

Running Scripts with the CScript and WScript Utilities

Contributed by Adi Bach
 Friday, 15 December 2006
 Last Updated Tuesday, 15 January 2008

Windows supports two methods of starting scripts. The CScript application works at the command prompt, while the WScript application works from within the graphical user environment. Both applications accomplish the same task—they provide a means for interpreting a script file you create.

{mosgoogle}

CScript and WScript use the same command line. You must provide a script name as the first command line argument. Most scripts have a VBE or JS file extension, but any extension is acceptable. For example, you can still use VBS files with Windows Scripting Host (WSH), but the icon won't look right, in some cases, and you can't double-click it to start the execution with newer Windows products.

The VBS extension is the right choice for older versions of Windows. The icon is yellow for VBE files and blue for JS files. These utilities use the following syntax:

```
CScript [] []
WScript [] []
```

The following list describes each of the command line arguments:

- `//?` Displays the currently documented command line switches. The newest versions of WSH tend to reject older switches, even those of the undocumented variety.
- `//B` Limits user interaction with the script. Batch mode suppresses all non-command line console user interface requests from the script. It also suppresses error message display (a change from previous versions).
- `//D` Activates debugging mode so you can fix errors in a script.
- `//H:CScript` Makes CSCRIPT.EXE the default application for running scripts. (WScript is the default engine.)
- `//H:WScript` Makes WSCRIPT.EXE the default application for running scripts.
- `//I` Allows full interaction with the user. Any pop-up dialog boxes will wait for user input before the script continues.
- `//Job: JobName` Executes a WSH job. A WSH job has a Windows Script File (WSF) extension. This file enables you to perform tasks using multiple scripting engines and multiple files. Essentially, this allows you to perform a "super batch" process. Creating WSF files is an advanced technique not discussed in this book because it isn't very useful in most cases. You can learn more about this topic at <http://msdn.microsoft.com/archive/en-us/wsh/htm/wsAdvantagesOfWs.asp>
- `//Logo` and `//NoLogo` WSH normally prints out a logo message. You'd use the `//NoLogo` switch to prevent WSH from displaying this message.
- `//S:` This command line switch allows you to save current command line options for a user. WSH will save the following options: `//B`, `//I`, `//Logo`, `//NoLogo`, and `//T:n`.
- `//T:TimeLimit` Limits the maximum time the script can run to the number of seconds specified. Normally, there isn't any timeout value. You'd use this switch in situations where a script might end up in a continuous loop or is unable to get the requested information for other reasons. For example, you might use this switch when requesting information on a network drive.

`//X` Starts the script in the debugger. This allows you to trace the execution of the script from beginning to end.

`//U` Outputs any console information using Unicode instead of pure ASCII. You use this switch on systems where you need to support languages other than English. This is a CScriptonly option.

Notice that all of these command line switches start with two slashes (`//`) to differentiate them from switches you may need for your script. WSH passes script arguments to your script for processing. Script arguments can be anything including command line switches of your own or values needed to calculate a result.

Users of older versions of CScript and WScript may remember the `//C` and the `//W` switches used to switch the default scripting engines. Newer versions of CScript and WScript replace these switches with the `//H` switch.

You'll also find the `//R` (reregister) and `//Entrypoint` switches missing from WSH because script developers no longer need the functionality.

Always use the correct command line switches for the version of Windows and WSH installed on your machine. You can work with WSH in either interactive or batch mode. Use batch mode when you need to perform tasks that don't require user input. For example, you might want to run Scan Disk every evening, but use different command line switches for it based on the day.

You could use Task Scheduler to accomplish this task, but using it in conjunction with a WSH script improves the flexibility you get when running the task.

Another kind of batch processing might be to send log files to your supervisor or perhaps set up a specific set of environment variables for a character-mode application based on the current user. On the other hand, interactive mode requires user interaction.

You'd use it for tasks such as cleaning the hard drive because you don't always know whether the user needs a particular file.

Such a script could ask the user a set of general questions, and then clean excess files from the hard drive based on the user input. The cleaning process would follow company guidelines and save the user time.

Because batch processing doesn't require any form of user input, it's usually a good idea to include the `//T` switch with the `//B` switch. This combination stops the script automatically if it runs too long. In most cases, using this switch setup stops an errant script before it corrupts the Windows environment or freezes the machine. However, you can't time some tasks with ease.

For example, any Web-based task is difficult to time because you can't account for problems with a slow connection. In this case, you'll need to refrain from using the `//T` switch or provide a worst-case scenario time interval.

The next set of command line switches to consider is `//Logo` and `//NoLogo`. There isn't any right or wrong time to use these switches, but you usually use the `//Logo` switch when testing a script and the `//NoLogo` switch afterward. The reason is simple.

During the testing process, you want to know about every potential source of problems in your script environment, including an old script engine that might cause problems. On the other hand, you don't want to clutter your screen with useless text after you debug the script. Using the `//NoLogo` switch keeps screen clutter to a minimum.