Autodesk University 2007

Windows Vista: What CAD Managers Need To Know

Tom Conti – Walt Disney Imagineering

CM115-1P This class takes a look at Windows Vista from a CAD manager's point of view. What's changed, what's new, and how will it affect you and your users? Will it make your job easier or harder? Come see how Vista will change your life. While some overlap is inevitable, the majority of the course content is NOT included in the Windows Vista: The New Frontier course.

About the Speaker:

Tom is a senior principal CAD systems analyst for Walt Disney Imagineering (WDI) in Glendale, California. He supports over 150 CAD users at multiple locations, and is an expert in system and CAD performance, as well as troubleshooting. Tom has been with WDI for over 18 years, and has over 30 years of computer experience. He is one of the most highly rated speakers at AU.
tom.f.conti@disney.com

As a CAD manager, your interest in the operating system is deeper than the average user's. You look for features that will help you do your job, things that may cause problems for your users, ways to do things faster and easier. Windows Vista has some significant changes from previous versions.

User Account Control

When you're setting up a new computer, the new User Account Control (UAC) feature will definitely make you say to yourself, "How do I turn this off?" Tasks that were simply and quickly done in XP now result in multiple dialogs. Try creating a folder in C:\Program Files, and then rename it. You go through 6 dialogs, and that's when you have administrative privileges!

But the UAC is a significant part of Vista's protection against viruses and intruder attacks. It reduces the inherent danger of using an administrator account for everyday tasks by requesting your consent when an application needs to do something with system-wide effect, which includes virtually all administrative tasks. Furthermore, architectural changes wrought by UAC make it practical for most people to use a standard account for daily computing.

UAC intercedes whenever a user or program attempts to perform a system administrative task and asks for the consent of a computer administrator before commencing what could be risky business.

To understand why UAC is effective you need to look at security before Windows Vista. Computer security experts have long preached the *least privilege* rule, that you should give only enough access for a person to perform his or her job. This basic security tenet is sometimes referred to as LUA, an acronym that, depending upon whom you ask, stands for "limited user account," "least user access," "least-privileged user account," or something similar. (I love acronyms. For a long time I collected them. If you do, too, one word of advice. Don't try to use the line, "Would you like to come up and see my acronyms?" It doesn't work.) In earlier versions of Windows, by default all accounts were set up as administrator accounts, with full privileges to do anything on the computer, including the ability to easily and inadvertently install viruses and perform other harmful tasks. This goes against what the proponents of LUA stand for (truth, justice, and a more secure computing experience!). As it turns out, however, using a limited account in Windows XP is practically impossible, primarily because most applications were written with the assumption that users would have full administrative privileges and don't run properly (or at all) with a limited account.

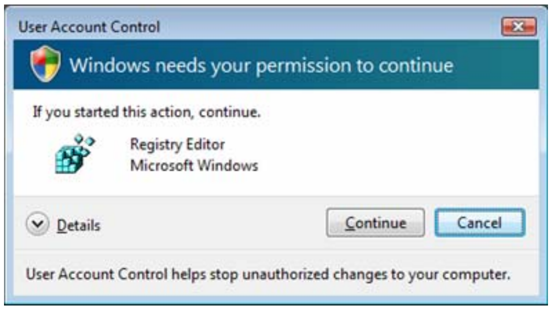
By contrast, in Windows Vista, all accounts, with the exception of the first one that gets created as part of the installation process, are non-administrator standard accounts by default. While they can carry out all the usual daily computing tasks, they're prevented from performing potentially harmful operations. These restrictions apply not just to the user; more important, they also apply to any programs launched by the user. Even administrator accounts run as "protected administrator" accounts, in which they run with standard-user privileges except when they need to perform administrative tasks.

So what triggers a UAC prompt? Any type of action that requires elevation to administrator status (and therefore displays a UAC elevation prompt) include those that make changes to system-wide settings or to files in %SystemRoot% (usually C:\Windows) or %ProgramFiles% (usually C:\Program Files). Among the actions that require elevation are:

- Installing and uninstalling applications
- Installing unsigned or not trusted device drivers
- Installing ActiveX controls
- Installing Windows Updates
- Changing settings for Windows Firewall
- Changing UAC settings
- Configuring Windows Update
- Adding or removing user accounts
- Changing a user's account type
- Configuring Parental Controls
- Running Task Scheduler
- Restoring backed-up system files
- Viewing or changing another user's folders and files

Here's an interesting bit of info. At logon, Windows creates a security access token that is used to identify the privilege levels of your account. Standard users get a standard token, but administrators get two: a standard token and an administrator token. The standard token is used to open Explorer.exe (the Windows shell), from which all subsequent programs are launched. Child processes inherit the token of the process that launches them so, by default, all applications run as a standard user, even when you're logged on with an administrator account. Certain programs request elevation to administrator privileges, and that's when the UAC prompt is displayed. If you provide administrator credentials, the program opens using the administrator token. And any processes that the successfully elevated program opens also run with administrative privileges.

As an elevation-requesting application attempts to open, UAC evaluates the application and the request and then displays an appropriate prompt. As an administrator, the most common prompt you're likely to see is the consent prompt. Read it, check the name of the program, click *Continue*, and carry on.


User Account Control

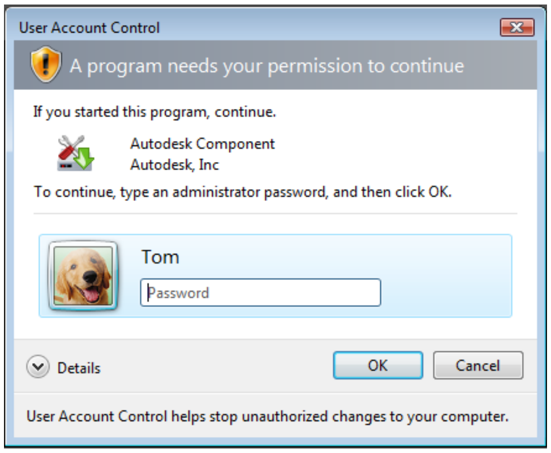
Windows needs your permission to continue

If you started this action, continue.

Registry Editor
Microsoft Windows

User Account Control helps stop unauthorized changes to your computer.

If you use a standard account, when a program requires elevation, you'll see the credentials prompt.



User Account Control

A program needs your permission to continue

If you started this program, continue.

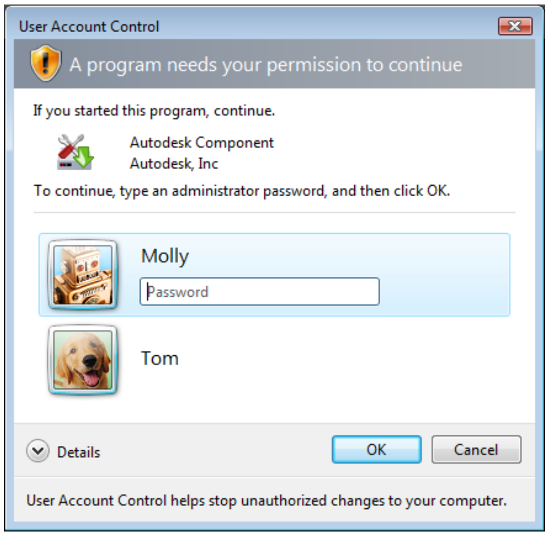
Autodesk Component
Autodesk, Inc

To continue, type an administrator password, and then click OK.

Tom
Password

User Account Control helps stop unauthorized changes to your computer.

Before proceeding I think it's worth noting a couple of things about this dialog. At the time I did this, there was one account, *Tom*, with admin rights on the computer. As such the dialog presented only that account for authentication. I added another account, *Molly*, and gave it admin rights. I did the same action and got the following dialog.



User Account Control

A program needs your permission to continue

If you started this program, continue.

Autodesk Component
Autodesk, Inc

To continue, type an administrator password, and then click OK.

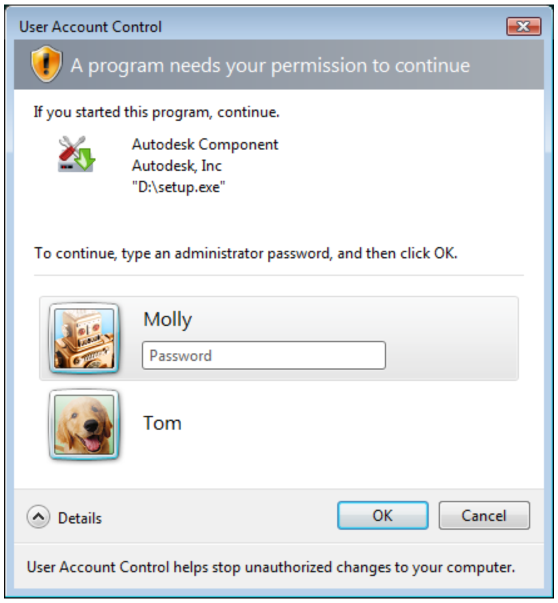
Molly
Password

Tom

User Account Control helps stop unauthorized changes to your computer.

This time, it presented both accounts to choose from. What I find interesting about this is that it's giving you the login names instead of just presenting a dialog asking for the login name and password. From a security viewpoint, it seems like you wouldn't want to provide the login names.

You'll notice that even though the dialog tells you that the dialog was prompted by a program, and who's program it is, it doesn't tell you which program is trying to run. To see this, click the Details button.


User Account Control

A program needs your permission to continue

If you started this program, continue.

Autodesk Component
Autodesk, Inc
"D:\setup.exe"

To continue, type an administrator password, and then click OK.

Molly
Password

Tom

User Account Control helps stop unauthorized changes to your computer.

Okay. Moving on. If the user is able to provide the credentials (user name and password, or smart card, or fingerprint, depending on how logon authentication is configured on the computer) of an administrator, the application opens using the administrator's access token.

Here's an interesting bit of information about UAC. When you see the UAC dialog, the desktop darkens behind it and you can't click on anything except in the UAC dialog. The interesting bit is that what you're seeing behind the dialog is not really the desktop, but rather a picture of it. The UAC dialog box creates a secure desktop, a separate process that no other application can interfere with. It does this so a malicious program can't put another dialog box in front of the UAC dialog box asking you to let the program proceed, or grab your keystrokes and learn your administrator logon password. When UAC invokes the secure desktop, it snaps a picture of the desktop, darkens it, and then displays that image behind the dialog box. Once the secure desktop is displayed, you can't switch tasks or click the windows on the desktop because they're not really there. Pretty darn clever, eh?

While you are doing an initial computer setup, you may want to turn off UAC. Once you have the computer set up, you should turn it back on. UAC is your first line of defense against malicious attacks.

One way to disable UAC is to use regedit.exe to go into the Registry and navigate to HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System. Change the value for *EnableLUA* from *1* to *0*. To enable it again, change the value back to *1*.

Another way to do this is to open *Control Panel*, *User Accounts*, go to an account, click *Turn User Account Control on or off*, turn on/off the check box for *Use User Account Control (UAC) to help protect your computer*.

A third way is to go to *Start…* type *msconfig* into the *Start Search* box… go to the *Tools* tab… double-click the *Disable UAC* option.

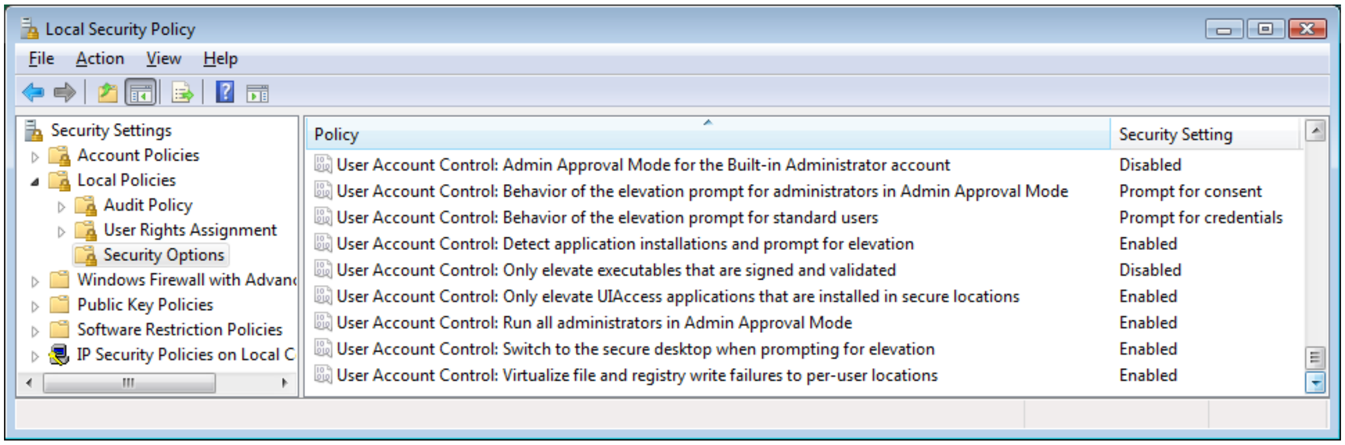
After you enable or disable UAC, you will have to reboot your computer for the changes to take effect.

Maybe you don't want to disable UAC for all users. Maybe you only want to disable it for administrators. In the same Registry location, HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System, there's a key value named *ConsentPromptBehaviorAdmin*. Change its value data from *2* to *0* and, when logged in with an account with administrative privileges, you won't get the consent dialogs any more.

If you normally log on with a standard user account, you can right-click an application and select *Run as Administrator*, and you'll be presented with a dialog to enter an administrator's credentials.

If you want to always run the Command Prompt in Administrator mode, you can change the properties of the shortcut that launches it. Right-click the shortcut, then *Properties…*, click the *Advanced* button and turn on the check box for *Run as administrator*.

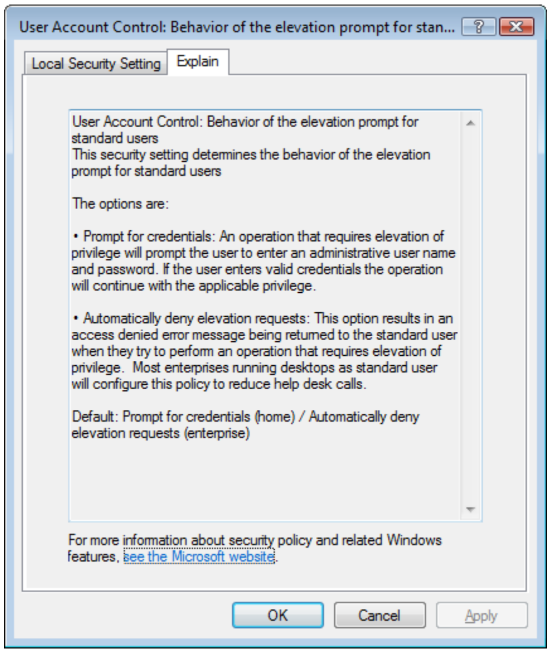
If you have the Business, Enterprise, or Ultimate edition of Vista, you can go into the Local Security Policy (*Start*, enter *secpol.msc*) and set a number of local policies that control the behavior of UAC. Here's the policy showing the UAC default settings.



Policy	Security Setting
User Account Control: Admin Approval Mode for the Built-in Administrator account	Disabled
User Account Control: Behavior of the elevation prompt for administrators in Admin Approval Mode	Prompt for consent
User Account Control: Behavior of the elevation prompt for standard users	Prompt for credentials
User Account Control: Detect application installations and prompt for elevation	Enabled
User Account Control: Only elevate executables that are signed and validated	Disabled
User Account Control: Only elevate UIAccess applications that are installed in secure locations	Enabled
User Account Control: Run all administrators in Admin Approval Mode	Enabled
User Account Control: Switch to the secure desktop when prompting for elevation	Enabled
User Account Control: Virtualize file and registry write failures to per-user locations	Enabled

The choice for all but two of the policies are *Enabled* or *Disabled*. For *Behavior of the elevation for administrators in Admin Approval Mode* the choices are *Elevate without prompting, Prompt with credentials,* and *Prompt for consent*. For *Behavior of the elevation prompt for standard users*, the choices are *Automatically deny elevation requests*, and *Prompt for credentials*. Be sure you understand what the policies do before you enable or disable them.

For any of the policies, when you double-click the policy you can get more information about it by going to the *Explain* tab.


User Account Control: Behavior of the elevation prompt for stan...
Local Security Setting
Explain
OK
Cancel
Apply

User Account Control: Behavior of the elevation prompt for standard users
This security setting determines the behavior of the elevation prompt for standard users

The options are:

• Prompt for credentials: An operation that requires elevation of privilege will prompt the user to enter an administrative user name and password. If the user enters valid credentials the operation will continue with the applicable privilege.

• Automatically deny elevation requests: This option results in an access denied error message being returned to the standard user when they try to perform an operation that requires elevation of privilege. Most enterprises running desktops as standard user will configure this policy to reduce help desk calls.

Default: Prompt for credentials (home) / Automatically deny elevation requests (enterprise)

For more information about security policy and related Windows features, see the Microsoft website.

SIDs

I'm sure that you CAD managers all know that Windows assigns a unique identifier, called a Security Identifier, or SID, to every account that gets created. The SID remains associated with that account until the account is deleted. And even if you create an account, later on delete it, and then create another account with the same exact properties, the SID will be different. SIDs are never re-used.

When a user logs into Windows, it validates the login name and password, then it creates a security access token. The security access token includes the user name and SID, plus information about any security groups to which the account belongs. As the user goes from folder to folder, Windows checks the token to see if the user has rights to the folder. Because the token is created every time a user logs on, if you add a user to a new security group, the change in rights doesn't take effect until the user logs in again.

Have you ever gone into a security dialog and saw SIDs instead of account names? Sometimes the SID shows up and then changes to the account name once Windows has had a chance to look up the account information. If the SID doesn't change to an account name, it may be because the account has been deleted, or it may be because the account is a network account and you're not connected to the network.

You'll find SIDs in the hidden protected operating system folder \Recycler. Each SID you see in this folder represents the Recycle Bin for a particular user. In the registry, the HKEY_USERS hive contains a key, identified by SID, for each user account on the computer.

The easiest way to determine your own SID is with Vista's new command line utility *whoami*. This utility can be used to get user name and group information along with the respective security identifiers (SID), privileges, and logon identifier (logon ID) for the current user on the local system. Enter *whoami /?* to see available arguments.

Not all SIDs are unique, although SIDs assigned to normal user accounts are always unique. A number of commonly used SIDs are constant among all Windows installations. For example, S-1-5-18 is the SID for the built-in Local System account, a hidden member of the Administrators group that is used by the operating system and by services that log on using the Local System account. For a list of what Microsoft calls "well-known" SIDs, see the Microsoft Knowledge Base article 243330 at http://support.microsoft.com/kb/243330.

Administrator Account

When you install Windows Vista, the Administrator account is disabled by default. There is one exception. If you upgrade from Windows XP and Administrator is the only active local account, then Administrator remains enabled.

Using Vista's default settings, the Administrator account has one unique capability, it's not subject to UAC, even when UAC is turned on for all other users. It runs with full administrative privileges at all times and never needs your consent for elevation, although you will still get the consent dialog to do the action. This obviously makes it easier for viruses, etc., to do bad things if you're using this account.

However, if you want to enable this account, you can do it either of two ways.

- *Control Panel… Administrative Tools… Computer Management… Local Users and Groups… Users…* open the properties for the Administrator account and clear the check box for *Account is disabled*
- At an elevated command prompt (in the *Start* menu search box type *cmd* and press *Ctrl-Shift-Enter*), enter the following command.

```
net user Administrator /active:yes
```

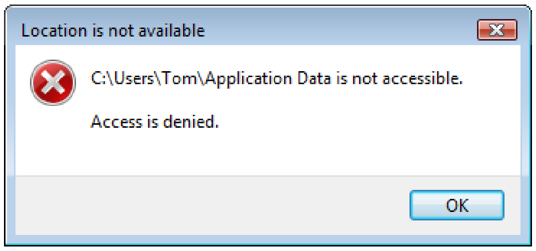
IMPORTANT SAFETY NOTE: the Administrator account does not initially have a password! If you do enable this account be sure to set a password for it.

In Windows XP, when you start the computer in Safe Mode, you have to logon using the local Administrator account. In Vista, on a computer that is not joined to a domain, as long as there is at least one other administrator account, if you need to use Safe Mode you must use one of the other administrator accounts to log on and perform administrative tasks. Standard users can log on in Safe Mode, but they face the same restrictions as when Windows is running normally. If you somehow manage to delete, disable, or demote the last administrator account (User Accounts in Control Panel won't let it happen, but it is possible with the other account management tools), then Safe Mode allows the Administrator account to log on, even if it's disabled.
On a computer that's joined to a domain, you can never log on in Safe Mode using a disabled Administrator account. Recovery in a domain environment relies on members of the global Domain Admins group. Any member of that group can log on and create a local administrator account for further repair work if necessary. If an account that's a member of Domain Admins has logged onto the computer previously, its cached credentials can be used to log on in Safe Mode. If the domain administrator has never logged onto the computer, cached credentials don't exist; in that situation, you must start Safe Mode With Networking.

How Windows Vista maintains compatibility with Windows XP

The folder structure used by Windows Vista is different than previous versions of Windows. Let's start with the user folder structure.

A number of the standard user folders have new names in Vista. To avoid problems with programs that expect the XP structure, Vista has a new object called a *junction* (short for *junction point*). A junction looks a folder shortcut. Junctions are hidden by default. To see them when you enter *DIR* at a Command Prompt, you have to use the */a* parameter. To see them in Explorer you have to turn on the *Show hidden files and folders* and *Hide protected operating system files (Recommended)* options. But try double-clicking a junction, even when logged on with an account with administrative rights, and you'll get a denial dialog.



C:\Users\Tom\Application Data is not accessible.

Access is denied.

That's because for all of these junctions, the Everyone group has a Deny access control entry (ACE) preventing users from listing folder contents. This Deny ACE may seem drastic, but it's Vista's way of saying, "Don't mess with the compatibility infrastructure." Note that the Deny ACE does not prevent you from deleting a junction, but you should never perform such a deletion unless you absolutely know what you are doing. Although a junction looks like an ordinary shortcut in Windows Explorer, it's not what it appears to be. Deleting a shortcut deletes a pointer, leaving what it was pointing to unchanged. Deleting a junction has the same effect as deleting the location to which it points.

And it's interesting to note that because the access-denied message is reminiscent of the messages displayed by UAC, you might think that UAC is causing the access problem. In fact, this is entirely an NTFS permissions issue, and has nothing to do with UAC. You can confirm it by turning off UAC, and you still won't have access to the junctions.

The junctions below the user folder point to the locations listed below.

Application Data	C:\Users\loginname\AppData\Roaming
Cookies	C:\Users\loginname\AppData\Roaming\Microsoft\Windows\Cookies
Local Settings	C:\Users\loginname\AppData\Local
My Documents	C:\Users\loginname\Documents
NetHood	C:\Users\loginname\AppData\Roaming\Microsoft\Windows\Network Shortcuts
PrintHood	C:\Users\loginname\AppData\Roaming\Microsoft\Windows\Printer Shortcuts
Recent	C:\Users\loginname\AppData\Roaming\Microsoft\Windows\Recent
SendTo	C:\Users\loginname\AppData\Roaming\Microsoft\Windows\SendTo
Start Menu	C:\Users\loginname\AppData\Roaming\Microsoft\Windows\Start Menu
Templates	C:\Users\loginname\AppData\Roaming\Microsoft\Windows\Templates

Let’s compare the folders below the %USERPROFILE% folder in XP and Vista.

Windows XP	Windows Vista
C:\Documents and Settings\loginname	C:\Users\loginname
	AppData
Application Data	Application Data *
	Contacts
Cookies	Cookies *
Desktop	Desktop
	Documents
Favorites	Favorites
	Links
Local Settings	Local Settings *
	Music
My Documents	My Documents *
NetHood	NetHood *
	Pictures
PrintHood	PrintHood *
Recent	Recent *
	Saved Games
	Searches
SendTo	SendTo *
Start Menu	Start Menu *
Templates	Templates *
UserData	
	Videos

* indicates a junction

You may be wondering how to create a new junction. If you right-click in Explorer and look at the *New* list, junction’s not there. So how do you make one? Vista has a new feature, the ability to use file-based symbolic links. The new command is MKLINK.

Enter *mklink /?* at a command prompt and you’ll get

```
Creates a symbolic link.

MKLINK [[/D] | [/H] | [/J]] Link Target

        /D      Creates a directory symbolic link.  Default is a file
                symbolic link.
        /H      Creates a hard link instead of a symbolic link.
        /J      Creates a Directory Junction.
        Link    specifies the new symbolic link name.
        Target  specifies the path (relative or absolute) that the new link
                refers to.
```

Using mklink without one of the three optional parameter is the same as using the */d* option and will create a soft link to a file or folder.

Be aware that Vista’s symbolic links are not usable by previous versions of Windows.

Windows Vista uses virtualization to provide compatibility with current legacy programs. And Microsoft does not promise to include it with future versions of Windows.

Many legacy applications write data (such as configuration information) to areas that are ordinarily inaccessible to standard accounts. This behavior presented few problems in Windows XP, because most users ran with administrative privileges. In Windows Vista, that is no longer the case. To avoid errors that would otherwise arise because users, even those with administrative accounts, are now expected to carry out most operations in a non-administrative security context, Windows Vista redirects writes (and subsequent reads) to per-user virtualized locations.

Let's say an application, running in your security context, attempts to write to a location within %ProgramFiles%, the write will be redirected to a comparable location within %LocalAppData%\VirtualStore. When the application subsequently reads what it has written, the read request is redirected to the same virtualized location. As far as the application is concerned, everything is perfectly normal, and the operating system has prevented standard-user access to the %ProgramFiles% folder.

A similar form of virtualization protects sensitive areas of the registry. Programmatic access to HKLM\Software is redirected to HKLM\Software\Classes\VirtualStore.

Virtualization does not affect administrative access to files or registry keys, 64-bit processes, and virtualized data does not move with roaming profiles.

Vista’s Graphics Subsystem

A brief history of OpenGL

OpenGL started as Silicon Graphic’s IRIS GL and introduced in 1992. It was first implemented in Windows in NT 3.5 in 1994. OpenGL wasn’t very fast back then (okay, it was out and out slow), so Microsoft came out with Direct X as their solution for fast graphics in Windows. Then Microsoft came out with Direct X 3D for better gaming. In 1996, Microsoft delivered OpenGL 1.1 for Windows NT 4.0 and Windows 95, and it’s been a part of Windows ever since. Until now.

What’s in Vista

In previous versions of the Windows operating system, the graphics subsystem was part of the Windows kernel where it had to contend with and share resources with other major operating system functions. In addition to consuming resources, display drivers, according to Microsoft, were responsible for up to 20 percent of all BSODs (Blue Screen Of Death) in XP. In Windows Vista, most of the graphics subsystem has been moved out of the kernel and given its own space, the Windows Presentation Foundation (WPF). While a lot of what you initially see in Vista may seem like so much visual fluff, the Aero interface, and the application user interfaces that take full advantage of the WPF graphics subsystem, can take advantage of all kinds of graphical effects, including 2-D, 3-D, 3-D animation, effortless scaling, vector-based text and shape rendering and motion, transparency, blurring, shadows in motion, object movements, and more.

For best performance, you want a PCI Express video card with at least 256MB of video memory, but you’ll get all of the features with 128MB.

And Vista has a new type of video driver, the Windows Display Driver Model (WDDM) driver. One of the benefits of the WDDM that you'll appreciate is that you no longer have to reboot the system when you update the driver. Obviously, you'll want to make sure that your graphics card supports WDDM.

Another huge benefit of WDDM is stability. There is still a small kernel mode component that is solely responsible for lower-level functionality. The user mode component provides the heavier functionality, like the translation from higher-level API constructs to direct graphics processing unit (GPU) commands while maintaining application compatibility. This greatly reduces the chance of a fatal blue screen and most graphics driver-related problems result in, at worst, one application being affected.

WDDM also provides fault-tolerance against display driver hangs. This enables Windows Vista to detect system hangs and restart the display driver again without the need of a system reboot.

Vista ships with Direct X version 9, and, according to Microsoft, OpenGL is only supported as a layer sitting on top of DirectX. I've read that if you disable the Aero interface, you can still use raw OpenGL, but I haven't tried this.

Driver Verifier Manager

Warning! Driver Verifier Manager is not intended for use by the casual user! This is a powerful tool and should only be used by a knowledgeable person! This tool is also available in Windows XP.

When your computer acts unpredictably, a buggy device driver may be the cause. If you're experiencing unexplained computer problems, a powerful troubleshooting tool called Driver Verifier Manager (verifier.exe) is a terrific way to identify flawed device drivers. Instead of your computer locking up at a most inopportune time with a misleading Blue Screen of Death, Driver Verifier Manager stops your computer predictably at startup with a BSOD that accurately explains the true problem. Although this doesn't sound like a huge improvement (your system still won't work, after all), Driver Verifier Manager performs a critical troubleshooting step: identifying the problem. You can then correct the problem by removing or replacing the offending driver.
Driver Verifier works at startup to thoroughly exercise each driver. It performs many of the same tests that are run by Microsoft's Windows Hardware Quality Lab (WHQL) as part of the certification and signing process, such as checking for the way the driver accesses memory.

If you're satisfied that the driver really is okay despite Driver Verifier Manager's warning, you can turn off Driver Verifier for all drivers or for a specific driver. Any driver that Driver Verifier chokes on should be regarded with suspicion, but some legitimate drivers bend the rules without causing problems.

Beware! If Driver Verifier Manager finds a nonconforming driver, even one that doesn't seem to be causing any problems, it will prevent your system from starting. Use Driver Verifier only if you're having problems. In other words, if it ain't broke ….

To begin working with Driver Verifier Manager, you must start it using credentials from an account in the Administrators group. Open a Command Prompt window using the *Run As Administrator* option, type *verifier* at the command line, and press *Enter*. In the Driver Verifier Manager dialog box, select *Create Standard Settings*. In the next dialog box, select the type of drivers you want to verify. Unsigned drivers are a likely cause of problems, as are those created for an older version of Windows.

When you click *Next*, you get a list of all currently installed drivers that match the conditions you specified. Note that the list might contain a mix of hardware drivers and some file-system filter drivers, such as those used by antivirus programs, CD burning software, and other low-level system utilities.

At this point you have two choices.
1) Go through the list and make a note of all drivers identified and then click *Cancel*. No changes are made to your system configuration, all you've done is gather a list of suspicious drivers, which you can then try to remove or disable manually. This is definitely your first course of action!
2) Click *Finish* to complete the wizard and restart your computer. Don't choose this option unless you're prepared to deal with the consequences, as explained in the remainder of this section. This is definitely your second choice!

If your computer stops with a blue screen when you next log on, you've identified a problem driver. The error message includes the name of the offending driver and an error code. For information about the error codes, see Microsoft Knowledge Base article 229903, "Partial List of Possible Error Codes With Driver Verifier." Although this article is specifically for Windows 2000, the information is valid for Windows XP and Windows Vista. To resolve the problem, boot into Safe Mode (press F8 during startup, which I assume you already know how to do) and disable or uninstall the problem driver. You'll then want to check with the device vendor to get a working driver that you can install.

To disable Driver Verifier so that it no longer performs verification checks at startup, run Driver Verifier Manager again and select *Delete Existing Settings* in the initial dialog box. Alternatively, at a command prompt, type *verifier /reset*. If you haven't yet solved the driver problem, of course, you'll be stopped at a BSOD, unable to disable Driver Verifier. In that case, boot into Safe Mode and then disable Driver Verifier.

You can configure Driver Verifier so that it checks only certain drivers. To do that, open Driver Verifier Manager, select *Create Standard Settings*, click *Next*, and select the last option, *Select Driver Names From A List*. This option lets you exempt a particular driver from Driver Verifier's scrutiny, such as one that Driver Verifier flags but you are certain is not the cause of your problem.

Automatic Logon

Do you have a computer that you want to set up to automatically log into Windows when it's turned on? I'm assuming that your computers are part of a domain, and as such will not autologon by default. When the computer is part of a workgroup and there's only one logon account, Vista will automatically logon as that user on startup.

You can manually set up the computer for automatic logon by making the following changes in the Registry.
- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Winlogon
- Set the value for DefaultUserName to your domain\loginname
- Set the value for DefaultPassword to your login password
- Set the value for AutoAdminLogon to 1

If you need to create any of the entries, they are all string values.

You'll need to restart the computer for the autologon to take effect.

Alternatively, you can download either of two command line utilities. They are both named *Autologon*, and must be run from an elevated Command Prompt window.
http://www.vista-io.com/1103, this was created by Mark Russinovich at Sysinternals (now part of Microsoft)
http://www.vista-io.com/1104, this created by the Windows Vista shell team

Or you can write a batch file that uses passed parameters to get the domain name, user name, and password, and uses reg.exe (see *Editing the Registry from the command line*) to add the information to the Registry. Or you can create a .reg file with the information.

You can bypass the automatic logon by holding down the Shift key as the system boots. This will cause Windows to stop at the login dialog.

You can also prevent users from bypassing the automatic logon. In the Registry, navigate to HKLM\Software\Microsoft\ WindowsNT\CurrentVersion\Winlogon. If the string value *IgnoreShiftOverride* doesn't exist, create it. Setting this value to *1* will ensure that your system always starts with its auto-logon account.

Microsoft Virtual PC

Microsoft Virtual PC runs on Windows Vista Business, Windows Vista Enterprise, Windows Vista Ultimate, Windows XP Professional, or Windows XP Tablet PC Edition.

Windows Vista Enterprise allows you to install up to four copies of the operating system in a virtual machine for a single user on a single device. Virtual PC is an extremely useful tool that lets you create separate virtual machines on your Windows desktop, each of which virtualizes the hardware of a complete physical computer. Use virtual machines to run an earlier version of Windows (as far back as Windows 98), or even DOS or OS/2.. You can run multiple operating systems at once on a single physical computer and switch between them with a mouse click.

You can use Virtual PC for legacy application support, training, or just for consolidating physical computers. Instead of having a dual-boot system or multi-boot system, use virtual PCs.

If you do documentation of installation or troubleshooting procedures, Virtual PC allows you to capture screens that you cannot capture any other way. For example, I captured the Vista installation screens for my *Windows Vista: The New Frontier* PowerPoint presentation using a Virtual PC (how else can you do that?!). I was also able to capture UAC dialogs, which you normally can't do because the desktop is running in a separate process (see the section on UAC for the details of what happens when a UAC dialog opens). And you can copy and paste between the real operating system and the virtual operating system(s).

To use virtual computers, your computer must have at least enough memory to cover the requirements of the host operating system and each virtual operating system that will run simultaneously. The more RAM you have available, the better.

And be sure to install Virtual Machine Additions (it comes with the virtual PC software) after setting up the virtual machine. VM Additions give improved performance of the guest operating system, integrated use of the mouse, optimized video drivers, and time synchronization.

Virtual PC isn't perfect, though. There are some components that don't share well between the host operating system and the virtual operating system(s). Like USB ports, network cards (you should really have one for each virtual machine), and the graphics card is a particular problem. My laptop has an ATI Mobility FireGL V5000 video card with 128MB of memory. Without VM Additions, the best display resolution I could get was 800x600 in the virtual machine. With VM Additions installed, I can get high resolution, up to 1600x1200. If I look at the adapter information, it shows VM Additions S3 Trio32/64 as the type, and 8MB of total available graphics memory. So you're not really getting a virtual copy of your computer with Virtual PC. Just FYI. Something to be aware of. But Virtual PC is something I recommend you look into. If you need to really have your entire computer configuration in multiple operating systems, then go with a multi-boot configuration.

You can download Microsoft Virtual PC at
http://www.microsoft.com/windows/downloads/virtualpc/default.mspx

Changing Boot Option Descriptors

If you decide to have a multi-boot computer instead of using virtual PCs, when you boot up you'll see a list of boot choices. By default, any earlier version of Windows will be named, logically, Earlier Version of Windows, and Vista will be named Windows Vista. But what if you need to have Windows 2000 installed, and Windows XP installed? They'll both have the same name in the boot menu.

In Windows XP, you would remove the read-only flag and edit the boot.ini file. Not so with Vista. The boot.ini file is no more (see the *Windows Vista's Boot Process* section of the handout for CM501-2, Windows Vista: The New Frontier, for more info).

On computers that are running only Windows Vista, the boot.ini text file is gone (not hidden) and any remnants of it on interim beta test builds are ignored. On computers with both earlier versions of Windows and with Windows Vista, the boot.ini file remains to support the older versions but it does not affect booting in Windows Vista.

So how do you change what you see when you have a multi-boot computer? Microsoft Vista includes a command line tool named bcdedit.exe (that stands for Boot Configuration Data Edit). For a complete list of BCD boot options, go to http://technet2.microsoft.com/WindowsVista/f/?en/library/08d64d13-4f45-4a05-bd86-c99211a93dd91033.mspx. BCDEdit.exe is, at best, not easy to use. It's entirely command line driven and the options are not exactly straightforward and easy to do. So what's a boy (or girl) to do? You download VistaBootPro from http://www.vistabootpro.org/index.php, that's what. This is a graphical editor for the BCD store and is the way to go if you need to edit the BCD.

Editing the Registry from the command line

All editions of Windows Vista include reg.exe (sometimes called the Console Registry Tool), a command-line tool that enables you to perform registry operations without using Registry Editor. reg.exe is also available in Windows XP.

By incorporating reg.exe commands in batch programs or scripts, you can automate registry activities, as well as take conditional actions based on the state of a local or remote registry, something you can't do using .reg files. For example, you can query a registry value and then edit that value (or take another action) if the data meets some criterion. Virtually the entire feature set of Registry Editor is available in reg.exe (one exception, the Export operation in reg.exe exports Unicode .reg files only).

And you can do at least one thing in reg.exe that's impossible to do in Registry Editor, change the data type of a key's default value.

For syntax information about reg.exe, open a Command Prompt window and type *reg /?*. As you'll see, the tool's basic syntax is

```
    reg operation [parameter list]
```

where operation is one of the 12 items listed below, and parameter list is one or more items (the name of a subkey or value, for example) pertinent to the specified operation. You can get additional syntax details about an operation by typing *reg operation /?*. For example, to learn more about how to use the Query operation, type *reg query /?*.

Here's a list of the available operations for reg.exe.

Add	Adds a key, value, or data item
Compare under	Compares one value with another or all values under a particular key with all values another key
Copy	Copies a value or key from one location in the registry to another
Delete	Deletes a key or value
Export	Exports a key as a Unicode .reg file
Flags	Displays or sets registry virtualization flags for subkeys of HKLM\Software
Import	Imports a .reg file (to the local registry only)
Load	Loads a hive file to a specified new key
Query	Returns the data associated with a specified value or with all values of a specified key
Restore	Loads a hive file into an existing key, overwriting that key
Save	Creates a hive file
Unload	Unloads a hive file

Some guidelines to note about reg.exe's syntax:

- reg.exe requires that you abbreviate the names of root keys. Use HKLM, for example, not HKEY_LOCAL_MACHINE.
- If a key or value name includes spaces, enclose it within quotation marks.
- If you're incorporating a reg.exe command in a batch program and that command includes an environment variable, enclose the variable name within two pairs of percent signs, not a single pair of percent signs. Otherwise, the command interpreter expands the variable name before passing it to reg.exe.

All reg.exe operations issue errorlevel values that can be tested in batch programs. For all operations except Compare, these values are 0 if the operation is successful and 1 if unsuccessful. Compare returns 0 if the operation is successful and all compared values are identical, 1 if the operation is unsuccessful, or 2 if the operation is successful and there are differences in the compared values.

Pardon me while I drift off into a soliloquy here about using errorlevels in batch files. I won't cover this material in class, but this is really good stuff if you do this kind of thing.

A program can return an errorlevel when it runs. Usually, errorlevel 0 means whatever it did was successful. Other errorlevels can mean whatever the programmer wants them to mean. Depending what you're going to do, based upon the errorlever, will determine if you check the errorlevel from lowest to highest or highest to lowest. If you're going to set a variable, you'll want to check from lowest to highest. If you're going to use a goto statement, you'll want to check from highest to lowest. The "if errorlevel *number*" check is really asking is the errorlevel number *equal or greater* than the *number* it's checking against. Allow me to demonstrate with the following batch file.

```
@echo off
:::  Let's assume that we're calling a program named ERRLVL.EXE, it returns an errorlevel from 1 to 4
ERRLVL

:::  In this section, we'll set a variable, myErr, to the errorlevel number.
:::  the variable will keep getting set as long as the errorlevel is equal or greater
:::  than the specified errorlevel number.
if errorlevel 1 set myErr =1
:::  if the errorlevel is 1, myErr is set to 1, all subsequent checks fail and don't change myErr
if errorlevel 2 set myErr =2
:::  if the errorlevel is 2, the first check (if errorlevel 1) is true and myErr is first set to 1
:::  then the second check (if errorlevel 2) is also true and myErr is set to 2
:::  all subsequent checks fail and don't change myErr
if errorlevel 3 set myErr =3
:::  continue extrapolating from the previous check
if errorlevel 4 set myErr =4
:::  and the same for this

:::  In this section, we use a goto command to jump to another location in the file.
:::  We need to check from highest to lowest since we are not continuing checking
:::  after the first match is found.
if errorlevel 4 goto myErr4
:::  if the errorlevel is 4, the program jumps to myErr4, bypassing all other checks
:::  any number smaller than 4 fails (not => 4) and doesn't do the goto command
if errorlevel 3 goto myErr3
:::  if the errorlevel is 3, the program jumps to myErr3, bypassing all other checks
:::  any number smaller than 3 fails (not => 3) and doesn't do the goto command
if errorlevel 2 goto myErr2
:::  if the errorlevel is 2, the program jumps to myErr2, bypassing all other checks
:::  any number smaller than 2 fails (not => 2) and doesn't do the goto command
if errorlevel 1 goto myErr1

: myErr1
:::  your code here
goto end

: myErr2
:::  your code here
goto end

:myErr3
:::  your code here
goto end
```

```
:myErr4
:::  your code here
goto end

:end
```

Okay. End of soliloquy. Here's an example showing adding a new value and checking the errorlevel in a batch file.

```
@echo off
reg ADD HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion /f /v Bogus /t REG_SZ /d "My Test"
if errorlevel = 1 goto failed

:::  we don't need to check for errorlevel 0
:::  since there are only two choices, if it's not 1 it must be 0
:::  so we do whatever we want to do if it doesn't fail
echo Registry add successful!
pause
goto end

:failed
echo Registry add failed!
pause

:end
```

What if we had written the batch file like this?

```
@echo off
reg ADD HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion /f /v Bogus :/t REG_SZ /d "My Test"
if errorlevel = 0 goto passed

:failed
echo Registry add failed!
pause
goto end

:passed
echo It reports it passed!
pause

:end
```

The add fails, but since errorlevel 1 is higher than 0, it processes the errorlevel check as if it's 0.

Here's a situation that I had recently. We were moving our license server to a new server. I use the IP address of the server instead of the server name for the ADSKFLEX_LICENSE_FILE system variable because I have users that access this from all over the world and you can't use a fully qualified name for the variable. By using an IP address, I know that they'll be able to reach the server from wherever they are. Because of its physical location, the new server was going to have a different IP address than the previous one. I had to update everyone's system variable. Every computer has the key HKLM\SYSTEM\CurrentControlSet. Some computers have HKLM\SYSTEM\ControlSet001 and HKLM\SYSTEM\ControlSet003, some have HKLM\SYSTEM\ControlSet002 and HKLM\SYSTEM\ControlSet003, etc. I created a batch file that used reg.exe to query for the existence of the key, and then, depending on the result of the query, set the variable for the control set or not. If the errorlevel was 1, the key wasn't found and no action was taken, if it was 0, the key was found and the value was set.

A few things I learned while creating the batch file.

- If the key name, value, or data contains any space characters, enclose it in double quotes.
- reg.exe's *Add* method has an option, /f, that forces overwriting the existing registry entry without prompt. Without this, the user will be prompted to confirm every overwrite. The syntax given by reg.exe's command line help is

```
REG ADD KeyName [/v ValueName | /ve] [/t Type] [/s Separator] [/d Data] [/f]
```

What I found is that the /f didn't do its job when it was at the end of the line. It did work when it was the first argument after the KeyName.

- If you're like me, I always start by exporting the key that I want to modify. This gives me the framework for the .reg file, or I can use the key information for reg.exe, etc. One of the values I wanted to edit was an expandable string value (REG_EXPAND_SZ) named *LastUsedSource*. In regedit, the original data looked like this.

n;1;\\139.104.241.28\deploy\ShowRide\AutoCAD\AutoCAD2007\AdminImage

When I exported it from regedit.exe, the .reg file looked like this.

```
Windows Registry Editor Version 5.00

[HKEY_CLASSES_ROOT\Installer\Products\7D2F387510059040002000060BECB6AB\SourceList]
"LastUsedSource"=hex(2):6e,00,3b,00,31,00,3b,00,5c,00,5c,00,31,00,33,00,39,00,\
  2e,00,31,00,30,00,34,00,2e,00,32,00,34,00,31,00,2e,00,32,00,38,00,5c,00,64,\
  00,65,00,70,00,6c,00,6f,00,79,00,5c,00,53,00,68,00,6f,00,77,00,52,00,69,00,\
  64,00,65,00,5c,00,41,00,75,00,74,00,6f,00,43,00,41,00,44,00,5c,00,41,00,75,\
  00,74,00,6f,00,43,00,41,00,44,00,32,00,30,00,30,00,37,00,5c,00,41,00,64,00,\
  6d,00,69,00,6e,00,49,00,6d,00,61,00,67,00,65,00,22,00,00,00
```

What I found was even though the .reg file shows the data in hex, when using reg.exe *Add* the data is passed as a regular text string.

Using .reg Files

You can also, of course, make Registry changes using .reg files. This is often the way to go when you're making a lot of changes and you don't need to check an entries current value or if it exists.

The first thing I always do is export the key I want to make changes to. As mentioned above, this gives me the framework needed for the .reg file. After that I can make the necessary changes and I'm ready to go.

Have you even wanted to delete an existing value in the Registry using a .reg file? No worries. Make the value's data a hyphen for that item in the .reg file. For example, let's say there's a value named "Bogus" in the [HKEY_CURRENT_USER\Software\Autodesk] key that you want to delete. The .reg file would look like this.

```
Windows Registry Editor Version 5.00

[HKEY_CURRENT_USER\Software\Autodesk]
"Bogus"=-
```

Environment Variables

Command Prompt gets its environment variables from three sources:

- Any variables set in your Autoexec.bat file
- System variables, as recorded in HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Environment
- User variables, as recorded in HKCU\Environment

When you log on, Windows Vista scans the Autoexec.bat file in the root folder of your boot drive for environment variables initialized with Set statements. If you don't want Windows Vista to scan your Autoexec.bat file for Set statements, open a registry editor and navigate to HKCU\Software\Microsoft\Windows NT\CurrentVersion\Winlogon. Then change the data associated with the *ParseAutoexec* value from *1* to *0*.

System and user variables are both stored in the registry, but you don't need to launch a registry editor to change them. Open *System* in *Control Panel* instead. Click *Advanced System Settings* to get to the *System Properties* dialog box. Click the *Advanced* tab and then the *Environment Variables* button. The top half of the dialog shows you User variables and the bottom half shows you System variables. Variables added to the User variables are only for the current user. Variables added to the System variables will show up for all users. It follows that a standard user can only add or change User variables. And of course adding or changing System variables requires administrator credentials.

Changes to environment variables made via Control Panel affect your next and subsequent Command Prompt sessions (not the current ones, of course). Changes made via Autoexec.bat or HKCU\Environment are not effective until the next time you log on. In case of conflicting assignments, user variables take precedence over system variables, which take precedence over variables declared in Autoexec.bat. The *Path* variable, however, is cumulative. That is, changes made in any venue are appended to any changes made elsewhere.

Within a given Command Prompt session, you can change environment variables by means of Set statements. Such statements affect only the current session and any applications (including additional Command Prompt sessions) spawned from the current session.

The Autoexec.nt file has no effect on the Command Prompt environment. Autoexec.nt affects MS-DOS-based applications only. Command Prompt, although it is the MS-DOS command interpreter, is itself a Windows application, cmd.exe.

You can access User and System variables at the command line or in a batch file by enclosing them in percent signs. For example, if you want to copy a file into the user's Documents folder, you can use the %USERPROFILE% variable.

```
    Copy c:\bogus.txt %USERPROFILE%\Documents
```

To see all of the current variables, go to a Command Prompt window and enter SET.

How Will Vista Change Your Life?

Some of what makes Vista better is what makes Vista worse. Everyone wants Windows to be more secure and less susceptible to virus and malicious attacks, but the price for this is stricter control over what gets installed and what's allowed to run.

As a CAD manager, anything that helps keep your users' computers safe from viruses and intruder attacks, is a good thing. Vista's UAC is a key element in its defense these attacks, but it'll be very tempting to turn this off. As annoying as the UAC elevation and consent dialogs can be, they're the first line of defense for your computer. In theory, once the system is set up, the dialogs should show up must less frequently, and when they do, you may be glad they did.

If you lock down your systems, Vista makes being a non-admin user more feasible than ever before.

Vista comes with thousands of drivers, but there are still many hardware devices that aren't supported. Before moving to Vista make sure your critical hardware is supported or the manufacturer has Vista drivers. And try to only use drivers that are WHQL-signed drivers. Don't try to use a driver intended for an earlier version of Windows. While it is possible to use unsigned and un-certified drivers, you should do so only if you are 100% sure of their reliability and safety. Stick with WHQL-signed drivers and your computers should be more stable than ever before. And that's definitely a good thing. Remember, good drivers are the key to happiness!

You may want to use Vista only on new systems you purchase. Unless you special order your computers with XP, they're going to come with Vista and hopefully were fully tested. If you're thinking about putting Vista on your existing computers, think of it this way, the older the computer, the more likely you're going to run into problems.

And you'll find that there are lots of applications that will need to be upgraded to be able to use them in Vista. Take AutoCAD. The only version that works in Vista is 2008. Before moving to Vista, make sure you can run all the software you need to run.

Microsoft is working on Service Pack 1 for Vista, which will be released in 2008, and will contain hundreds of fixes and improvements. If you're not in a hurry, waiting for SP1 may be the way to go for you.