Command Shell and Filesystems: Section 1 Transcript

Introduction

Welcome to the Command Shell and File Systems module. The command shell is the user interface for accessing the operating system's services without the use of a graphic user interface (GUI). The Windows file system structure is a roadmap of how Microsoft tracks the location folders and files. What's the connection? Most administrators will only use a command line interface to view a user's file system. In this module, we will learn the basics of both.

Throughout this module, you'll be presented with opportunities to assess and apply what you've learned.

At the end of this module, you will be able to:

- Demonstrate basic familiarization with the Command-Line Interface (CLI),
- Identify environmental variables,
- Demonstrate basic scripting concepts,
- Identify the main components of the NTFS File System,
- Utilize Windows Management Instrumentation Command-Line (WMIC) commands to obtain and evaluate responses, and
- Utilize PsExec commands to obtain and evaluate responses.

Bypass Exam Introduction

If you are already familiar with the subject matter presented in this module, you can choose to take a Bypass Exam to skip this module.

The Bypass Exam option provides a single opportunity to successfully demonstrate your competence with the material presented within the module. If you pass, you'll receive credit for completing the module, unlocking the content within, and you will be free to proceed to the next module. If you do not pass, you will need to successfully complete the module, including all exercises and the Module Exam, to receive credit.

Click the Next Section button to continue.

Command Shell and Filesystems: Section 2 Transcript

Windows Command Shell

The Windows Command Shell uses a Command-Line Interface (CLI), which is a very simple, low overhead interface with character-based applications and utilities. It provides direct communication between the user and the operating system. The command shell can be used locally or remotely for administration, monitoring, or troubleshooting.

There are different command shells, and we will discuss only a few: cmd.exe, Windows PowerShell, and Windows Management Instrumentation Command-Line (WMIC).

CMD.exe

Based from the legacy shell, command.com, CMD.exe is the command shell used with Windows XP and newer operating systems. CMD.exe loads applications, directs the flow of information between applications, and translates user input into a form that the operating system understands.

The executable (CMD.exe) is located in the %systemroot% directory as follows:

- 32-bit shell on 32-bit system, located in %systemroot%\system32,
- 32-bit shell on 64-bit system, located in %systemroot%\syswow64, and
- 64-bit shell on 64-bit system, located in %systemroot%\system32.

On a typical Windows installation, the default value for %systemroot% is c:\Windows. %systemroot% is an environmental variable which we will discuss later in the module.

More features of CMD.exe include the following:

- Operates in batch mode,
- Supports long file names,
- Has a built-in buffer to recall past commands,
- Has two categories of native commands built in by Microsoft: internal commands, which exist in the shell and include commands such as copy, move, dir, set, date; and standard external commands, which has its own executable file normally found in the %systemroot%\system32 directory, and
- Uses Help or /? to show available commands.

Windows PowerShell

The Windows PowerShell is a full featured command shell that uses "cmdlets" (built-in commands), built-in programming features, and standard command-line utilities. Developed on the .NET Framework, it allows administrators to manage servers and workstations in the enterprise from the command line. Cmdlets are instances of .NET Framework classes; they are not stand-alone

executables. Cmdlets are named using verb-noun pairs in which the verb identifies the action performed and the noun identifies the resource on which the action is performed. For example, the cmdlet "get-command" is used to get all the cmdlets that are registered in the command shell. A complete list of cmdlets is available from the Windows PowerShell prompt by typing help *-*. Documentation on a specific cmdlet is available by help followed by the cmdlet name, such as "help get-command".

Windows PowerShell is included as part of the operating system starting with Windows 7 and Windows Server Pack 2008 R2. It is also available for download on Windows XP and later versions.

Windows Management Instrumentation Command-Line

Another tool primarily for system administrators, the Windows Management Instrumentation Command-Line, or WMIC, is a command-line and scripting interface that simplifies the use of Windows Management Instrumentation (WMI) and system management through WMI. The WMIC shell prompt opens to root\cli and enables an administrator to enter commands to the Windows Management Interface without needing to understand the native query language of WMI. It is also compatible with existing shells and utility commands. WMIC is supported by Windows XP or Server 2003 and newer systems.

Command Shell Operations

The command shells can operate in interactive mode or in batch mode. During interactive mode, identified by a blinking cursor, the user enters commands manually. In batch mode, commands are read and executed one by one, typically from a script file. There are many function keys for the Windows Command Shells. You can view the list of common commands by typing help at the command prompt and then press Enter. In fact, there are just too many commands to show, but they are listed in the reference materials for this module.

It is simply important to know that you can customize the appearance and function of the command session by increasing the command history buffer size, changing the font size and color, turning the echo on and off and changing the prompt. You can also view the command history and add macro functionality. You can activate auto-complete for the current command session or activate that command permanently.

In the following exercise, you will practice using some of these key commands.

Exercise Introduction

It is time for an Exercise. In order to successfully complete the Exercise, you are expected to use your notes and any available resources presented throughout the course, in addition to conducting your own Internet research.

Command Shell and Filesystems: Section 3 Transcript

Environmental Variables

Environmental variables are strings that are used by Windows to communicate information about the location of system files, folders, and programs as they pertain to the currently logged on user. Systems can be configured differently, either by default or by individual choice. Environmental variables enable universal programming codes to be compatible to the variety of potential system configurations. This is possible because the values of these variables are not hardcoded to specific locations, but are allowed to be changeable based on the system configuration. System environmental variables, such as the path to the Windows files, are defined by Windows at installation and they apply to all computer users. Changes to the system environment are written to the registry, usually require a restart to become effective, and can only be modified by an administrator. The percent symbol identifies the variable as an environmental variable.

Environmental variables are contained in HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Environment. These environmental variables are pre-defined. Here are a few commands:

- set displays all current environmental variables,
- setx creates or modifies environmental variables in the user or system environment, without requiring programming or scripting,
- echo %systemroot% shows where Windows files are located, and
- echo %homepath% shows the \Documents and Settings\"user profile" or \Users\"user profile" in Windows Vista and later systems.

Exercise Introduction

It is time for an Exercise. In order to successfully complete the Exercise, you are expected to use your notes and any available resources presented throughout the course, in addition to conducting your own Internet research.

Command Shell and Filesystems: Section 4 Transcript

Commands and Syntax—Redirects

Different types of commands require the use of special characters. Let's talk about these types of commands and the syntax to implement them.

Redirects are used to direct input and output using Windows commands. Using the "less than" or "greater than" special characters, one can send the output of a command to a file or device; for example, command1 outputs to a file or device. A redirect may often involve receiving input or directing output to or from a given path; for example, command writes its output to the specified file. Note that input is represented by the "less than" symbol, while output is denoted by the "greater than" symbol. Information already in the file is overwritten, and, if the file does not exist, it will be created.

Double "greater than" characters denote that information should be appended to a file instead of overwritten. For example, command can accept input from a file and then append its output to the end of the specified file.

Command and Syntax—Pipes

At the Windows command line, we use "pipes" to send the output of the first command as the input for the second command. Piping allows for multiple redirects in which a number of commands are strung together. For example, command1 | command2 | command3 denotes that the output of command 1 should be the input for command 2 and the output for command 2 should be the input for command 3.

Commands and Syntax—Chaining and Grouping

The ampersand separates multiple commands on one command line. Cmd.exe runs the first command, and then the second command. This is called chaining and looks like this: (command1 & command2).

A double ampersand runs the command following the symbol only if the command preceding the symbol is successful. Cmd.exe runs the first command, and then runs the second command only if the first command completed successfully.

Conversely, a double pipe symbol runs the command following the symbol only if the command preceding the symbol fails. Cmd.exe runs the first command, and then runs the second command only if the first command did not complete successfully, in other words, receives an error code greater than zero.

Parentheses group a set of commands for conditional execution based on success such as

((command1 & command2) && (command3)) or based on failure ((command1 & command2) || (command3)).

Escape Characters

The escape character can be used for a few reasons.

- It can be added before a command symbol to allow the command symbol to be treated as ordinary text, or
- It can be used to make long commands more readable by splitting them into multiple lines and escaping the Carriage Return + Line Feed (CR/LF) at the end of the line.

For example, the command `echo <escape>` will result in an error whereas the command `echo ^<escape^>` will print <escape> to the screen. One thing to note: a stray space at the end of the line (after the escape character) will break the command, which is hard to detect unless you have a text editor that displays spaces and tab characters.

Wildcard Characters

A wildcard is a symbol used to replace or represent zero or more characters. Wildcards or wild characters are either an asterisk (*), which represents zero or more characters or question mark (?), which represents zero or one character.

Exercise Introduction

It is time for an Exercise. In order to successfully complete the Exercise, you are expected to use your notes and any available resources presented throughout the course, in addition to conducting your own Internet research.

Command Shell and Filesystems: Section 5 Transcript

New Technology File System (NTFS)

Let's shift gears and discuss the New Technology File System (NTFS). The key NTFS components include the sector, cluster, logical cluster number (LCN), virtual cluster number (VCN), the file system format, and the metadata. Click the key NTFS components to learn about them.

New Technology File System (NTFS)

Sector: Sectors are hardware-addressable blocks on a storage medium. They are typically 512 bytes and are defined in the Partition Boot Sector.

Cluster: Clusters are the addressable blocks that many file system formats use. They are always a multiple of the sector size, typically 4096 bytes, (or 8 sectors), in Windows and are defined in the Partition Boot Sector.

Logical Cluster Number (LCN): The Logical Cluster Number is the cluster address relative to the storage media.

Virtual Cluster Number (VCN): The Virtual Cluster Number is the cluster address relative to the start of the file.

File System Format: The file system format defines the way that file data is stored on storage media.

Metadata: Metadata is the data stored on the volume that defines how the file system is implemented within the volume.

NTFS Advanced Features

NTFS has many advanced features that are noteworthy. We won't discuss all of them; however, you should review the documents located in Resources. Click the significant advanced features to learn about them. Note that we will discuss an important advanced feature, robust file permissions, in a later module.

Multiple Data Streams: With multiple data streams, each unit of information associated with a file is implemented as a file attribute. Each attribute consists of a single stream. This implementation makes it easy to add more attributes.

Unicode-Based Names: Unicode-based naming is a 16-bit character coding scheme that allows each character in each of the world languages to be uniquely represented. This makes it easy to move data from one country to another. Each complete filename path can be up to 255 characters in length.

Dynamic Bad-Cluster Remapping: This feature is enabled when a bad cluster is identified and the data is remapped to a new cluster. The process is as follows: the NTFS sends a warning that the cluster is bad, and then a good copy of the sector's data is retrieved. A new cluster is allocated, and the data is copied to the new cluster. The bad cluster is flagged and no longer used.

Per User Volume Quotas: This feature allows for per-user quota specification of quota enforcement, which is useful for usage tracking and tracking when a user reaches warning and threshold limits.

Encryption: This process is transparent to the user and is performed using the Encrypting File System (EFS). Encryption is accomplished using private/public key pairs.

Defragmentation: This feature is enabled when files occupy non-contiguous space on the hard disk. The NTFS has tools to make the files contiguous.

Read-Only Support: This feature was introduced with Windows XP and allows mounted volumes to be read-only.

Master File Table (MFT)

The Master File Table (MFT) is the heart of the NTFS. It contains information about all files and directories on a machine. Every file and directory has at least one entry in the MFT, also referred to as a file record. Entries are likely 1KB in size and are considered to be very simple in structure. The MFT entry contains attributes that define and describe the MFT and its sub components. We will talk about these attributes later in this module.

Metadata Files

The file system's administrative data are stored in files on the volume. These files are called metadata files. Microsoft reserves the first 16 entries for its standard system metadata files. Entries 17 through 23 are sometimes used as overflow when the reserved entries are not sufficient. The first entry that is allocated to a user file or directory is entry 24. We'll detail the first 12 metadata File entries. Click the metadata file entries to learn more.

$MFT (Entry 0): This entry is for the MFT itself.

$MFTMirr (Entry 1): Contains a backup of the first entries of the MFT

$LogFile (Entry 2): Contains the journal that records the metadata transactions

$Volume (Entry 3): Contains volume information such as label, identifier, and version

$AttrDef (Entry 4): Contains the attribute information, such as identifier values, names, and sizes

\ **(Entry 5)**: Contains the root directory of the file system

$Bitmap (Entry 6): Contains the allocation status of each cluster in the file system. If the bit is set to 1, the cluster is allocated. If set to 0, it is not.

$Boot (Entry 7): Contains the boot sector and boot code for the file system; provides basic information such as the size of each cluster, number of sectors in the file system, starting cluster address of the MFT, the size of each MFT entry, and the serial number for the file system.

$BadClus (Entry 8): Contains the clusters that have bad sectors

$Secure (Entry 9): Contains information about the security and access control for the files (Windows 2000 and XP versions only)

$Upcase (Entry 10): Contains the uppercase version of every Unicode character

$Extend (Entry 11): Contains files for optional extensions (Note: Microsoft does not typically place the files in this directory into the reserved MFT entries)

MFT Entry Layout

The NTFS file system does not have strict layout requirements, but Windows uses some general guidelines when formatting the file system. The MFT Zone is a collection of clusters not used to store file or directory content. This zone is created to be as small as possible and only expands when it is needed for additional entries. It can also become easily fragmented.

The on-disk allocation usually varies depending on the Windows operating system. The exception is the first cluster, which is allocated to the $Boot file on all Windows versions.

MFT Entry Standard Attributes

Everything in a file is stored as an attribute, even the data itself. Attributes can be standard or well-known. Standard attributes are present in every file, whereas well-known attributes may or may not exist with every file. Standard attributes include $STANDARD_INFORMATION, $FILE_NAME, and $DATA. Let's discuss each of these standard attributes in more detail.

$STANDARD_INFORMATION

$STANDARD_INFORMATION has a type ID of 16 and is 72 bytes in Windows XP. It is always the first attribute in the MFT entry because it has the lowest Type ID. It contains ownership, security, quota, and time stamp information. Its four time stamp values include creation time, which is the time that the file was created; modified time, which is the time that the contents of the $DATA or $INDEX attribute were last modified (The $INDEX attribute provides information describing how the file is stored in the nodes of a tree.); the MFT modified time, which is the time that the metadata of the file was last modified; and the accessed time, which is the time that the contents of the file were last accessed.

$FILE_NAME

The $FILE_NAME attribute has a Type ID of 48 and is 66 bytes plus the file name. It occurs at least once per file MFT entry. It contains the same temporal time stamps as $STANDARD_INFORMATION, but is not typically updated. The time stamps usually correspond to

when the file was created, moved, or renamed.

$DATA

The $DATA attribute has a Type ID of 128 and a variable size. Every file has a $DATA attribute which contains the file content. If the content is 700 bytes or less in size, it is resident. If the content is greater than 700 bytes in size, it becomes non-resident, and its content is saved in external clusters.

A file may contain more than one $DATA attribute. These additional $DATA attributes are referred to as Alternate Data Streams. The original $DATA attribute does not have a name associated with it, but additional $DATA attributes must be named.

Opportunistic Locks

The Microsoft online library defines an opportunistic lock, also known as an oplock, as "a lock placed by a client on a file residing on a server." In most cases, a client requests an opportunistic lock so it can cache data locally, thus reducing network traffic and improving apparent response time.

Microsoft also states that "opportunistic locks coordinate data caching and coherency between clients and servers and among multiple clients. Data that is coherent is data that is the same across the network. In other words, if data is coherent, data on the server and all the clients is synchronized."

"Opportunistic locks are not commands by the client to the server. They are requests from the client to the server. From the point of view of the client, they are opportunistic. In other words, the server grants such locks whenever other factors make the locks possible."

For more information on oplocks, please see the Resources for this module.

Windows File Location

The last thing we will discuss are the different Windows File Locations. Remember that you will most likely access these files from the command-line interface, so it is important to recognize what usually is contained in these locations. Also, when surveying the machine, you can begin to learn a lot about the configuration and the user. For example, what the machine is used for by identifying the software files in Program Files, seeing common users on the network, learning where specific users keep their files, or if the computer is used by one person or shared by more. Click each file location to learn more.

Program Files - contains installed programs

Users - contains user profiles (Windows Vista +)

Documents and Settings - contains user profiles (user hives) (Windows XP)

Program Files (x86) - contains installed 32-bit programs on 64-bit architectures

\Windows\Prefetch - on XP and up; stores pathname and mapping of last time application was run. For example, Acrobat reader will load all dlls the first time it is run. It will be pre-fetched, if possible for the second time.

\Windows\SoftwareDistribution— contains the Windows downloaded updates

\Windows\$NT*— Location of replaced files after a system update or service pack so a user can roll back to previous versions

\Windows\system32\config - contains registry files

\Windows\system32\dllcache - contains backed up critical system files

\Windows\system32\drivers\etc - contains system (tcp/ip) network files

\Windows\system32\Repair— contains the backup, or off-line copy, of registry files

Exercise Introduction

It is time for an Exercise. In order to successfully complete the Exercise, you are expected to use your notes and any available resources presented throughout the course, in addition to conducting your own Internet research.

Command Shell and Filesystems: Section 6 Transcript

Windows Management Instrumentation Introduction

Windows Management Instrumentation, or WMI, defines a set of environment-independent specifications which allow management information to be shared between management applications. Installed on all computers since Windows Millennium Edition, WMI is the Microsoft implementation of Web Based Enterprise Management, or WBEM, which is built on the Common Information Model, or CIM, a computer industry standard.

The Windows Management Instrumentation Command-Line (WMIC) tool is a command-line and scripting interface that simplifies the use of WMI and systems managed through WMI.

WMIC can be used either by typing an entire command at the Command Shell prompt, like this one (C:> wmic useraccount get), or by dropping into the WMIC shell. To enter the WMIC shell, enter the command 'wmic' (C:\> wmic) at the Command Shell prompt.

You will enter the WMIC shell and receive the prompt seen here (wmic:root\cli>), and then you can enter a command (useraccount get).

This and more information on WMIC can be found in the Resources for this module.

WMIC Commands

WMIC is based on aliases. Aliases make the primary data provided by WMI available without having to understand WMI-specific concepts. WMI data and many WMI features are also accessible through WMI without aliases using the CLASS and PATH commands. You can list the available aliases by using WMIC /? help.

Within its library, Microsoft states that WMIC commands are composed of ALIASes, VERBs, and SWITCHes. Click each to learn more.

Alias: An alias is a friendly renaming of a class, property, or method that makes WMI easier to use and read. You can determine what aliases are available for WMIC through the /? command. You can also determine the aliases for a specific class using the <className> /? command.

Verb: To use verbs in WMIC, enter the alias name followed by the verb. If an alias does not support a verb, you receive the message "provider is not capable of the attempted operation."

Switch: A switch is a WMIC option you can set globally or optionally.

WMIC Commands

For example, the processes running on the current system are available from the PROCESS alias.

To view all of the processes that are currently running on the computer, type PROCESS in the WMIC utility. To list a specific process, type a command such as PROCESS WHERE (Description="explorer.exe"). To receive specific properties for the processes, type a command such as PROCESS GET Name, Handle, PageFaults.

To execute a command on a remote system, the /NODE, /USER, and /PASSWORD switches can be used.

Without using aliases, you can use the same options with the PATH command, as you see here (PATH Win32_Process GET Name, Handle, PageFaults).

However, you must determine the name of the class from other sources. To do the equivalent of the 'alias where' clause, you must use this command (PATH Win32_Process.Description="explorer.exe").

Exercise Introduction

It is time for an Exercise. In order to successfully complete the Exercise, you are expected to use your notes and any available resources presented throughout the course, in addition to conducting your own Internet research.

Command Shell and Filesystems: Section 7 Transcript

PsExec Intro

1/4

Mark Russinovich has written some helpful information about PsExec. He explains that utilities like Telnet and remote control programs let you execute programs on remote systems, but they can be a pain to set up and require that you install client software on the remote systems that you wish to access. PsExec is a light-weight telnet-replacement that lets you execute processes on other systems, complete with full interactivity for console applications, without having to manually install client software. PsExec's most powerful uses include launching interactive command-prompts on remote systems and remote-enabling tools like 'ipconfig' that otherwise do not have the ability to show information about remote systems.

You'll find more information from Mark Russinovich in the Resources for this module.

PsExec Syntax Options

2/4

The most basic usage of the command is seen here (psexec \\remote-computer -u username -p password command-to-be-executed). Note that the username used must have the appropriate privileges to run the command to be executed and must be in the form of (domain or host)\username. Also, the -p option can be left off causing the user to be prompted and the response to be hidden. To review the other syntax options, see the Resources for this section.

PsExec Commands

3/4

To display the IP interface configuration of a remote system, you can run this command (psexec \\remote.widget.com -u administrator -p LetM3In ipconfig /all).

It is also possible to open an interactive shell so that you can run multiple commands without continually opening up the connection (psexec \\remote.widget.com -u administrator -p LetM3In cmd).

Some options for the psexec command will allow you to open programs on the remote system and have them interact directly with the remote desktop.

One important option to note is '-accepteula'. This option suppresses the dialog box asking to accept the End User License Agreement. This comes in handy when you are in a situation when a pop-up dialog box is not possible.

Exercise Introduction

4/4

It is time for an Exercise. In order to successfully complete the Exercise, you are expected to use your notes and any available resources presented throughout the course, in addition to conducting

your own Internet research.

Command Shell and Filesystems: Section 8 Transcript

Summary

You have completed the Command Shell and File Systems module.

You should now be able to:

- Demonstrate basic familiarization with the Command-Line Interface (CLI),
- Identify environmental variables,
- Demonstrate basic scripting concepts,
- Identify the main components of the NTFS File System
- Utilize Windows Management Instrumentation Command-Line (WMIC) commands to obtain and evaluate responses, and
- Utilize PsExec commands to obtain and evaluate responses.

To receive credit and advance to the next module, you must achieve a passing score on the Module Exam. Click the Next Section button to begin the Module Exam.