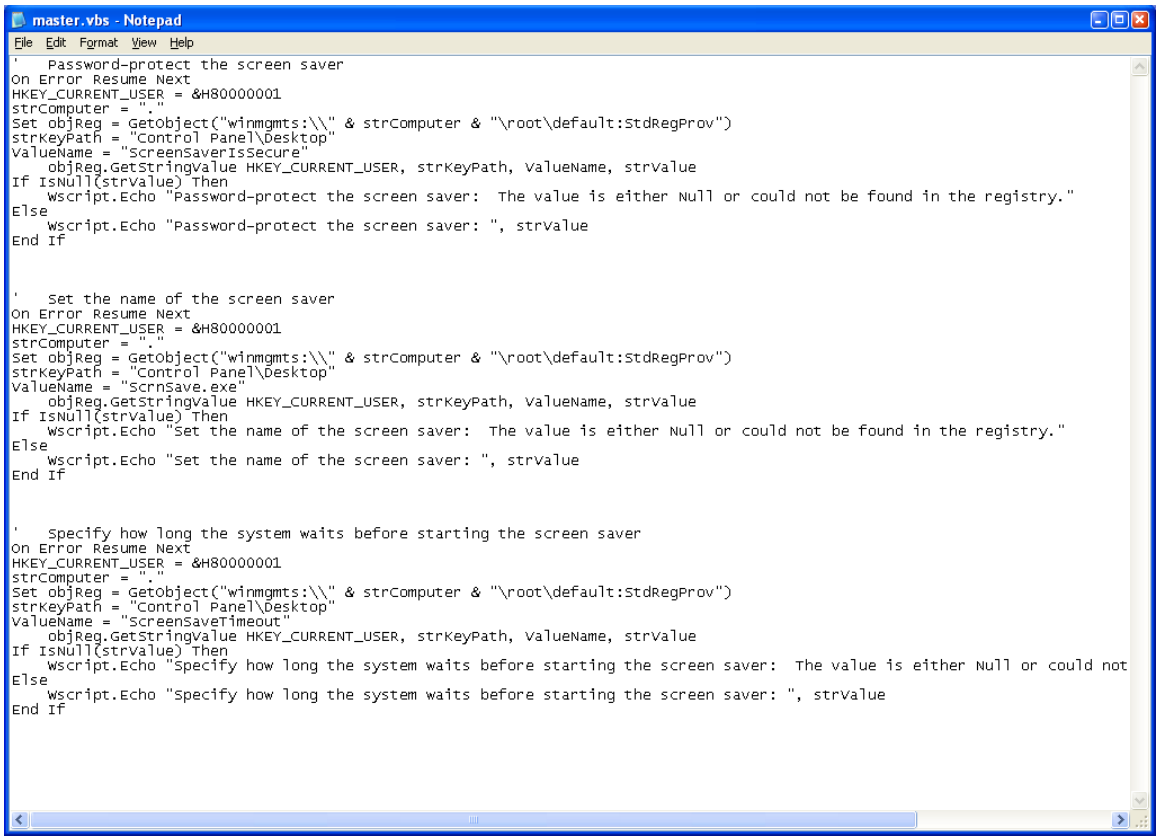
Master Script button

This is actually a good idea, but not a very good name. Initially we had only the **Save Script** button, which allowed you to save the current script to a file. That was nice, but it occurred to us that—considering the nature of what the Tweakomatic does—well, maybe you don't want hundreds of individuals scripts (that is, one script that tells you whether **Run** appears on the Start menu, another that tells you whether **My Documents** appears on the Start menu, another that tells you whether **Printers and Faxes** appears on the Start menu, etc.). Instead, you might like to have one script that carries out *all* these functions.

Therefore, we came up with the idea of a "master" script. Click the **Master Script** button (and, yes, there are two of them, one for retrieval, one for configuration), and the script currently onscreen will automatically be added to your "master" script. (Unless you haven't told the Tweakomatic the name of your master script. In that case, the first time you click this button you'll need to enter a file name.) We think if you use this a few times you'll see that it's actually quite handy; we just aren't sure that the name Master Script is the name we were looking for. (But it could be worse: initially this was called the Append Script, we guess because you appended data to it.)

Show Script button

This simply pops up Notepad and displays the appropriate Master Script. (Hey, what more would you *want* it to do?) That can be handy if you've been adding a bunch of scripts to the master script and now want to review what you have and haven't done:



Set Computer Name button

By default each time the Tweakomatic writes a script for you, it sets the name of the computer to a dot (**.),** which, in WMI-speak, represents the local computer. Suppose you'd rather have it, by default, set the name of the computer to **Computer1**. No problem; click **Set Computer Name**, type **Computer1** in the resulting dialog box, and then click **OK**. If you ever want to switch back, click the button, type **.** in the dialog box, and click **OK**.

Set Configuration Master Script button

Ok, again, not the most obvious of button titles, but this one allows you to specify the name of the "Master Script" for scripts that actually configure a setting.

Set Retrieval Master Script button

And, as you might have guessed, this one allows you to specify the name of the "Master Script" for scripts that retrieve the current value of a setting.

That should do it. If you have questions or comments about the Tweakomatic, please send them to the Scripting Guys at scripter@microsoft.com. (Of course, if anyone wearing a TweakUI shirt asks, you never heard of the Tweakomatic. Just to be on the safe side)