At a glance:
Running as a standard user
File and registry virtualisation
Elevating account status

Inside Windows Vista User Account Control

Mark Russinovich

User Account Control (UAC) is an often misunderstood feature in Windows Vista. In my previous *TechNet* articles on Windows Vista kernel changes, available at *www.microsoft.com/uk/technetmagazine*, I didn't cover UAC because I felt it merited its own article. In a two-part series, concluding next issue, I discuss the problems UAC solves and describe the architecture and implementation of its component technologies. These technologies include the refactoring of operations that previously required administrative rights, lightweight virtualisation to help programs run correctly without administrative rights, the ability for programs to explicitly request administrative rights, and isolation of administrative processes from non-administrative processes running on the same user desktop.

UAC's goal

UAC is meant to enable users to run with standard user rights, as opposed to administrative rights. Administrative rights give users the ability to read and modify any part of the operating system, including the code and data of other users – and even Windows itself. Without administrative rights users cannot accidentally (or deliberately) modify system settings, malware can't alter system security settings or disable antivirus software, and users can't compromise the sen-

32_36_Vista.desfin1.indd 32 9/8/07 15:23:19

sitive information of other users on shared computers. Running with standard user rights can therefore reduce urgent help desk calls in corporate environments, mitigate the impact of malware, keep home computers running more smoothly and protect sensitive data on shared computers.

UAC had to address several problems to make it practical to run with a standard user account. First, prior to Windows Vista, the Windows usage model had been one of assumed administrative rights. Software developers assumed their programs could access and modify any file, registry key or operating system setting. Even when Windows NT introduced security and differentiated between accesses granted to administrative and standard user accounts, users were guided through a setup process that encouraged them to use the built-in Administrator account or one that was a member of the Administrators group.

The second problem UAC had to address was that users sometimes need administrative rights to perform such operations as installing software, changing the system time and opening ports in the firewall.

The UAC solution to these problems is to run most applications with standard user rights, obviate the need for administrator rights all the time and encourage software developers to create applications that run with standard user rights. UAC accomplishes these by requiring administrative rights less frequently, enabling legacy applications to run with standard user rights, making it convenient for standard users to access administrative rights when they need them and enabling even administrative users to run as if they were standard users.

Running as a standard user

A full audit of all administrative operations during the development of Windows Vista identified many that could be enabled for standard users without compromising the security of the system. For example, even corporations that adopted standard user accounts for their Windows XP desktop systems were unable to remove their travelling users from the Administrators group for the sole reason that Windows XP does not differentiate changing the time zone from changing the system time. A laptop user who wants to configure the local time zone so that their appointments show correctly in their calendar when they travel must have the 'Change the system time' privilege (internally called SeSystemTimePrivilege), which by default is only granted to administrators.

Time is commonly used in security protocols like Kerberos, but the time zone only affects the way that time is displayed, so Windows Vista adds a new privilege, 'Change the time zone' (SeTimeZonePrivilege), and assigns it to the Users group, as seen in **Figure 1**. This makes it possible for many corporations to have their laptop users run under standard user accounts.

Windows Vista also lets standard users configure WEP settings when they connect to wireless networks, create VPN connections, change power management settings and install critical Windows updates. In addition, it introduces Group Policy settings that enable standard users to install printer and other device drivers approved by IT administrators and to install ActiveX® controls from administrator-approved sites.

What about consumer and line of business (LOB) applications that do not run correctly in standard user accounts? While some software legitimately requires administrative rights, many programs needlessly store user data in system-global locations. Microsoft recommends that global application installers that expect to run with administrative rights create a directory under the %ProgramFiles% directory to store their application's executable files and auxiliary data and create a key under HKEY_LOCAL_MACHINE\Software for their application settings. When an application executes, it can be running in different user accounts and it should therefore store user-specific data in the per-user %AppData% directory and save per-user

Prior to Windows Vista, the Windows usage model had been one of assumed administrative rights

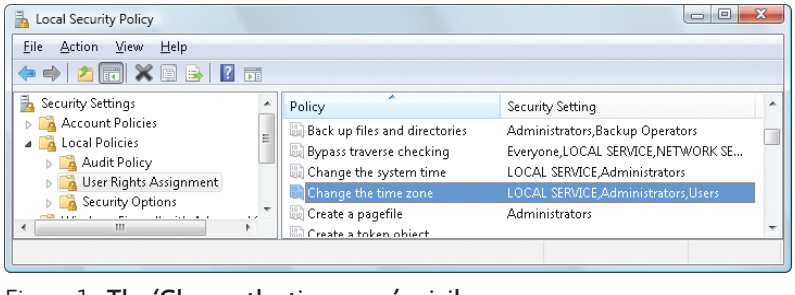

Figure 1 **The 'Change the time zone' privilege**

32_36_Vista.desfin1.indd 33 9/8/07 15:23:22

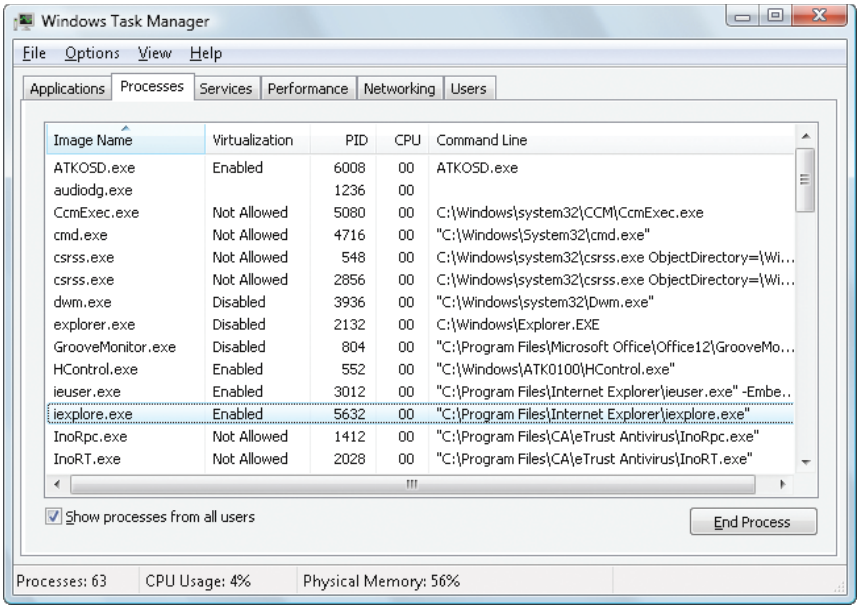

Figure 2 **Task Manager shows virtualisation status**

The effects of virtualisation

You can change the virtualisation status of a process by selecting Virtualization from the context menu that appears when you right-click it in Task Manager. **Figure A** shows the behaviour of a command prompt when its virtualisation status changes. It starts out with virtualisation disabled because it has a Windows Vista manifest. Because it's running with standard user rights, it is unable to create a file in the \Windows directory, but after it becomes virtualised with Task Manager it appears to create the file successfully. When its virtualisation returns to its disabled state it can't find the file, which is actually in the user's virtual store.

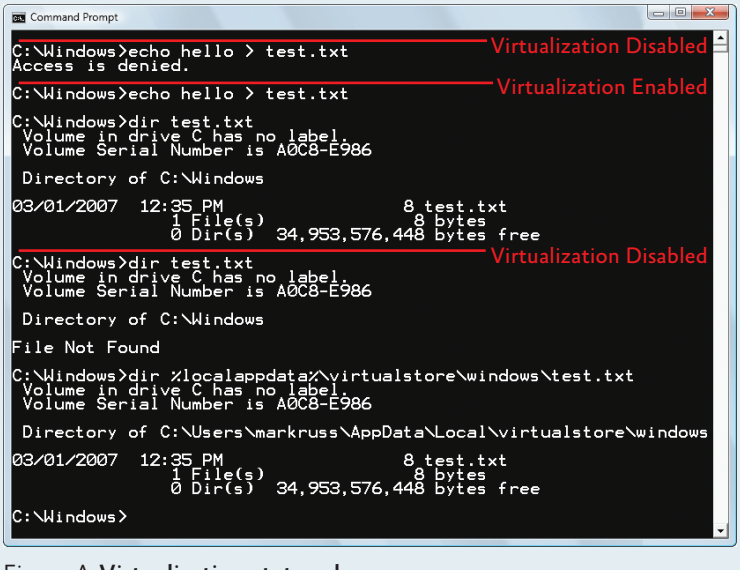

Figure A **Virtualisation status change**

settings in the user's registry profile under HKEY_CURRENT_USER\Software. Standard user accounts don't have write-access to the %ProgramFiles% directory or HKEY_LOCAL_MACHINE\Software, but because most Windows systems are single-user and most users have been administrators up until Windows Vista, apps that incorrectly save user data and settings to these locations work anyway.

Windows Vista enables these legacy applications to run in standard user accounts through the help of file system and registry namespace virtualisation. When an application modifies a system-global location in the file system or registry and that operation fails because access is denied, Windows redirects the operation to a per-user area; when the application reads from a system-global location, Windows first checks for data in the per-user area and, if none is present, permits the read attempt from the global location.

For the purposes of this virtualisation, Windows Vista treats a process as legacy if it's 32-bit (versus 64-bit),

> Vista enables legacy apps to run in standard user accounts through virtualisation

is not running with administrative rights and does not have a manifest file indicating that it was written for Windows Vista. Any operations not originating from a process classified as legacy according to this definition, including network file sharing accesses, are not virtualised. A process's virtualisation status is stored as a flag in its token, which is the kernel data structure that tracks the security context of a process, including its user account, group memberships and privileges.

32_36_Vista.desfin1.indd 34 9/8/07 15:23:32

You can see the virtualisation status of a process by adding the Virtualisation column to Task Manager's Processes page. **Figure 2** shows that most Windows Vista components, including Desktop Window Manager (Dwm.exe), Client Server Runtime Subsystem (Csrss.exe) and Explorer, either have virtualisation disabled because they have a Windows Vista manifest or are running with administrative rights and hence do not allow virtualisation. Internet Explorer (iexplore.exe) has virtualisation enabled because it can host multiple ActiveX controls and scripts and must assume that they were not written to operate correctly with standard user rights.

The file system locations that are virtualised for legacy processes are %ProgramFiles%, %ProgramData% and %SystemRoot%, excluding some specific subdirectories. However, any file with an executable extension, including .exe, .bat, .scr, .vbs, and others, is excluded from virtualisation. This means that programs that update themselves from a standard user account fail instead of creating private versions of their executables that aren't visible to an administrator running a global updater. To add additional extensions to the exception list, enter them in the following registry key and reboot:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\
Services\Luafv\Parameters\
ExcludedExtensionsAdd
```

Use a multi-string type to delimit multiple extensions and do not include a leading dot in the extension name.

Modifications to virtualised directories by legacy processes redirect to the user's virtual root directory, %LocalAppData%\VirtualStore. For example, if a virtualised process that is running on my system creates C:\Windows\Application.ini, the file it actually creates is C:\Users\Markruss\AppData\Local\VirtualStore\Windows\Application.ini. The Local component of the path highlights the fact that virtualised files don't roam with the rest of the profile when the account has a roaming profile.

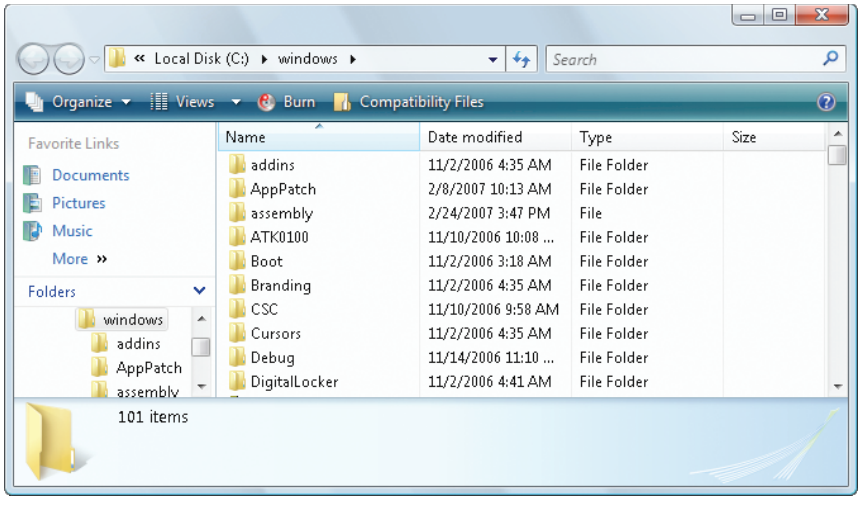

Figure 3 **Compatibility Files button indicates virtualised files nearby**

If you navigate in Explorer to a directory containing virtualised files, Explorer displays a button labelled Compatibility Files in its toolbar, as shown in **Figure 3**. Clicking the button navigates you to the corresponding VirtualStore subdirectory to show you the virtualised files.

Figure 4 shows how the UAC File Virtualization Filter Driver (%SystemRoot%\System32\Drivers\Luafv.sys) implements file system virtualisation. Because it's a file system filter driver, it sees all file system operations, but it only implements functionality for operations from legacy processes. You can see that it changes the target file path for a legacy process that creates a file in a system-global location, but does not for a process running a Windows Vista application with standard user rights. The legacy process believes that the operation succeeds when it really created the file in a location fully accessible by the user, but default permissions on the \Windows directory deny access to the application written for Windows Vista.

Registry virtualisation is implemented slightly differently from file system virtualisation. Virtualised registry keys include most of the HKEY_LOCAL_MACHINE\Software branch, but

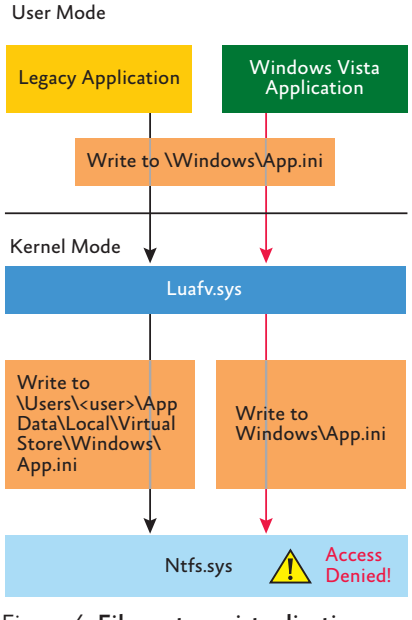

Figure 4 **File system virtualisation**

32_36_Vista.desfin1.indd 35 9/8/07 15:23:36

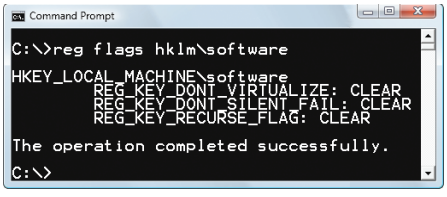

Figure 5 **The Reg.exe utility shows virtualisation flags**

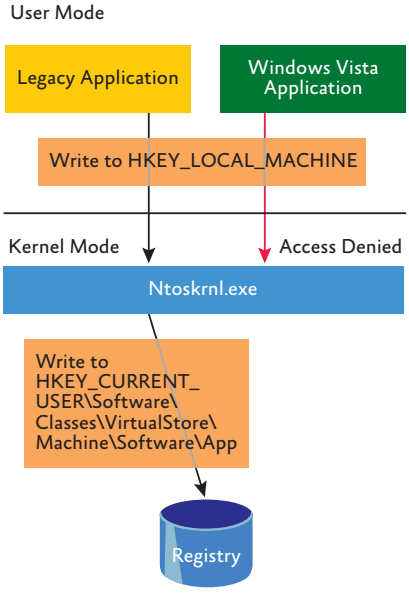

Figure 6 **Registry virtualisation**

there are numerous exceptions, such as the following:

```
HKLM\Software\Microsoft\Windows
HKLM\Software\Microsoft\Windows NT
HKLM\Software\Classes
```

Only keys that are commonly modified by legacy applications, but that don't introduce compatibility or interoperability problems, are virtualised. Windows redirects modifications of virtualised keys by a legacy application to a user's registry virtual root at HKEY_CURRENT_USER\Software\Classes\VirtualStore. The key is located in the user's Classes hive, %LocalAppData%\Microsoft\Windows\UsrClass.dat, which, like any other virtualised file data, does not roam with a roaming user profile.

Instead of keeping a fixed list of virtualised locations as Windows does for the file system, the virtualisation status of a key is stored as a flag, REG_KEY_DONT_VIRTUALIZE, in the key itself. The Reg.exe utility can show the flag and the two other virtualisation-related flags, REG_KEY_DONT_SILENT_FAIL and REG_KEY_RECURSE_FLAG (**Figure 5**). When REG_KEY_DONT_SILENT_FAIL is set and the key is not virtualised (REG_KEY_DONT_VIRTUALIZE is set), a legacy application that would be denied access performing an operation on the key is instead granted any access the user has to the key rather than the ones the application requested. REG_KEY_RECURSE_FLAG indicates if new subkeys inherit the virtualisation flags of the parent instead of the default flags.

Figure 6 shows how registry virtualisation is implemented by the Configuration Manager, which manages the registry in the operating system kernel, Ntoskrnl.exe. As with file system virtualisation, a legacy process creating a subkey of a virtualised key is redirected to the user's registry virtual root, but a Windows Vista process is denied access by default permissions.

Some apps require additional help to run correctly with standard user rights

In addition to file system and registry virtualisation, some applications require additional help to run correctly with standard user rights. For example, an application that tests the account in which it's running for membership in the Administrators group might otherwise work, but won't run if it's not in that group. Windows Vista therefore defines a number of application-compatibility shims so that such applications work anyway. The shims most commonly applied to legacy applications for operation with standard rights are shown in **Figure 7**. Corporate IT professionals can use tools like the Application Compatibility Toolkit (ACT, from **technet.microsoft.com/windowsvista/aa905066.aspx**) and its Standard User Analyzer (SUA) utility, or Aaron Margosis's LUA Buglight (**blogs.msdn.com/aaron_margosis/archive/2006/08/07/LuaBuglight.aspx**) to identify the shim requirements for their LOB applications. They assign shims to an application using the Compatibility Administrator, also part of ACT, and then deploy the resulting compatibility database (.sdb file) to their desktops via Group Policy. Note that, if required, virtualisation can be completely disabled for a system using a local security policy setting. ■

Figure 7 **Common user shims**

Shim	Purpose
ElevateCreateProcess	Changes CreateProcess to handle ERROR_ELEVATION_REQUIRED errors by calling the Application Information Service to prompt for elevation.
ForceAdminAccess	Spoofs queries of administrator group membership.
VirtualizeDeleteFile	Spoofs successful deletion of global files and directories.
LocalMappedObject	Forces global section objects into the user's namespace.
VirtualizeHKCRLite, VirtualizeRegisterTypeLib	Redirects global registration of COM objects to a per-user location.

32_36_Vista.desfin1.indd 36 9/8/07 15:23:40