At a glance:

Run as Administrator for third-party scripting tools

Run as Another User

Prompt Here as System for CMD and Windows PowerShell

Drag-and-drop Elevation Gadget

New Elevation PowerToys for Windows Vista

Michael Murgolo

Welcome to another edition of the Elevation PowerToys for Windows Vista. I took a close look at this in the June 2007 issue of *TechNet Magazine*. Here we are a year later. This time around, I want to show you how I expanded the Run as Administrator functionality to some of my favourite third-party scripting tools, demonstrate how you can replace a nice Windows XP feature that was removed from Windows Vista, and look at some of the handy tools that are included in Elevation PowerToys.

Run as Administrator for additional scripting tools

One topic I discussed in the previous article (available at http://technet.microsoft.com/magazine/cc162321.aspx) was enabling the Run as Administrator option for the native Windows scripting tools. For this article, I've created Run as Administrator PowerToys for some third-party scripting tools:

- AutoIt v3 (www.hiddensoft.com)
- AutoHotkey (www.autohotkey.com)
- ActivePerl (www.activestate.com)
- KiXtart 2010 (www.kixtart.org)

The code for each of these is included in the download for this article, which you can find at http://technetmagazine.com. The actual files are named ElevateAutoIt3.inf, El-

evateAutoHotKey.inf, ElevatePerlScript.inf, and ElevateKiXtart.inf, respectively. For AutoIt v3, AutoHotkey and ActivePerl, setup is quite simple. Just download the corresponding application and install it in the default location. After you've installed the applications you are interested in, you can then just install the appropriate Run as Administrator PowerToy for each tool.

Unfortunately, KiXtart 2010 does not provide an installer. Therefore, in order to make sure KiXtart is installed in a standard location so my PowerToy will work properly, I've provided an INF file that will installKiXtart 2010 (v 4.60) into Program Files\KiXtart and register the .kix file extension.

Go to www.kixtart.org/?p=downloads, download KiX2010_460.zip, and unzip it into a folder. Copy the Install_KiXtart.inf file (included in the code download that accompanies this article) into that same folder. Then right-click on Install_KiXtart.inf and select Install. After that, you can simply install the ElevateKiXtart.inf PowerToy.

Run as another user powertoy

User Account Control (UAC) was created to make the operating system less vulnerable to malware by having users, even those who are Administrators, run most applications with standard user privileges. UAC offers elevation potential for administrative tasks and other app functions. This elevation potential is provided through the Run as Administrator option, which you get when right-clicking on executable files. The Elevation PowerToys that I discussed in the June 2007 issue extended this capability to work on other file and object types.

The functionality built into Windows Vista works quite well for many administrative tasks. However, one important scenario was left out for Windows Vista. Many IT departments have a policy where network administrators use one user account for their everyday tasks (using e-mail, creating documents, and the like) and another account that is used only for network administration (or local computer administration).

This is done to help lower the risk that if a network administrator accidentally runs malware doing his everyday tasks, he will not compromise his entire system – or the domain he is on. This was accomplished on Windows XP using the Run as... right-click option. But this option is gone in Windows Vista because it was replaced with the Run as Administrator option.

The runas command-line tool, however, still exists in Windows Vista. Unfortunately, it cannot be used for the most common dual account tasks – running Microsoft Management Console (MMC) snap-ins. For example, say you have been delegated some account management tasks in Active Directory. You are running as a standard user for everyday tasks, and your network administration account is also a member of the local administrators group (so you can install network management tools when needed) on a Windows Vista computer with UAC enabled. Now you want to start Active Directory User & Computers (ADU&C) with your Active Directory administrative account, so you try the runas command, as follows:

> UAC was created to make the OS less vulnerable to malware by having users run apps with standard privileges

```
runas /user:mydomain\admin
"mmc.exe %windir%\system32\dsa.msc"
```

Unfortunately, this does not cause ADU&C to launch. Instead, you receive a runas error that says "The requested operation requires elevation." What is happening in this case is that the MMC executable is marked to run at the highestAvailable privilege level. Since the highestAvailable level for your network administration account is as administrator, launching ADU&C in this manner would require elevation. Since runas does not cause a prompt for elevation, the error occurs.

So Windows Vista makes this scenario difficult by not providing a context menu item

for Run as... and by providing no built-in means to run a process as another user that requires elevation.

This would be a frustrating article if there was no solution to this, but as luck would have it, one of my original Elevation Power-

> Windows Vista offers no context menu item for Run as... and no built-in way to run a process as another user to be elevated

Toys provides the key to solving the second problem, and I've whipped up another one to solve the first. (I wish I could claim that I thought up the solution to the elevation problem, but this was hit upon by Gov Maharaj of the Windows AppCompat team.)

It turns out that the Elevate Command PowerToy can be used with the runas command. Where the previous command failed to cause an elevation prompt, the following will cause the prompt:

```
runas /user:mydomain\admin
"elevate mmc.exe%windir%\system32\dsa.msc"
```

This causes runas to launch elevate.cmd (technically, the process being launched is cmd.exe) as the other user, and the elevate command takes care of launching mmc.exe with an elevation prompt.

Finally, I've taken this trick, combined it with file associations for .exe and .msc files, and given it an HTML Application UI to create a PowerToy that creates a Run as Another User option available through the right-click menu. When you select Run as Another User, you get an HTML Application like the one shown in **Figure 1**.

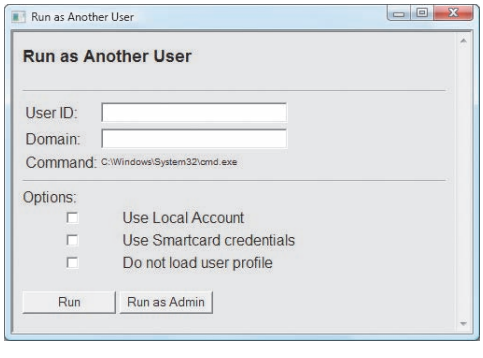

Figure 1 **Run as Another User tool**

Here, just enter the user name and domain – for an account on the local computer, check the Use Local Account checkbox. You can then click the Run button to launch the application as a standard user, or you can click the Run as Admin button to launch the application with elevated privileges. After you click either of these two buttons, runas.exe will run and prompt for a password or smart card pin.

Since this PowerToy uses the Elevate Command PowerToy, you need to install that first. Then right-click on the RunAs.inf file, select Install, and approve the elevation. To uninstall the tool, use the Programs and Features Control Panel.

You will find that some of the shortcuts to .msc files in the Administrative Tools (such as Computer Management) will work with this PowerToy. But note that if you install the Windows Server 2003 administrative tools using adminpak.msi, the shortcuts that are created are not standard shortcuts to the .msc files. Instead, they are Windows Installer shortcuts, and as a result Windows Explorer will not display the Run as Another User option for those shortcuts.

For those shortcuts you will either have to find the actual .msc files and right-click on them or create new shortcuts to the .msc files. Additionally, runas does not work with Internet Explorer® due to the way Internet Explorer was re-architected for Protected Mode in Windows Vista (you can find additional information concerning this issue at http://support.microsoft.com/?id=922980).

Note: between the time I finished these PowerToys and the publication of this article, Windows Sysinternals has released a new tool that is functionally very similar to my Run as Another User PowerToy. It's called ShellRunas and can be found on the Windows Systernals site: http://technet.microsoft.com/sysinternals/cc300361.

Since the folks at Sysinternals actually write real code for a living, you may prefer their tool for your own use. I decided to leave

mine in the article as an example of how this task in particular and shell extensions in general can be done using HTML Applications with script code.

CMD and PowerShell Prompt Here as System

There are times when it is necessary to run programs in the Local System context. For example, many software distribution tools, such as System Center Configuration Manager (SCCM), use a client agent that runs as Local System to accomplish its tasks.

To test the behaviour of a software installation program running as Local System before attempting a distribution with a product such as SCCM, it can be helpful to start the installer using a command prompt running as Local System. Therefore, I set out to create my CMD and PowerShell Prompt Here as System PowerToys.

When using Windows XP, I used to accomplish a function like this with a command shell script:

```
@echo off
sc create CmdAsSystem type= own type= interact
binPath= "cmd /c start cmd /k (cd c:\ ^& colour ec ^&
title ***** SYSTEM *****)"
net start CmdAsSystem
sc delete CmdAsSystem
```

However, if you try to run this from an elevated command prompt on Windows Vista, you get the following error message and the command prompt running as System will not appear:

```
WARNING:  The service CmdAsSystem is configured as
interactive whose support is being deprecated. The
service may not function properly.
```

The problem is that this script tries to create and start an interactive service. Interactive services will not function correctly due to Session 0 Isolation in Windows Vista. (For an explanation of Session 0 Isolation, see the "Services in Windows Vista" whitepaper available at www.microsoft.com/uk/vistaservices)

To work around this limitation, I used the Psexec tool that was developed by Sysinternals (see http://technet.microsoft.com/sysinternals/bb897553.aspx). This tool allows processes to be started in the System context. Unfortunately, most Sysinternals tools do not include an installer. Therefore, I've provided an INF file (available in the code download) to install the whole Sysinternals Suite (which includes Psexec) into Program Files\Sysinternals Suite. As an added bonus, this INF file creates Start Menu shortcuts for the Suite's graphical tools.

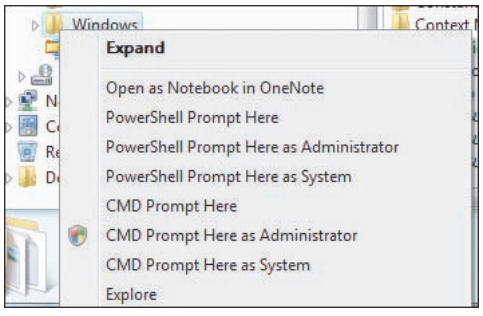

Figure 2 **CMD Prompt Here as System and PowerShell Prompt Here as System options**

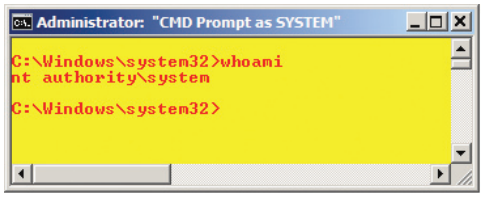

Figure 3 **CMD Prompt as System must be used responsibly**

Figure 4 **The drag-and-drop Elevation Gadget**

To install the suite, first download SysinternalsSuite.zip from http://technet.microsoft.com/sysinternals/bb842062.aspx and unzip it into a folder. Copy my INF file (Install_SysinternalsSuite.inf) into that folder, right-click Install_SysinternalsSuite.inf, and select Install. Since these new PowerToys use the Elevate Command PowerToy, install that next. After doing that, you can install CmdHereAsSystem.inf and PowerShellHereAsSystem.inf. Once these PowerToys are installed, you will then have the CMD Prompt Here as System and PowerShell Prompt Here as System options available as right-click items for folders and drives in Windows Explorer, as you can see in **Figure 2**.

Figure 3 shows a CMD prompt running as system. I selected the bright colours as a reminder that this prompt is running as System and can do unexpected (and damaging) things to the system if the wrong commands are entered.

Finally, these PowerToys add commands to the system so that these prompts can be started in the Run dialog or a CMD prompt. For example, executing the following command from the Run box will start a CMD prompt as System in the Windows folder:

```
cmdassystem "c:\windows"
```

The equivalent command for Windows PowerShell is psassystem. You should note that I have also modified the CMD and PowerShell Prompt Here as Administrator PowerToys to install similar commands – cmdasadmin and psasadmin, respectively.

> You can drag and drop an executable or a script onto the Elevation Gadget and the item will launch as elevated

Elevation Gadget

Most of my PowerToys require clicking the right mouse button. But as a bonus for this update, I have included something a bit more fun. This is a Windows Sidebar Gadget, which I call the Elevation Gadget (shown in **Figure 4**). It is a drag-and-drop target. Just drag an executable or a script from Windows Explorer that has a runas action defined and it will launch as elevated.

If you have installed my previous Elevation PowerToys, then this will work for Windows Script Host scripts, Windows PowerShell scripts, HTML Applications, and Windows Installer packages and patches (as well as executables and command shell scripts that have a runas action defined by default in Windows Vista). And you can also drag more than one item at a time. (Just try dragging a folder to the gadget and see what happens.)

To install the gadget, double-click on Elevation.gadget (available in the code download). If you would like to look at the code for the gadget, just add the .cab extension to the file name. You can then extract the contents from the Cab file.

Wrapping up

The download for this article contains both the new PowerToys and the ones from the original article. I've made minor changes to a few of the old ones. For example, the original versions of Windows PowerShell Prompt Here as Administrator and Elevate WSH Script both installed their own copies of elevate.cmd and elevate.vbs. Since several of the new PowerToys also depend on these files, I've changed these tools to require Elevate Command PowerToy to be installed (and then they share that copy). To determine if a PowerToy requires the installation of the Elevate Command PowerToy, check the header in the INF file.

Some other original PowerToys have minor, non-functional changes as well. Always uninstall the old version of a PowerToy before installing the latest. Since this collection has now expanded to 17 tools, I've included command shell scripts to install and uninstall the entire collection (InstallAllPowerToys.cmd and UninstallAllPowerToys.cmd, respectively). You can customise these to install and uninstall only the tools that you need to use.

You should keep in mind that InstallAllPowerToys.cmd does not install the Run as Administrator PowerToys for third-party scripting tools by default. You can modify this script to install only those PowerToys for which you have installed the software. When you run either of these, it will relaunch itself as elevated. UninstallAllPowerToys.cmd should remove all the old versions of these tools as well.

As with all my PowerToys, these are unsupported, use-at-your-own-risk tools. And they are not official Microsoft products – they are my own personal creations. These were only tested by me and a few other volunteers on 32-bit Windows Vista with US English as the default language. Finally, it is possible that any or all of these PowerToys may not work with future Windows updates, service packs

Michael Murgolo *is a Senior Infrastructure Consultant for Microsoft Consulting Services. He focuses on operating systems, deployment, network services, Active Directory, systems management, automation and patch management. He is a subject matter expert in the area of desktop deployment and migration.*