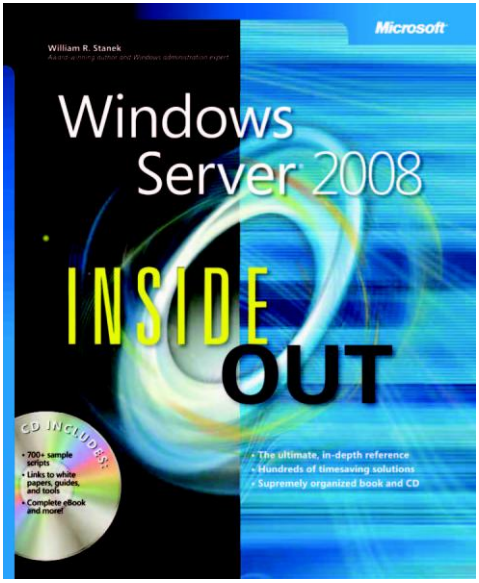


Windows Server® 2008 Inside Out

William R. Stanek

To learn more about this book, visit Microsoft Learning at
http://www.microsoft.com/MSPress/books/11448.aspx

9780735624382

Microsoft® Press

Table of Contents

Part 3: Managing Windows Server 2008 Storage and File Systems

Part 4: Managing Windows Server 2008 Networking and Print Services

Part 5: Managing Active Directory and Security

CHAPTER 9

Managing the Registry

Everyone who accesses a computer, whether in a workgroup or on a domain, at one time or another has worked with the Windows Registry whether the person realizes it or not. Whenever you log on, your user preferences are read from the Registry. Whenever you make changes to the system configuration, install applications or hardware, or make other changes to the working environment, the changes are stored in the Registry. Whenever you uninstall hardware, applications, or system components, these changes are recorded in the Registry as well.

The Registry is the central repository for configuration information in Microsoft Windows. Applications, system components, device drivers, and the operating system kernel all use the Registry to store settings and to obtain information about user preferences, system hardware configuration, and system defaults. The Registry also stores information about security settings, user rights, local accounts, and much more. Unlike Microsoft Windows NT, in domains, later versions of Windows do not store information about domain accounts or network objects in the Registry; these settings are managed by Active Directory Domain Services as discussed in Part 5, "Managing Active Directory and Security."

With so much information being read from and written to the Registry, it is not only important for administrators to understand its structures and uses, it is essential. You should know the types of data the Registry works with, what type of data is stored where, and how to make changes if necessary. This is important because often when you must fine-tune system configuration or correct errors to stabilize systems, you may be instructed to access the Registry and make such and such a change. Generally, the instructions assume you know what you're doing. Unfortunately, if you attempt such a change and really don't know what you're doing, you could make it so the system won't boot at all. So, with this in mind, let's look at how the Registry works and how you can work with it.

Introducing the Registry

The Registry is written as a binary database with the information organized in a hierarchy. This hierarchy has a structure much like that used by a file system and is an inverted tree with the root at the top of the tree. Any time the Windows operating system must obtain system default values or information about your preferences, it obtains this information from the Registry. Any time you install programs or make changes in Control Panel, these changes usually are written to the Registry.

> **Note**
>
> I say "usually" because in Windows domains some configuration information is written to Active Directory directory service. For example, beginning with Microsoft Windows 2000, information about user accounts and network objects is stored in Active Directory. In addition, when you promote a member server to a domain controller, key Registry settings that apply to the server, such as the default configuration values, are transferred to Active Directory and thereafter managed through Active Directory. If you were later to demote the domain controller, the original Registry settings would not be restored either. Instead, the default settings are restored as they would appear on a newly installed server.

The Registry's importance is that it stores most of a system's state. If you make preference and settings changes to a system, these changes are stored in the Registry. If a system dies and cannot be recovered, you don't have to install a new system and then configure it to look like the old one. You could instead install Windows Server 2008 and then restore a backup of the failed system's Registry. This restores all the preferences and settings of the failed system on the new system.

Although it's great that the Registry can store settings that you've made, you might be wondering what else the Registry is good for. Well, in addition to storing settings that you've made, the Registry stores settings that the operating system makes as well. For example, the operating system kernel stores information needed by device drivers in the Registry, including the driver initialization parameters, which allows the device drivers to configure themselves to work with the system's hardware.

Many other system components make use of the Registry as well. When you install Windows Server 2008, the setup choices you make are used to build the initial Registry database. Setup modifies the Registry whenever you add or remove hardware from a system. Similarly, application setup programs modify the Registry to store the application installation settings and to determine whether components of the application are already installed. Then, when you run applications, the applications make use of the Registry settings.

Unlike previous releases of Windows, however, Windows Vista and Windows Server 2008 don't always store application settings directly in the Registry and may in fact read some settings from a user's profile. This behavior is new and occurs because of User Account Control (UAC). Of the many features UAC implements, there are two key features that change the way Windows installs and runs applications: application run levels and application virtualization.

To support run levels and virtualization, all applications that run on Windows Vista and Windows Server 2008 have a security token. The security token reflects the level of privileges required to run the application. Applications written for Windows Vista and Windows Server 2008 can have either an *administrator* token or a *standard user* token. Applications with administrator tokens require elevated privileges to run and perform core tasks. After it's started in elevated mode, an application with an administrator token can perform tasks that require administrator privileges and can also write to system locations of the Registry and the file system.

On the other hand, applications with standard user tokens do not require elevated privileges to run and perform core tasks. After it's started in standard user mode, an application with a standard user token must request elevated privileges to perform administration tasks. For all other tasks, the application should not run using elevated privileges. Further, the application should write data only to nonsystem locations of the Registry and the file system.

Standard user applications run in a special compatibility mode and use file system and Registry virtualization to provide virtualized views of resources. When an application attempts to write to a system location, Windows Vista and Windows Server 2008 give the application a private copy of the file or Registry value. Any changes are then written to the private copy and this private copy is in turn stored in the user's profile data. If the application attempts to read or write to this system location again, it is given the private copy from the user's profile to work with. By default, if an error occurs when working with virtualized data, the error notification and logging information show the virtualized location rather than the actual location the application was trying to work with.

INSIDE OUT

The Transactional Registry

Windows Server 2008 implements transactional technology in the kernel to preserve data integrity and handle error conditions when writing to the NTFS file system and the Registry. Applications that are written to take advantage of the Transactional Registry can use transactions to manage Registry changes as discrete operations that can be committed if successful or rolled back if unsuccessful. While a transaction is active, Registry changes are not visible to users or other applications—it is only when Windows Server 2008 commits the transaction that the changes are applied fully and become visible. Transactions used with the Registry can be coordinated with any other transactional resource, such as Microsoft Message Queuing (MSMQ). If the operating system fails during a transaction, work that has started to commit is written to the disk and incomplete transactional work is rolled back.

INSIDE OUT

Controlling virtualization

In Local Security Policy, Security Options can enable or disable Registry virtualization. With Windows Vista and Windows Server 2008, a new security setting is provided for this purpose: User Account Control: Virtualize File And Registry Write Failures To Per-User Locations. This security setting enables the redirection of legacy application write failures to defined locations in the Registry and file system. This feature is designed to allow legacy programs that require administrator privileges to run. When enabled as per the default setting, this setting allows redirection of application write failures to defined user locations for both the file system and the Registry. When you disable this setting, applications that write data to protected locations silently fail.

To view or modify this setting in the Local Security Settings console, click Start, click Administrative Tools, and then click Local Security Policy. This opens the Local Security Policy console. Expand the Local Policies node in the left pane and then select the Security Options node. In the main pane, you should now see a list of policy settings. Scroll down through the list of security settings. Double-click User Account Control: Virtualize File And Registry Write Failures To Per-User Locations. On the Local Policy Setting tab of the dialog box, you'll see the current enabled or disabled state of the setting. To change the state of the setting select Enabled or Disabled as appropriate and then click OK.

Understanding the Registry Structure

Many administrative tools are little more than friendly user interfaces for managing the Registry, especially when it comes to Control Panel. So, rather than having you work directly with a particular area of the Registry, Microsoft provides tools that you can use to make the necessary changes safely and securely. Use these tools—that's what they are for.

CAUTION!

The importance of using the proper tools to make Registry changes cannot be overstated. If there's a tool that lets you manage an area of the Registry, you should use it. Don't fool around with the Registry just because you can. Making improper changes to the Registry can cause a system to become unstable, and in some cases, it could even make it so the system won't boot.

As you can see, nearly everything you do with the operating system affects the Registry in one way or another. That's why it's so important to understand what the Registry is used for, how you can work with it, how you can secure it, and how you can maintain it.

The Registry is first a database. Like any other database, the Registry is designed for information storage and retrieval. Any Registry value entry can be identified by

specifying the path to its location. For example, the path HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\ServerManager\DoNotOpenServerManagerAtLogon specifies a Registry value that you can use to enable or disable the automatic display of Server Manager at log on.

Figure 9-1 shows this value in the Registry. Because of its hierarchical structure, the Registry appears to be organized much like a file system. In fact, its structure is often compared to that of a file system. However, this is a bit misleading because there is no actual folder/file representation on a system's hard disk to match the structure used by the Registry. The Registry's actual physical structure is separate from the way Registry information is represented. Locations in the Registry are represented by a logical structure that has little correlation to how value entries are stored.

Unlike Windows 2000 and Windows NT, Windows Server 2003 and Windows Server 2008 support larger Registry sizes than were previously possible and no longer keep the entire Registry in paged pool memory. Instead, 256-kilobyte (KB) views of the Registry are mapped into system cache as needed. This is an important change from the original architecture of the Registry, which effectively limited the Registry to about 80 percent of the total size of paged pool memory. The new Registry implementation is limited only by available space in the paging file.

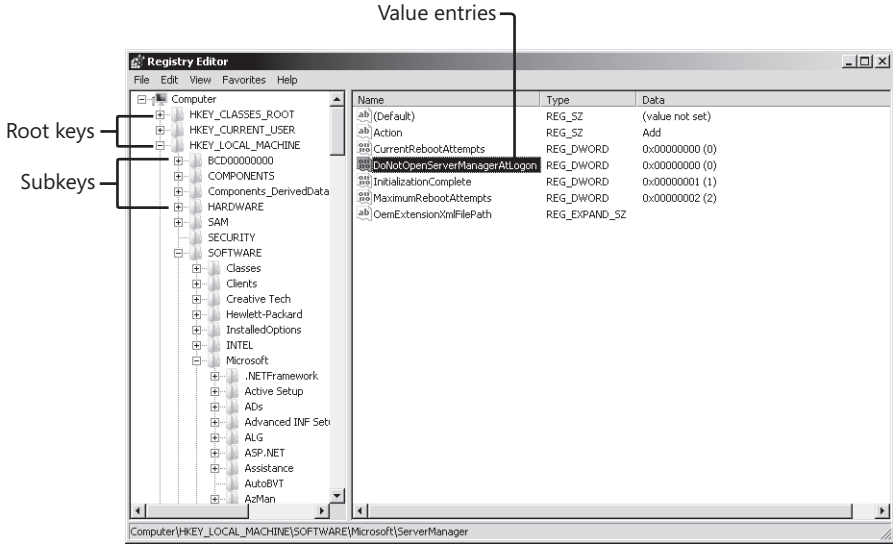


Figure 9-1 Accessing a value according to its path in the Registry.

At startup, 256-KB mapped views of the Registry are loaded into system cache so that Windows Server 2008 can quickly retrieve configuration information. Some of the Registry's information is created dynamically based on the system hardware configuration at startup and doesn't exist until it is created. For the most part, however, the Registry is stored in persistent form on disk and read from a set of files called hives. *Hives* are binary files that represent a grouping of keys and values. You'll find the hive files in the %SystemRoot%\System32\Config directory. Within this directory, you'll also find .sav, .log files, which serve as backup files for the Registry.

INSIDE OUT

Windows Server 2008 manages the Registry size and memory use

Windows NT and Windows 2000 store the entire Registry in paged, pooled memory. For 32-bit systems, this limits the Registry to approximately 160 megabytes (MB) because of the layout of the virtual address space in the operating system kernel. Unfortunately, in this configuration as the Registry grows in size it uses a considerable amount of paged, pooled memory and can leave too little memory for other kernel-mode components.

Windows Server 2003 and Windows Server 2008 resolve this problem by changing the way the Registry is stored in memory. Under the new implementation, 256-KB mapped views of the Registry are loaded into the system cache as necessary by the Cache Manager. The rest of the Registry is stored in the paging file on disk. Because the Registry is written to system cache, it can exist in system random access memory (RAM) and be paged to and from disk as needed. In previous versions of the Windows operating system, the operating system allowed you to control the maximum amount of memory and disk space that could be used by the Registry. With the improved memory management features, the operating system has now taken over control of managing how much memory the Registry uses. Most member servers use between 20 and 25 MB of memory for the Registry. Domain controllers or servers that have many configuration components, services, and applications can use considerably more. That said, however, one of my key domain controllers uses only 25 to 30 MB of memory for the Registry. This represents quite a change from the old architecture, when the in-memory requirements of the Registry could be up to 160 MB.

To read the Registry you need a special editor. The editor provided in Windows Server 2008 is Registry Editor. By using Registry Editor, you can navigate the Registry's logical structure from the top of the database to the bottom. From the top down, the levels of the database are defined as root keys, subkeys, and value entries.

INSIDE OUT

Regedit replaces Regedt32

Unlike previous versions of the Windows operating system that included two versions of Registry Editor, Windows Server 2003 and Windows Server 2008 ship with a single version. This version, Regedit.exe, integrates all of the features of both the previous Registry editors. From the original Regedit.exe it gets its core features. From Regedt32.exe, which is no longer available, it gets its security and Favorites features. By using the Permissions feature, you can view and manage permissions for Registry values. By using the Favorites feature, you can create and use favorites to quickly access stored locations within the Registry.

Regedt32 *really* is gone—although I, like many administrators, still refer to it. It is, after all, the editor administrators used because it gave us the ability to manage Registry security and it is the one that was recommended for administrators over Regedit. Because old habits die hard, Windows Server 2008 still has a stub file for Regedt32. However, if you run Regedt32, the operating system in fact starts Regedit.

At the top of the Registry hierarchy are the root keys. Each root key contains several subkeys, which contain other subkeys and value entries. The names of value entries must be unique within the associated subkey, and the value entries correspond to specific configuration parameters. The settings of those configuration parameters are the values stored in the value entry. Each value has an associated data type that controls the type of data it can store. For example, some value entries are used to store only binary data, while others are used to store only strings of characters, and the value's data type controls this.

We can now break down the Registry path HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\AllowMultipleTSSessions so that it is more meaningful. Here, *HKEY_LOCAL_MACHINE* is the root key. Each entry below the root key until we get to *AllowMultipleTSSessions* represents a subkey level within the Registry hierarchy. Finally, *AllowMultipleTSSessions* is the actual value entry.

The Registry is very complex and it is often made more confusing because documentation on the subject uses a variety of different terms beyond those already discussed. When reading about the Registry in various sources, you might see references to the following:

- **Subtrees** A *subtree* is a name for the tree of keys and values stemming from a root key down the Registry hierarchy. In documentation, you often see root keys referred to as subtrees. What the documentation means when it refers to a subtree is the branch of keys and values contained within a specified root key.
- **Keys** Technically, root keys are the top of the Registry hierarchy, and everything below a root key is either a subkey or a value entry. In practice, subkeys are often referred to as keys. It's just easier to refer to such and such a key—sort of like when we refer to "such and such a folder" rather than saying "subfolder."
- **Values** A *value* is the lowest level of the Registry hierarchy. For ease of reference, value entries are often simply referred to as values. Technically, however, a value entry comprises three parts: a name, a data type, and a value. The name identifies the configuration setting. The data type identifies the format for the data. The value is the actual data within the entry.

Now that you know the basics of the Registry's structure, let's dig deeper, taking a closer look at the root keys, major subkeys, and data types.

Registry Root Keys

The Registry is organized into a hierarchy of keys, subkeys, and value entries. The root keys are at the top of the hierarchy and form the primary branches, or subtrees, of Registry information. There are two physical root keys, HKEY_LOCAL_MACHINE and HKEY_USERS. These physical root keys are associated with actual files stored on the disk and are divided into additional logical groupings of Registry information. As shown in Table 9-1, the logical groupings are simply subsets of information gathered from HKEY_LOCAL_MACHINE and HKEY_USERS.

Table 9-1 **Registry Subtrees**

Subtree	Description
Physical Subtree	
HKEY_LOCAL_MACHINE (HKLM)	Stores all the settings that pertain to the hardware currently installed on the machine.
HKEY_USERS (HKU)	Stores user profile data for each user who has previously logged on to the computer locally as well as a default user profile.
Logical Subtree	
HKEY_CLASSES_ROOT (HKCR)	Stores all file associations and object linking and embedding (OLE) class identifiers. This subtree is built from HKEY_LOCAL_MACHINE\SOFTWARE\Classes and HKEY_CURRENT_USER\SOFTWARE\Classes.
HKEY_CURRENT_CONFIG (HKCC)	Stores information about the hardware configuration with which you started the system. This subtree is built from HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Hardware Profiles\Current, which in turn is a pointer to a numbered subkey that has the current hardware profile.
HKEY_CURRENT_USER (HKCU)	Stores information about the user currently logged on. This key has a pointer to HKEY_USERS\UserSID, where UserSID is the security identifier for the current user as well as for the default profile discussed previously.

INSIDE OUT

The Registry on 64-bit Windows systems

The Registry on 64-bit Windows systems is divided into 32-bit and 64-bit keys. Many keys are created in both 32-bit and 64-bit versions, and although the keys belong to different branches of the Registry, they have the same name. On these systems, Registry Editor (Regedit.exe) is designed to work with both 32-bit and 64-bit keys. The 32-bit keys, however, are represented with the WOW64 Registry redirector and appear under the HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node key. If you want to work directly with the 32-bit keys, you can do so by using the 32-bit Registry editor located in the file path %SystemRoot%\Syswow64\Regedit.

To support both 32-bit and 64-bit interoperability through the Component Object Model (COM) and the use of 32-bit programs, the WOW64 redirector mirrors COM-related Registry keys and values between the 64-bit and 32-bit Registry views. In some cases, the keys and values are modified during the reflection process to adjust pathnames and other values that might be version-dependent. This, in turn, means that the 32-bit and 64-bit values might differ.

HKEY_LOCAL_MACHINE

HKEY_LOCAL_MACHINE, abbreviated as HKLM, contains all the settings that pertain to the hardware currently installed on a system. It includes settings for memory, device drivers, installed hardware, and startup. Applications are supposed to store settings in HKLM only if the related data pertains to everyone who uses the computer.

As Figure 9-2 shows, HKLM contains the following major subkeys:

- COMPONENTS
- HARDWARE
- SAM
- SECURITY
- SOFTWARE
- SYSTEM

These subkeys are discussed in the sections that follow.

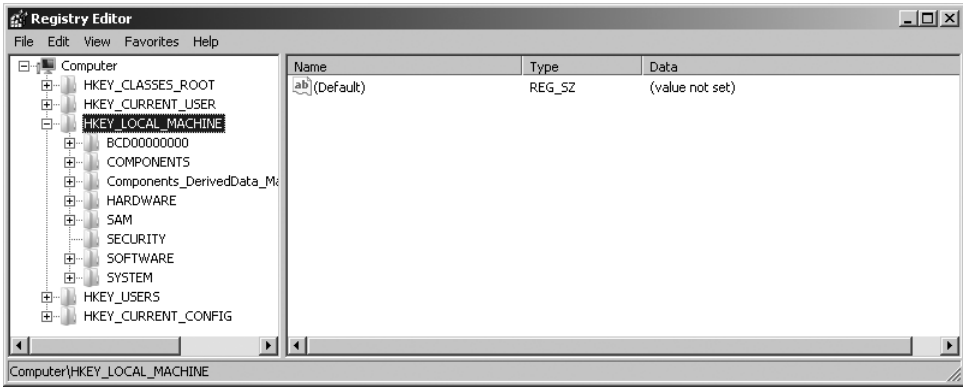


Figure 9-2 Accessing HKEY_LOCAL_MACHINE in the Registry.

HKLM\COMPONENTS

Windows Vista and Windows Server 2008 store information about updates and Windows features in a data store. These operating systems use the HKLM\COMPONENTS key to store information regarding the configuration and state of the data store, including the store architecture and format version. Windows Vista and Windows Server 2008 make changes to this data store whenever you download or install updates as well as when you add or remove features.

> **Note**
>
> If the component data store becomes corrupted you may see error code 0x80073712 whenever you try to install an update using the Windows Update Web site or you may find that Windows Features are not listed when you try to add or remove features. In this case, you can tell Windows that the store has become corrupted and should be rebuilt by typing the following command at an elevated command prompt: **reg delete HKLM\COMPONENTS /v StoreDirty**. See Microsoft Knowledge Base article 931712 for more information (*http://support.microsoft.com/kb/931712*).

HKLM\HARDWARE

HKLM\HARDWARE stores information about the hardware configuration for the computer. This key is re-created by the operating system each time you start Windows Server 2008, and it exists only in memory, not on disk. To build this key, the operating system enumerates every device it can find by scanning the system buses and by searching for specific classes of devices, such as serial ports, keyboards, and pointer devices.

Under HKLM\HARDWARE, you'll find four standard subkeys that are dynamically created at startup and contain the information gathered by the operating system. These subkeys are as follows:

- **ACPI** Contains information about the Advanced Configuration Power Interface (ACPI), which is a part of system BIOS that supports Plug and Play and advanced power management. This subkey doesn't exist on non-ACPI-compliant computers.
- **DESCRIPTION** Contains hardware descriptions including those for the system's central processor, floating-point processor, and multifunction adapters. For portable computers, one of the multifunction devices lists information about the docking state. For any computer with multipurpose chip sets, one of the multifunction devices lists information about the controllers for disks, keyboards, parallel ports, serial ports, and pointer devices. There's also a catchall category for other controllers, such as when a computer has a PC Card controller.
- **DEVICEMAP** Contains information that maps devices to device drivers. You'll find device mappings for keyboards, pointer devices, parallel ports, Small Computer System Interface (SCSI) ports, serial ports, and video devices. Of particular note is that within the VIDEO subkey is a value entry for the VGA-compatible video device installed on the computer. This device is used when the computer must start in VGA display mode.
- **RESOURCEMAP** Contains mappings for the hardware abstraction layer (HAL), for the Plug and Play Manager, and for available system resources. Of particular note is the Plug and Play Manager. It uses this subkey to record information about devices it knows how to handle.

Additional nonstandard subkeys can exist under HKLM\HARDWARE. The subkeys are specific to the hardware used by the computer.

HKLM\SAM

HKLM\SAM stores the Security Accounts Manager (SAM) database. When you create local users and groups on member servers and workstations, the accounts are stored in HKLM\SAM as they were in Windows NT. This key is also used to store information about built-in user and group accounts, as well as group membership and aliases for accounts.

By default, the information stored in HKLM\SAM is inaccessible through Registry Editor. This is a security feature designed to help protect the security and integrity of the system.

HKLM\SECURITY

HKLM\SECURITY stores security information for the local machine. It contains information about cached logon credentials, policy settings, service-related security settings, and default security values. It also has a copy of the HKLM\SAM. As with the HKLM\SAM subkey, this subkey is inaccessible through Registry Editor. This is a security feature designed to help protect the security and integrity of the system.

HKLM\SOFTWARE

HKLM\SOFTWARE stores machine-wide settings for every application and system component installed on the system. This includes setup information, executable paths, default configuration settings, and registration information. Because this subkey resides under HKLM, the information here is applied globally. This is different from the HKCU\SOFTWARE configuration settings, which are applied on a per-user basis.

As Figure 9-3 shows, you'll find many important subkeys within HKLM\SOFTWARE, including the following:

- **Classes** Contains all file associations and OLE class identifiers. This is also the key from which HKEY_CLASSES_ROOT is built.
- **Clients** Stores information about protocols and shells used by every client application installed on the system. This includes the calendar, contacts, mail, media, and news clients.
- **Microsoft** Contains information about every Microsoft application and component installed on the system. This includes their complete configuration settings, defaults, registration information, and much more. You'll find most of the graphical user interface (GUI) preferences in HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion. You'll find the configuration settings for most system components, language packs, hot fixes, and more under HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion.

- **ODBC** Contains information about the Open Database Connectivity (ODBC) configuration on the system. It includes information about all ODBC drives and ODBC file Data Source Names (DSNs).
- **Policies** Contains information about local policies for applications and components installed on the system.

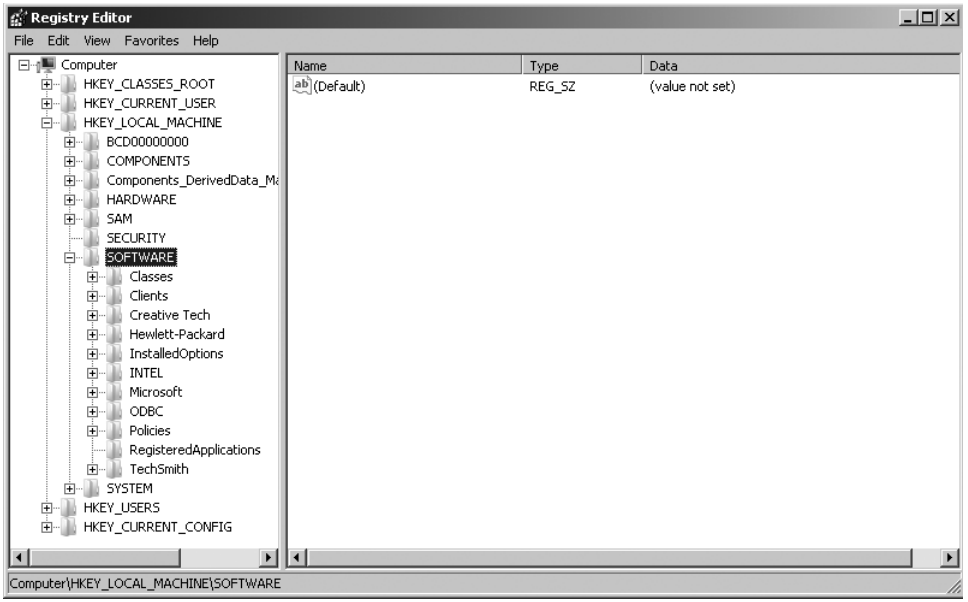


Figure 9-3 Accessing HKEY_LOCAL_MACHINE\SOFTWARE in the Registry.

HKLM\SYSTEM

HKLM\SYSTEM stores information about device drivers, services, startup parameters, and other machine-wide settings. You'll find several important subkeys within HKLM\SYSTEM. One of the most important is HKLM\SYSTEM\CurrentControlSet, as shown in Figure 9-4.

CurrentControlSet contains information about the set of controls and services used for the last successful boot of the system. This subkey always contains information on the set of controls actually in use and represents the most recent successful boot. The operating system writes the control set as the final part of the boot process so that it updates the Registry as appropriate to reflect which set of controls and services were last used for a successful boot. This is, in fact, how you can boot a system to the Last Known Good Configuration after it crashes or experiences a Stop error.

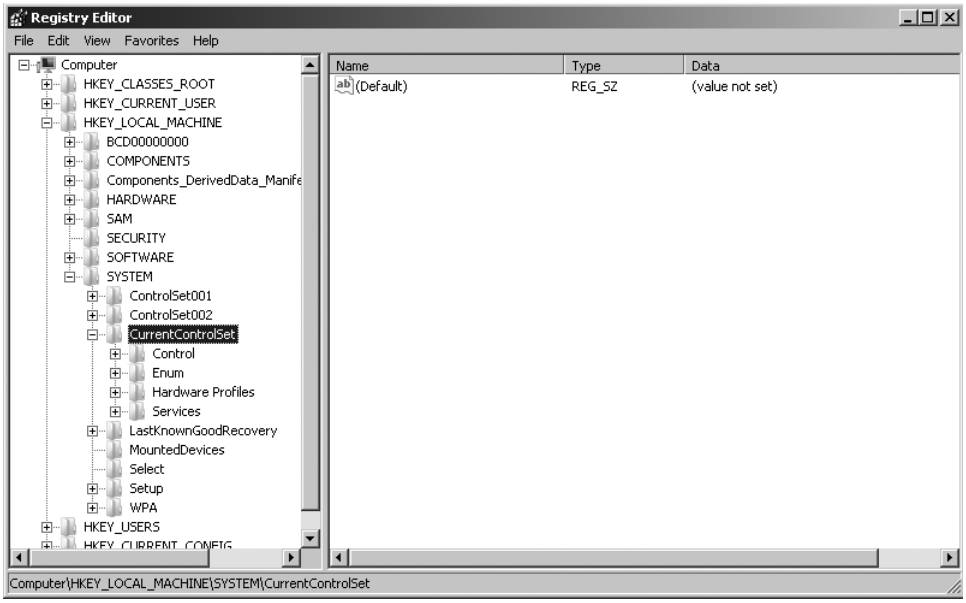


Figure 9-4 Accessing HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet in the Registry.

HKLM\SYSTEM also contains previously created control sets. These are saved under the subkeys named ControlSet001, ControlSet002, and so forth. Within the control sets, you'll find four important subkeys:

- **Control** Contains control information about key operating system settings, tools, and subcomponents, including the HAL, keyboard layouts, system devices, interfaces, and device classes. Under BackupRestore, you'll find the saved settings for Backup, which include lists of Automated System Recovery (ASR) keys, files, and Registry settings not to restore. Under the SafeBoot subkey, you'll find the control sets used for minimal and network-only boots of the system.
- **Enum** Contains the complete enumeration of devices found on the computer when the operating system scans the system buses and searches for specific classes of devices. This represents the complete list of devices present during startup of the operating system.
- **Hardware Profiles** Contains a subkey for each hardware profile available on the system. The first hardware profile, 0000, is an empty profile. The other numbered profiles, beginning with 0001, represent profiles that are available for use on the system. The profile named Current always points to the profile being used currently by the operating system.
- **Services** Contains a subkey for each service installed on the system. These subkeys store the necessary configuration information for their related services, which can include startup parameters as well as security and performance settings.

Another interesting subkey is HKLM\SYSTEM\MountedDevices. The operating system creates this key and uses it to store the list of mounted and available disk devices. Disk devices are listed according to logical volume configuration and drive letter designator.

HKEY_USERS

HKEY_USERS, abbreviated as HKU, contains user profile data for every user who has previously logged on to the computer locally, as well as a default user profile. Each user's profile is owned by that user unless you change permissions or move profiles. Profile settings include the user's desktop configuration, environment variables, folder options, menu options, printers, and network connections.

User profiles are saved in subkeys of HKEY_USERS according to their security identifiers (SIDs). There is also a *SecurityID*_Classes subkey that represents file associations that are specific to a particular user. For example, if a user sets Adobe Photoshop as the default program for .jpeg and .jpg files and this is different from the system default, there are entries within this subkey that show this association.

When you use Group Policy as discussed in Part 5, the policy settings are applied to the individual user profiles stored in this key. The default profile specifies how the machine behaves when no one is logged on and is also used as the base profile for new users who log on to the computer. For example, if you wanted to ensure that the computer used a password-protected screen saver when no one was logged on, you would modify the default profile accordingly. The subkey for the default user profile is easy to pick out because it is named HKEY_USERS\.DEFAULT.

> **Note**
>
> The profile information stored in HKU is loaded from the profile data stored on disk. The default location for profiles is %SystemDrive%\Users*UserName,* where *UserName* is the user's pre–Windows 2000 logon name.

HKEY_CLASSES_ROOT

HKEY_CLASSES_ROOT, abbreviated as HKCR, stores all file associations that tell the computer which document file types are associated with which applications, as well as which action to take for various tasks, such as open, edit, close, or play, based on a specified document type. For example, if you double-click a .doc file, the document typically is opened for editing in Microsoft Word. This file association is added to HKCR when you install Microsoft Office or Microsoft Word. If Microsoft Office or Microsoft

Word isn't installed, a .doc file is opened instead in WordPad because of a default file association created when the operating system is installed.

HKCR is built from HKEY_LOCAL_MACHINE\SOFTWARE\Classes and HKEY_CURRENT_USER\SOFTWARE\Classes. The former provides computer-specific class registration, and the latter, user-specific class registration. Because the user-specific class registrations have precedence, this allows for different class registrations for each user of the machine. This is different from previous versions of the Windows operating system for which the same class registration information was provided for all users of a particular machine.

HKEY_CURRENT_CONFIG

HKEY_CURRENT_CONFIG, abbreviated as HKCC, contains information about the hardware configuration with which you started the system, which is also referred to as the machine's boot configuration. This key contains information about the current device assignments, device drivers, and system services that were present at boot time.

HKCC is built from HKEY_LOCAL_MACHINE \SYSTEM\CurrentControlSet\Hardware Profiles\Current, which in turn is a pointer to a numbered subkey that contains the current hardware profile. If a system has multiple hardware profiles, the key points to a different hardware profile, depending on the boot state or the hardware profile selection made at startup.

HKEY_CURRENT_USER

HKEY_CURRENT_USER, abbreviated as HKCU, contains information about the user currently logged on. This key has a pointer to HKEY_USERS*UserSID*, where *UserSID* is the security identifier for the current user as well as for the default profile discussed previously. Microsoft requires that applications store user-specific preferences under this key. For example, Microsoft Office settings for individual users are stored under this key. Additionally, as discussed previously, HKEY_CURRENT_USER\SOFTWARE\Classes stores the user-specific settings for file associations.

> **Note**
>
> If you don't want users to be able to set their own file associations, you could change the permissions on HKLM\SOFTWARE\Classes so users can't alter the global settings you want them to have. For more information about Registry permissions, see "Securing the Registry" on page 276.

Registry Data: How It Is Stored and Used

Now that you know more about the Registry's structure, let's take a look at the actual data within the Registry. Understanding how Registry data is stored and used is just as important as understanding the Registry structure.

Where Registry Data Comes From

As mentioned previously, some Registry data is created dynamically during startup of the operating system and some is stored on disk so it can be used each time you boot a computer. The dynamically created data is volatile, meaning that when you shut down the system, it is gone. For example, as part of the startup process, the operating system scans for system devices and uses the results to build the HKEY_LOCAL_MACHINE\HARDWARE subkey. The information stored in this key exists only in memory and isn't stored anywhere on disk.

On the other hand, Registry data stored on disk is persistent. When you shut down a system, this Registry data remains on disk and is available the next time you boot the system. Some of this stored information is very important, especially when it comes to recovering from boot failure. For example, by using the information stored in HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet, you can boot using the Last Known Good Configuration. If the Registry data was corrupted, however, this information might not be available and the only way to recover the system would be to try repairing the installation or reinstalling the operating system.

To help safeguard the system and ensure that one section of bad data doesn't cause the whole Registry to fail to load, Windows Server 2008 has several built-in redundancies and fail safes. For starters, the Registry isn't written to a single file. Instead, it is written to a set of files called hives. There are six main types of hives, each representing a group of keys and values. Most of the hives are written to disk in the %SystemRoot%\System32\Config directory. Within this directory, you'll find these hive files:

- .DEFAULT, which corresponds to the HKEY_USERS\.DEFAULT subkey
- SAM, which corresponds to the HKEY_LOCAL_MACHINE\SAM subkey
- SECURITY, which corresponds to the HKEY_LOCAL_MACHINE\SECURITY subkey
- SOFTWARE, which corresponds to the HKEY_LOCAL_MACHINE\SOFTWARE subkey
- SYSTEM, which corresponds to the HKEY_LOCAL_MACHINE\SYSTEM subkey

The remaining hive files are stored in individual user profile directories with the default name of Ntuser.dat. These files are in fact hive files that are loaded into the Registry and used to set the pointer for the HKEY_CURRENT_USER root key. When no user is logged on to a system, the user profile for the default user is loaded into the Registry. When an actual user logs on, this user's profile is loaded into the Registry.

Note

The root keys not mentioned are HKEY_CURRENT_CONFIG and HKEY_CLASSES_ROOT. The on-disk data for HKEY_CURRENT_CONFIG comes from the subkey from which it is built: HKEY_LOCAL_MACHINE \SYSTEM\CurrentControlSet\Hardware Profiles\Current. Similarly, the on-disk data for HKEY_CLASSES_ROOT comes from HKEY_LOCAL_MACHINE \SOFTWARE\Classes and HKEY_CURRENT_USER\SOFTWARE\Classes.

Every hive file has associated log files—even Ntuser.dat. Windows Server 2008 uses the log files to help protect the Registry during updates. When a hive file is to be changed, the operating system writes the change to a log file and stores this log file on disk. The operating system then uses the change log to write the changes to the actual hive file. If the operating system were to crash while a change is being written to a hive file, the change log could later be used by the operating system to roll back the change, resetting the hive to its previous configuration.

INSIDE OUT

How Windows Server 2008 starts over with a clean Registry

Examine %SystemRoot%\System32\Config closely and you'll see several files with the .sav extension. These files represent the postinstallation state of the Registry. If you ever wonder how Windows Server 2008 can reset the Registry to that of a clean install after you demote a domain controller, this is the answer. By loading these files into the Registry and then writing them to disk as the original hive files, the server is returned to its postinstallation state with a clean Registry.

Types of Registry Data Available

When you work your way down to the lowest level of the Registry, you see the actual value entries. Each value entry has a name, a data type, and a value associated with it. Although value entries have a theoretical size limit of 1024 KB, most value entries are less than 1 KB in size. In fact, many value entries contain only a few bits of data. The type of information stored in these bits depends on the data type of the value entry.

The data types defined include the following:

- **REG_BINARY** Raw binary data without any formatting or parsing. You can view binary data in several forms, including standard binary and hexadecimal. In some cases, if you view the binary data, you will see the hexadecimal values as well as the text characters these values define.
- **REG_DWORD** A binary data type in which 32-bit integer values are stored as 4-byte-length values in hexadecimal. REG_DWORD is often used to track values

that can be incremented, 4-byte status codes, or Boolean flags. With Boolean flags, a value of 0 means the flag is off (false) and a value of 1 means the flag is on (true).

- **REG_QWORD** A binary data type in which 64-bit integer values are stored as 8-byte-length values in hexadecimal. REG_QWORD is often used to track large values that can be incremented, 8-byte status codes, or Boolean flags. With Boolean flags, a value of 0 means the flag is off (false) and a value of 1 means the flag is on (true).
- **REG_SZ** A fixed-length string of Unicode characters. REG_SZ is used to store values that are meant to be read by users and can include names, descriptions, and so on, as well as stored file system paths.
- **REG_EXPAND_SZ** A variable-length string that can include environment variables that are to be expanded when the data is read by the operating system, its components, or services, as well as installed applications. Environment variables are enclosed in percentage signs (%) to set them off from other values in the string. For example, %SystemDrive% refers to the SystemDrive environment variable. A REG_EXPAND_SZ value that defines a path to use could include this environment variable, such as %SystemDrive%\Program Files\Common Files.
- **REG_MULTI_SZ** A multiple-parameter string that can be used to store multiple string values in a single entry. Each value is separated by a standard delimiter so that the individual values can be picked out as necessary.
- **REG_FULL_RESOURCE_DESCRIPTOR** A value with an encoded resource descriptor, such as a list of resources used by a device driver or a hardware component. REG_FULL_RESOURCE_DESCRIPTOR values are associated with hardware components, such as a system's central processors, floating-point processors, or multifunction adapters.

The most common data types you'll see in the Registry are REG_SZ and REG_DWORD. The vast majority of value entries have this data type. The most important thing to know about these data types is that one is used with strings of characters and the other is used with binary data that is normally represented in hexadecimal format. And don't worry, if you have to create a value entry—typically you do so because you are directed to by a Microsoft Knowledge Base article in an attempt to resolve an issue—you are usually told which data type to use. Again, more often than not, this data type is either REG_SZ or REG_DWORD.

Working with the Registry

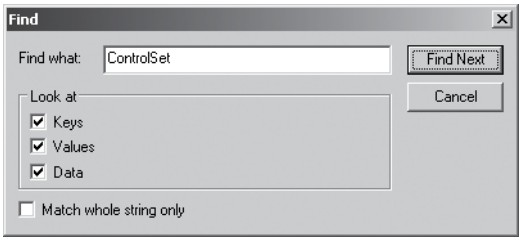
Windows Server 2008 provides several tools for working with the Registry. The main tool, of course, is Registry Editor, which you start by typing **regedit** or **regedt32** at the command line or in the Run dialog box. Another tool for working with the Registry is the REG command. Both tools can be used to view and manage the Registry. Keep in mind that although both tools are considered editors, Windows Server 2008 applies any changes you make immediately. Thus, any change you make is applied automatically to the Registry without you having to save the change.

CAUTION!

As an administrator, you have permission to make changes to most areas of the Registry. This allows you to make additions, changes, and deletions as necessary. However, before you do this, you should always make a backup of the system state along with the Registry first, as discussed in "Backing Up and Restoring the Registry" on page 272. This helps ensure that you can recover the Registry in case something goes wrong when you are making your modifications.

Searching the Registry

One of the common tasks you'll want to perform in Registry Editor is to search for a particular key. You can search for keys, values, and data entries using the Find option on the Edit menu (see the following screen).



Don't let the simplicity of the Find dialog box fool you—there is a bit more to searching the Registry than you might think. So, if you want to find what you're looking for, do the following:

- The Find function in Registry Editor searches from the current node forward to the last value in the final root key branch. So, if you want to search the complete Registry, you must select the Computer node in the left pane before you select Find on the Edit menu or press Ctrl+F.
- Type the text you want to find in the Find What box. You can search only for standard American Standard Code for Information Interchange (ASCII) text. So, if you're searching for data entries, Registry Editor searches only string values (REG_SZ, REG_EXPAND_SZ, and REG_MULTI_SZ) for the specified text.
- Use the Look At options to control where Registry Editor looks for the text you want to find. You can search on key names, value names, and text within data entries. If you want to match only whole strings instead of searching for text within longer strings, select the Match Whole String Only check box.

After you make your selections, click Find Next to begin the search. If Registry Editor finds a match before reaching the end of the Registry, it selects and displays the matching item. If the match isn't what you'tre looking for, press F3 to search again from the current position in the Registry.

Modifying the Registry

When you want to work with keys and values in the Registry, you typically are working with subkeys of a particular key. This allows you to add a subkey and define its values and to remove subkeys and their values. You cannot, however, add or remove root keys or insert keys at the root node of the Registry. Default security settings within some subkeys might also prohibit you from working with their keys and values. For example, by default you cannot create, modify, or remove keys or values within HKLM\SAM and HKLM\SECURITY.

Modifying Values

The most common change you'll make to the Registry is to modify an existing value. For example, a Knowledge Base article might recommend that you change a value from 0 to 1 to enable a certain feature in Windows Server 2008 or from 1 to 0 to disable it. To change a value, locate the value in Registry Editor, and then in the right pane double-click the value name. This opens an Edit dialog box, the style of which depends on the type of data you are modifying.

The most common values you'll modify are REG_SZ, REG_MULTI_SZ, and REG_DWORD. Figure 9-5 shows the Edit String dialog box, which is displayed when you modify REG_SZ values. In the dialog box, you would typically replace the existing value shown in the Value Data box with the value you need to enter.

Figure 9-5 Using the Edit String dialog box.

Figure 9-6 shows the Edit Multi-String dialog box, which is displayed when you modify REG_MULTI_SZ values. In this example, there are three separate string values. In the dialog box, each value is separated by a new line to make the values easier to work with. If directed to change a value, you would typically need to replace an existing value, making sure you don't accidentally modify the entry before or after the entry you are working with. If directed to add a value, you would begin typing on a new line following the last value.

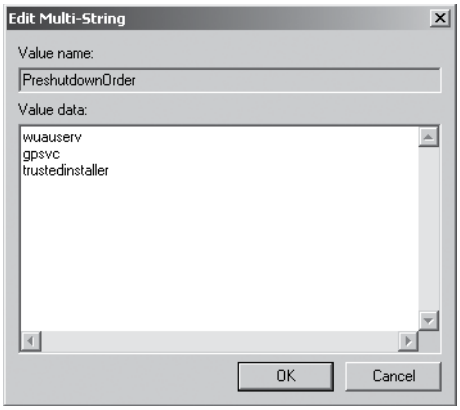


Figure 9-6 Using the Edit Multi-String dialog box.

Figure 9-7 shows the Edit DWORD Value dialog box, which is displayed when you modify REG_DWORD values. In this example, the value is displayed in hexadecimal format. Typically, you won't need to worry about the data format. You simply enter a new value as you've been directed. For example, if the Current value entry represents a flag, the data entry of 1 indicates the flag is on (or true). To turn off the flag (switch it to false), you would replace the 1 with a 0.

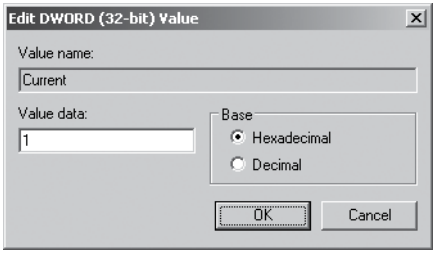


Figure 9-7 Using the Edit DWORD Value dialog box.

> **Note**
>
> The Windows Clipboard is available when you are working with Registry Editor. This means you can use the Copy, Cut, and Paste commands just as you do with other Windows programs. If there is a value in a Knowledge Base article that's difficult to type, you might want to copy it to the Clipboard and then paste it into the Value Data box of the Edit dialog box.

Adding Keys and Values

As noted previously, you can add or remove keys in most areas of the Registry. The exceptions pertain to the root node, the root keys, and areas of the Registry where permissions prohibit modifications.

You add new keys as subkeys of a selected key. Access the key with which you want to work, and then add the subkey by right-clicking the key and selecting Edit, New, and then Key. Registry Editor creates a new key and selects its name so that you can set it as appropriate. The default name is New Key #1.

The new key has a default value entry associated with it automatically. The data type for this default value is REG_SZ. Just about every key in the Registry has a similarly named and typed value entry, so don't delete this value entry. Either set its value by double-clicking it to display the Edit String dialog box, or create additional value entries under the selected key.

To create additional value entries under a key, right-click the key, then select New followed by one of these menu options:

- **String Value** Used to enter a fixed-length string of Unicode characters; type REG_SZ
- **Binary Value** Used to enter raw binary data without any formatting or parsing; type REG_BINARY
- **DWORD (32-bit) Value** Used to enter binary data type in which 4-byte integer values are stored; type REG_DWORD
- **QWORD (64-bit) Value** Used to enter binary data type in which 8-byte integer values are stored; type REG_QWORD
- **Multi-String Value** Used to enter a multiple-parameter string; type REG_MULTI_SZ
- **Expandable String Value** Used to enter a variable-length string that can include environment variables that are to be expanded when the data is read; type REG_EXPAND_SZ

Creating a new value adds it to the selected key and gives it a default name of New Value #1, New Value #2, and so on. The name of the value is selected for editing so that you can change it immediately. After you change the value name, double-click the value name to edit the value data.

Removing Keys and Values

Removing keys and values from the Registry is easy but should never be done without careful forethought to the possible consequences. That said, you delete a key or value by selecting it, and then pressing the Delete key. Registry Editor will ask you to confirm the deletion. After you do this, the key or value is permanently removed from the Registry. Keep in mind that when you remove a key, Registry Editor removes all subkeys and values associated with the key.

Modifying the Registry of a Remote Machine

You can modify the Registry of remote computers without having to log on locally. To do this, select Connect Network Registry on the File menu in Registry Editor, then use the Select Computer dialog box to specify the computer with which you want to work. In most cases, all you must do is type the name of the remote computer and then click OK. If prompted, you might need to enter the user name and password of a user account that is authorized to access the remote computer.

After you connect, you get a new icon for the remote computer under your Computer icon in the left pane of Registry Editor. Double-click this icon to access the physical root keys on the remote computer (HKEY_LOCAL_MACHINE and HKEY_USERS). The logical root keys aren't available because they are either dynamically created or simply pointers to subsets of information from within HKEY_LOCAL_MACHINE and HKEY_USERS. You can then edit the computer's Registry as necessary. When you are done, you can select Disconnect Network Registry on the File menu and then choose the computer from which you want to disconnect. Registry Editor then closes the Registry on the remote computer and breaks the connection.

When working with remote computers, you can also load or unload hives as discussed in "Loading and Unloading Hive Files" on page 270. If you're wondering why you would do this, the primary reason is to work with a specific hive, such as the hive that points to Dianne Prescott's user profile because she inadvertently changed the display mode to an invalid setting and can no longer access the computer locally. With her user profile data loaded, you could then edit the Registry to correct the problem and then save the changes so that she can once again log on to the system.

Importing and Exporting Registry Data

Sometimes you might find that it is necessary or useful to copy all or part of the Registry to a file. For example, if you've installed a service or component that requires extensive configuration, you might want to use it on another computer without having to go through the whole configuration process again. So, instead, you could install the service or component baseline on the new computer, then export the application's Registry settings from the previous computer, copy them over to the other computer, and then import the Registry settings so that the service or component is properly configured. Of course, this technique works only if the complete configuration of the service or component is stored in the Registry, but you can probably see how useful being able to import and export Registry data can be.

By using Registry Editor, it is fairly easy to import and export Registry data. This includes the entire Registry, branches of data stemming from a particular root key, and individual subkeys and the values they contain. When you export data, you create a .reg file that contains the designated Registry data. This Registry file is a script that can then be loaded back into the Registry of this or any other computer by importing it.

> **Note**
>
> Because the Registry script is written as standard text, you could view it and, if necessary, modify it in any standard text editor as well. Be aware, however, that double-clicking the .reg file launches Registry Editor, which prompts you as to whether you want to import the data into the Registry. If you are concerned about this, save the data to a file with the .hiv extension because double-clicking files with this extension won't start Registry Editor. Files with the .hiv extension must be manually imported (or you could simply change the file extension to .reg when it is time to use the data).

To export Registry data, right-click the branch or key you want to export, and then select Export. You can also right-click the root node for the computer you are working with, such as Computer for a local computer, to export the entire Registry. Either way, you'll see the Export Registry File dialog box as shown in Figure 9-8. Use the Save In selection list to choose a save location for the .reg file, and then type a file name. The Export Range panel shows you the selected branch within the Registry that will be exported. You can change this as necessary or select All to export the entire Registry. Then click Save to create the .reg file.

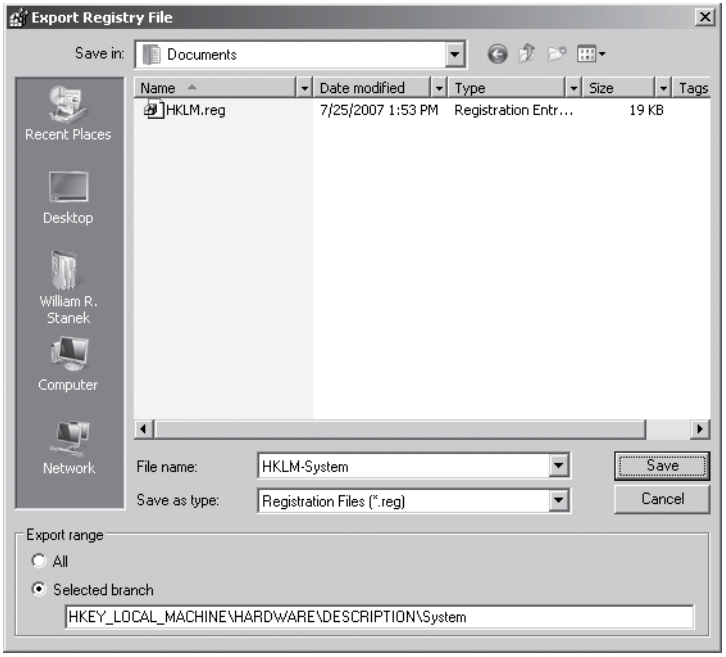


Figure 9-8 Exporting Registry data to a .reg file so that it can be saved and, if necessary, imported on this or another computer.

INSIDE OUT

Want to export the entire Registry quickly?

You can export the entire Registry at the command line by typing **regedit /e SaveFile**, where *SaveFile* is the complete file path to the location where you want to save the copy of the Registry. For example, if you wanted to save a copy of the Registry to C:\Corpsvr06-regdata.reg, you would type **regedit /e C:\corpsvr06-regdata.reg**.

You can also extend this technique to rapidly determine the exact Registry values the operating system modifies when you make a change to a system or application setting. Start by opening the application of the System utility you want to work with as well as a command prompt window. Next, export the Registry prior to making the change you want to track. Then immediately and without doing anything else, make the change that you want to track and export the Registry to a different file using the command prompt window you opened previously. Finally, use the file comparison tool (fc.exe) to compare the two files. For example, if you saved the original Registry to orig.reg and the changed Registry to new.reg, you could type the following command at a command prompt to write the changes to a file called changes.txt: **fc /u orig.reg new.reg > changes.txt**. When you examine the changes.txt file in a text editor, you'll see a comparison of the Registry files and the exact differences between the files.

Importing Registry data adds the contents of the Registry script file to the Registry of the computer you are working with, either creating new keys and values if they don't already exist or overwriting keys and values if they do exist. You can import Registry data in one of two ways. You can double-click the .reg file, which starts Registry Editor and prompts you as to whether you want to import the data. Or you can select Import on the File menu, then use the Import Registry File dialog box to select and open the Registry data file you want to import.

INSIDE OUT

Using export and import processes to distribute Registry changes

The export and import processes provide a convenient way to distribute Registry changes to users. You could, for example, export a subkey with an important configuration change and then mail the associated .reg file to users so they could import it simply by double-clicking it. Alternatively, you could copy the .reg file to a network share where users could access and load it. Either way, you have a quick and easy way to distribute Registry changes. Officially, however, distributing Registry changes in this manner is frowned upon because of the potential security problems associated with doing so. The preferred technique is to distribute Registry changes through Group Policy as discussed in Part 5.

Loading and Unloading Hive Files

Just as you sometimes must import or export Registry data, you'll sometimes need to work with individual hive files. The most common reason for doing this, as discussed previously, is when you must modify a user's profile to correct an issue that prevents the user from accessing or using a system. Here, you would load the user's Ntuser.dat file into Registry Editor and then make the necessary changes. Another reason for doing this would be to change a particular part of the Registry on a remote system. For example, if you needed to repair an area of the Registry, you could load the related hive file into the Registry of another machine and then repair the problem on the remote machine.

Loading and unloading hives affects only HKEY_LOCAL_MACHINE and HKEY_USERS, and you can perform these actions only when you select one of these root keys. Rather than replacing the selected root key, the hive you are loading then becomes a subkey of that root key. HKEY_LOCAL_MACHINE and HKEY_USERS are of course used to build all the logical root keys used on a system, so you could in fact work with any area of the Registry.

After you select either HKEY_LOCAL_MACHINE or HKEY_USERS in Registry Editor, you can load a hive for the current machine or another machine by selecting Load Hive on the File menu. Registry Editor then prompts you for the location and name of the previously saved hive file. Select the file, and then click Open. Afterward, enter a name for the key under which you want the hive to reside while it is loaded into the current system's Registry, and then click OK.

> **Note**
>
> You can't work with hive files that are already being used by the operating system or another process. You could, however, make a copy of the hive and then work with it. At the command line, type **reg save** followed by the abbreviated name of the root key to save and the file name to use for the hive file. For example, you could type **reg save hkcu c:\curr-hkcu.hiv** to save HKEY_CURRENT_USER to a file called Curr-hkcu.hiv on drive C. Although you can save the logical root keys (HKCC, HKCR, HKCU) in this manner, you can save only subkeys of HKLM and HKU using this technique.

When you are finished working with a hive, you should unload it to clear it out of memory. Unloading the hive doesn't save the changes you've made—as with any modifications to the Registry, your changes are applied automatically without the need to save them. To unload a hive, select it, and choose Unload Hive on the File menu. When prompted to confirm, click Yes.

Working with the Registry from the Command Line

If you want to work with the Registry from the command line, you can do so using the REG command. REG is run using the permissions of the current user and can be used to access the Registry on both local and remote systems. As with Registry Editor, you can work only with HKEY_LOCAL_MACHINE and HKEY_USERS on remote computers. These keys are, of course, used to build all the logical root keys used on a system, so you can in fact work with any area of the Registry on a remote computer.

REG has different subcommands for performing various Registry tasks. These commands include the following:

- **REG ADD** Adds a new subkey or value entry to the Registry
- **REG COMPARE** Compares Registry subkeys or value entries
- **REG COPY** Copies a Registry entry to a specified key path on a local or remote system
- **REG DELETE** Deletes a subkey or value entries from the Registry
- **REG EXPORT** Exports Registry data and writes it to a file

> **Note**
>
> These files have the same format as files you export from Registry Editor. Typically, however, they are saved with the .hiv extension so double-clicking files with this extension won't start Registry Editor.

- **REG IMPORT** Imports Registry data and either creates new keys and value entries or overwrites existing keys and value entries
- **REG LOAD** Loads a Registry hive file
- **REG QUERY** Lists the value entries under a key and the names of subkeys (if any)
- **REG RESTORE** Writes saved subkeys and entries back to the Registry
- **REG SAVE** Saves a copy of specified subkeys and value entries to a file
- **REG UNLOAD** Unloads a Registry hive file

You can learn the syntax for using each of these commands by typing **reg** followed by the name of the subcommand you want to learn about and then **/?**. For example, if you wanted to learn more about REG ADD, you would type **reg add /?** at the command line.

Backing Up and Restoring the Registry

By now it should be pretty clear how important the Registry is and that it should be protected. I'll go so far as to say that part of every backup and recovery plan should include the Registry. Backing up and restoring the Registry normally isn't done from within Registry Editor, however. It is handled through the Windows Server Backup utility or through your preferred third-party backup software. Either way, you have an effective means to minimize downtime and ensure that the system can be recovered if the Registry becomes corrupted.

You can make a backup of the entire Registry very easily at the command line. Simply type **regedit /e *SaveFile***, where *SaveFile* is the complete file path to the save location for the Registry data. Following this, you could save a copy of the Registry to C:\Backups\Regdata.reg by typing **regedit /e c:\backups\regdata.reg**. You would then have a complete backup of the Registry.

You can also easily make backups of individual root keys. To do this, you use REG SAVE. Type **reg save** followed by the abbreviated name of the root key you want to save and the file name to use. For example, you could type **reg save hkcu c:\backups\hkcu.hiv** to save HKEY_CURRENT_USER to a file in the C:\Backups directory. Again, although you can save the logical root keys (HKCC, HKCR, HKCU) in this manner, you can save only subkeys of HKLM and HKU using this technique.

Okay, so now you have your fast and easy backups of Registry data. What you do not have, however, is a sure way to recover a system in the event the Registry becomes corrupted and the system cannot be booted. Partly this is because you have no way to boot the system to get at the Registry data.

In Windows Server 2008, you create a system state backup to help you recover the Registry and get a system to a bootable state. The system state backup includes essential system files needed to recover the local system as well as Registry data. All computers have system state data, which must be backed up in addition to other files to restore a complete working system.

Normally, you back up the system state data when you perform a normal (full) backup of the rest of the data on the system. Thus, if you are performing a full recovery of a server rather than a repair, you use the complete system backup as well as system state data to recover the server completely. Techniques for performing full system backups and recovery are discussed in Chapter 41, "Backup and Recovery."

That said, you can create separate system state backups. The fastest and easiest way to do so is to use Wbadmin, the command-line counterpart to Windows Server Backup. You create a system state backup using Wbadmin by entering the following command at an elevated command prompt:

```
wbadmin start systemstatebackup -backuptarget StorageDrive
```

where *StorageDrive* is the drive letter for the storage location, such as:

```
wbadmin start systemstatebackup -backuptarget d:
```

Maintaining the Registry

The Registry is a database, and like any other database it works best when it is optimized. Optimize the Registry by reducing the amount of clutter and information it contains. This means uninstalling unnecessary system components, services, and applications. One way to uninstall components, services, and applications is to use the Uninstall Or Change A Program utility in Control Panel. This utility allows you to remove Windows components and their related services safely as well as applications installed using the Windows Installer. In Control Panel, click the Uninstall A Program link under the Programs heading to access the Uninstall Or Change A Program utility.

Most applications include uninstall utilities that attempt to remove the application, its data, and its Registry settings safely and effectively as well. Sometimes, however, applications either do not include an uninstall utility or for one reason or another do not fully remove their Registry settings, and this is where Registry maintenance utilities come in handy.

At the Microsoft Download Center on the Web, you'll find a download package for the Windows Installer Clean Up Utility. This download package includes several files as well as a helper application called Windows Installer Zapper. The Windows Installer Clean Up Utility calls Windows Installer Zapper to perform clean up operations on the Windows Installer configuration management information. Although not to be used by novice administrators, you can also work directly with Windows Installer Zapper.

Before you download and work with these utilities, you should refer to Microsoft Knowledge Base Article 29031 (*http://support.microsoft.com/kb/290301/en-us*). This article also includes a download link for obtaining the installer package. After you download the installer package, right-click it and then select Run As Administrator. You can then follow the prompts to install the Clean Up utilities. In the %SystemDrive%\Program Files\Windows Installer Clean Up folder, you'll find Windows Installer Clean Up Utility (msicuu.exe), Windows Installer Zapper (msizap.exe), and a read me file (readme.txt).

> **Note**
>
> There are two versions of Windows Installer Zapper: MsiZapA.exe is for use in Windows 95, Windows 98, and Windows Me, and MsiZapU.exe is for use in all other versions of Windows. When you install the Windows Installer Clean Up Utility, the installation process installs the correct version automatically and renames the .exe as Msizap.exe.

Both tools are designed to work with programs installed using the Windows Installer and must be run using an account with Administrator permissions. In addition to being able to clear out Registry settings for programs you've installed and then uninstalled, you can use these utilities to recover the Registry to the state it was in prior to a failed

or inadvertently terminated application installation. This works as long as the application used the Windows Installer.

Using the Windows Installer Clean Up Utility

Windows Installer Clean Up Utility removes Registry settings for applications that were installed using the Windows Installer. It is most useful for cleaning up Registry remnants of applications that were partially uninstalled or whose uninstall failed. It is also useful for cleaning up applications that can't be uninstalled or reinstalled because of partial or damaged settings in the Registry. It isn't, however, intended to be used as an uninstaller because it won't clean up the application's files or shortcuts and will make it necessary to reinstall the application to use it again.

> **Note**
>
> Keep in mind that the profile of the current user is part of the Registry. Because of this, the Windows Installer Clean Up Utility will remove user-specific installation data from this profile. It won't, however, remove this information from other profiles.

If you've already run the installer package, you can start this utility by clicking Start, All Programs, Windows Installer Clean Up. When the Windows Installer Clean Up Utility dialog box is displayed, select the program or programs to clean up, and then click Remove. The Windows Installer Clean Up Utility keeps a log file to record the applications that users delete in this manner. The log is stored in the %SystemDrive%\Users*UserName*\AppData\Local \Temp directory and is named Msicuu.log.

> **Note**
>
> The Windows Installer Clean Up Utility is a GUI for the Windows Installer Zapper discussed in the next section. When you use this utility, it runs the Windows Installer Clean Up Utility with the /T parameter to delete an application's Registry entries. It has an added benefit because it creates a log file, which is not used with Windows Installer Zapper.

CAUTION!

The Windows Installer Clean Up Utility is meant to be used as a last resort only. Don't use this program if you can uninstall programs by other means.

Using the Windows Installer Zapper

The Windows Installer Zapper (Msizap.exe) is an advanced command-line utility for removing Registry settings for applications that were installed using the Windows Installer. Like the Windows Installer Clean Up Utility, it can be used to clean up Registry settings for applications that were partially uninstalled or for which the uninstall failed, as well as applications that can't be uninstalled or reinstalled because of partial or damaged settings in the Registry. Additionally, it can be used to remove Registry settings related to failed installations or failed rollbacks of installations. It can also be used to correct failures related to multiple instances of a setup program running simultaneously and in cases when a setup program won't run. Because you can inadvertently cause serious problems with the operating system, only experienced administrators should use this utility.

You'll find the Windows Installer Zapper in the %SystemDrive%\Program Files\Windows Installer Clean Up folder. The complete syntax for the Windows Installer Zapper is as follows:

```
msizap [*] [!] [A] [M] [P] [S] [W] [T] [G] [AppToZap]
```

where

- **AppToZap** Specifies an application's product code or the file path to the application Windows Installer (.msi) program
- ***** Deletes all Windows Installer configuration information on the computer, including information stored in the Registry and on disk. Must be used with the ALLPRODUCTS flag
- **!** Turns off warning prompts asking you to confirm your actions
- **A** Gives administrators Full Control permissions on the applicable Windows Installer data so that it can be deleted even if the administrator doesn't have specific access to the data
- **M** Deletes Registry information related to managed patches
- **P** Deletes Registry information related to active installations
- **S** Deletes Registry information saved for rollback to the previous state
- **T** Used when you are specifying a specific application to clean up
- **W** Examines all user profiles for data that should be deleted
- **G** Removes orphaned Windows Installer files that have been cached for all users

CAUTION!

Windows Installer Zapper is meant as a last resort only. Don't use this program if you can uninstall programs by other means.

Removing Registry Settings for Active Installations That Have Failed

Application installations can fail during installation or after installation. When applications are being installed, an InProgress key is created in the Registry under the HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Installer subkey. In cases when installation fails, the system might not be able to edit or remove this key, which could cause the application's setup program to fail the next time you try to run it. Running Windows Installer Zapper with the P parameter clears out the InProgress key, which should allow you to run the application's setup program.

After installation, applications rely on their Registry settings to configure themselves properly. If these settings become damaged or the installation becomes damaged, the application won't run. Some programs have a repair utility that can be accessed simply by rerunning the installation. During the repair process, the Windows Installer might attempt to write changes to the Registry to repair the installation or roll it back to get back to the original state. If this process fails for any reason, the Registry can contain unwanted settings for the application. Running Windows Installer Zapper with the S parameter clears out the rollback data for the active installation. Rollback data is stored in the HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Installer\Rollback key.

Any running installation also has rollback data, so you typically use the P and S parameters together. This means you would type **msizap ps** at an elevated command line.

Removing Partial or Damaged Settings for Individual Applications

When an application can't be successfully uninstalled you can attempt to clean up its settings from the Registry using the Windows Installer Zapper. To do this, you need to know the product code for the application or the full path to the Windows Installer file used to install the application. The installer file ends with the .msi extension and usually is found in one of the application's installation directories.

You then type **msizap t** followed by the product code or .msi file path. For example, if the installer file path is C:\Apps\KDC\KDC.msi, you would type **msizap t c:\apps\kdc\kdc.msi** at the command line to clear out the application's settings. Because the current user's profile is a part of the Registry, user-specific settings for the application will be removed from this profile. If you want to clear out these settings for all user profiles on the system, add the W parameter, such as **msizap wt c:\apps\kdc\kdc.msi**.

Securing the Registry

The Registry is a critical area of the operating system. It has some limited built-in security to reduce the risk of settings being inadvertently changed or deleted. Additionally, some areas of the Registry are available only to certain users. For example, HKLM\SAM and HKLM\SECURITY are available only to the LocalSystem user. This security in some cases might not be enough, however, to prevent unauthorized access to the

Registry. Because of this, you might want to set tighter access controls than the default permissions, and you can do this from within the Registry. You can also control remote access to the Registry and configure access auditing.

Preventing Access to the Registry Utilities

One of the best ways to protect the Registry from unauthorized access is to make it so users can't access the Registry in the first place. For a server, this means tightly controlling physical security and allowing only administrators the right to log on locally. For other systems or when it isn't practical to prevent users from logging on locally to a server, you can configure the permissions on Regedit.exe and Reg.exe so that they are more secure. You could also remove Registry Editor and the REG command from a system, but this can introduce other problems and make managing the system more difficult, especially if you also prevent remote access to the Registry.

To modify permissions on Registry Editor, access the %SystemRoot% folder, right-click Regedit.exe, and then select Properties. In the Regedit Properties dialog box, click the Security tab, as shown in Figure 9-9. Add and remove users and groups as necessary, then set permissions as appropriate. Permissions work the same as with other types of files. You select an object and then allow or deny specific permissions. See Chapter 14, "File Sharing and Security," for details.

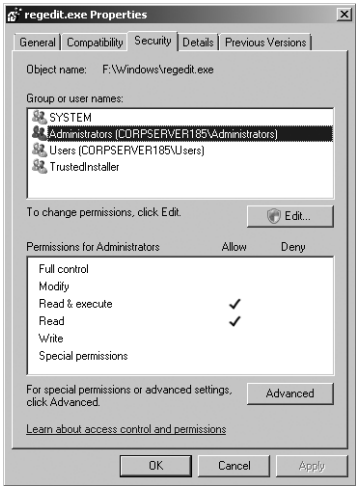


Figure 9-9 Tighten controls on Registry Editor to limit access to it.

To modify permissions on the REG command, access the %SystemRoot%\System32 folder, right-click Reg.exe, and then select Properties. In the Reg Properties dialog box, click the Security tab. As Figure 9-10 shows, this command by default can be used by users as well as administrators. Add and remove users and groups as necessary, then set permissions as appropriate.

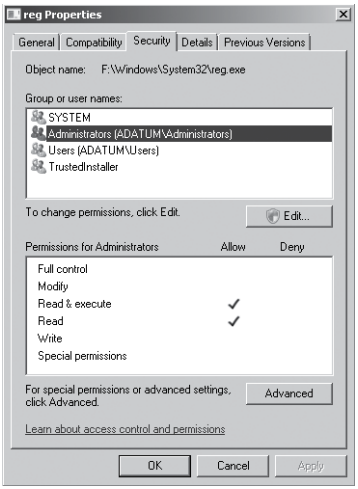


Figure 9-10 Reg.exe is designed to be used by users as well as administrators and to be run from the command line; its permissions reflect this.

> **Note**
>
> I'm not forgetting about Regedt32. It's only a link to Regedit.exe, so you don't really need to set its access permissions. The permissions on Regedit.exe will apply regardless of whether users attempt to run Regedt32 or Regedit.exe.

Applying Permissions to Registry Keys

Keys within the Registry have access permissions as well. Rather than editing these permissions directly, I recommend you use an appropriate security template as discussed in Chapter 36, "Managing Group Policy." Using the right security template locks down access to the Registry for you, and you won't have to worry about making inadvertent changes that will prevent systems from booting or applications from running.

That said, you might in some limited situations want to or have to change permissions on individual keys in the Registry. To do this, start Registry Editor and then navigate to the key you want to work with. When you find the key, right-click it, and select Permissions, or select the key, then choose Permissions on the Edit menu. This displays a Permissions For dialog box similar to the one shown in Figure 9-11. Permissions work the same as for files. You can add and remove users and groups as necessary. You can select an object and then allow or deny specific permissions.

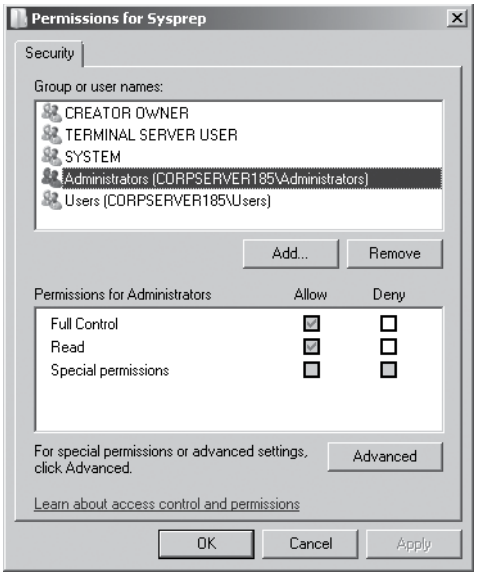


Figure 9-11 Use the Permissions For dialog box to set permissions on specific Registry keys.

Many permissions are inherited from higher-level keys and are unavailable. To edit these permissions, you must access the Advanced Security Settings dialog box by clicking the Advanced button. As Figure 9-12 shows, the Advanced Security Settings dialog box has four tabs:

- **Permissions** The Inherited From column on the Permissions tab shows from where the permissions are inherited. Usually, this is the root key for the key branch you are working with, such as CURRENT_USER. You can use the Add and Edit buttons on the Permissions tab to set access permissions for individual users and groups. Table 9-2 shows the individual permissions you can assign.

CAUTION!

Before you click OK to apply changes, consider whether you should clear the Include Inheritable Permissions From This Object's Parent option. If you don't do this, you'll change permissions on the selected key and all its subkeys.

- **Auditing** Allows you to configure auditing for the selected key. The actions you can audit are the same as the permissions listed in Table 9-2. See "Registry Root Keys" on page 251.

- **Owner** Shows the current owner of the selected key and allows you to reassign ownership. By default, only the selected key is affected, but if you want the change to apply to all subkeys of the currently selected key, choose Replace Owner On Subcontainers And Objects.

CAUTION

Be sure you understand the implications of taking ownership of Registry keys. Changing ownership could inadvertently prevent the operating system or other users from running applications, services, or application components.

- **Effective Permissions** Lets you see which permissions would be given to a particular user or group based on the current settings. This is helpful because permission changes you make on the Permissions tab aren't applied until you click OK or Apply.

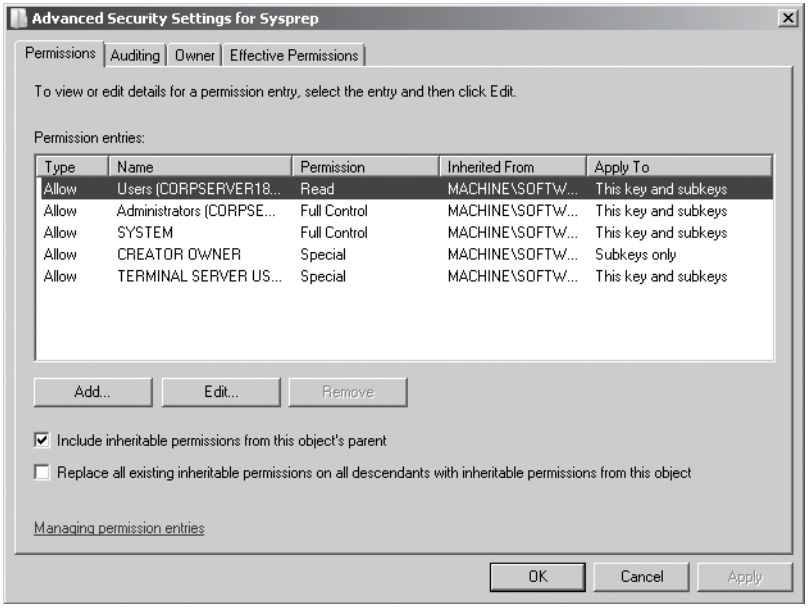


Figure 9-12 Use the Advanced Security Settings dialog box to change the way permissions are inherited or set and to view auditing settings, ownership, and effective permissions.

Table 9-2 **Registry Permissions and Their Meanings**

Permission	Meaning
Full Control	Allows user or group to perform any of the actions related to any other permission
Query Value	Allows querying the Registry for a subkey value
Set Value	Allows creating new values or modifying existing values below the specified key
Create Subkey	Allows creating a new subkey below the specified key
Enumerate Subkeys	Allows getting a list of all subkeys of a particular key
Notify	Allows registering a callback function that is triggered when the select value changes
Create Link	Allows creating a link to a specified key
Delete	Allows deleting a key or value
Write DAC	Allows writing access controls on the specified key
Write Owner	Allows taking ownership of the specified key
Read Control	Allows reading the discretionary access control list (DACL) for the specified key

Controlling Remote Registry Access

Hackers and unauthorized users can attempt to access a system's Registry remotely just like you do. If you want to be sure they are kept out of the Registry, you can prevent remote Registry access. One way remote access to a system's Registry can be controlled is through the Registry key HKLM\SYSTEM\CurrentControlSet\Control\SecurePipeServers\Winreg. If you want to limit remote access to the Registry, you can start by changing the permissions on this key.

If this key exists, then the following occurs:

1. Windows Server 2008 uses the permissions on the key to determine who can access the Registry remotely, and by default any authenticated user can do so. In fact, authenticated users have Query Value, Enumerate Subkeys, Notify, and Read Control permissions on this key.
2. Windows Server 2008 then uses the permissions on the keys to determine access to individual keys.

If this key doesn't exist, Windows Server 2008 allows all users to access the Registry remotely and uses the permissions on the keys only to determine which keys can be accessed.

INSIDE OUT

Services might need remote access to the Registry

Some services require remote access to the Registry to function correctly. This includes the Directory Replicator service and the Spooler service. If you restrict remote access to the Registry, you must bypass the access restrictions. Either add the account name of the service to the access list on the Winreg key or list the keys to which services need access in the Machine or Users value under the AllowedPaths key. Both values are REG_MULTI_SZ strings. Paths entered in the Machine value allow machine (LocalSystem) access to the locations listed. Paths entered in the Users value allow users access to the locations listed. As long as there are no explicit access restrictions on these keys, remote access is granted. After you make changes, you must restart the computer so that Registry access can be reconfigured on startup.

Windows Vista and Windows Server 2008 disable remote access to all Registry paths by default. As a result, the only Registry paths remotely accessible are those explicitly permitted as part of the default configuration or by an administrator. In Local Security Policy, you can use Security Options to enable or disable remote Registry access. With Windows Vista and Windows Server 2008, two new security settings are provided for this purpose:

- Network Access: Remotely Accessible Registry Paths
- Network Access: Remotely Accessible Registry Paths And Sub-Paths

These security settings determine which Registry paths and subpaths can be accessed over the network, regardless of the users or groups listed in the access control list (ACL) of the Winreg Registry key. A number of default paths are set, and you should not modify these default paths without carefully considering the damage that changing this setting may cause.

You can follow these steps to access and modify these settings in the Local Security Policy console:

1. Click Start, click Administrative Tools, and then click Local Security Policy. This opens the Local Security Policy console.
2. Expand the Local Policies node in the left pane and then select the Security Options node.
3. In the main pane, you should now see a list of policy settings. Scroll down through the list of security settings. As appropriate, double-click Network Access: Remotely Accessible Registry Paths or Network Access: Remotely Accessible Registry Paths And Sub-Paths.
4. On the Local Policy Setting tab of the Properties dialog box, you'll see a list of remotely accessible Registry paths or a list of remotely accessible Registry paths and subpaths depending on which security setting you are working with. You can

now add or remove paths or subpaths as necessary. Note that the default settings are listed on the Explain tab.

> **Note**
>
> Windows Server 2008 has an actual service called Remote Registry service. This service does in fact control remote access to the Registry. You want to disable this service only if you are trying to protect isolated systems from unauthorized access, such as when the system is in a perimeter network and is accessible from the Internet. If you disable Remote Registry service before starting the Routing and Remote Access service, you cannot view or change the Routing and Remote Access configuration. Routing and Remote Access reads and writes configuration information to the Registry, and any action that requires access to configuration information could cause Routing and Remote Access to stop functioning. To resolve this, stop the Routing and Remote Access service, start the Remote Registry service, and then restart the Routing and Remote Access service.

Auditing Registry Access

Access to the Registry can be audited as can access to files and other areas of the operating system. Auditing allows you to track which users access the Registry and what they're doing. All the permissions listed previously in Table 9-1 can be audited. However, you usually limit what you audit to only the essentials to reduce the amount of data that is written to the security logs and to reduce the resource burden on the affected server.

Before you can enable auditing of the Registry, you must enable the auditing function on the system you are working with. You can do this either through the server's local policy or through the appropriate Group Policy Object. The policy that controls auditing is Computer Configuration\Windows Settings\Security Settings\Local Policies\Audit Policy. For more information on auditing and Group Policy, see Chapter 14 and Chapter 36, respectively.

After auditing is enabled for a system, you can configure how you want auditing to work for the Registry. This means configuring auditing for each key you want to track. Thanks to inheritance, this doesn't mean you have to go through every key in the Registry and enable auditing for it. Instead, you can select a root key or any subkey to designate the start of the branch for which you want to track access and then ensure the auditing settings are inherited for all subkeys below it (this is the default setting).

Say, for example, you wanted to audit access to HKLM\SAM and its subkeys. To do this, you would follow these steps:

1. After you locate the key in Registry Editor, right-click it, and select Permissions, or select the key, then choose Permissions on the Edit menu. This displays the Permissions For SAM dialog box.

2. In the Permissions For SAM dialog box, click the Advanced button.
3. In the Advanced Security Settings dialog box, click the Auditing tab.
4. Click Add to select a user or group whose access you want to track.
5. After you select the user or group, click OK. The Auditing Entry For SAM dialog box is displayed, as shown in Figure 9-13.

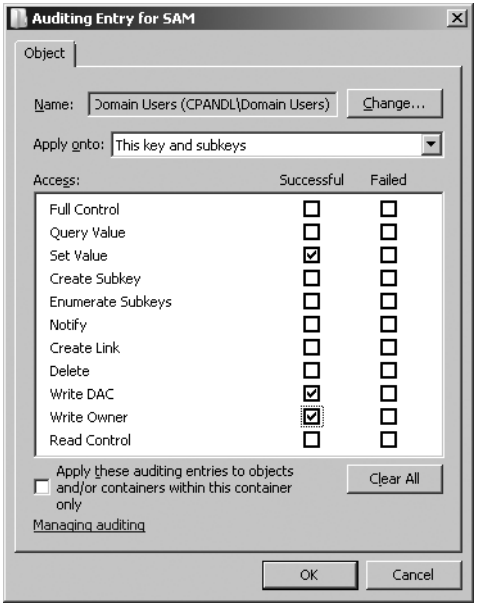


Figure 9-13 Use the Auditing Entry For dialog box to specify the permissions you want to track.

6. For each permission, select the type of auditing you want to track. If you want to track successful use of the permission, select the adjacent Successful check box. If you want to track failed use of the permission, select the adjacent Failed check box. Click OK to close the dialog box.
7. Repeat Step 6 to audit other users or groups.
8. If you want auditing to apply to subkeys, ensure the Include Inheritable Auditing Entries From This Object's Parent check box is selected.
9. Click OK twice.

CHAPTER 29

Active Directory Architecture

Active Directory is an extensible directory service that enables you to manage network resources efficiently. A directory service does this by storing detailed information about each network resource, which makes it easier to provide basic lookup and authentication. Being able to store large amounts of information is a key objective of a directory service, but the information must be also organized so that it is easily searched and retrieved.

Active Directory provides for authenticated search and retrieval of information by dividing the physical and logical structure of the directory into separate layers. Understanding the physical structure of Active Directory is important for understanding how a directory service works. Understanding the logical structure of Active Directory is important for implementing and managing a directory service.

Active Directory Physical Architecture

Active Directory's physical layer controls the following features:

- How directory information is accessed
- How directory information is stored on the hard disk of a server

Active Directory Physical Architecture: A Top-Level View

From a physical or machine perspective, Active Directory is part of the security subsystem (see Figure 29-1). The security subsystem runs in user mode. User-mode applications do not have direct access to the operating system or hardware. This means that requests from user-mode applications have to pass through the executive services layer and must be validated before being executed.

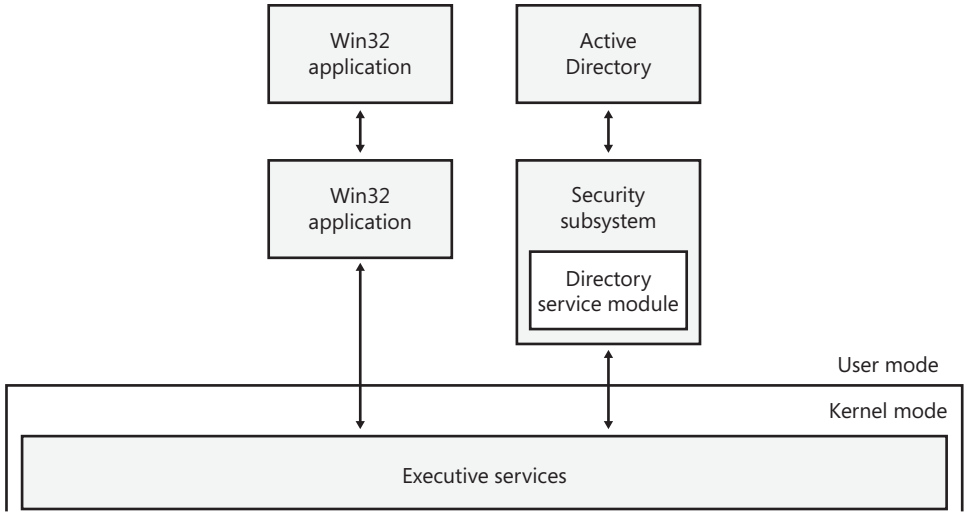


Figure 29-1 Top-level overview of Active Directory architecture.

> **Note**
>
> Being part of the security subsystem makes Active Directory an integrated part of the access control and authentication mechanism built into Windows Server 2008. Access control and authentication protect the resources in the directory.

Each resource in Active Directory is represented as an object. Anyone who tries to gain access to an object must be granted permission. Lists of permissions that describe who or what can access an object are referred to as access control lists (ACLs). Each object in the directory has an associated ACL.

You can restrict permissions across a broader scope by using Group Policy. The security infrastructure of Active Directory uses policy to enforce security models on several objects that are grouped logically. Trust relationships between groups of objects can also be set up to allow for an even broader scope for security controls between trusted groups of objects that need to interact. From a top-level perspective, that's how Active Directory works, but to really understand Active Directory, you need to delve into the security subsystem.

Active Directory Within the Local Security Authority

Within the security subsystem, Active Directory is a subcomponent of the Local Security Authority (LSA). As shown in Figure 29-2, the LSA consists of many components that provide the security features of Windows Server 2008 and ensure that access

control and authentication function as they should. Not only does the LSA manage local security policy, it also performs the following functions:

- Generates security identifiers
- Provides the interactive process for logon
- Manages auditing

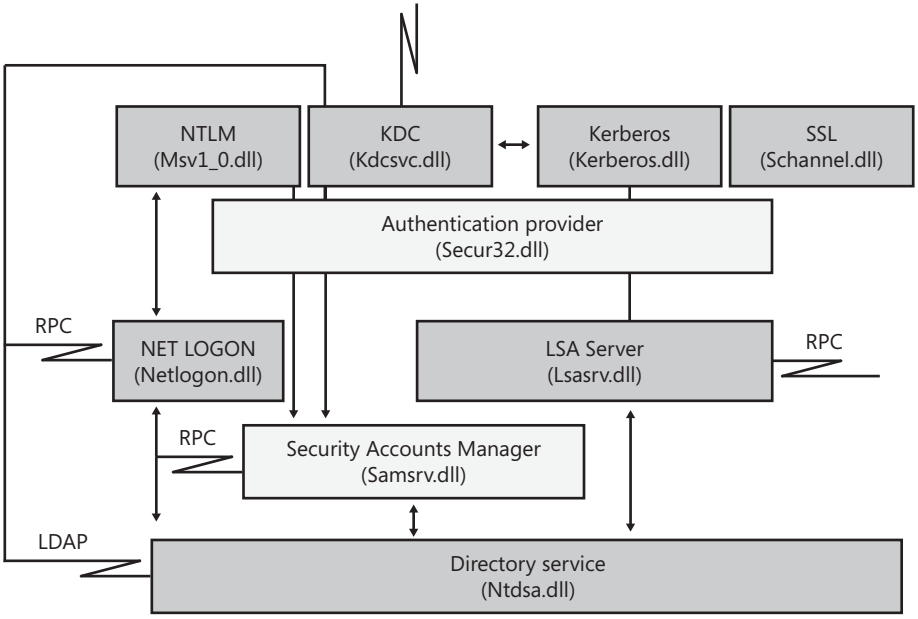


Figure 29-2 Windows Server 2008 security subsystem using Active Directory.

When you work through the security subsystem as it is used with Active Directory, you'll find the three following key areas:

- Authentication mechanisms
  - NTLM (Msv1_0.dll) used for Windows NT LAN Manager (NTLM) authentication
  - Kerberos (Kerberos.dll) and Key Distribution Center (Kdcsvc.dll) used for Kerberos V5 authentication
  - SSL (Schannel.dll) used for Secure Sockets Layer (SSL) authentication
  - Authentication provider (Secur32.dll) used to manage authentication
- Logon/access control mechanisms
  - NET LOGON (Netlogon.dll) used for interactive logon via NTLM. For NTLM authentication, NET LOGON passes logon credentials to the directory service module and returns the security identifiers for objects to clients making requests.

- LSA Server (Lsasrv.dll) used to enforce security policies for Kerberos and SSL. For Kerberos and SSL authentication, LSA Server passes logon credentials to the directory service module and returns the security identifiers for objects to clients making requests.
- Security Accounts Manager (Samsrv.dll) used to enforce security policies for NTLM.

- Directory service component
  - Directory service (Ntdsa.dll) used to provide directory services for Windows Server 2008. This is the actual module that allows you to perform authenticated searches and retrieval of information.

As you can see, users are authenticated before they can work with the directory service component. Authentication is handled by passing a user's security credentials to a domain controller. After they are authenticated on the network, users can work with resources and perform actions according to the permissions and rights they have been granted in the directory. At least, this is how the Windows Server 2008 security subsystem works with Active Directory.

When you are on a network that doesn't use Active Directory or when you log on locally to a machine other than a domain controller, the security subsystem works as shown in Figure 29-3. Here, the directory service is not used. Instead, authentication and access control are handled through the Security Accounts Manager (SAM). This is, in fact, the model used for authentication and access control in Microsoft Windows NT 4. In this model, information about resources is stored in the SAM, which itself is stored in the Registry.

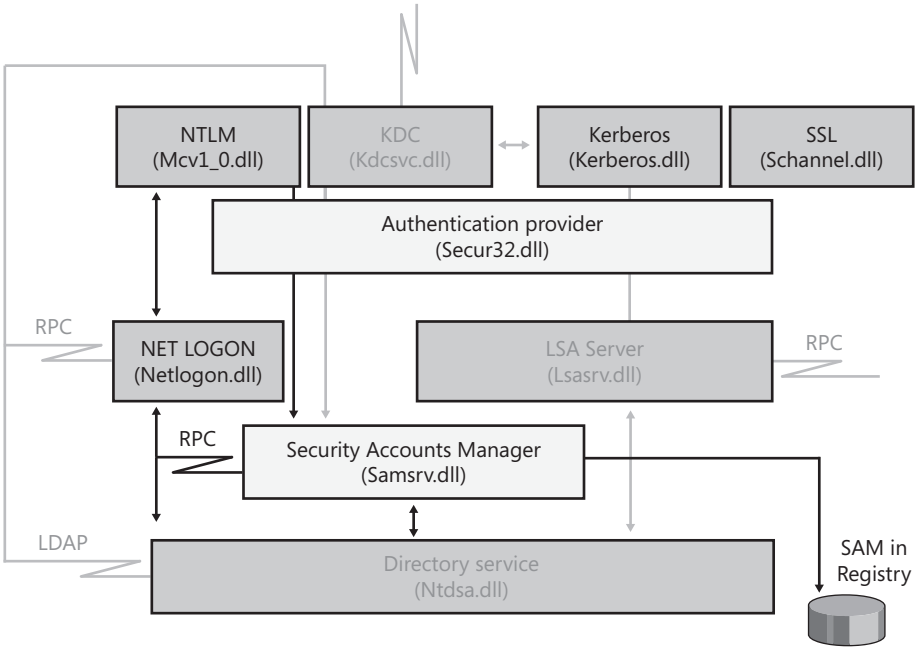


Figure 29-3 Windows Server 2008 security subsystem without Active Directory.

Directory Service Architecture

As you've seen, incoming requests are passed through the security subsystem to the directory service component. The directory service component is designed to accept requests from many different kinds of clients. As shown in Figure 29-4, these clients use specific protocols to interact with Active Directory.

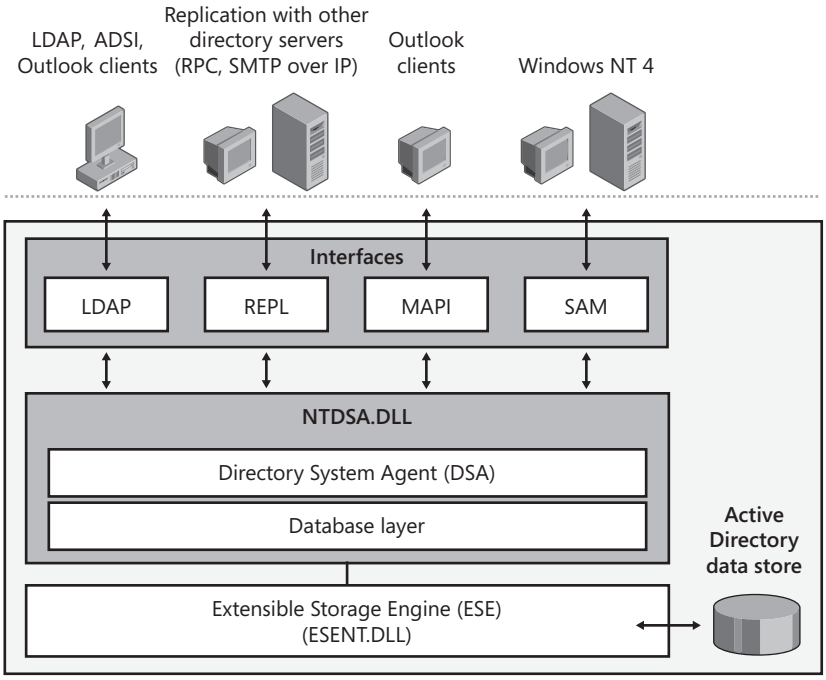


Figure 29-4 The directory service architecture.

Protocols and Client Interfaces

The primary protocol for Active Directory access is Lightweight Directory Access Protocol (LDAP). LDAP is an industry-standard protocol for directory access that runs over TCP/IP. Active Directory supports LDAP versions 2 and 3. Clients can use LDAP to query and manage directory information, depending on the level of permissions they have been granted, by establishing a TCP connection to a domain controller running the directory service. The default TCP port used by LDAP clients is 389 for standard communications and 636 for SSL.

Active Directory supports intersite and intrasite replication through the REPL interface, which uses either Remote Procedure Calls (RPCs) or Simple Mail Transport Protocol over Internet Protocol (SMTP over IP), depending on how replication is configured. Each domain controller is responsible for replicating changes to the directory to other domain controllers, using a multimaster approach. Unlike Windows NT 4, which used a single primary domain controller and one or more backup domain controllers,

the multimaster approach used in Active Directory allows updates to be made to the directory, via any domain controller, and then replicated to other domain controllers. For Windows Server 2008, the algorithms used for replication have been improved to reduce the performance impact on domain controllers and improve the overall replication performance.

For older messaging clients, Active Directory supports the Messaging Application Programming Interface (MAPI). MAPI allows messaging clients to access Active Directory (which is used by Microsoft Exchange for storing information), primarily for address book lookups. Messaging clients use Remote Procedure Calls (RPCs) to establish connection with the directory service. UDP port 135 and TCP port 135 are used by the RPC Endpoint Mapper. Current messaging clients use LDAP instead of RPC.

For clients running Windows NT 4, Active Directory supports the Security Accounts Manager (SAM) interface, which also uses RPCs. This allows Windows NT 4 clients to access the Active Directory data store the same way they would access the SAM database. The SAM interface is also used during replication with Windows NT 4 backup domain controllers.

Directory System Agent and Database Layer

Clients and other servers use the LDAP, REPL, MAPI, and SAM interfaces to communicate with the directory service component (Ntdsa.dll) on a domain controller. From an abstract perspective, the directory service component consists of the following:

- Directory System Agent (DSA), which provides the interfaces through which clients and other servers connect
- Database Layer, which provides an Application Programming Interface (API) for working with the Active Directory data store

From a physical perspective, the DSA is really the directory service component, and the database layer resides within it. The reason for separating the two is that the database layer performs a vital abstraction. Without this abstraction, the physical database on the disk would not be protected from the applications the DSA interacts with. Furthermore, the object-based hierarchy used by Active Directory would not be possible. Why? Because the data store is in a single data file using a flat (record-based) structure, while the database layer is used to represent the flat file records as objects within a hierarchy of containers. Like a folder that can contain files as well as other folders, a container is simply a type of object that can contain other objects as well as other containers.

Each object in the data store has a name relative to the container in which it is stored. This name is aptly called the object's relative distinguished name (RDN). An object's full name, also referred to as an object's distinguished name (DN), describes the series of logical containers, from the highest to the lowest, of which the object is a part.

To make sure every object stored in Active Directory is truly unique, each object also has a globally unique identifier (GUID), which is generated when the object is created. Unlike an object's RDN or DN, which can be changed by renaming an object or moving it to another container, the GUID can never be changed. It is assigned to an object by the DSA and it never changes.

The DSA is responsible for ensuring that the type of information associated with an object adheres to a specific set of rules. This set of rules is referred to as the *schema*. The schema is stored in the directory and contains the definitions of all object classes and describes their attributes. In Active Directory, the schema is the set of rules that determine the kind of data that can be stored in the database, the type of information that can be associated with a particular object, the naming conventions for objects, and so on.

INSIDE OUT

The schema saves space and helps validate attributes

The schema serves to separate an object's definition from its actual values. Thanks to the schema, Active Directory doesn't have to write information about all of an object's possible attributes when it creates the object. When you create an object, only the defined attributes are stored in the object's record. This saves a lot of space in the database. Furthermore, as the schema not only specifies the valid attributes but also the valid values for those attributes, Active Directory uses the schema both to validate the attributes that have been set on an object and to keep track of what other possible attributes are available.

The DSA is also responsible for enforcing security limitations. It does this by reading the security identifiers (SIDs) on a client's access token and comparing it with that of the SID for an object. If a client has appropriate access permissions, it is granted access to an object. If a client doesn't have appropriate access permissions, it is denied access.

Finally, the DSA is used to initiate replication. Replication is the essential functionality that ensures that the information stored on domain controllers is accurate and consistent with changes that have been made. Without proper replication, the data on servers would become stale and outdated.

Extensible Storage Engine

The Extensible Storage Engine (ESE) is used by Active Directory to retrieve information from and write information to the data store. The ESE uses indexed and sequential storage with transactional processing, as follows:

- **Indexed storage** Indexing the data store allows the ESE to access data quickly without having to search the entire database. In this way, the ESE can rapidly retrieve, write, and update data.
- **Sequential storage** Sequentially storing data means that the ESE writes data as a stream of bits and bytes. This allows data to be read from and written to specific locations.
- **Transactional processing** Transactional processing ensures that changes to the database are applied as discrete operations that can be rolled back if necessary.

Any data that is modified in a transaction is copied to a temporary database file. This gives two views of the data that is being changed: one view for the process changing the data and one view of the original data that is available to other processes until the transaction is finalized. A transaction remains open as long as changes are being processed. If an error occurs during processing, the transaction can be rolled back to return the object being modified to its original state. If Active Directory finishes processing changes without errors occurring, the transaction can be committed.

As with most databases that use transactional processing, Active Directory maintains a transaction log. A record of the transaction is written first to an in-memory copy of an object, then to the transaction log, and finally to the database. The in-memory copy of an object is stored in the version store. The version store is an area of physical memory (RAM) used for processing changes. If a domain controller has 400 megabytes (MB) of RAM or more, the version store is 100 MB. If a domain controller has less than 400 MB of RAM, the version store is 25 percent of the physical RAM.

The transaction log serves as a record of all changes that have yet to be committed to the database file. The transaction is written first to the transaction log to ensure that even if the database shuts down immediately afterward, the change is not lost and can take effect. To ensure this, Active Directory uses a checkpoint file to track the point up to which transactions in the log file have been committed to the database file. After a transaction is committed to the database file, it can be cleared out of the transaction log.

The actual update of the database is written from the in-memory copy of the object in the version store and not from the transaction log. This reduces the number of disk I/O operations and helps ensure that updates can keep pace with changes. When many updates are made, however, the version store can reach a point where it is overwhelmed. This happens when the version store reaches 90 percent of its maximum size. When this happens, the ESE temporarily stops processing cleanup operations that are used to return space after an object is modified or deleted from the database.

Because changes need to be replicated from one domain controller to another, an object that is deleted from the database isn't fully removed. Instead, most of the object's attributes are removed and the object's Deleted attribute is set to TRUE to indicate that it has been deleted. The object is then moved to a hidden Deleted Objects container where its deletion can be replicated to other domain controllers. In this state, the object is said to be *tombstoned*. To allow the tombstoned state to be replicated to all domain controllers, and thus removed from all copies of the database, an attribute called tombstoneLifetime is also set on the object. The tombstoneLifetime attribute specifies how long the tombstoned object should remain in the Deleted Objects container. The default lifetime is 180 days.

The ESE uses a garbage-collection process to clear out tombstoned objects after the tombstone lifetime has expired and performs automatic online defragmentation of the database after garbage collection. The interval at which garbage collection occurs is a factor of the value set for the garbageCollPeriod attribute and the tombstone lifetime. By default, garbage collection occurs every 12 hours. When there are more than 5,000 tombstoned objects to be garbage-collected, the ESE removes the first 5,000 tombstoned objects, and then uses the CPU availability to determine if garbage collection

can continue. If no other process is waiting for the CPU, garbage collection continues for up to the next 5,000 tombstoned objects whose tombstone lifetime has expired and the CPU availability is again checked to determine if garbage collection can continue. This process continues until all the tombstoned objects whose tombstone lifetime has expired are deleted or another process needs access to the CPU.

Data Store Architecture

After you have examined the operating system components that support Active Directory, the next step is to see how directory data is stored on a domain controller's hard disks. As Figure 29-5 shows, the data store has a primary data file and several other types of related files, including working files and transaction logs.

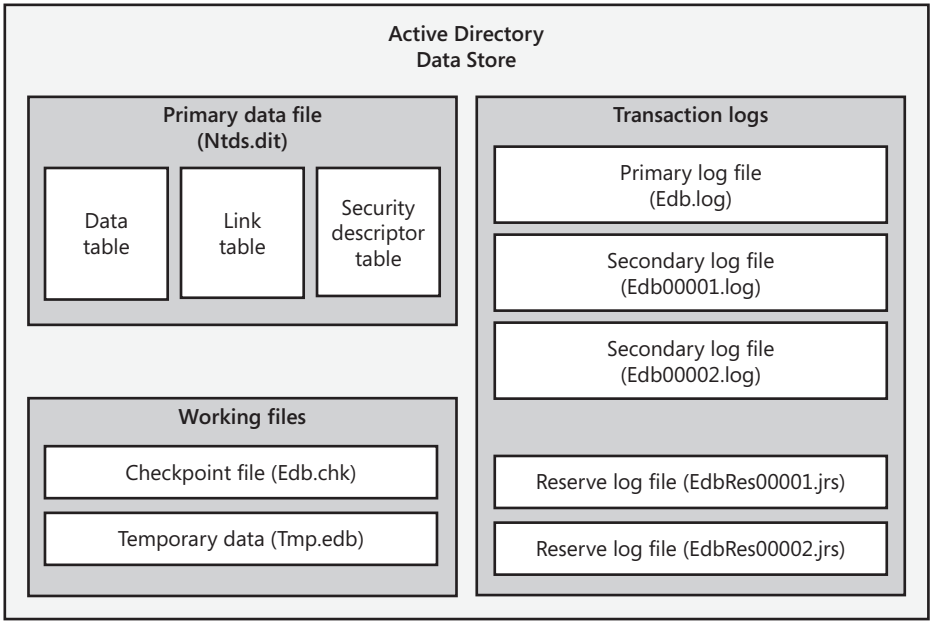


Figure 29-5 The Active Directory data store.

These files are used as follows:

- **Primary data file (Ntds.dit)** Physical database file that holds the contents of the Active Directory data store
- **Checkpoint file (Edb.chk)** Checkpoint file that tracks the point up to which the transactions in the log file have been committed to the database file
- **Temporary data (Tmp.edb)** Temporary workspace for processing transactions
- **Primary log file (Edb.log)** Primary log file that contains a record of all changes that have yet to be committed to the database file

- **Secondary log files (Edb00001.log, Edb00002.log, ...)** Additional logs files that are used as needed
- **Reserve log files (EdbRes00001.jrs, EdbRes00002.jrs, ...)** Files that are used to reserve space for additional log files if the primary log file becomes full

The primary data file contains three indexed tables:

- **Active Directory data table** The data table contains a record for each object in the data store, which can include object containers, the objects themselves, and any other type of data that is stored in Active Directory.
- **Active Directory link table** The link table is used to represent linked attributes. A linked attribute is an attribute that refers to other objects in Active Directory. For example, if an object contains other objects (that is, it is a container), attribute links are used to point to the objects in the container.
- **Active Directory security descriptor table** The security descriptor table contains the inherited security descriptors for each object in the data store. Windows Server 2008 uses this table so that inherited security descriptors no longer have to be duplicated on each object. Instead, inherited security descriptors are stored in this table and linked to the appropriate objects. This makes Active Directory authentication and control mechanisms much more efficient than they were in Microsoft Windows 2000.

Think of the data table as having rows and columns; the intersection of a row and a column is a field. The table's rows correspond to individual instances of an object. The table's columns correspond to attributes defined in the schema. The table's fields are populated only if an attribute contains a value. Fields can be a fixed or a variable length. If you create an object and define only 10 attributes, only these 10 attributes will contain values. Although some of those values might be fixed length, other might be variable length.

Records in the data table are stored in data pages that have a fixed size of 8 kilobytes (KB, or 8,192 bytes). Each data page has a page header, data rows, and free space that can contain row offsets. The page header uses the first 96 bytes of each page, leaving 8,096 bytes for data and row offsets. Row offsets indicate the logical order of rows on a page, which means that offset 0 refers to the first row in the index, offset 1 refers to the second row, and so on. If a row contains long, variable-length data, the data may not be stored with the rest of the data for that row. Instead, Active Directory can store an 8-byte pointer to the actual data, which is stored in a collection of 8-KB pages that aren't necessarily written contiguously. In this way, an object and all its attribute values can be much larger than 8 KB.

The primary log file has a fixed size of 10 MB. When this log fills up, Active Directory creates additional (secondary) log files as necessary. The secondary log files are also limited to a fixed size of 10 MB. Active Directory uses the reserve log files to reserve space on disk for log files that may need to be created. As several reserve files are already created, this speeds up the transactional logging process when additional logs are needed.

By default, the primary data file, working files, and transaction logs are all stored in the same location. On a domain controller's system volume, you'll find these files in the %SystemRoot%\NTDS folder. Although these are the only files used for the data store, there are other files used by Active Directory. For example, policy files and other files, such as startup and shutdown scripts used by the DSA, are stored in the %SystemRoot%\Sysvol folder.

Note

A distribution copy of Ntds.dit is also placed in the %SystemRoot%\System32 folder. This is used to create a domain controller when you install Active Directory on a server running Windows Server 2008. If the file doesn't exist, the Active Directory Installation Wizard will need the installation CD to promote a member server to be a domain controller.

INSIDE OUT

The log files have attributes you can examine

When you stop Active Directory Domain Services, you can use the Extensible Storage Engine Utility (esentutl.exe) to examine log file properties. At an elevated command prompt, enter **esentutl.exe -ml *LogName*** where *LogName* is the name of the log file to examine, such as edb.log, to obtain detailed information on the log file, including base name, creation time, format version, log sector sizes, and logging parameters. While Active Directory Domain Services is offline, you can also use esentutl.exe to perform defragmentation, integrity checks, copy, repair, and recovery operations. To learn more about this utility, enter **esentutl.exe** at an elevated command prompt. Following the prompts, you can then enter the letter corresponding to the operation you want to learn more about. For example, enter **esentutl.exe** and then press the D key to learn the defragmentation options.

Active Directory Logical Architecture

The logical layer of Active Directory determines how you see the information contained in the data store and also controls access to that information. The logical layer does this by defining the namespaces and naming schemes used to access resources stored in the directory. This provides a consistent way to access directory-stored information regardless of type. For example, you can obtain information about a printer resource stored in the directory in much the same way that you can obtain information about a user resource.

To better understand Active Directory's logical architecture, you need to understand the following topics:

- Active Directory objects
- Active Directory domains, trees, and forests
- Active Directory trusts
- Active Directory namespaces and partitions
- Active Directory data distribution

Active Directory Objects

Because so many different types of resources can be stored in the directory, a standard storage mechanism was needed and Microsoft developers decided to use the LDAP model for organizing data. In this model, each resource that you want to represent in the directory is created as an object with attributes that define information you want to store about the resource. For example, the user object in Active Directory has attributes for a user's first name, middle initial, last name, and logon name.

An object that holds other objects is referred to as a *container object* or simply a *container*. The data store itself is a container that contains other containers and objects. An object that can't contain other objects is a *leaf object*. Each object created within the directory is of a particular type or class. The object classes are defined in schema and include the following types:

- User
- Group
- Computer
- Printer
- Organizational unit

When you create an object in the directory, you must comply with the schema rules for that object class. Not only do the schema rules dictate the available attributes for an object class, they also dictate which attributes are mandatory and which attributes are optional. When you create an object, mandatory attributes must be defined. For example, you can't create a user object without specifying the user's full name and logon name. The reason is that these attributes are mandatory.

Some rules for attributes are defined in policy as well. For example, the default security policy for Windows Server 2008 specifies that a user account must have a password and the password must meet certain complexity requirements. If you try to create a user account without a password or with a password that doesn't meet these complexity requirements, the account creation will fail because of the security policy.

The schema can be extended or changed as well. This allows administrators to define new object classes, to add attributes to existing objects, and to change the way attributes are used. However, you need special access permissions and privileges to work directly with the schema.

Active Directory Domains, Trees, and Forests

Within the directory, objects are organized using a hierarchical tree structure called a *directory tree*. The structure of the hierarchy is derived from the schema and is used to define the parent-child relationships of objects stored in the directory.

A logical grouping of objects that allows central management of those objects is called a *domain*. In the directory tree, a domain is itself represented as an object. It is in fact the parent object of all the objects it contains. Unlike Windows NT 4.0, which limited the number of objects you could store in a domain, an Active Directory domain can contain millions of objects. Because of this, you probably do not need to create separate user and resource domains as was done commonly with Windows NT 4.0. Instead, you can create a single domain that contains all the resources you want to manage centrally. In Figure 29-6, a domain object is represented by a large triangle and the objects it contains are as shown.

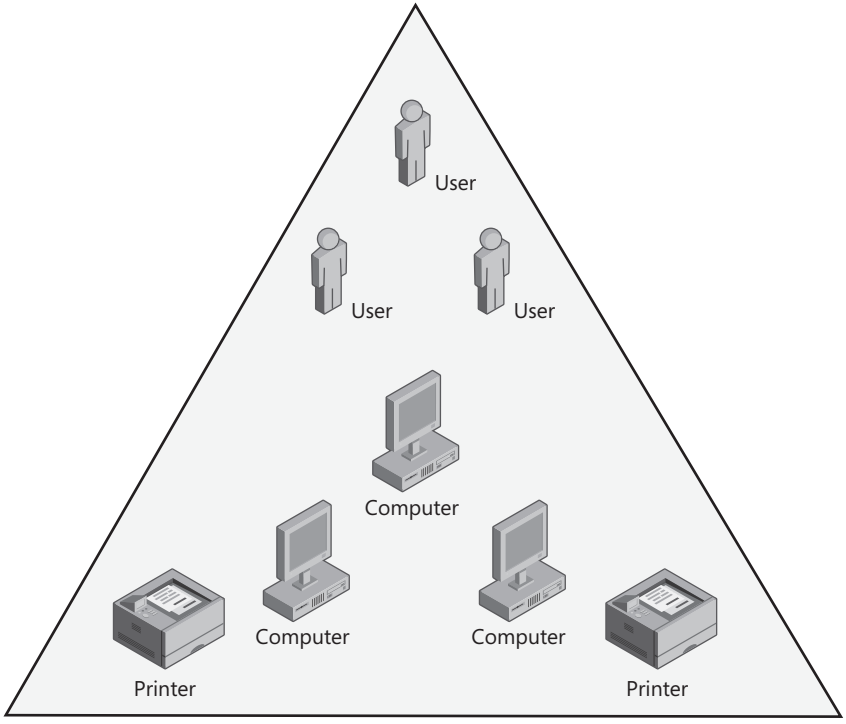


Figure 29-6 An Active Directory domain.

Domains are only one of several building blocks for implementing Active Directory structures. Other building blocks include the following:

- Active Directory trees, which are logical groupings of domains
- Active Directory forests, which are logical groupings of domain trees

As described above, a directory tree is used to represent a hierarchy of objects, showing the parent-child relationships between those objects. Thus, when we're talking about a domain tree, we're looking at the relationship between parent and child domains. The domain at the top of the domain tree is referred to as the *root domain (think of this as an upside-down tree)*. More specifically, the root domain is the first domain created in a new tree within Active Directory. When talking about forests and domains, there is an important distinction made between the first domain created in a new forest—a forest root domain—and the first domain created in each additional tree within a forest—a root domain.

In the example shown in Figure 29-7, cohovineyard.com is the root domain in an Active Directory forest with a single tree, that is, it is the forest root domain. As such, cohovineyard.com is the parent of the sales.cohovineyard.com domain and the mf.cohovineyard.com domain. The mf.cohovineyard.com domain itself has a related subdomain: bottling.mf.cohovineyard.com. This makes mf.cohovineyard.com the parent of the child domain bottling.mf.cohovineyard.com.

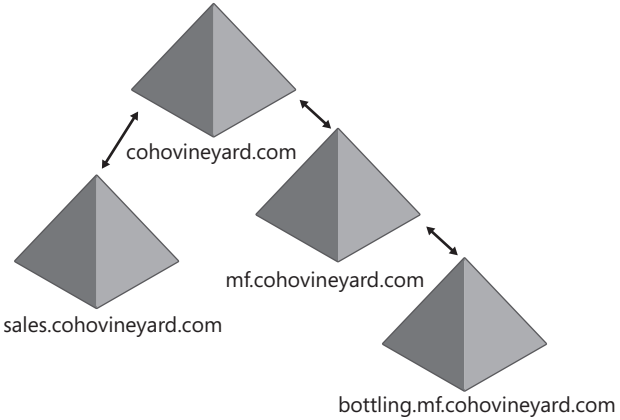


Figure 29-7 An Active Directory forest with a single tree.

The most important thing to note about this and all domain trees is that the namespace is contiguous. Here, all the domains are part of the cohovineyard.com namespace. If a domain is a part of a different namespace, it can be added as part of a new tree in the forest. In the example shown in Figure 29-8, a second tree is added to the forest. The root domain of the second tree is cohowinery.com, and this domain has cs.cohowinery.com as a child domain. The forest root domain does not change; cohovineyard.com remains the forest root domain.

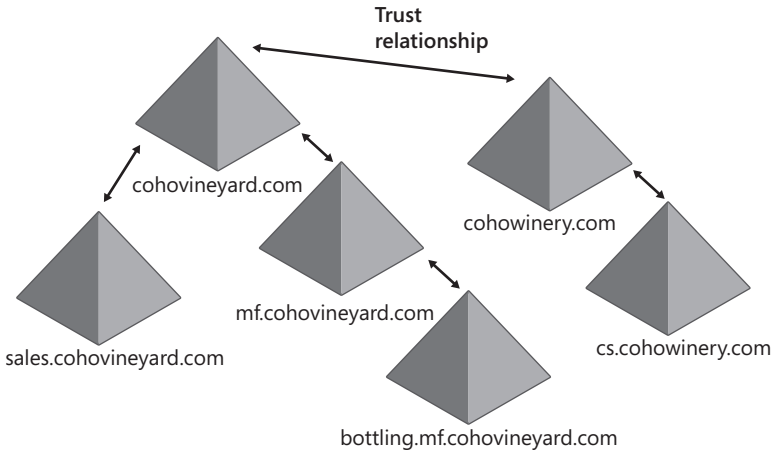


Figure 29-8 An Active Directory forest with multiple trees.

You create a forest root domain by installing Active Directory on a stand-alone server and establishing the server as the first domain controller in a new forest. To add an additional tree to an existing forest, you install Active Directory on a stand-alone server and configure the server as a member of the forest, but with a domain name that is not part of the current namespace being used. You make the new domain part of the same forest to allow associations called trusts to be made between domains that belong to different namespaces.

Active Directory Trusts

In Active Directory, two-way transitive trusts are established automatically between domains that are members of the same forest. Trusts join parent and child domains in the same domain tree and join the roots of domain trees. Because trusts are transitive, this means that if domain A trusts domain B and domain B trusts domain C, domain A trusts domain C as well. As all trusts in Active Directory are two-way and transitive, by default every domain in a forest implicitly trusts every other domain. It also means that resources in any domain are available to users in every domain in the forest. For example, with the trust relationships in place, a user in the sales.cohovineyard.com domain could access a printer or other resources in the cohovineyard.com domain or even the cs.cohowinery.com domain.

However, the creation of a trust doesn't imply any specific permission. Instead, it implies only the ability to grant permissions. No privileges are automatically implied or inherited by the establishment of a trust relationship. The trust doesn't grant or deny any permission. It only exists to allow administrators to be able to grant permissions.

There are several key terms used to describe trusts, including the following:

- **Trusting domain** A domain that establishes a trust is referred to as a trusting domain. Trusting domains allow access by users from another domain (the trusted domain).

- **Trusted domain** A domain that trusts another domain is referred to as a trusted domain. Users in trusted domains have access to another domain (the trusting domain).

To make it easier for administrators to grant access throughout a forest, Active Directory allows you to designate two types of administrators:

- **Enterprise administrators** Enterprise administrators, which are the designated administrators of the enterprise. Enterprise administrators can manage and grant access to resources in any domain in the Active Directory forest.
- **Domain administrators** Domain administrators, which are the designated administrators of a particular domain. Domain administrators in a trusting domain can access user accounts in a trusted domain and set permissions that grant access to resources in the trusting domain.

Going back to the example, an enterprise administrator in this forest could grant access to resources in any domain in the forest. If Jim, in the sales.cohovineyard.com domain, needed access to a printer in the cs.cohowinery.com domain, an enterprise administrator could grant this access. As cs.cohowinery.com is the trusting domain and sales.cohovineyard.com is the trusted domain in this example, a domain administrator in the cs.cohowinery.com could grant permission to use the printer as well. A domain administrator for sales.cohovineyard.com could not grant such permissions, however, as the printer resource exists in a domain other than the one the administrator controls.

To continue working with Figure 29-8, take a look at the arrows that designate the trust relationships. For a user in the sales.cohovineyard.com domain to access a printer in the cs.cohowinery.com domain, the request must pass through the following series of trust relationships:

1. The trust between sales.cohovineyard.com and cohovineyard.com
2. The trust between cohovineyard.com and cohowinery.com
3. The trust between cohowinery.com and cs.cohowinery.com

The *trust path* defines the path that an authentication request must take between the two domains. Here, a domain controller in the user's local domain (sales.cohovineyard.com) would pass the request to a domain controller in the cohovineyard.com domain. This domain controller would in turn pass the request to a domain controller in the cohowinery.com domain. Finally, the request would be passed to a domain controller in the cs.cohowinery.com domain, which would ultimately grant or deny access.

In all, the user's request has to pass through four domain controllers—one for each domain between the user and the resource. Because the domain structure is separate from your network's physical structure, the printer could actually be located right beside the user's desk and the user would still have to go through this process. If you expand this scenario to include all the users in the sales.cohovineyard.com domain, you could potentially have many hundreds of users whose requests have to go through a similar process to access resources in the cs.cohowinery.com domain.

Omitting the fact that the domain design in this scenario is very poor—because if many users are working with resources, those resources are ideally in their own domain or a domain closer in the tree—one solution for this problem would be to establish a shortcut trust between the user's domain and the resource's domain. With a shortcut trust, you could specify that cs.cohowinery.com explicitly trusts sales.cohovineyard.com. Now when a user in the sales.cohovineyard.com requests a resource in the cs.cohowinery.com domain, the local domain controller knows about cs.cohowinery.com and can directly submit the request for authentication. This means that the sales.cohovineyard.com domain controller sends the request directly to a cs.cohowinery.com domain controller.

Shortcut trusts are meant to help make more efficient use of resources on a busy network. On a network with a lot of activity, the explicit trust can reduce the overhead on servers and on the network as a whole. Shortcut trusts shouldn't be implemented without careful planning. They should only be used when resources in one domain will be accessed by users in another domain on a regular basis. They don't need to be used between two domains that have a parent-child relationship, because a default trust already exists explicitly between a parent and a child domain.

With Active Directory, you can also make use of external trusts that work the same they did in Windows NT 4. External trusts are manually configured and are always nontransitive. One of the primary reasons for establishing an external trust is to create a trust between an Active Directory domain and a legacy Windows NT domain. In this way, existing Windows NT domains continue to be available to users while you are implementing Active Directory. For example, you could upgrade your company's main domain from Windows NT 4 to Windows Server 2008, and then create external trusts between any other Windows NT domains. You should create these external trusts as two-way trusts to ensure that users can access resources as their permissions allow.

Active Directory Namespaces and Partitions

Any data stored in the Active Directory database is represented logically as an object. Every object in the directory has a relative distinguished name (RDN). That is, every object has a name relative to the parent container in which it is stored. The relative name is the name of the object itself and is also referred to as an object's *common name*. This relative name is stored as an attribute of the object and must be unique for the container in which it is located. Following this, no two objects in a container can have the same common name, but two objects in different containers could have the same name.

In addition to an RDN, objects also have a distinguished name (DN). An object's DN describes the object's place in the directory tree and is logically the series of containers from the highest to the lowest of which the object is a part. It is called a distinguished name because it serves to distinguish like-named objects and as such must be unique in the directory. No two objects in the directory will have the same distinguished name.

Every object in the directory has a parent, except the root of the directory tree, which is referred to as the rootDSE. The rootDSE represents the top of the logical namespace for a directory. It has no name *per se*. Although there is only one rootDSE, the information stored in the rootDSE specifically relates to the domain controller on which the

directory is stored. In a domain with multiple domain controllers, the rootDSE will have a slightly different representation on each domain controller. The representation relates to the capability and configuration of the domain controller in question. In this way, Active Directory clients can determine the capabilities and configuration of a particular domain controller.

Below the rootDSE, every directory tree has a root domain. The root domain is the first domain created in an Active Directory forest and is also referred to as the forest root domain. After it is established, the forest root domain never changes, even if you add new trees to the forest. The LDAP distinguished name of the forest root domain is: DC=*ForestRootDomainName* where DC is an LDAP identifier for a domain component and *ForestRootDomainName* is the actual name of the forest root domain. Each level within the domain tree is broken out as a separate domain component. For example, if the forest root domain is cohovineyard.com, the domain's distinguished name is DC=cohovineyard,DC=com.

When Active Directory is installed on the first domain controller in a new forest, three containers are created below the rootDSE:

- Forest Root Domain container, which is the container for the objects in the forest root domain
- Configuration container, which is the container for the default configuration and all policy information
- Schema container, which is the container for all objects, classes, attributes, and syntaxes

From a logical perspective, these containers are organized as shown in Figure 29-9. The LDAP identifier for an object's common name is CN. The DN for the Configuration container is CN=configuration,DC=*ForestRootDomainName* and the DN for the Schema container is CN=schema,CN=configuration,DC=*ForestRootDomainName*. In the cohovineyard.com domain, the DNs for the Configuration and Schema containers are CN=configuration,DC=cohovineyard,DC=com and CN=schema,CN=configuration,DC=cohovineyard,DC=com, respectively. As you can see, the distinguished name allows you to walk the directory tree from the relative name of the object you are working with to the forest root.

As shown in the figure, the forest root domain and the Configuration and Schema containers exist within their own individual partitions. Active Directory uses partitions to logically apportion the directory so that each domain controller does not have to store a complete copy of the entire directory. To do this, object names are used to group objects into logical categories so that the objects can be managed and replicated as appropriate. The largest logical category is a directory partition. All directory partitions are created as instances of the domainDNS object class.

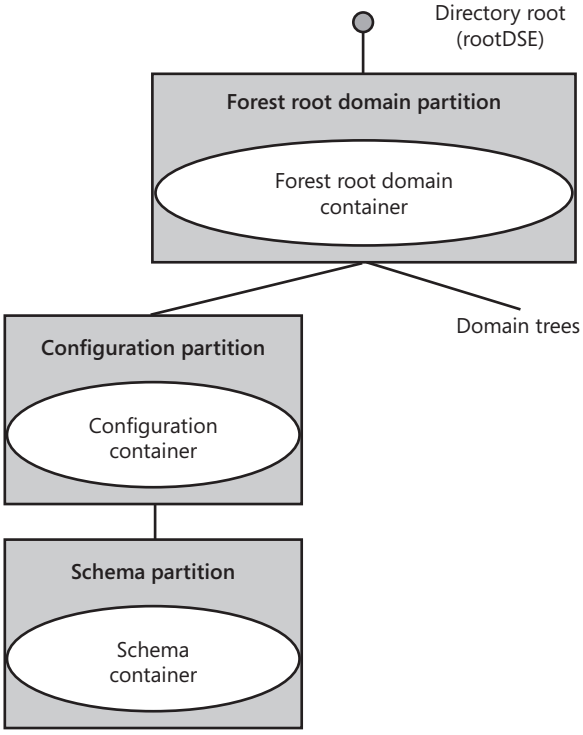


Figure 29-9 The directory tree in a new forest.

As far as Active Directory is concerned, a domain is a container of objects that is logically partitioned from other container objects. When you create a new domain in Active Directory, you create a new container object in the directory tree, and that container is in turn contained by a domain directory partition for the purposes of management and replication.

Active Directory Data Distribution

Active Directory uses partitions to help distribute three general types of data:

- Domain-wide data, which is data replicated to every domain controller in a domain
- Forest-wide data, which is data replicated to every domain controller in a forest
- Application data, which is data replicated to an arbitrary set of domain controllers

Every domain controller stores at least one domain directory partition as well as two forest-wide data partitions: the schema partition and the configuration partition. Data in a domain directory partition is replicated to every domain controller in the domain as a writable replica.

Forest-wide data partitions are replicated to every domain controller in the forest. The configuration partition is replicated as a writable replica. The schema partition is replicated as a read-only replica and the only writable replica is stored on a domain controller that is designated as having the schema operations master role. Other operations master roles are defined as well.

Active Directory can replicate application-specific data that is stored in an application partition such as the default application partitions used with zones in Domain Name System (DNS) that are integrated with Active Directory. Application partition data is replicated on a forest-wide, domain-wide, or other basis to domain controllers that have a particular application partition. If a domain controller doesn't have an application partition, it doesn't receive a replica of the application partition.

> **Note**
>
> Application partitions can be created on domain controllers running only Windows Server 2003 and later. Domain controllers running Windows 2000 or earlier versions of Windows do not recognize application partitions.

In addition to full replicas that are distributed for domains, Active Directory distributes partial replicas of every domain in the forest to special domain controllers designated as global catalog servers. The partial replicas stored on global catalog servers contain information on every object in the forest and are used to facilitate searches and queries for objects in the forest. Because only a subset of an object's attributes is stored, the amount of data replicated to and maintained by a global catalog server is significantly smaller than the total size of all object data stored in all the domains in the forest.

Every domain must have at least one global catalog server. By default, the first domain controller installed in a domain is set as that domain's global catalog server. You can change the global catalog server, and you can designate additional servers as global catalog servers as necessary.