



F1 Solutions November 2005



::REM

Hello All,



The Techmentor conference in San Jose was pretty successful I thought. We had a good turnout of enthusiastic scripting administrators. If you couldn't make it I hope you'll try and come to the Orlando conference at the end of March. I expect I will be back as part of the scripting track, plus there's the possibility of a post-conference session on Virtual Server 2005, the Virtual Server Migration Toolkit and Automated Deployment Services. More on that in future issues if it comes to fruition.

The advanced VBScript manuscript is being edited now. We're still hoping for an early 2006 release by Microsoft Press. I'll be sure to let you know when to watch for it as soon I know anything. I have tentative plans for another major writing project which I'll share with you as soon as it is official.

This month we finish up our short series on WMI events by taking a look at asynchronous queries. This is an expanded tutorial so I've cut out a few sections including the 10 Minute Script. Don't worry. I'll be back next month with a quick and dirty 10 minute script.

For my U.S. readers (yes, I have a number of international subscribers), I hope you have a pleasant, happy and healthy Thanksgiving surrounded by friends and family.

As always, I appreciate your continued support and welcome all comments, suggestions and feedback at jhicks@jdhitsolutions.com.

~Jeff

Cobian Backup 7



I don't have very complicated backup needs and since money is always a consideration, I'm always looking for freeware solutions. Lately I've been testing a freeware backup solution called Cobian Backup 7. This small yet flexible application can be run as either a service or as an end-user application. The current version supports Win9x and later.

One thing to mention right off is that this is not a backup program like NTBackup. You can create full, incremental and differential backups to files, but there is no granular restore. The program takes your file and directory selections and basically creates an archive file using either ZIP or SQX compression. If you need to restore a single file, you need to decompress the entire archive to a temporary directory and then get the file or files you need. Still, for some purposes this is not necessarily a bad thing. My backup needs tend to be business continuity purposes or rebuilding systems.

Volume 2, Issue 9

November 2005

Special points of interest:

- Cobian Backup 7
- Geek Humor
- WMI Events Part 3

Inside this issue:

- | | |
|------------|---|
| Geek Humor | 2 |
| Tech Tutor | 2 |

Cobian Backup- cont'd.

Compression is pretty good. I was able to do a full backup of 139MB. Using a ZIP archive I got 25.27MB and with SQX 22.74MB. You can password protect the backups as well as encrypt them with strong ciphers such as BlowFish.

You can backup directly to an FTP server or even backup to multiple locations. This last feature is very intriguing. I was able to run a backup with 2 destinations and it didn't really take any longer than a single destination. You could run a backup to a local folder and one to a network share

for extra protection. Cobian includes some pretty good logging and the option to send an email report when a backup finishes. Scheduling backup jobs is very easy and you can configure Cobian to retain X number of backup sets.

The application has the ability to run remote backups as well but I have not tested that feature yet.

You can download the latest version at <http://www.cobian.se>

“Cobian Backup is a multi-threaded program that can be used to schedule and backup your files and directories from their original location to other directories/drives in the same computer or other computer in your network. FTP backup is also supported in both ways (upload and download).”

“There is no great genius without a tincture of madness.”

—Seneca

Geek Humor — Planet DeBella

New Messages for Microsoft Windows (<http://www.jdbshow.com/geek2.html#win2000>)

1. Enter any 11-digit number to continue.
2. Press any key except... no,No,NO, NOT THAT ONE!
3. This will end your Windows session. Do you want to play another game?
4. File not found. Should I fake it? (Y/N)
5. Runtime error 6D at417A:32CF: Incompetent user.
6. Windows Virus Scan 1.0- Windows found: Remove (Y/Y)
7. Your Hard Drive has been scanned and all stolen software titles have been deleted. The police are on the way.

Tech Tutor—WMI Events Part 3

In last month's tutorial we looked at using a notification query and blocking for the next event. This works fine for single events but rarely are we interested in just monitoring for a single event. The other drawback to using blocking is that the script can't do anything else until an event fires. WMI events can also be queried asynchronously. This means that while the query is running the script can continue to run and do other things. However, just as with a synchronous query, the script must be running when the events fire or you won't be notified. Let's take a look through this script here.

```
Dim oWMI,oEventSrc,NTEvent,objSINK
strComputer="."
strQuery="Select * from __InstanceCreationEvent WHERE TargetInstance " & _
"ISA 'Win32_NTLogEvent'"

'need security privilege in case event is a security event
Set oWMI=GetObject("winmgmts:{(security)}\\" & strComputer & "\root\cimv2")
```



Tech Tutor — abcd

```

Set objSINK = wscript.CreateObject("WbemScripting.SWbemSink","SINK_")

oWMI.ExecNotificationQueryAsync objSINK, strQuery

MsgBox "Waiting for an event to happen. Do NOT press OK until you " & _
"get your results or you won't see anything.",vbokonly+vbinformation, _
"WMI Sink Demo"

'Cancel SINK since we no longer need it around to receive information.
objSINK.Cancel

wscript.quit

'callback subroutine
'this is code to happen when we get a return from the async Call
Sub SINK_OnObjectReady(NTEvent,refContext)

WScript.Echo NTEvent.Path_
logtime=NTEvent.TargetInstance.TimeGenerated
logyr = Left(logtime,4)
logmo = mid(logtime,5,2)
logdy = mid(logtime,7,2)
logtm = mid(logtime,9,6)

WScript.Echo "  Event ID: " & NTEvent.TargetInstance.EventCode & _
"    Source: " & NTEvent.TargetInstance.SourceName & vbTab & logmo & _
"/" & logdy & "/" & logyr & " [" & FormatDateTime(left(logtm,2) & _
":"&Mid(logtm,3,2) & ":" & Right(logtm,2),3) & "]" & vbCrLf & _
NTEvent.TargetInstance.message

End Sub

```

Much of the script is what we've been working with. One thing you'll notice is that the method used to execute the query is **ExecNotificationQueryAsync**. With this query method, WMI sends off the query without waiting for a response. The script can continue executing other code. In this demo script there's not much else to do but keep the script alive. We do this with a simple *MsgBox* function. As long as we don't click OK, the script will run and events will be returned to the script. But how does the server know how to talk back to the script? Without getting into the ugly DCOM details, this is done with a special type of object called a *sink*. Like a kitchen sink, everything gets drawn to it and funneled down. The same thing happens here. The remote server sends it's response back to the sink and any code within the sink gets executed.

We instantiate the sink by creating a **WbemScripting.SWbemSink** object. The method requires a second parameter which will be the name of the sink. DCOM tradition is that the sink name end in an underscore.

```
Set objSINK = wscript.CreateObject("WbemScripting.SWbemSink","SINK_")
```

The query notification method ties the query to the sink object.

```
oWMI.ExecNotificationQueryAsync objSINK, strQuery
```

When the remote computer responds with an event it sends an *OnReady* call to the originating server. The script's sink is waiting for this call and sends the response to a subroutine or function whose name is a combination of the sink name and *OnReady*. Hence, we have subroutine called *Sink_OnReady*. Just as I demonstrated last time, WMI sends an object, in this case an **NTEvent** object and a context reference object which we can ignore. The subroutine executes everytime the script receives an *OnReady* call.

Unlike the simple notification query with blocking, this script will continue to display event information as long as the script is running, which in our demo script will be until the message box is dismissed.

(Continued on Page 4)

“Against stupidity
the very gods
Themselves
contend in vain.”

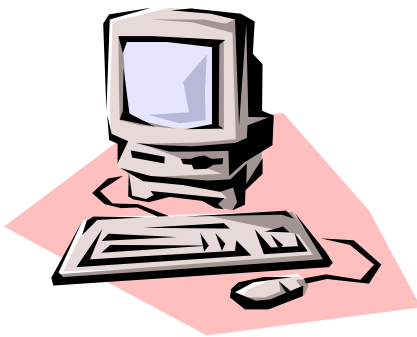
— Schiller

JDH Information Technology Solutions

6349 Tulipwood Lane
Jamesville, NY 13078

Phone: (315) 256-7023
Fax: (315) 295-2534
E-mail: jhicks@jdhitsolutions.com

WE'RE ON THE WEB AT
HTTP://
WWW.JDHITSOLUTIONS.COM



If you wish to no longer receive this newsletter, please send an email to:

newsletters@jdhitsolutions.com

Use a subject line of Unsubscribe.

**This newsletter was created with
Microsoft Publisher 2003**

**Copyright 2005 All Rights Reserved
JDH Information Technology Solutions, Inc.**

**All trademark names are property of their
respective owners**

Disclaimer:

**All code and script samples are provided
'as is' with no warranty, either expressed
or implied.**

**Use at your own risk and test thoroughly in
a non-production environment.**

Mission Statement

Our mission is to provide outstanding information technology consulting services and solutions to our clients utilizing a value-oriented approach. We recognize that most information technology projects are goal not hour driven. Our aim is to leverage technology to solve our clients' business challenges in the most cost-effective manner possible. We succeed when they succeed.

WMI Events – continued

Open a command prompt and run the script. Move the message box out of the way and leave it alone. Now stop and start services which will generate event log entries. Each entry will be displayed on the command line for as long as you let the script run. When you close the message box the script quits and you no longer receive any notifications.

You can have as many sinks as you might need in a script and more than one computer can use the same sink. What this means is that you could process a text list of servers executing an asynchronous query for each one. Depending on the number of servers and what types of events might be happening, you could be getting responses back to the sink while still querying other servers. That's the power of an asynchronous query.

Granted a message box isn't the best way to keep a script alive. You might use `wscript.sleep` if you know you only want to monitor for specific period of time. Suppose you want to monitor for 5 minutes, you could use the command

```
wscript.sleep 3000000
```

Remember you specify a time value in milliseconds with the sleep method.

Or you might use some sort of loop in place of the message box.

```
For x=1 to 1000
    Wscript.sleep 3000
Next
```

If I've done my math right, this loop accomplishes the same thing as the previous sleep command. You might also use a Do While loop and check for the existence of a file or process. As long the file or process doesn't exist, keep looping. Be careful of using an infinite loop otherwise you'll end up needing Task Manager to kill the script.

As written, this script relies on the command prompt window to display results. You could just as easily use an instance of Internet Explorer, send results to a file, use the `WshShell.Popup` method, or create an HTA. Perhaps in a future tutorial we'll try out one of these options.

With asynchronous event queries you have many options and enormous flexibility.. These types of scripts can be some of the most powerful in your library. If you have a great script that takes advantage of asynchronous queries, I hope you can share it with me and others in a future newsletter.

If you've liked this little series, please let me know. I'd also like to hear about other topics that interest you.

[Next month I'll have another 10 Minute Script]