



realtimepublishers.com™

Tips and Tricks Guide™ To

Securing .NET Server



Roberta Bragg

Note to Reader: This book presents tips and tricks for eight Windows .NET Server security topics. For ease of use, the questions and their solutions are divided into chapters based on topic, and each question is numbered based on the chapter, including:

- Chapter 1: Understanding and Utilizing PKI in .NET
- Chapter 2: Securing Web Services and Web Servers—the Administrative Perspective
- Chapter 3: Understanding Active Directory Foundations
- Chapter 4: Fulfilling the Promises of Group Policy
- Chapter 5: Administrative Authority
- Chapter 6: Triple A’s—Authentication, Authorization, and Audit
- Chapter 7: Remote Access
- Chapter 8: Security Tools, Mechanisms, and Emerging Issues.

Chapter 1: Understanding and Utilizing PKI in .NET	1
Q 1.4: What purpose does a certificate revocation list serve?	1
Certificate Revocation	2
Certificate List Revocation Publishing	6
Introduction to Delta CRLs.....	7
Delta CRL Implementation.....	8
Chapter 2: Securing Web Services and Web Servers—the Administrative Perspective.....	9
Q 2.4: The use of Internet-downloadable components has been a major security issue. As we move to Windows .NET, how can we prevent malicious code from running on our systems?	9
Code Groups	11
Named Permission Sets.....	12
Creating Custom Permission Sets.....	13
Policy Assemblies.....	16
Security Policy	17
The Microsoft .NET Framework Configuration Tool	18
Process	19
Chapter 3: Understanding Active Directory Foundations	21
Q 3.4: What possible advantage is there to implementing universal groups?	21
Win2K Mixed Domain Functional Level Group Management Practice	21
Win2K Native or Windows .NET Domain Functional Level and Universal Groups	24
Chapter 4: Fulfilling the Promises of Group Policy	25
Q. 4.4: How can I ensure that the proper system file and registry permissions are maintained across all Windows .NET servers in my domain?	25

Determining Appropriate Files and Registry Permission Settings	26
Using Group Policy to Maintain File and Registry Permission Settings.....	27
Using Secedit to Maintain File and Registry Permission Settings	30
Chapter 5: Administrative Authority	34
Q. 5.4: Our policy is to let users in sensitive departments maintain the backups for servers within their departments. To allow users in the Accounting department to back up their servers, I created a global group for the users and a local group on each server. The local group is assigned the right to back up files and directories in the local security policy of each server, but these users cannot back up the servers. What is wrong?.....	34
Location, Location, Location.....	34
Chapter 6: Triple A's—Authentication, Authorization, and Audit	37
Q. 6.4: What is the difference between user rights, permissions, and privileges?.....	37
Logon Rights.....	39
Privileges.....	40
Chapter 7: Remote Access	42
Q. 7.4: I want to setup dial-up remote access and a virtual private network connection, but I don't want to configure Active Directory. Is this setup possible?	42
Understanding the Difference Between Authentication and Authorization	42
Choosing an Authentication Provider	43
Authentication Methods.....	45
MS-CHAP and MS-CHAPv2	47
CHAP	48
SPAP	48
PAP	48
Data Encryption	50
Remote Access Policy.....	51
Policy Attributes	52
Account Lockout.....	54
Installation and Configuration	54
Chapter 8: Security Tools, Mechanisms, and Emerging Issues.....	54
Q. 8.4: I'm always a little leery of using secedit to apply a security template. What if the system is not working the way I want it to after I apply the template?	54

Copyright Statement

© 2002 Realtimepublishers.com, Inc. All rights reserved. This site contains materials that have been created, developed, or commissioned by, and published with the permission of, Realtimepublishers.com, Inc. (the "Materials") and this site and any such Materials are protected by international copyright and trademark laws.

THE MATERIALS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. The Materials are subject to change without notice and do not represent a commitment on the part of Realtimepublishers.com, Inc or its web site sponsors. In no event shall Realtimepublishers.com, Inc. or its web site sponsors be held liable for technical or editorial errors or omissions contained in the Materials, including without limitation, for any direct, indirect, incidental, special, exemplary or consequential damages whatsoever resulting from the use of any information contained in the Materials.

The Materials (including but not limited to the text, images, audio, and/or video) may not be copied, reproduced, republished, uploaded, posted, transmitted, or distributed in any way, in whole or in part, except that one copy may be downloaded for your personal, non-commercial use on a single computer. In connection with such use, you may not modify or obscure any copyright or other proprietary notice.

The Materials may contain trademarks, services marks and logos that are the property of third parties. You are not permitted to use these trademarks, services marks or logos without prior written consent of such third parties.

If you have any questions about these terms, or if you would like information about licensing materials from Realtimepublishers.com, please contact us via e-mail at info@realtimepublishers.com.

Chapter 1: Understanding and Utilizing PKI in .NET

Q 1.4: What purpose does a certificate revocation list serve?

A: A certificate revocation list (CRL), pronounced krill, is a signed list of certificates that have been revoked. A Windows .NET Certification Authority (CA) assigns an expiration date to every certificate that it issues. The application using the certificate should be programmed to check the date and not allow the use of an expired certificate. For example, a certificate presented for authentication, if expired, will not be accepted. In this way, the periodic need to obtain a new certificate ensures that certificates do not remain valid and useful forever. You can imagine the chaos, and potential for intrusion, if authentication certificates for employees who left the company several years ago were still valid.

How should you handle certificates that are known to be compromised, or those certificates belonging to employees who just quit today? You would hardly want to wait until these certificates expire for them to be invalid. For situations such as these, CA administrators have the ability to revoke certificates.

But how do you discover that a certificate has been revoked? Asking every application to query the CA would be inefficient, if not unworkable. Attempting to somehow round up every copy of a certificate and mark it revoked is equally ridiculous. Enter the CRL. The CRL provides a list of revoked certificates. When a certificate is presented, its CRL is checked to ensure that the certificate has not been revoked. To assist in this process, each certificate can indicate the location of its CA CRL. A Windows 2000 (Win2K) or Windows .NET CRL can be located on a Web site, FTP site, within a shared folder, or present in Active Directory (AD). Default locations can be found by examining the CRL Distribution Point (CDP) properties of the CA, as Figure 1.14 shows. If necessary, the default locations can be changed and/or locations can be added. Should an application need to check the revocation status of a particular certificate, it can query the location, download the list, and check it.

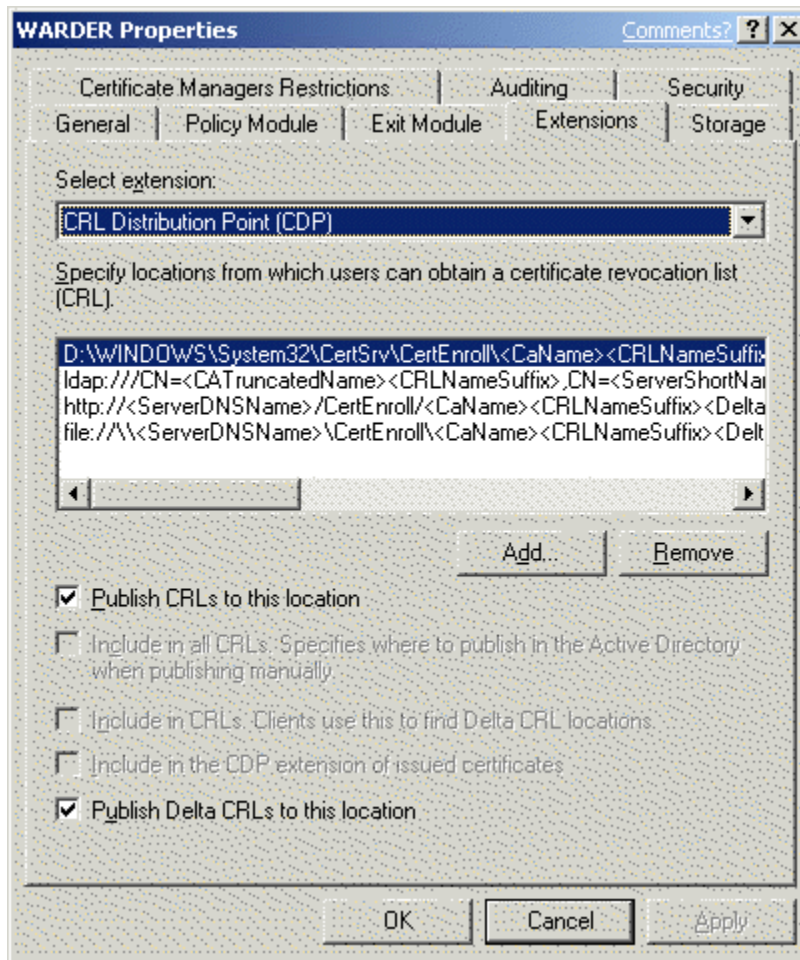


Figure 1.14: The CDP lists the locations at which CRLs are published. These locations will be included in the certificates issued by this CA.

Although this arrangement sounds perfect, several revocation and CRL considerations exist:

- When should a certificate be revoked and for what reason?
- Can a revoked certificate be “un-revoked”?
- The CRL, once downloaded, is cached at the client. The CA periodically publishes an up-to-date CRL, but the client will not download this new list immediately. Thus, the information about revoked certificates can be out of date. How do you deal with this situation?
- As the number of issued certificates increases, the CRL grows. At some point, such a list might become rather cumbersome. How does Windows .NET’s Delta CRL manage this problem?

Certificate Revocation

You can manually revoke certificates by right-clicking a valid certificate in the CA Issued Certificates container. To revoke a certificate, a CA administrator must provide a reason for the revocation (see Figure 1.15).



Figure 1.15: A reason must be selected when revoking a certificate.

The following list describes acceptable reasons for revocation:

- **Key Compromise**—The location of the private key of the certificate has been compromised. The private key is necessary for digital signature and for decrypting data encrypted with its paired public key. If the storage area of the private key is now in the possession of unauthorized parties, the certificate should be revoked immediately.
- **CA Compromise**—The disk location, or token, at which the CA’s private key is stored has been compromised. Unauthorized individuals are in possession of the private key. Revoking the CA certificate means that all certificates signed by this private key are considered to be revoked. The rationale behind this action is the ability of the individual in possession of the CA to issue unauthorized certificates.
- **Affiliation Changed**—When an employee is fired or resigns, any certificates that the employee held should be revoked. In some institutions, the affiliation change might mean that the individual transferred to another location or to another department. Your security policy will dictate whether this revocation is required.
- **Superseded**—A new certificate is issued for the user before the assigned certificate expires. Perhaps the individual’s smart card does not work, the user has changed his or her name, or the user has forgotten the pin or other identification token required to use the smart card. Another reason for using this revocation is that a change has been made to the certificate template (for example, the re-issuing of Encryption File Certificates after the template has been redesigned to include key archival).
- **Cessation of Operations**—When a CA is decommissioned, you provide this reason for revoking the certificate. If a CA is no longer issuing certificates but is still publishing a CRL, the CA certificate should NOT be revoked.
- **Certificate Hold**—When the reason for revocation is not clear or there is a chance that the certificate should not be revoked, use this temporary revocation reason. This revocation assignment is the only one that can be reversed.
- **Remove From CRL**—If a revoked certificate is “un-revoked,” it remains in the CRL, but is marked as RemoveFromCRL.
- **Unspecified**—If a certificate needs to be revoked but doesn’t fit the previously named categories, you can classify it as Unspecified. However, unspecified revocations should not be “un-revoked” as there is no audit trail listed about why the certificates were revoked the first time.

When a certificate is revoked, its serial number, the reason for revocation, and the time it was revoked are added to the CRL on the CA. The CRL is periodically published to the locations indicated in the CDP. The information stays on the CRL until at least one publication period after the certificate expires. The published list is signed with the CA certificate. You can view the current CRL by selecting the View CRLs tab of the Revoked Certificates Properties page, which Figure 1.16 shows.

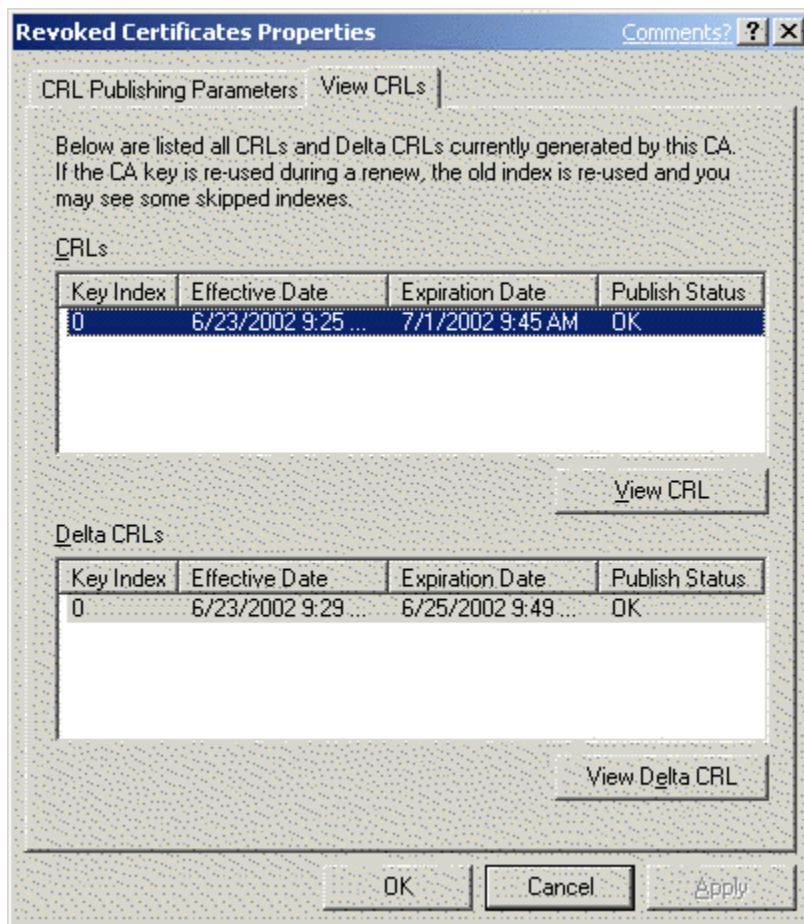


Figure 1.16: The View CRLs tab of the Revoked Certificates Properties page.

On this tab, click View CRL to see a list similar to the one that Figure 1.17 shows.

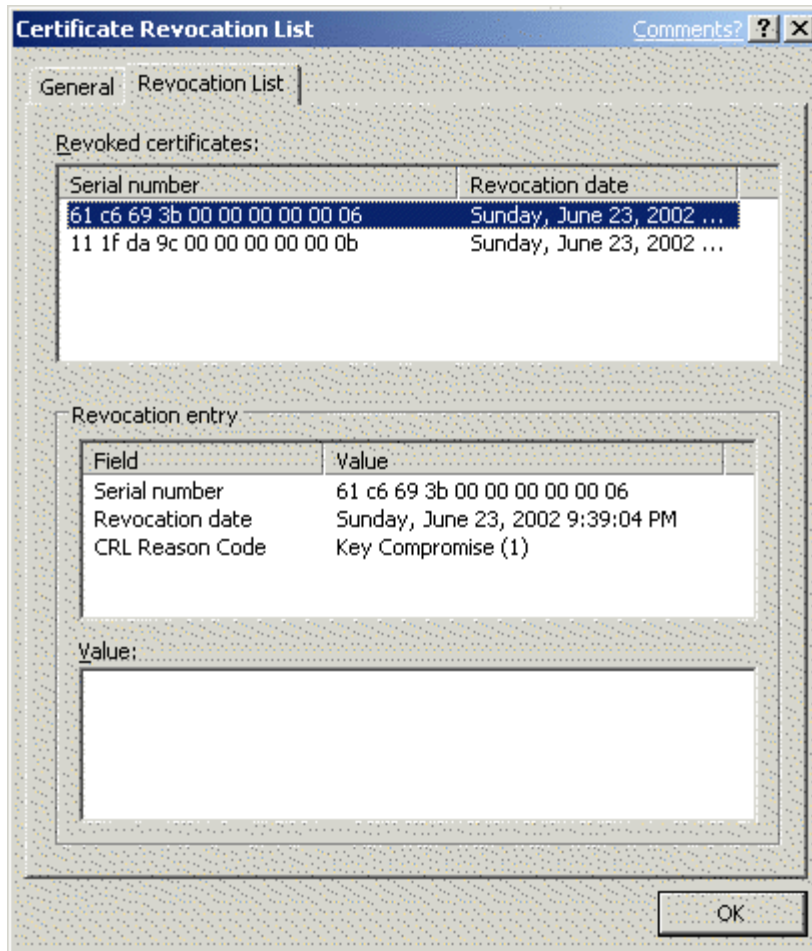


Figure 1.17: The CRL.


When a certificate is presented for validation, it is checked for a CDP extension (the location of a CRL). The CDP will indicate which protocol (LDAP, FILE, HTTP, FTP) is to be used to retrieve the CRL and its location. The CRL will be downloaded and cached, unless a cached, valid version exists. A cached version of the CRL will always be used if it is valid. (To delete a cached CRL, delete all temporary Internet files.)

Some interesting logic is used to determine the validity of the CRL:

- Validate the signature—Win2K and Windows XP can only natively utilize the CRL signed by the same CA that signed the certificate in question. Windows .NET can utilize information from revocation providers such as those using the OCSP, SCVP, and XKMS protocols. This support must be programmatically configured.
- Determine the expiration date of the CRL—This date can be checked by examining the next update field of the CRL. If this date has already passed, the CRL is considered expired.

- Check the CRL for the certificate serial number—If the CRL has not expired and the serial number of the certificate exists on the CRL, the CRL is considered valid and the certificate is considered revoked. If the CRL has expired and the certificate serial number is on the expired CRL, unless the revocation reason is Certificate Hold, the certificate is considered revoked and a new CRL is not downloaded. If the serial number of the certificate is not on the CRL or the reason for revocation is Certificate Hold and the CRL is expired, a new CRL is downloaded and checked for the certificate serial number. (If the reason is Certificate Hold, a new CRL must be checked to see whether the certificate has been removed from hold status.)

If the CRL must be downloaded and can't be found, a server offline error is generated. You should program an application in how to respond to this error. In Win2K and Windows .NET, when a certificate is being used for authentication and the CRL cannot be located, the certificate is not accepted and authentication is denied.

 Each CA in a CA hierarchy generates its own CRL, and each Win2K and Windows .NET CRL will only contain records of its own revoked certificates. Although some certificate models do so, Win2K and Windows .NET don't support any mechanism in which a CRL signed by anything other than the issuing CA is valid. That is, a CA may not report certificate revocation information about another CA. A revoked certificate from the root CA will only be noted on a CRL issued and signed by the root CA, and each subordinate CA will provide its own CRL with information about revoked certificates that it issued.

Certificate List Revocation Publishing


When a CA is installed, the publishing period of the CRL is one week. That is, a new CRL will be published each week. This period can be modified. The validity period of the CRL should always be longer than the publishing period to allow for AD. Because the new CRL might take time before it is available, keeping the old CRL valid beyond the new CRL publication date makes sense. By default, the validity period is extended by 10 percent. You can adjust the publishing period through the CA properties, and you can modify both the publishing period and validity period by editing the registry. The values to modify are HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Certsvc\Configuration\CAName\CRLOverlapPeriod and HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Certsvc\Configuration\CAName\CRLOverlapUnits. The difference can be a maximum of 12 hours, and the time period must be at least 1.5 times the Kerberos skew time as indicated in the Kerberos policy for the domain. Use certutil to make these changes. For example, the following command sets the overlap period to 2 days:

```
Certutil -setreg ca CRLOverlapPeriod = days Certutil -setreg ca
CRLOverlapUnits = 2
```

The administrator might manually publish a new CRL, but this publication won't cause a client to download a new CRL and will not affect the set publishing period.

In addition to the timing issues associated with the latency inherent in the infrequent publishing of CRLs and the time that may be taken to replication the CRL information across the domain, CRLs can become large and further slow processing. A base or full CRL is the complete list of revoked certificates. Over time, this list might become quite large and unwieldy. Although the best practice might require frequent updating of the CRL and frequent downloading of the CRL

to the client, as the list grows, this practice becomes impractical. In a large, distributed AD infrastructure, the CRL must replicate through the structure; the latency involved might make it impossible to appropriately manage frequent publication and distribution of the CRL.

 The CRL can become large (30,000 revoked certificates can equal 1MB). If the publishing period is short, this file being published will add substantially to the replication traffic. If the publishing period is set longer, there is risk of not having an up-to-date CRL. Remember, an administrator can revoke a certificate at any time, but the list is not published unless the administrator manually publishes it or the publishing time is reached. Even if an administrator publishes a list sooner, the client will not download it until its currently cached list has expired.

To help solve this problem, Windows .NET provides the ability to use Delta CRLs.

Introduction to Delta CRLs

A Delta CRL is an interim CRL published between the publication of base CRLs. As I previously mentioned, a base CRL is a list of every revoked certificate. A Delta CRL is a smaller list of just those certificates revoked since the last CRL or the last Delta CRL. (Delta CRLs are defined in Request for Comments—RFC—2459.) When a client needs to download a CRL, the client will automatically get the most up-to-date listing. If the base CRL has expired, the client will download the current base CRL and any Delta CRLs that have been published since the base CRL was published. If the base CRL is still current, the client will only download those Delta CRLs published after the current CRL. Thus, the client needs to download the full CRL at longer intervals, and yet the revocation information can be kept up to date by downloading smaller, Delta CRLs at shorter intervals.

In addition, the client can be forced to download Delta CRLs even though the base CRL has not expired. When the CRL publishing time is reached, a new base CRL is created that includes the revoked certificates published on the interim Delta CRLs. Figure 1.18 shows an example of this arrangement. In this figure, the base CRL is published on Tuesday and is not due for publication again until Friday. A new Delta CRL is scheduled for publication each day. Each day additional certificates are revoked, and each day a new Delta CRL is published. On Friday, a new base CRL is published that includes the revoked certificates published on all Delta CRLs.

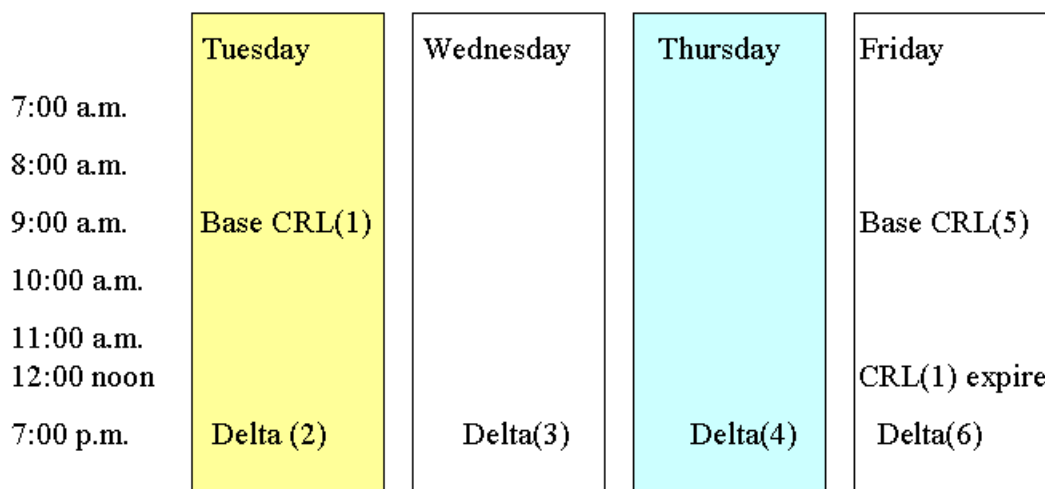



Figure 1.18: Delta CRLs are issued between regular, base CRL, publication.

This model is quite simple and only meant to present the concepts behind the use of Delta CRLs. A number of considerations are used to determine whether a Delta CRL will be downloaded:

- Only Windows XP and Windows .NET Server can utilize Delta CRLs. If Delta CRLs are implemented, the legacy clients will continue to use the base CRLs only. This behavior needs to be taken into consideration in planning the time between base CRL publication.
- The CRL extension CRL Publication Period tells the client the frequency of CRL publication. This setting is used by the client to determine when it should look for the next base or Delta CRL.
- The administrator can implement a freshness policy that defines the maximum permissible age of the base or Delta CRL. A new CRL must be retrieved if the CRL is older than this age. The age is computed as the difference between the current time and the ThisUpdate field of the CRL.
- The base CRL is stored by default in AD. Directory replication latency must be taken into account when planning publication periods.
- Delta CRL publication periods should not be less than the time it takes for AD replication. Otherwise, it might be possible for a retrieved Delta CRL to reference a base CRL that isn't available to the client for download because it hasn't replicated to the domain controller that the client uses.
- Delta CRL publication periods should not be less than the AD replication interval configured for remote sites. Otherwise, because the base CRL of a Delta CRL is indicated in the Delta CRL, it might be possible that a delta CRL would reference a base CRL that was not available to the client in the remote site.
- When a new base CRL is published (either automatically or on demand) it is not immediately downloaded by the client. A new base CRL will be downloaded the next time the client looks for a CRL and finds that it either has no base CRL in its cache or discovers that the base CRL it possesses has expired.

 To determine which base CRL and which Delta CRLs are appropriate at any time, the CRL Number and Delta CRL Indicator fields of the Delta CRL are inspected. The CRL Number is a monotonically increasing number issued in sequence by the CA for each CRL (base and Delta) it publishes. The Delta CRL Indicator identifies the base CRL that can be used with a Delta CRL. By inspecting the CRL Number, the client can know whether it has all the interim Delta CRLs between the base CRL and the highest number Delta CRL in its cache. If the base CRL cached by the client has a CRL Number less than the Delta CRL Indicator of any of its Delta CRLs, the client will attempt to retrieve a new base CRL.

Delta CRL Implementation

To implement Delta CRLs, you configure their publishing schedule. To do so, open Administrative Tools, Certification Authority, right-click Revoked Certificates, and select Properties. On the Revoked Certificates Properties page, which Figure 1.19 shows, select the Publish Delta CRLs check box. Enter the publication interval. Click OK.

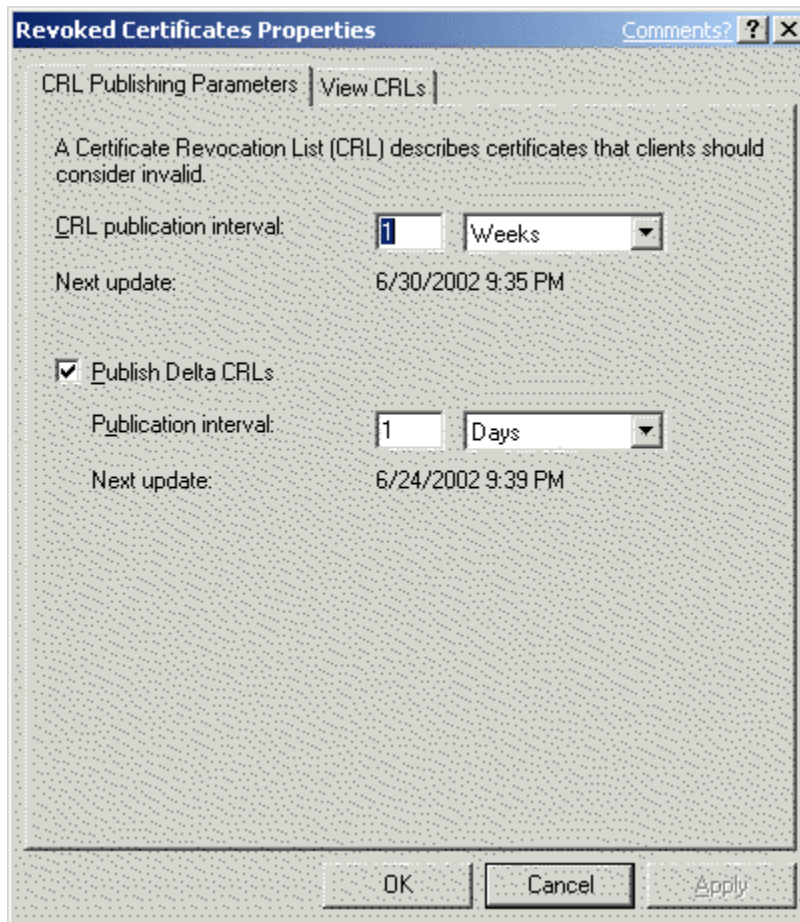


Figure 1.19: Requiring Delta CRL publication.

For more information about CRLs, check out the following Web sites:

http://csrc.nist.gov/pki/documents/sliding_window.pdf

<http://csrc.nist.gov/pki/documents/acsac99.pdf>

<http://www.microsoft.com/technet/treeview/default.asp?url=/TechNet/security/prodtech/dbsql/tshtcrl.asp>

Chapter 2: Securing Web Services and Web Servers—the Administrative Perspective

Q 2.4: The use of Internet-downloadable components has been a major security issue. As we move to Windows .NET, how can we prevent malicious code from running on our systems?

A: Although there is no guarantee that you can prevent malicious code, as part of its security framework, Windows .NET includes two security mechanisms that you can manage administratively. These mechanisms are the use of a role-based security model and the ability to enforce code access rights. Role-based security lets the programmer assign permissions to code

objects according to defined roles. In many cases, each role will correspond to a Windows user group. Administrators can manage who has access to what code by adding user accounts to specific groups that correspond to the role that the users play. You should be aware that a program might ignore Windows groups and create its own internal roles. These roles are then programmatically assigned to users or groups. In this instance, the administrator does not control the membership in the role. Enforcing code access rights allows an administrator grant permissions to code assemblies at runtime. Assemblies, which are units of code, are the building blocks of the Windows .NET framework. Thus, role-based security restricts who can run what code, and code access rights restrict what code can do on a system no matter who runs the code.

Code access security is a flexible process that can be customized for each computer and each user of a computer. It allows the administrator to set the Security Policy for managed code operating on a computer. Security Policy is the set of rules used by the Common Language Runtime (CLR) to determine which type of access is allowed to code. The CLR is the engine that executes the managed code.

The first thing you must realize about the process of enforcing code access rights is that this mechanism only applies to assemblies that consist of well-managed code. If code is not managed, the safeguards that you have designed won't apply. Of course, you do have the option of preventing unmanaged code from running at all.

The second thing is that code access permissions do not supersede the control mechanisms provided by the operating system (OS). For example, permissions assigned to files and folders in NTFS-formatted file systems will prevent code from accessing files, even if the code has been granted full access to the file system. If access to the file system is denied to the code assembly but permitted by NTFS permissions, access is still denied. To determine whether access will be granted, you need to determine the order in which permissions are determined. If the assembly is allowed to run, its code access permissions are determined. At this time, the system does not know which files the assembly will attempt to access, it just knows that the system gives the assembly the right to read and write files. When the assembly actually attempts to open a file for reading or writing, the normal file permissions are applied. Alternatively, an assembly that has been denied file access will not be able to access files; file permissions are never explored.

The third consideration with this model is that it requires the cooperation of both programmer and administrator to make it work. If the administrator does not set the proper access permissions, code will run under the default settings that allow all code all permissions that it requests. If the programmer does not write the code to ask for only the permissions necessary for execution, and the organization demands that the code be run, the administrator will have to relax the security permissions to let the code run. The first step for both programmer and administrator is to gain an understanding of code groups, code access permissions or named permission sets, code security policy, and process.

Code Groups

Code groups are simply a means of organizing code based on some condition. Within the Security Policy, this organization results in a hierarchical structure that has the code group All Code at the top of each Security Policy level. By grouping code, named set of permissions can be more easily applied to a body of code rather than specifying code's application on each individual assembly. At runtime, the CLR examines the identifying characteristics of the code to determine membership in the code group. Administrators configure enterprise, machine, and user policy levels with a hierarchy of code groups, the root of which is a code group that contains all code.

Figure 2.4 shows a snapshot of the .NET Framework Configuration Tool. You can see that the All Code group has representation at the enterprise, machine, and user levels.

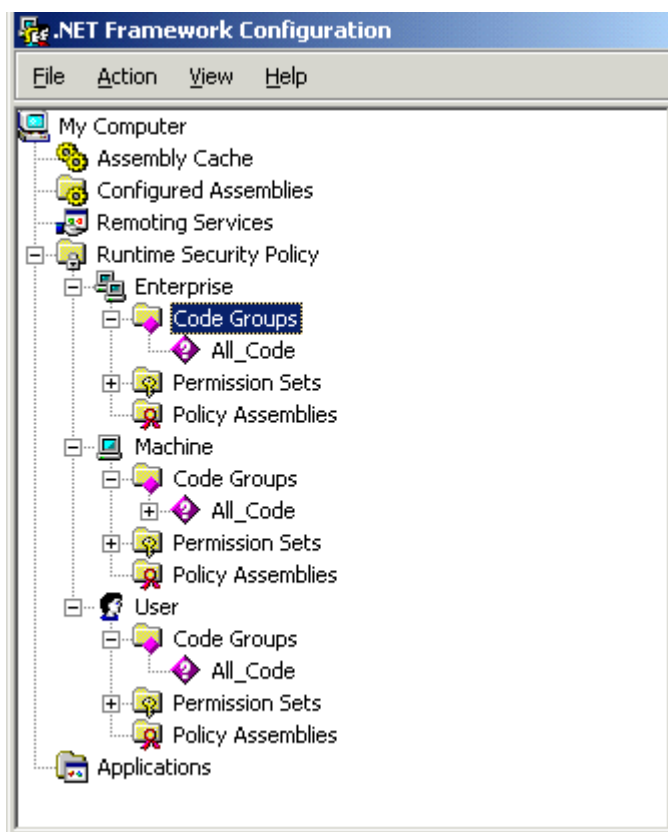


Figure 2.5: By default, each level includes the All Code group.

Code group membership may be determined by examining the evidence. Evidence is just information that the CLR uses to determine membership in a code group and the appropriate Security Policy. Evidence information can be embedded in the assembly (such as a digital signature) and can be examined directly by the CLR or presented to the CLR by a trusted application domain host (such as a URL). An application domain host is the computer that creates the application domain and loads the assembly into it. (A trusted host is one that has been given permission to share this information with the CLR—other types of application domain hosts are defined in Table 2.3. The application domain host knows from where the code originated, has the digital signature of the assembly, and can specify a Security Policy for the

code that runs within it. This policy cannot expand the permission set assigned by enterprise, machine, and user policy, but it can reduce it. The following information is presented as evidence:

- All code (AllMembershipCondition)—All code will always match this condition
- Application directory (ApplicationDirectoryMembershipCondition)—Tells where the application is installed
- Cryptographic hash (HashMembershipCondition)—MD5, SHA1, or other cryptographic hash is presented
- Software publisher (PublisherMembershipConditions)—Public key of a valid Authenticode signature
- Site membership (SiteMembershipCondition)—HTTP, HTTPS, and FTP site from which the code originated
- Strong name (StrongNameMembershipCondition)—Cryptographically strong signature
- URL (UrlMembershipCondition)—Contains the URL from where the code originated
- Zone (ZoneMembershipCondition)—The Internet Explorer (IE) zone from which the code originated

Application Domain Host	Example	Description
Browser host	IE	Code is run within the context of a Web site
Custom designed host		Creates domain and/or load assemblies into a domain; might also load dynamic assemblies; might be managed or unmanaged code
Server host	ASP.NET	Handles requests submitted to a server
Shell host	Windows shell	Runs applications such as executables in the shell

Table 2.3: Application hosts.

Named Permission Sets

A named permission set is a predefined set of permissions that an administrator can assign to a code group. Built-in named permission sets cannot be modified, but custom permission sets can be created. Built-in named permission sets are:

- Nothing—Code cannot run
- Execution—Code can run but has no access to protected resources
- Internet—A default policy for content of unknown origin
- LocalIntranet—A default policy for the enterprise
- Everything—All permission except skip verification
- FullTrust—Full access to all resources

These sets of code access permissions are defined to deal with the most common requests for access to resources; however, you can create custom permissions. Code-access permissions work because the assembly includes code that requests the appropriate permissions for the access it needs. So, for example, each new request to read a different file is accompanied by a request for permission to read the file, and each new request to append data to a different file is accompanied by a request for read and write permissions.

These FileIOPermissions have caused problems in the past. The programmer uses built-in classes that let the system know which sensitive operations it will want to perform. The assembly, in essence, enumerates for the system which files it may access and for what purpose, which registry keys it may read or modify, whether it needs to add or modify environmental variables, and so on. The administrator sets the Security Policy that details which code has which permissions. If there is a match between requests and approved access, the code works as expected. If not, an error occurs. The code should have explicit actions defined for what to do if these errors occur. Such actions might be as simple as a graceful exit or might be in the form of messages or event log entries that detail the reason that the code didn't work.

Creating Custom Permission Sets

When a new code group is configured, you can select a default set of code access permissions or a pre-selected permission set or you can design a new set. A new code group is created by right-clicking the code group's parent-to-be, and selecting Create New Code Group. A permission set is selected in the Assign a Permission Set to the Code Group dialog box, which Figure 2.5 shows.

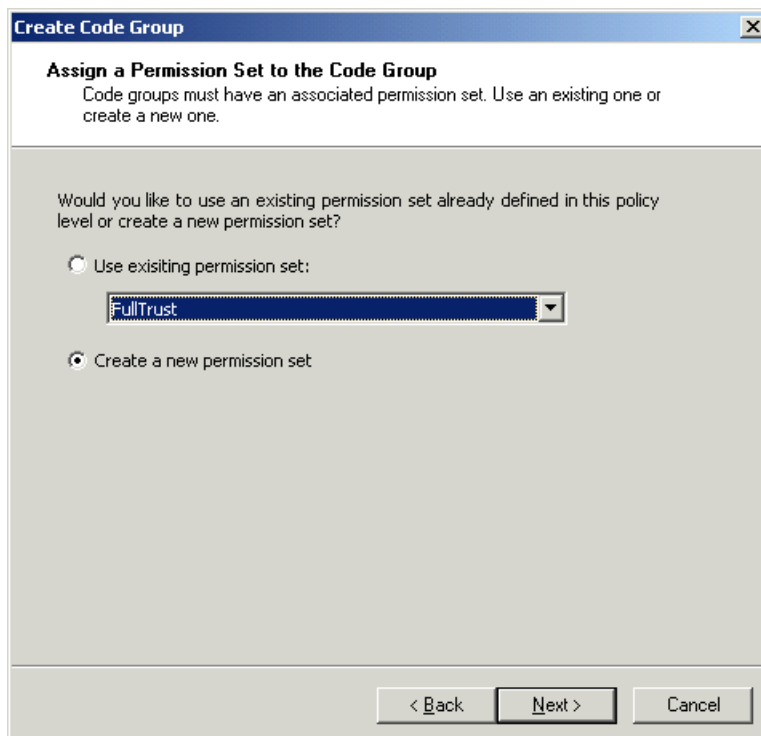


Figure 2.5: A pre-defined permission set (either default or custom designed) can be selected for a new code group or you can create a new permission set.

To create a new permission set, you must name the new set and select the set's permissions from the Assign Individual Permissions to Permission Set dialog box, which Figure 2.6 shows.

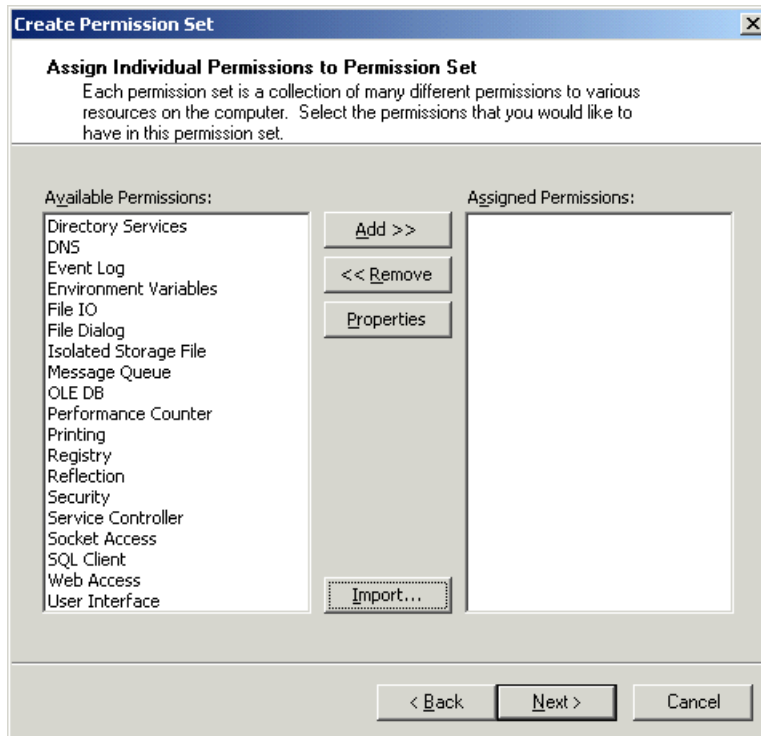


Figure 2.6: Permission sets are created by selecting individual permissions.

Table 2.4 defines the available permissions.

Permission	Access
Directory Services	Browse or write
DNS	None or ALL
Event Log	Unrestricted to all or none, browse, instrument, or audit per specific log
Environmental Variables	Unrestricted to all, or read or write on individual variables
File IO	Read, write, append, path disc. on stated paths
File Dialog	Unrestricted or open, save or open, and save
Isolated Storage File	Unrestricted access to the file system storage or set a quota for use; also set administration by user or roaming user, isolation by user, roaming user or domain
Message Queue	Unrestricted to all message queues or restricted to those identified by path or other element such as machine name, category, and label; access granted is either none, browse, peek, send, receive or administer
OLE DB	Unrestricted or to listed OLE DB providers; the option to allow blank passwords is available with a check box, but applies to all
Performance Counter	Unrestricted or limited to machine and category; access is either none, browse, or instrument
Printing	Unrestricted or no, safe (from a restricted print dialog box), default (only one job from other than default printer), or all printers
Registry	Unrestricted or per registry key; access is read, write, or create

Reflection	Unrestricted or find info about members, find info about types (classes), or script engines and compilers are allowed to generate assemblies
Security	Unrestricted or by selecting allowed security permissions; permissions possible are displayed in Figure 2.7
Service Controller	Unrestricted or by identification and assignment of access—either none, browse, or control
Socket Access	Unrestricted or allowed by identification of host, port, direction, and selection of TCP or UDP
SQL Client	Unrestricted access to SQL Server or limited access to SQL server using ADO.net; you must select the check box if you want to allow blank passwords
Web Access	Unrestricted or by listing host and accept or connect
User Interface	Unrestricted or specify access to windows and clipboard; access to windows is either all windows and events, only to top level safe windows, only to safe sub windows or no windows; clipboard access is to all, own, or none

Table 2.4: Available permissions.



Figure 2.7: Security permissions for assemblies.

Although a program asks for security permissions and administrators define them, an application can use the following security actions to determine what to do with the defined security permissions, including completely ignore them:

- **LinkDemand**—This action takes place at just-in-time (JIT) compile time and checks the immediate caller of the class or method to determine whether the caller has the appropriate privileges.

- **InheritanceDemand**—Evaluated at load time; checks the subclasses.
- **Demand**—Evaluated at load time; checks all callers.
- **Assert**—Evaluated at run time; seeks to excuse the particular class or method from adherence to the Security Policy. This condition is useful if it helps a trusted application’s performance. However, it opens a security hole. What if an untrusted application calls this method? Administratively, all asserts can be denied in the Security Policy causing this class or method not to run.
- **Deny**—Evaluated at run time; seeks to explicitly prevent the use of permissions that the assembly already has. This condition is useful if the assembly calls code that the assembly doesn’t want to use its permission set. For example, if the assembly calls a script that has unrestricted access. The CLR will check the caller (your assembly) for permissions requested by the script. If your assembly has denied some of its own permission set, the script will not be given access.
- **PermitOnly**—Evaluated at run time.
- **RequestMinimum**—Evaluated at grant time; the assembly might specify in its permissions list what it wants, what would be nice if available, and what permissions to refuse if they are granted but exceeds what the assembly needs. The three request security actions (**RequestMinimum**, **RequestOptional**, and **RequestRefuse**) define which the assembly requests. **RequestMinimum** says “I have to have these or I can’t run at all.”
- **RequestOptional**—Evaluated at grant time; the assembly might use these permissions if they are granted, but if the permissions aren’t granted, the assembly will still run.
- **RequestRefuse**—Evaluated at grant time; the assembly identifies permissions that it does not require and will not use even if granted.

Policy Assemblies

Policy assemblies are assemblies used during policy evaluation. By default, system assemblies used in the evaluation of default permissions are automatically included, as Figure 2.8 shows. If you create a custom permission, be sure to add the assembly to the policy assemblies.

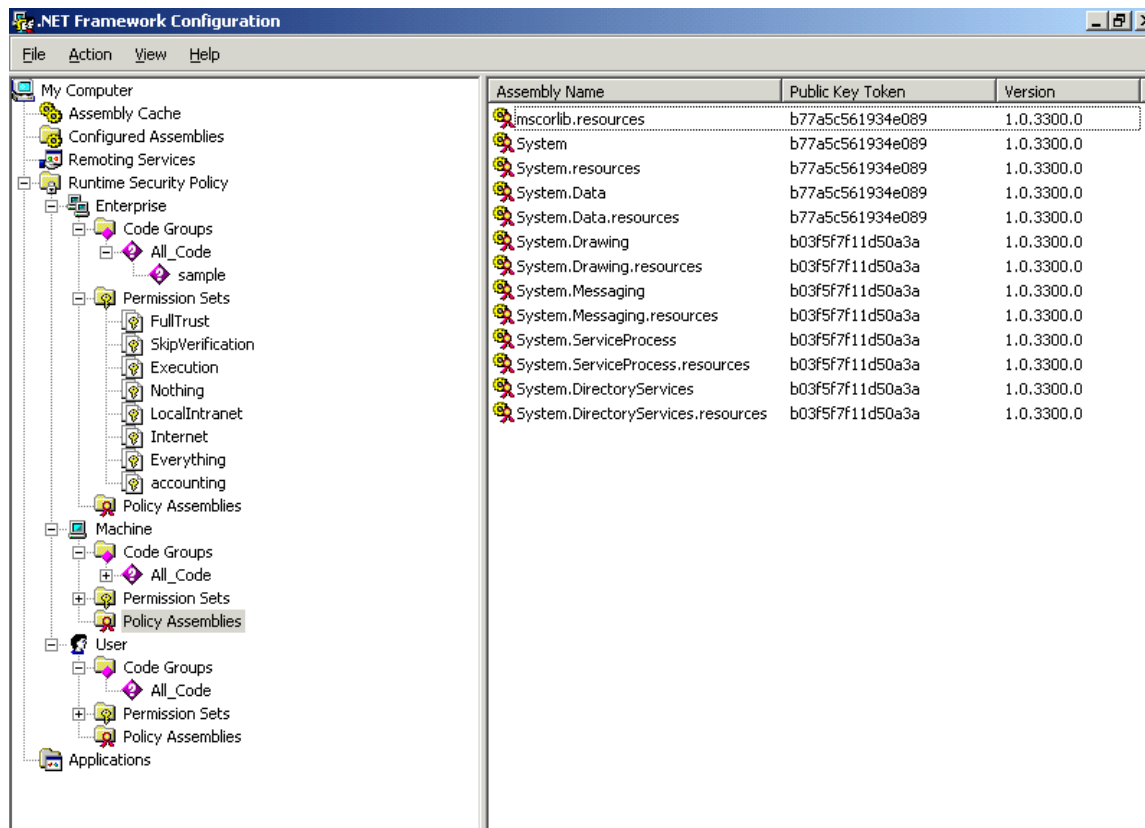


Figure 2.8: The policy assemblies in the machine node of Security Policy for the framework.

Security Policy

When an assembly is run, the Security Policy asks the questions “Where did the code come from?” and “Who wrote the code?” The answers to these questions determines how the code maps to various permission structures (in which level and code access group does it belong?). You can think of the Security Policy as a collection of statements that might sound like these:

- Code from www.somewebsite.com can access files in folder D:\folderforstuff\
- Code running on my local machine can use the Clipboard
- Code from www.wevehadtroublefromthembefore.com cannot read or write to disk or the registry, use the clipboard, or read or modify environmental variables

As administrator, you write the Security Policy using one of two tools: caspol.exe, the code access security policy tool (caspol is a command-line tool); and the Microsoft .NET Framework Configuration Tool, a GUI tool. Both tools let you view policy, code groups, and permissions sets as well as create, modify, and delete them. You also can assign permissions to code groups and analyze security settings on assemblies. In addition, these tools let you directly edit the information.

The Microsoft .NET Framework Configuration Tool

The Microsoft .NET Framework Configuration Tool is a multipurpose tool that lets you manage code access security. To load the tool either access its console from Start, Programs, Administrative Tools; from Start, Run, and enter `mscorcfg.msc`; or from the command-line

```
<systemdrive>\<system folder>\Microsoft.NET\Framework\<version number>\mscorcfg.msc
```

Next, expand the RunTime Security Policy node. There are three subnodes—enterprise, machine, and user—which represent Security Policy levels. Each subnode has configurable code groups, permission sets, and policy assemblies. You can use a handy wizard to assign permission sets to code groups. In the following example, we will create a new code group and assign a permission set:

1. Right-click the *All code* container under the machine node, and select New.
2. On the *Identify new Code Group* page of the wizard, enter a code group name and description. (You can also import a code group from an XML file.) Click Next.
3. On the *Choose a Condition Type* page, use the drop-down box to choose which condition will make an assembly a member of this code group. The conditions available match the evidence categories I previously discussed. You can also choose the custom category.
4. Depending on the category chosen, the appropriate text boxes and/or check boxes are displayed for your entry. So, for example, strong name allows the entry of a public key, while URL allows the entry of a URL (see Figure 2.9). Click Next.
5. On the *Assign a Permission Set to the Code Group* page, you can use the drop-down list to select a permission set or choose *Create a permission set*. For this example, we'll select the option to create a permission set. Click Next.
6. On the *Identify the New Permission Set* page, enter a name and description or choose to add a permission set by importing an XML file. Click Next.
7. On the *Assign Individual Permission to Permission Set* page, select permissions from the left window, then click Add to add them to the window on the right. When you are done, click Next. Click Finish to end the wizard and create the code group.

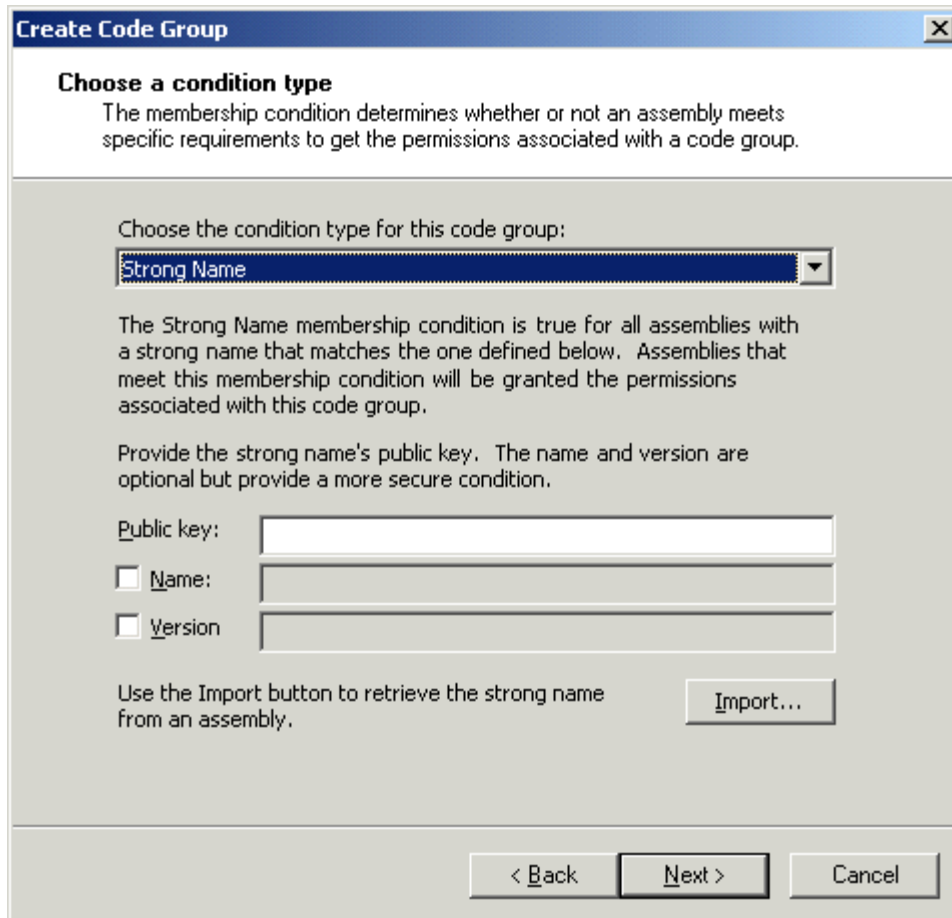


Figure 2.9 Strong Name as the evidence of membership in the code group allows entry of a public key.

Alternatively, you can create a custom permission set, then assign that set to a particular code group. To create a permission set:

1. Right-click the permission set node, and select New.
2. On the *Identify Permission Set* page, add a name and description or import a permission set from a prepared XML file. Click Next.
3. On the *Assign Individual Permissions to Permission Set* page, select permissions from the left window and click Add to move the permissions to the right pane. Each permission will ask for qualifying information. When you are done, click Finish.

Process

The following section explains how the code access security process works. At run time, the list of demanded permissions for the assembly is read and evaluated against the Security Policy. The first step involves gathering evidence, that is, determining the author (who signed it) of the code, the source (URL, site, zone) of the code, and any special identity such as the strong name assigned to the code. The evidence is gathered by the CLR and trusted application hosts. Trusted application hosts are IE, ASP.NET, and the shell. Evidence is divided into classes:

- Site—Web site
- URL—A specific location on some site
- ApplicationDirectory—The folder(s) in the file system that hold the program code for the application
- Zone—IE zones such as intranet, trusted sites, and so on
- StrongName—Cryptographic name
- Publisher—Who signed the code

The evidence is given to the Security Policy. Membership in a code group is determined. A code group may be the set of all code from a particular Web site or from particular authors. Next, the assemblies' requests are judged against those granted in the Security Policy according to the code group and the three hierarchical Security Policy levels: enterprise, machine, and user. See Table 2.5 for more information about these policy levels. A hierarchy of code groups is present at each level. The resulting permissions are the intersection of grants set at all levels. The enterprise level grants the total permissions that an assembly can have. The machine and user level grants cannot expand these permissions; they can only make the Security Policy more restrictive. The final resultant set of permissions is the set assigned to the assembly on this computer being used by this user in this application domain.

Policy Level (in hierarchical order)	Can be Modified by	Comments
Enterprise	Administrator	An enterprise policy configuration file is distributed so that managed code in an enterprise setting is effected
Machine	Administrator	All managed code that runs on this computer
User	Administrator or user	All code is associated with the current user on this OS
Application domain policy	Application domain host code	This policy represents the resultant set of the three previous policy levels.

Table 2.5: Policy level descriptions.

An administrator can set a machine-level policy that doesn't allow the other policy levels to be evaluated. However, other factors might also limit what the assembly can do. During runtime, each caller of the code will also be checked against the Security Policy; if any of the callers don't have the appropriate permissions, the particular access will not occur. However, `assert` can be used to allow simple, limited code to always run. No matter the simplicity or the limitations placed on code, there may be a way to misuse it. Although the programmer can assert that the code should always be allowed to use the permissions granted to it, the administrator can deny `assert`, and thus prevent abuse. It will be up to each organization to establish the Security Policy that works best for them, and programmers and administrators will need to implement it. If no attempt has been made to edit Security Policy, the default Security Policy is applied to the all code group. More restrictive policies are defined for code groups at the machine and user level.

Chapter 3: Understanding Active Directory Foundations

Q 3.4: What possible advantage is there to implementing universal groups?

A: The successful and advantageous use of universal groups depends on the understanding and usage of other Windows groups. In an environment in which no disciplined use of other Windows groups is implemented and assignment of rights and resource access is made to individual user accounts, there is little to no advantage to be gained by adding universal groups to the mix. However, in environments in which global groups and local groups have been properly implemented, the careful and appropriate addition of universal groups is a logical extension that can ease the administration and audit of large Active Directory (AD) environments.

To obtain the full benefits of universal groups, multiple domains in the forest must be at the Windows 2000 (Win2K) native or .NET domain functional level. For a domain to be in Win2K domain functional level, an environment must contain no Windows NT domain controllers and at least one Windows .NET domain controller. A domain cannot be set to Windows .NET functional level until all domain controllers are Windows .NET systems. In either case, the domain functional level must then be set; it will not automatically be changed. For a forest functional level of Windows .NET, all domains must be at Windows .NET domain functional level, and the forest must be set to Windows .NET forest functional level.

A Win2K domain controller is not aware of domain or forest functional level. Instead, a Win2K domain exists in mixed or native mode. When a Windows .NET server is promoted to domain controller, its domain functional level is Win2K mixed. An AD domain could consist of NT, Windows .NET, and Win2K domain controllers. We classify the functional level of this type of domain as Win2K mixed. The functional level can be raised to either Win2K native or Win2K .NET depending on the types of domain controllers present. Different group types, scope, and membership are dependent on the forest functional level.



An additional domain functional level, Windows .NET interim, can be selected when an NT domain controller is upgraded to become the first Windows .NET domain controller in a forest. The Windows .NET interim functional level provides improved replication algorithms and group management. Win2K servers cannot be promoted to become domain controllers if the Windows .NET domain functional level is interim. The interim functional level is only meant to exist until all NT domain controllers in the domain have been upgraded or retired.

Win2K Mixed Domain Functional Level Group Management Practice

Best practices for group administration in NT and Win2K mixed mode is clearly illustrated in the acronym, AGLP: placing accounts (A) into global groups (G), nesting global groups in domain local groups (L), and assigning resource access permissions (P) to domain local groups. Figure 3.14 illustrates this concept. In the figure, Nancy and Carol are members in the DomainClerks global group. On each server A, B, and C, a local group Clerks exists. DomainClerks is a member of each of these Clerks groups. On each server, the local group Clerks has been granted resource access to folders that the members of the DomainClerks need to do their job. Because



the local group has the permission and therefore its members have this access, Nancy and Carol, by extension, have the access required to do their jobs.

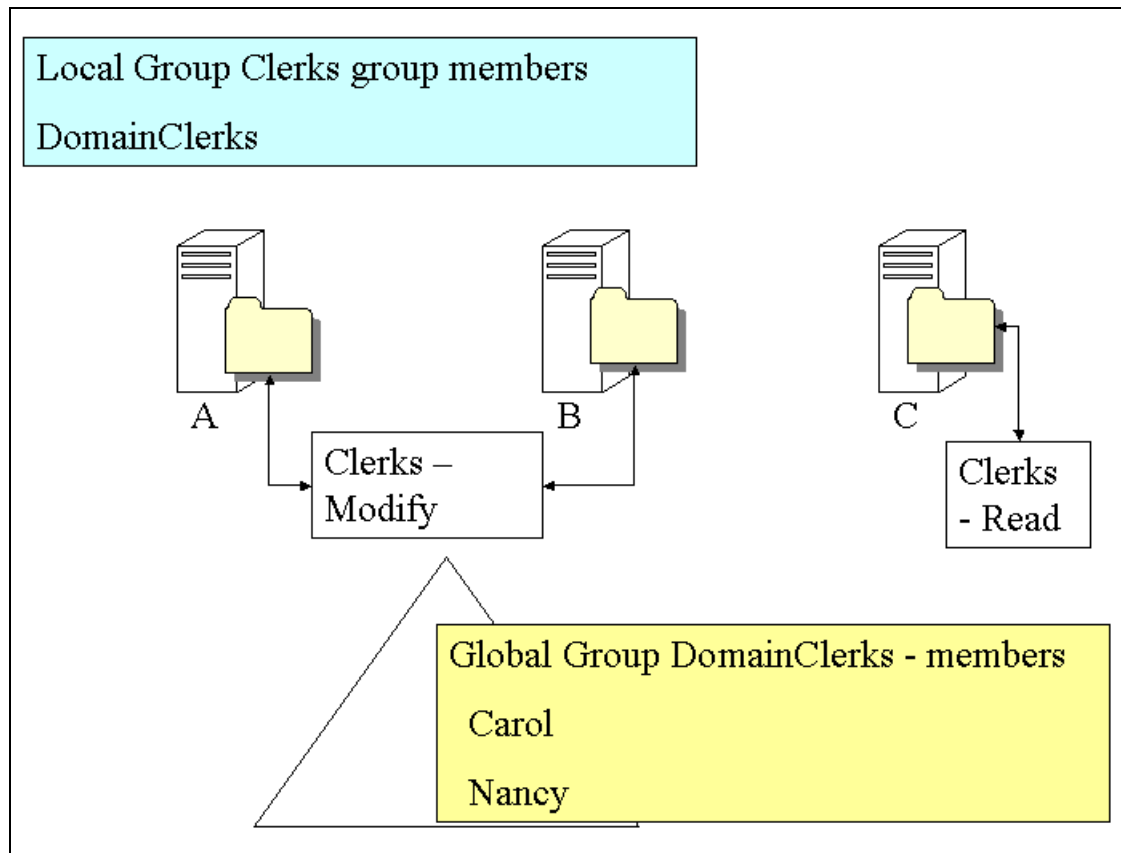


Figure 3.14: Assignment of access to clerks using AGLP.

For those who are new this concept, it might initially seem easier to simply add Nancy and Carol to each of the local Clerks groups on the three servers, or if there are few required resources, to directly give Nancy and Carol the required permissions on the resources. However, in all but the smallest environment, this process becomes unmanageable very quickly. When new clerks are hired, they too must be given access to resources. When new servers or resources are added, all of the clerks must be added to the access lists, or at least to the new servers' local groups. What happens when a clerk quits or is fired? To remove the clerk's access from multiple servers or multiple resources can become quite a chore. It could take months, even years, to determine to which resources Carol has access and remove that access. In addition to being messy and at some point a detriment to performance, this scenario creates a security vulnerability. Unmanaged access to resources exists that might be discovered and utilized by an attacker.

However, if best practice (the AGLP process) is followed, the management of users and resource access is immensely simplified. If John joins the company as a new clerk, you simply need to add his account to the global group DomainClerks, and he immediately has access to the resources he needs to do his job. If Carol quits, the administrator can simply remove her account from the DomainClerks group, and she, or anyone able to log on to her account, has no access to these resources. This setup is also easily auditable. You simply identify the appropriate resources for clerks, examine the resources to determine the groups, then trace back to the global group and

its membership to determine access. To determine whether access has been granted only to those who should have access, you simply review the account list.

This practice is also extensible to multiple domains where trust relationships are correctly configured. In Figure 3.15, a second domain has been added to our example.

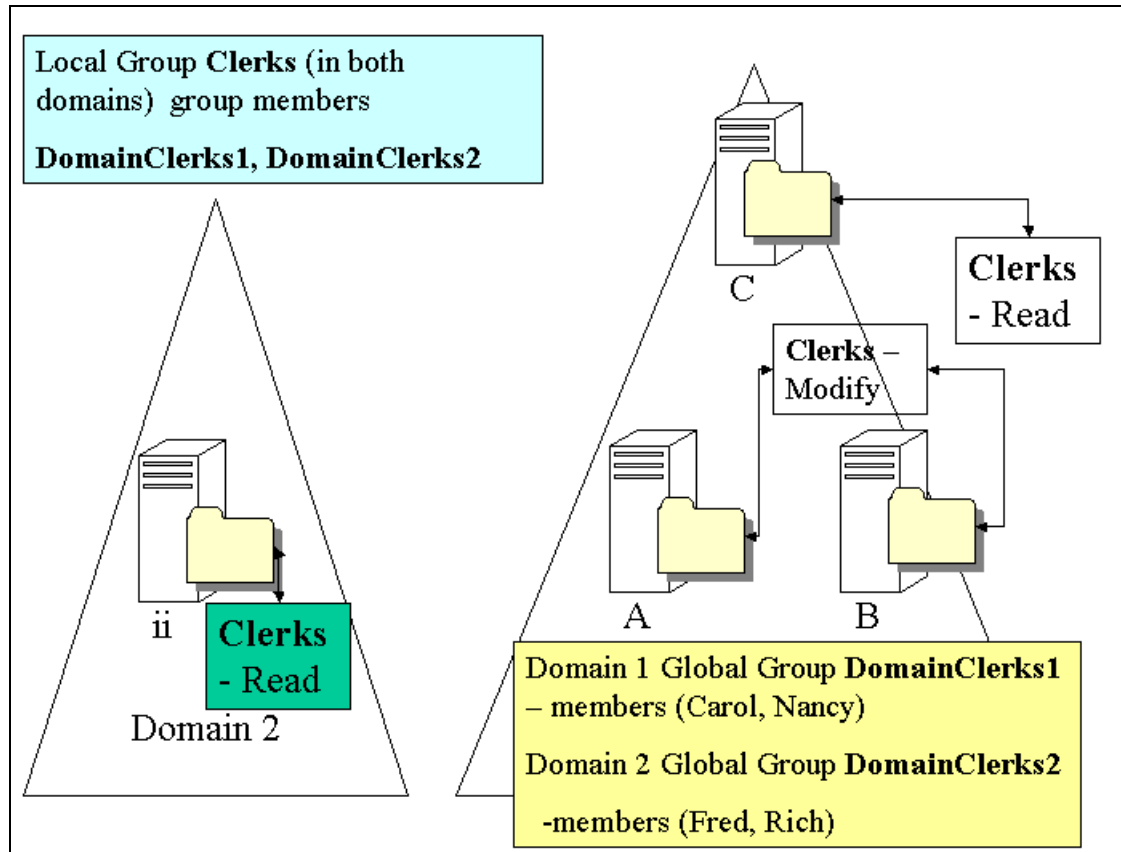


Figure 3.15: Extending AGLP to multiple domains.

If the members of the DomainClerks group need access to resources in this new domain, you can provide it by

- Establishing a trust between the two domains. If both domains are Win2K domains in the same forest, this trust is already established. If they are not, an NT-style trust must be created.
- Creating a local group Clerks on each server in the domain on which there are resources to which clerks need access (or in a Win2K functional level or Windows .NET functional level domain, creating a domain local group on the domain controller).
- Creating access control lists (ACLs) on the appropriate resources, and granting the Clerks group appropriate access.
- Adding the global group DomainClerks to each of these local groups.

In addition, should the new domain have users who need access to the same resources, you can create a new DomainClerks global group for that domain and make it a member of each local Clerks group. Adding domain2 users to the domain2 DomainClerks global group provides the

users with access to the resources in domain2. Should they also require access to resources in domain1, you can simply add the DomainClerks group from domain2 to the local Clerks groups on domain1 servers. You can repeat this process for each new domain. If you have used the AGLP formula to implement groups in your mixed mode forest, you will recognize this structure. When you convert domains to native mode, you do not have to change this configuration to continue providing appropriate access.

Win2K Native or Windows .NET Domain Functional Level and Universal Groups

When all NT domain controllers have been upgraded to Win2K or decommissioned, a Win2K domain may be changed to native mode. (If the Win2K domain was established as the result of a clean install and no NT domain controllers are part of the domain, a Win2K domain can immediately be changed to native mode. However, this change doesn't occur automatically.) If a Windows .NET domain controller is part of the domain, the domain functional level is Win2K mixed by default, but in the absence of NT domain controllers can be raised to Win2K native. In native mode, or in Win2K native domain functional level, universal groups and additional group nesting rules are available. Table 3.2 lists the groups, scope, and membership available in a Win2K mixed mode domain or a Win2K mixed functional level domain as well as in Win2K native functional level.

Group Type	Scope	Membership
Win2K Mixed Mode or Win2K Mixed Functional Level		
Global	Assigned permission in any domain	Accounts in the same domain
Domain Local	Assigned permission only in the same domain	Accounts and global groups from any domain
Win2K Native or Windows .NET Functional Level		
Universal	Assigned permission in any domain	Accounts, global groups, and universal groups from any domain
Global	Assigned permission in any domain	Accounts and global groups in the same domain
Domain Local	Assigned permission only in the same domain	Accounts, global groups, and universal groups from any domain; domain local groups from the same domain

Table 3.2: Groups, scope, and membership.

To illustrate the use of universal groups, I'll return to the previous example. We still have domains A and B, but now the domain functional level has been raised to Win2K native. We don't have to modify group type, membership, and so on to continue to use the resources in the domain and to manage additional resources and users. Everything works as before. However, with a simple change, we could reduce future user and group administration activity.

To do so, you can create and use a UnivClerks group. Instead of adding multiple domain global groups to the domain local groups, we add the single universal group to the domain local groups. The global groups are added to the universal group, and our users Carol and Nancy have access to the resources they need. Users are added to the global groups, so the domain administrators still maintain control over the users who have access to resources in the domain administrators'

Windows 2000 (Win2K); you will have to write, implement, and maintain any batch file or custom application to manage the application in any other way. The first step for any of the three options is to ensure that you have the best possible set of file and registry permission settings.

Determining Appropriate Files and Registry Permission Settings

The default settings for the root of the system drive, root system files, and registry keys in Windows .NET is different than those established in previous builds of Windows. The root of the system drive is C if the operating system (OS) has been installed on this drive. An improvement is the application of default security permissions on the system drive. Table 4.5 identifies these settings. In previous versions of Windows, the default setting was Everyone Full Control.

Group	Permission	Apply To
Administrators	Full Control	This folder, subfolders and files
SYSTEM	Full Control	This folder, subfolders and files
CREATOR OWNER	Full Control	Subfolders and files only
Users	Read and Execute	This folder, subfolders and files
Users	Create Folders/Append Data	This folder and subfolders
Users	Create Files/Write Data	Subfolders only
Everyone	Read and Execute	This folder only

Table 4.5: Default system drive root permissions.

System files in the root of the drive either receive inherited permissions or are explicitly permissioned. Inspecting the permissions on these files (many of which are displayed in Table 4.6) will help you understand the result of the default folder permissions on files placed in the root as well as consider the reasoning behind the more restrictive permissions set. The wise administrator doesn't assume that the default permissions on system folders, files, and registry keys are always going to be the best possible settings.

Files	Group	Permissions	Inherited
Boot.ini, NTDETECT.COM,ntldr	Administrators and SYSTEM	Full Control	No
Boot.ini, NTDETECT.COM,ntldr	Power Users	Read and Execute	No
Config.sys, autoexec.bat	Administrators and SYSTEM	Full Control	
Config.sys, autoexec.bat	Power Users	Modify	
Config.sys, autoexec.bat	Users	Read and Execute	
IO.SYS, MSDOS.SYS	Administrators and SYSTEM	Full Control	Yes
IO.SYS, MSDOS.SYS	Users	Read and Execute	Yes

Table 4.6: Default drive root file permissions.

The installation of new software or the addition of services might add folders and files for which permissions are automatically set or for which there is need to consider changes to the inherited permissions. (When the Windows .NET server is promoted to domain controller, an additional set of permissions is applied.)

To affect the settings that are applied at install, you can modify the `defltsv.inf` file in the installation files folder. This `inf` file is used to apply file and folder permissions during installation. After installation, to return the permissions settings to those applied at install time, use the Security Configuration and Analysis console and select the `setup security.inf` security template. (By default, security templates are stored at `%systemroot%\security\templates`.) To reapply only the system drive root security settings apply the `rootsec.inf` security template. This template can also be used to apply similar settings to the root of other disks. To reapply the settings applied when a server is promoted to a domain controller, use the `DC security.inf` security template. However, when reapplying default permissions using these templates, you should realize that any configuration settings that you may have applied using templates, Group Policy, or manually set will be overwritten if a default template setting exists. Instead of using these default templates, make a copy of them after set up and maintain both copies with official settings changes. A copy can either be made using Explorer or the Security Configuration and Analysis console. In this way, you have a backup of the current security settings. And should you need to return to the default settings, the original default template is available.

Using Group Policy to Maintain File and Registry Permission Settings

Group Policy in Windows .NET, like Group Policy in Win2K, allows administrators to centrally configure and push security policy and other administrative settings to an entire site, domain, or organizational unit—OU. Policies, officially named Group Policy Objects (GPOs), are linked to these containers. It is rarely efficient or practical to implement site policies, so most policies will be implemented at the domain or OU level. In addition to domain policies, a local Group Policy is configured and can be adjusted on individual workstations or servers.

GPOs are only applied to the computer or users whose accounts exist within the container to which they are linked. Thus, because all user and computer accounts for a domain are within the domain container, all GPOs linked at the domain level are applied to all users and computers in the domain unless otherwise filtered. OU policies are applied only to the user and computer accounts that are within the OU.

File and registry permission settings are only configurable within the Computer Settings portion of the Group Policy. When paths are added to the policy, the discretionary access control lists (DACLs) applied to them are applied to the objects on the computer when the policy is applied. Figure 4.9 illustrates a domain with several OUs.

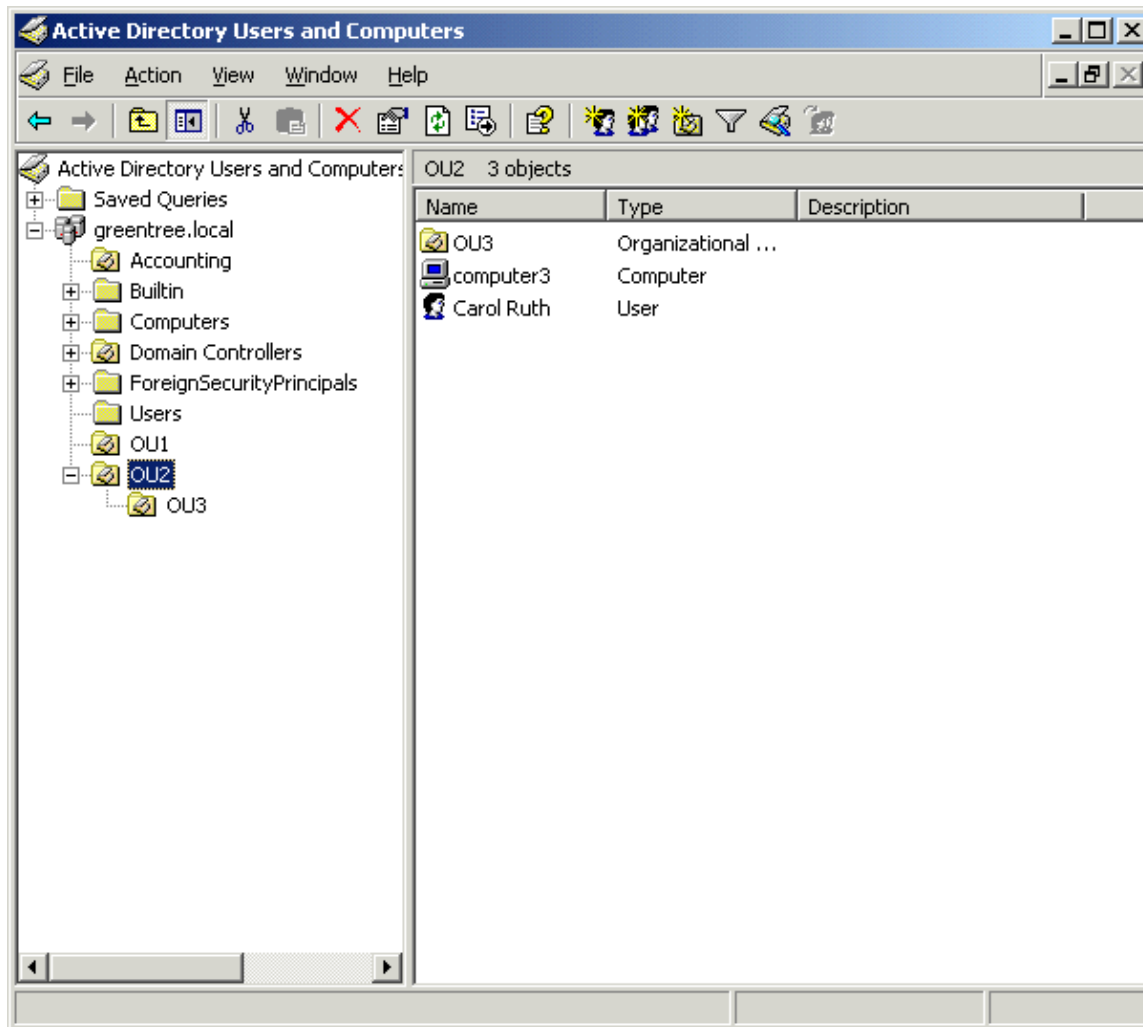


Figure 4.9: OU structure in the Active Directory Computers and Users snap-in.

Figure 4.10 shows more of its logical structure. In addition to a domain policy and a domain controller policy, each OU has its own policy. In this example, multiple policies affect the settings for Computer1 and users JohnC and CarolM. Each policy is applied during boot or logon. First the local policy is applied, then the domain policy, then the OU1 policy. The policies linked to OU2 and OU3 are not applied to the users and computers listed in OU1. Computer2, whose account is in OU2, will receive its local policy, the domain policy, and the policy for OU2. It will not receive the policy for OU1 or OU3.

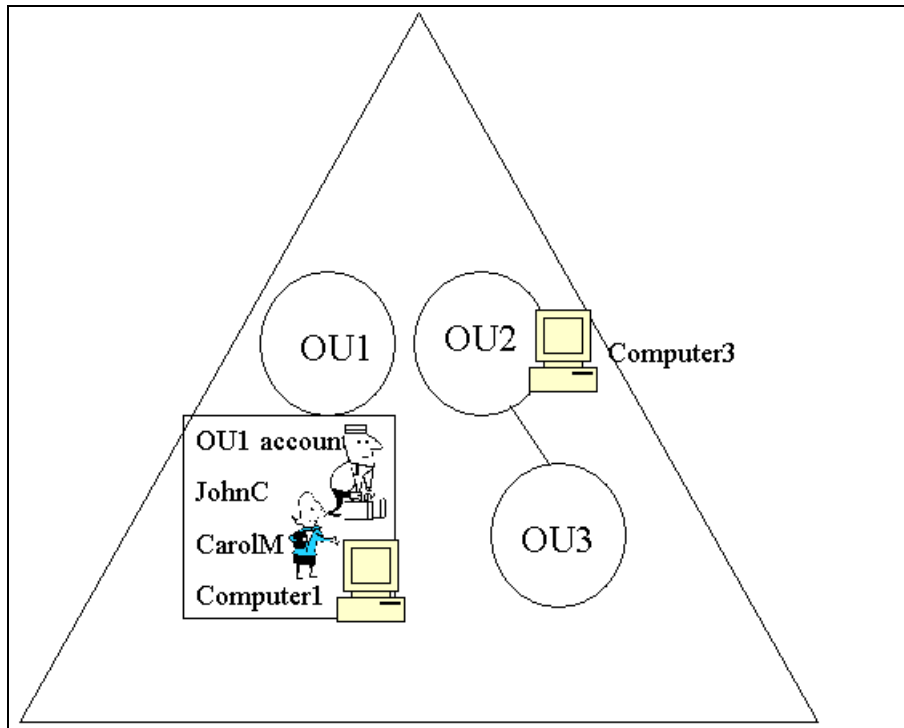



Figure 4.10: Logical OU diagram.


GPO application is cumulative. The settings applied in a GPO will be applied in addition to the previous GPO settings unless there is a conflict. That is, if a setting is not configured in a previous GPO, the new GPO's setting will be applied. If the new GPO and the old GPO have a conflicting setting, the conflict is resolved by applying the most recently applied GPO's setting. If the new GPO's setting for a previously set configuration is not configured, the previous setting will remain.

You can use Group Policy to maintain registry and file permission settings. The permissions can be configured within the Security Policy part of a security template and imported into the GPO or entered directly. Should permissions be set differently on a server, the correct settings will be reapplied at the next policy refresh. (Computers' policy is refreshed at boot and periodically if policy has been changed.) The problem with using Group Policy is that the file and registry permission settings increase the size of the GPO and are reapplied at each boot regardless of whether they have changed. So there is some inefficiency and performance loss. In a small environment, this activity might not be an impediment; in a larger infrastructure that contains multiple domains and many domain controllers, the size of the GPO adds to the replication burden.

 The portion of a GPO labeled Security Settings (and sometimes referred to as the Security Policy) is refreshed periodically. By default, this setting is 16 hours plus a randomized offset. The entire GPO is not refreshed unless changes have been made, unless requested using `gpupdate` or default settings have been modified to do so. For information about how to modify this default setting, see the Microsoft article "How to Delay Security Policies from Being Applied" (Q277543).

Using Secedit to Maintain File and Registry Permission Settings

Alternatively, to maintain file and registry permission settings, you can use the command-line tool `secedit`. Many of you are familiar with the Security and Analysis Configuration GUI, which you can use to analyze or apply security settings. `Secedit` is the command-line version of this tool. Like its GUI counterpart, `secedit` can be used to analyze the security settings on a server and apply new security settings by utilizing a template. In addition, `secedit` can be used to apply a single node from a security template. Thus, to re-apply your preferred file permissions, you can use a single command-line command. To re-apply your preferred registry permissions, you can use another line. Put both commands in a batch file or write a simple script, and you can re-apply both file permissions and registry permissions across multiple servers. And you can use the scheduling service to periodically refresh these settings without any replication burden.

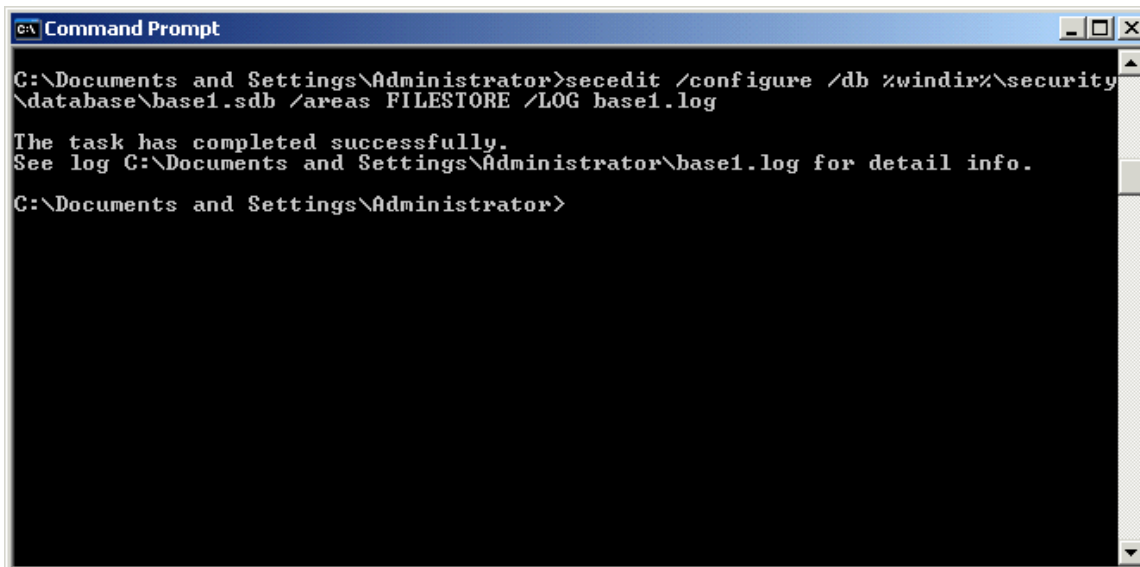
 The following example shows how to use a batch file and the scheduler to updates a server's file and folder permissions settings. If you want to update file permissions on more machines, you will need to place the security template and batch file on those machines and schedule a task to run.

For the following example, the security template name is `Base1`. It is the result of copying the setup security template and maintaining file and registry permissions settings. Before you can refresh the settings, you must import the template into a database. The following statement will do so for you

```
secedit /import /db base1.sdb /cfg base1.inf /overwrite
```

To refresh the file permissions settings on a server, use the following statement at the command line, as Figure 4.11 shows

```
secedit /configure /db %windir%\security\database\base1.sdb  
/areas FILESTORE /log base1.log
```



```

C:\Documents and Settings\Administrator>secedit /configure /db %windir%\security
\database\base1.sdb /areas FILESTORE /LOG base1.log

The task has completed successfully.
See log C:\Documents and Settings\Administrator\base1.log for detail info.

C:\Documents and Settings\Administrator>

```

Figure 4.11: Using `secedit` to refresh the file permissions on a server.

To refresh the registry permissions settings on a server, use the following statement at the command line

```
secedit /configure /db %windir%\security\database\basel.sdb /cfg
basel.inf /areas REGKEYS /log basel.log
```

You can, of course, do both at the same time

```
secedit /configure /db %windir%\security\database\basel.sdb
/areas FILESTORE REGKEYS /log basel.log
```

And even import the database

```
secedit /configure /db %windir%\security\database\basel.sdb /cfg
%windir%\security\security\basel.inf /overwrite /areas FILESTORE
REGKEYS /log basel.log
```

When the command successfully completes, inspect the log file for confirmation (see Figure 4.12).

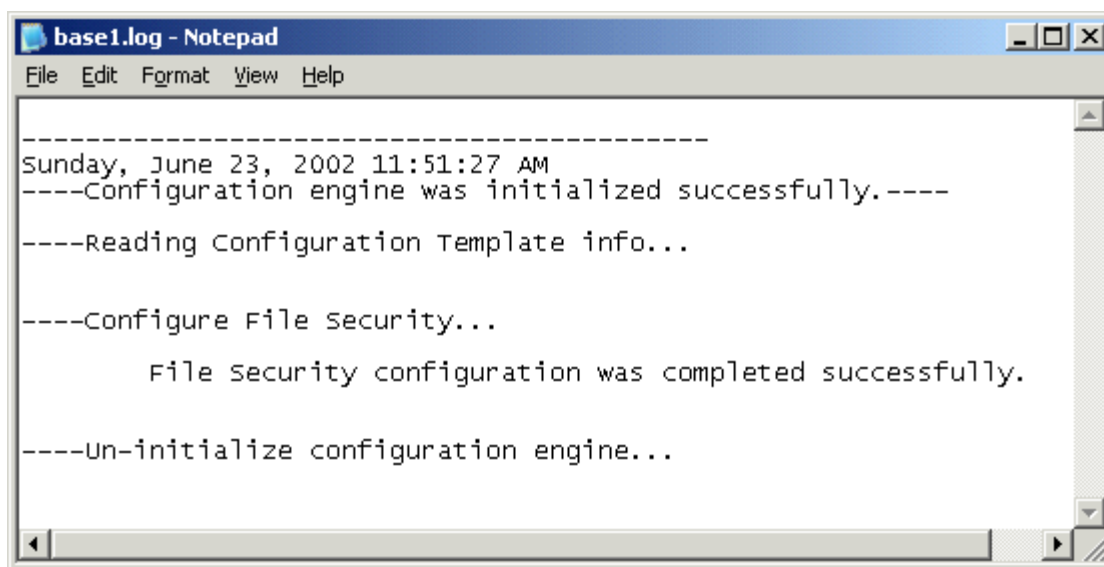


Figure 4.12: The log file created by a successful application.

After testing the statements, you can schedule a periodic refresh by putting both commands (or the combination line) in a batch file. Test the batch file. If successful, use the task scheduler or `schtasks.exe` to schedule the refresh. The following example command line schedules a batch file to run every Monday evening at 10PM under the authority of the system. Table 4.7 provides an explanation of the switches; additional switches are available.

```
schtasks.exe /create /tn filepermit /tr baselconfig.bat /sc
weekly /d MON /ru SYSTEM
```

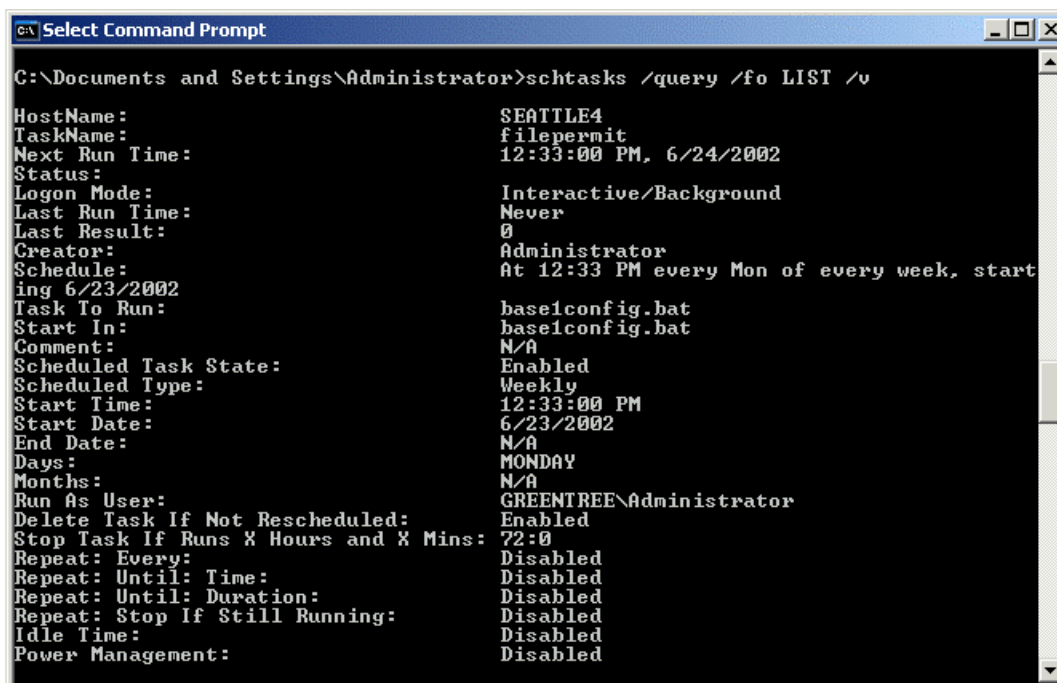
Switch	Description
/create	Create a task
/tn	The name of the new task
/tr	The name of the batch file or command to run
/sc	When to schedule the repetitive event (once, every <i>n</i> times a month, every month, every <i>n</i> times a day, at this time every day, and so on)
/d	Which day of the week; Monday is the default, so I could have left out this switch in the example; /d * runs the process every day
/ru	Under whose authority; if a user account name is entered here (use the domainname\username format), the password is entered using the /rp switch; to use a local computer account use the \machine switch and \u and \p parameters (when the SYSTEM account is used, no password is entered)

Table 4.7: The switches for schtasks.exe.

To provide a specific user account, use the /ru switch to enter the account name and the /rp switch to enter the password. After the task is entered, you can query the task by using the query switch

```
Schtasks.exe /query /fo LIST /v
```

The /fo switch is used to specify the format of the output, and the /v switch reports the advanced settings for each task. Figure 4.13 shows the results. (The user's password is not displayed.)



```

C:\Documents and Settings\Administrator>schtasks /query /fo LIST /v
HostName:                SEATTLE4
TaskName:                filepermit
Next Run Time:          12:33:00 PM, 6/24/2002
Status:
Logon Mode:              Interactive/Background
Last Run Time:          Never
Last Result:            0
Creator:                 Administrator
Schedule:                At 12:33 PM every Mon of every week, starting 6/23/2002
Task To Run:             base1config.bat
Start In:                base1config.bat
Comment:                 N/A
Scheduled Task State:    Enabled
Scheduled Type:          Weekly
Start Time:              12:33:00 PM
Start Date:              6/23/2002
End Date:                N/A
Days:                    MONDAY
Months:                  N/A
Run As User:             GREENTREE\Administrator
Delete Task If Not Rescheduled: Enabled
Stop Task If Runs X Hours and X Mins: 72:0
Repeat: Every:           Disabled
Repeat: Until: Time:     Disabled
Repeat: Until: Duration: Disabled
Repeat: Stop If Still Running: Disabled
Idle Time:               Disabled
Power Management:        Disabled

```

Figure 4.13: Querying scheduled tasks.

You can examine the same tasks in the GUI by selecting All Programs from the Start menu, Accessories, System Tools, Scheduled Tasks. The GUI doesn't display the user password either (see Figure 4.14).

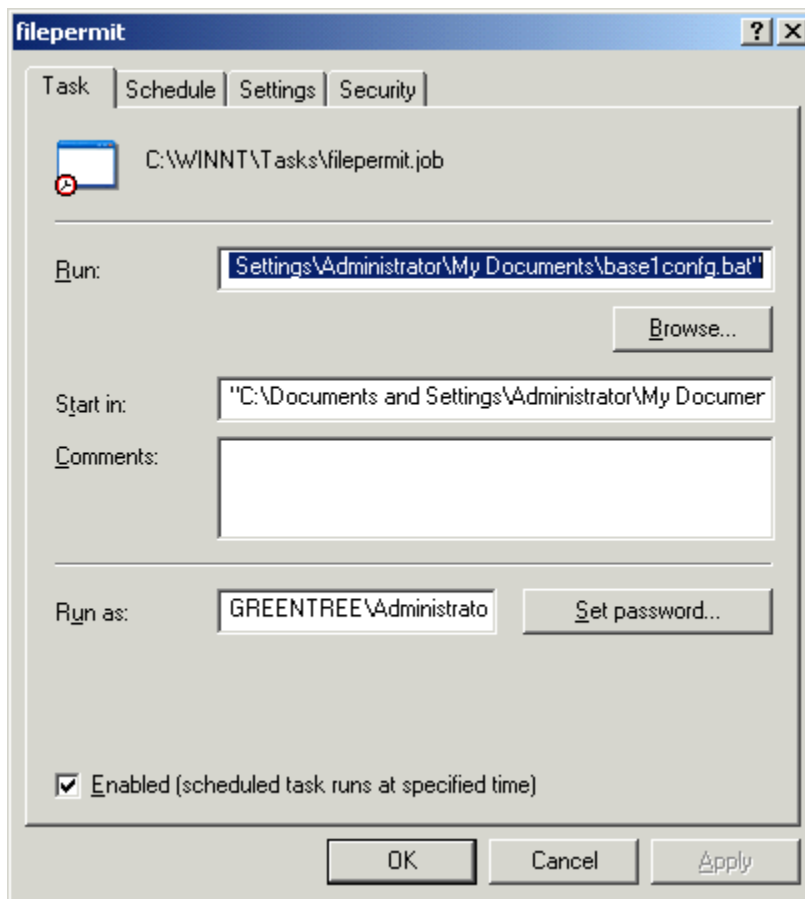


Figure 4.14: Querying scheduled tasks through the GUI.

The GUI has the advantage of showing the last time the task ran, as Figure 4.15 shows. You can use both the command-line `schtasks.exe` tool and the GUI Scheduled Tasks, creating the tasks in one and displaying or modifying the task in the other.

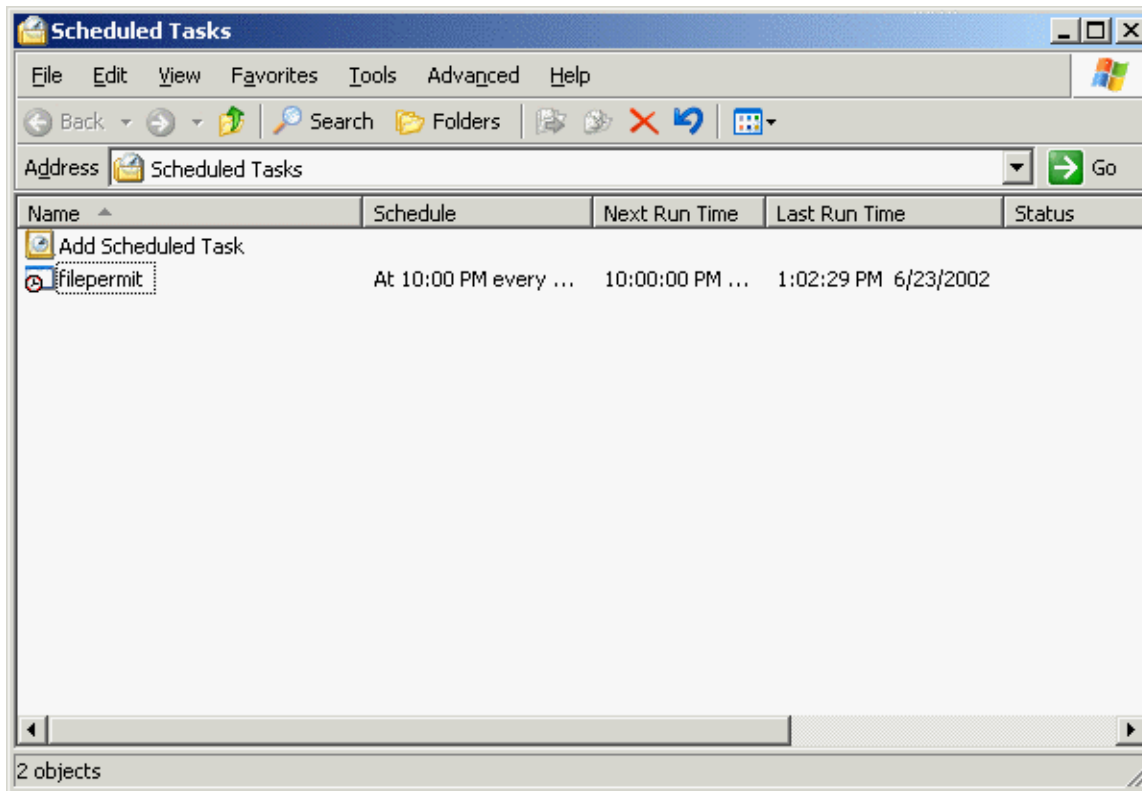


Figure 4.15: Displaying the status of scheduled tasks.

Chapter 5: Administrative Authority

Q. 5.4: Our policy is to let users in sensitive departments maintain the backups for servers within their departments. To allow users in the Accounting department to back up their servers, I created a global group for the users and a local group on each server. The local group is assigned the right to back up files and directories in the local security policy of each server, but these users cannot back up the servers. What is wrong?

A: User rights are easily assigned in Windows .NET; the challenge is to assign them in the right place to accomplish what you want to do. User rights are assigned in Group Policy, but multiple levels exist. Which level will give you the results you want? To select the appropriate location, you must understand how Group Policy affects user rights and how Group Policy processing for user rights varies from Group Policies' usual application.

Location, Location, Location

On a Windows .NET server that is not joined in a domain, user rights are assigned in the local security policy. These rights affect the local users. When, and if, the server is joined to a domain, these rights still affect local users' accounts. If a user logs on with a local user account, the user

will receive the rights assigned in the local security policy. However, users that use a domain account to access the server are subject to the more complex rules of Group Policy.

Several Group Policies have the potential to affect the user rights assigned to a domain account. Table 5.3 shows four basic levels that can do so in the order in which they are applied.

Level	GPO	Description
1	Local Security Policy	Affects only the local computer on which it resides
2	Site	Not recommended for use
3	Domain	Affects all computers in the domain, set in the default domain controller policy linked to the Domain Controller OU
4	OU	Affects only the computers with accounts in this OU or a child OU

Table 5.3: Group Policy levels.

If you inspect the default domain controller security settings Group Policy Object (GPO—located at Windows Settings, Security Settings, Local Policies, User Rights Assignment), you will find the backup files and directories user right assigned to Administrators, Server Operators, and Backup Operators. Inspect the GPO for the domain, and you will find that all user rights are not defined by default (see Figure 5.6). This setting seems to imply that all computers in the domain can be backed up by users with membership in these local groups assigned at the local level in the local security policy. However, if you use local security policy and add a group to the groups that can back up files and folders, then use the Resultant Set of Policy to determine who has the right to back up files and directories, you will see that it remains Administrator, Server Operators, and Backup Operators. Setting local policy doesn't seem to have any affect.



Figure 5.6: Default domain security policy.

To understand what's happening in this example and how to interpret what will happen to other locally set security settings, note that Group Policies are applied in order. Although one policy's settings can complement another policy's setting, such only happens when one of the policies does not have a setting defined. If more than one policy has conflicting settings, the policy applied last will be used. Domain-level Group Policy is applied after the local security policy, so any policy set locally will be overridden by the domain policy unless the domain policy doesn't have the setting configured.

Many settings are not defined in the domain policy, so when domain policy is applied, no changes are made to those items. However, the default domain policy is not the only one applied. The default domain controller GPO is applied as is any policy created at the OU or parent OU for the computer. Changes in these policies change the computer and user settings from those applied by the local Group Policy. If you inspect the default domain controller GPO, you will find that it has the user right *Backup Files and Directories* defined. Hence, when you set user rights in the local security policy of a computer joined in a domain, the user rights will only affect users logging on to the computer with a local account.

So how can we create a policy that lets us define the group that has the Backup Files and Directories right? We could make a change at the default domain policy, but that would affect every computer in the domain. Instead, we'll create a different user rights policy for different computers in the domain by creating a policy that is applied at the OU level. The policy will only be applied to computers whose accounts exist within the OU. OU policies are applied after the domain policy, so their settings will be used when a conflict exists. They will override the domain policy for all computers in the OU.

To give the right to back up the servers in the Accounting OU to a select group of users in the Accounting department, use the following steps: Create an Accounting OU, and place the computer accounts for Accounting servers within this OU. On each of the Accounting servers, create a local group Abackup. In the domain, create the global group GABackup, and make the accounting employees authorized to back up the Accounting servers members of this group. Make the GABackup group a member of the local group Abackup. In the GPO for the Accounting OU, open the properties page for the Backup Files and Directories user right (located at Windows Settings, Security Settings, Local Policies, User Rights). Select the *Define these policy settings* check box, and add the Abackup group, then click OK (see Figure 5.7). Close the Group Policy.

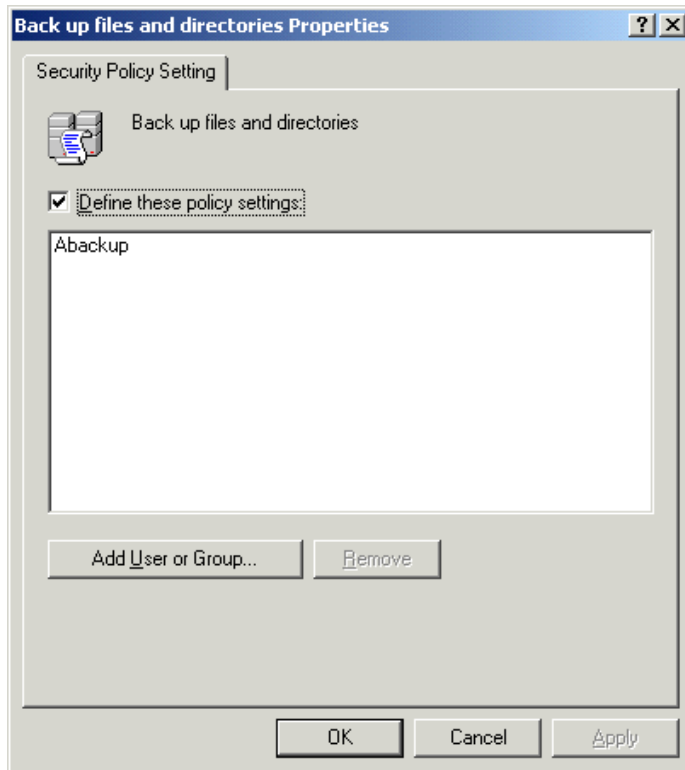


Figure 5.7: Defining user rights for the Accounting GPO.

After the policy has refreshed, use Resultant Set of Policies to inspect the user rights for one of the servers in the Accounting OU. You will find that the effective policy lists only the Abackup group under the user right Backup Files and Directories. On other servers in the domain, the default setting will still be Server Operators, Backup Operators, and Administrators. Remember, when a policy is applied and settings are not defined, the new settings are added to the old. However, if settings exist in more than one policy, the most recently applied policy will be used.

Chapter 6: Triple A's—Authentication, Authorization, and Audit

Q. 6.4: What is the difference between user rights, permissions, and privileges?

A: This subject is confusing for two reasons. First, because documentation from Microsoft and others has sometimes ambiguously defined these words. It's easy to find various interpretations. Second, because we quite naturally think of these things in everyday terms. If we look them up in a thesaurus, we find that privilege is a synonym for right. Our gut reaction is that rights and privileges and maybe even permissions mean the same thing. They're not.

The first distinction I'll make is between permissions and rights. Permissions detail the level of access that a user, computer, or group has to an object. The easiest example is file access permissions. A group or user account can be given read, write, delete, execute, or other combinations of permissions that give the group or user various levels of access to a file.

Permissions can also be set on registry keys, printers, and directory objects. Permissions set on directory objects can also dictate a user's ability to go far beyond the typical read, modify, and delete functionality. Directory object permissions include such things as resetting a password and unlocking an account. These permissions are sometimes hard to reconcile with our definition because we tend to think of them in terms of rights granted to administrators in previous versions of Windows. If you have trouble, just remember that permissions change access to objects. When you reset the password of a user account, you are manipulating one characteristic of that object, much as you might change the attributes of a file to read only.

Rights dictate broad ability to manage or access systems. A right enables some system-wide, or perhaps even domain- or forest-wide, ability. Examples of rights are the ability to logon or shut down a system remotely. In Windows 2000 (Win2K) and Windows .NET, the user rights category is a broad category that is divided into logon rights and privileges.

The sheer number of permissions makes the topic too broad to discuss in detail in this Q&A. Permission are expandable, as new permission may be created for objects within the Active Directory (AD), and new objects may be added to the directory. User rights, however, are finite. It's important to get a firm understanding of what they are. User rights, which consist of logon rights and privileges, are configured in the User Rights section of the local security policy (see Figure 6.13) of a standalone Windows .NET server or Windows XP computer. User rights for a domain are configured in the default domain controller Group Policy.

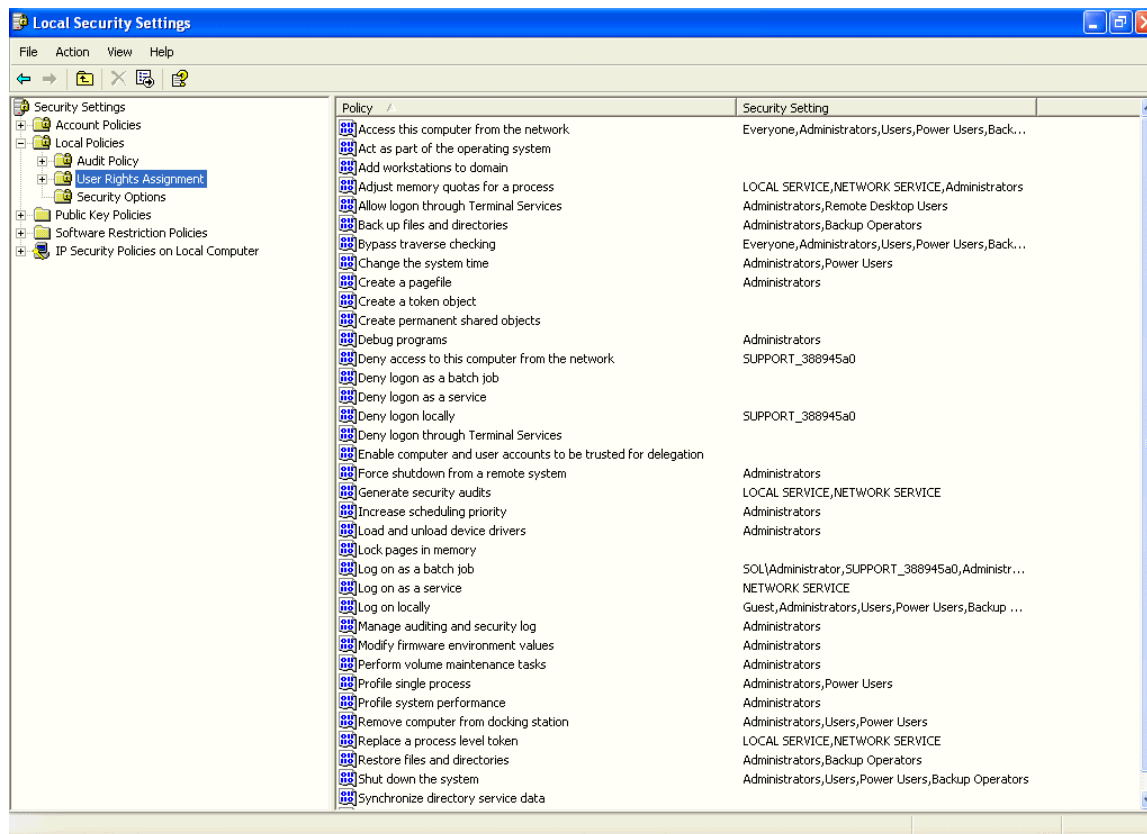


Figure 6.13: Configuring user rights.

Logon Rights

Logon rights affect the ability of a user or computer account to authenticate. Two types of logon rights exist—allow and deny. Table 6.5 describes logon rights.

Right	Description	Default Membership
Access this computer from a network	Affects which user and computer accounts can connect to another computer from the network. Although not obvious, also affects the use of administrative utilities, such as Group Policy, that use network connections even if accessed from the local interface of a domain controller. For this reason, the Enterprise Domain Controllers group is included here. (If the Everyone group is removed, even computers cannot access other computers from the network. Should someone remove the group Everyone, an administrator will still be able to manage Group Policy through the GUI and reset the policy to allow access to administrative tools.	Domain: Administrators, Authenticated Users, Enterprise Domain Controllers, Everyone Workstation/Server: Administrators, Everyone, Power Users, Backup Operators
Logon as a batch job	Allows logon using a batch utility	None; if IIS is installed, default IIS account
Logon as a service	All accounts used for services must have this right.	LocalSystem, LocalService, NetworkService
Logon locally	Use Ctrl+Alt+Del at the console to logon. Also known as an interactive logon.	Domain: Administrators, Server Operators, Account Operators, Backup Operators, Printer Operators Workstation/Server: Administrators, Power Users, Users, Guests, Backup Operators
Allow logon through Terminal Services	Users that can use Terminal Services. (This right has no effect on systems earlier than Win2K Service Pack 2—SP2.)	Domain: Administrators Workstation/Server: Administrator, Remote Desktop Users
Deny access to this computer from the network	Explicitly prevents an account or group from remotely connecting to this computer across the network.	
Deny log on as a batch job	Explicitly prevent an account from logging on as a batch job.	None
Deny log on as a service	Explicitly prevent an account from being used as a service account.	None
Deny logon locally	Explicitly prevent an account from logging on interactively.	None
Deny logon through Terminal Services	Explicitly prevent logon using Terminal Services. (This right has no effect on systems earlier than Win2K SP2.)	None

Table 6.5: Logon rights.

Privileges

The remaining user rights are privileges. The following list provides important things to remember about privileges:

- Some privileges might override permissions. For example, the right to back up files and directories will enable users with this right to back up a file even if there are explicit permissions on this file that deny access to the user.
- Some permissions extend or override privileges. For example, giving a user the Create Computer permission on an organizational unit (OU) or on the computer container in AD allows the user to create more than 10 computer accounts in the domain regardless of whether the user has the *Add workstations to domain* privilege.
- Administrators can regain control of all objects regardless of settings through user rights. That is, administrators can take ownership of an object and give themselves back any access they need.
- Some privileges are more powerful than others. Take a hint from the default settings. Those rights assigned only to administrators or not assigned at all are unusually set that way for a reason.
- System process might already have these privileges inherently and do not need to be assigned to them.
- An administrator's right to take ownership of objects cannot be removed.

Table 6.6 describes privileges.

Privilege	Description	Default Membership
Act as a part of the operating system	Process can gain access as user and access all resources. Calling process can request other privileges and an identity that is not tracked in the audit log.	None
Add workstations to a domain	Add 10 computers to a domain. A user can also be given the Create Computer permission on an OU or computer container in AD. This permission gives the user the right to add more than 10 computers to the domain.	Domain: Authenticated users Workstation/Server: Not relevant
Adjust memory quotas for a process	Use a process that has write property access to another process (to increase the processor quota of the process).	Domain: Administrators, NETWORKSERVICE, LOCALSERVICE
Backup Files and Directories	Make a backup.	Administrators, Server Operators, Backup Operators
Bypass Traverse Checking	Travers folders to follow a path, even though a user has no privileges on them (the user cannot list the contents of the folder, just traverse them).	Domain: Everyone, Authenticated Users, Administrators Workstation/Servers: Backup Operators, Users, Power Users, Everyone, Administrators
Change the system time	Set the internal clock of the computer.	Domain: Administrators, Server Operators

		Workstation/Servers: Administrators, Power Users
Create a Pagefile	Create and change the size of a pagefile.	Administrators
Create a Token Object	Allows a process to create a token that can then be used to access resources.	
Create Permanent Shared Objects	Create a directory object.	
Debug Programs	Attaché a debugger to any process.	Administrators
Enable Computer and User accounts to be trusted for delegation	Change the Trusted for Delegation setting on a user or computer account. Doing so allows access in a multi-tiered application of a front-end process to use a user's credentials to run backend processes and gain access to resources and privileges. The computer and user account involved must be trusted for delegation. Use of this service can leave resources vulnerable to attacks.	Domains: Administrators Server/Workstation: Not applicable.
Force shutdown from a remote system	Remotely shutdown a computer.	Domain: Administrators, Server Operators Workstation/Server: Administrators
Generate Security Audits	Add entries to the Security log.	LOCALSERVICE, NETWORKSERVICE
Increase Scheduling Priority	Change the scheduling priority of a process.	Administrators
Load and Unload Device Drivers	Install/uninstall Plug-and-Play (PnP) device drivers. (Non PnP device drivers can only be installed or uninstalled by administrators regardless of this setting). This privilege might be abused to add malicious code disguised as a device driver.	Administrators
Lock Pages in Memory	The OS manages pages in memory, paging some out as necessary to make room for others. Some pages are not removed from memory as they are critical for operations. This privilege can be used to keep other pages in memory. The OS would then not be able to remove them. This setting could make processes that use these pages run faster, but would lead to degradation of operations as less memory is available for management by the OS.	None
Manage Auditing and Security Log	View and clear the Audit log. Add or remove auditing of objects, registry keys, files, and folders.	Administrators
Modify firmware environmental variables	Change system environmental variables.	Administrators
Perform volume maintenance tasks	Run processes such a Disk Defragmenter and Disk Cleanup (not compatible with Win2K SP2 and earlier).	Not defined
Profile single process	Monitor non-system processes using Performance Monitor.	Administrators
Profile system performance	Monitor system processes using Performance Monitor	Administrators
Remove computer	Undock portable computer (use Eject from the Start	Administrators

from docking stations	menu).	
Replace process-level token	Replace the token that a process started its processing with. (and thus change its ability to access resources)	LOCALSERVICE, NETWORKSERVICE
Restore files and directories	Restore files and folders from backup media.	Administrators, Server Operators, Backup Operators, Print Operators, Account Operators
Shutdown the system	Shutdown the system from the console.	Domain: Administrators, Server Operators, Print Operators, Account Operators Server: Administrators, Power Users, Backup Operators Workstation: Users, Backup Operators, Power Users, Administrators, Everyone
Synchronize Directory Service Data	A process can provide directory synchronization (only relevant on domain controllers).	
Take ownership of files or other objects	Take ownership of a securable object (such as a file, folder, registry key, directory object, printers, processes, threads).	Administrators

Table 6.6: Privileges.

Chapter 7: Remote Access

Q. 7.4: I want to setup dial-up remote access and a virtual private network connection, but I don't want to configure Active Directory. Is this setup possible?

A. Windows .NET Routing and Remote Access (RRAS) can be established in either a domain or workgroup environment. So yes, you can set up dial-up remote access and a virtual private network (VPN) without Active Directory (AD). RRAS, however, has additional options when established in a Windows .NET AD domain. Before you begin either setup, there are decisions to be made about the authentication mechanisms to choose, the remote access policies that will be used, and whether you need and how you will accomplish data encryption. You must also determine which users will be allowed to remotely access the network.

Understanding the Difference Between Authentication and Authorization

Understanding the difference between authentication and authorization is a critical skill for those who will set up, configure, and troubleshoot Windows .NET RRAS. Although enabling the service is simply a few mouse-clicks away, ensuring that clients that should be able to connect can and that those that shouldn't be able to can't will take a little more of your time. Let's examine how a RRAS client gets connected. During the initial connection request, the client

must complete two steps: First, the client must authenticate (with one exception discussed later) or prove a valid identity to the RRAS server. Although hardware devices may participate in this process, the typical authentication process includes the use of a user name and password. Next, if the client is successful in authentication, authorization comes into play. The client has proven who he or she is and must now be authorized to connect. To complete your RRAS setup, you must configure both authentication methods and authorization information. Authentication must be configured on both the client and the server and can be configured via Windows .NET remote access policies for greater granularity.

Choosing an Authentication Provider

During installation, you must choose whether Windows will oversee authentication and authorization or if a central Remote Authentication Dial-In User Service (RADIUS) server will do so. In a workgroup, choose Windows Authentication. The local Security Accounts Manager (SAM) database of the Windows .NET RRAS server will be used. Should you later decide to implement AD and make the RRAS server a member of a domain, AD accounts can be used for authentication. You might also choose to switch to RADIUS as the authentication provider and set up Windows Internet Authentication Service (IAS) to centrally control both authentication (using domain accounts) and authorization.

There is no need in either case for a separate user database. Figure 7.12, 7.13, and 7.14 illustrate the three possibilities for authentication. Figure 7.12 shows a standalone RRAS and the request for access from many clients. Because the users have accounts on the server, they can be authenticated locally.

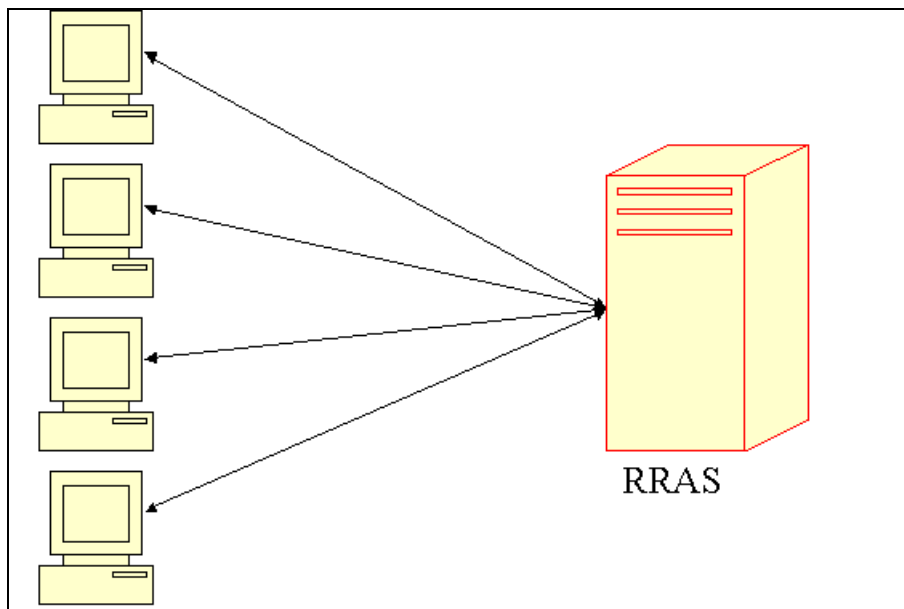


Figure 7.12: Standalone RRAS in a workgroup.

In Figure 7.13, domain accounts are used.

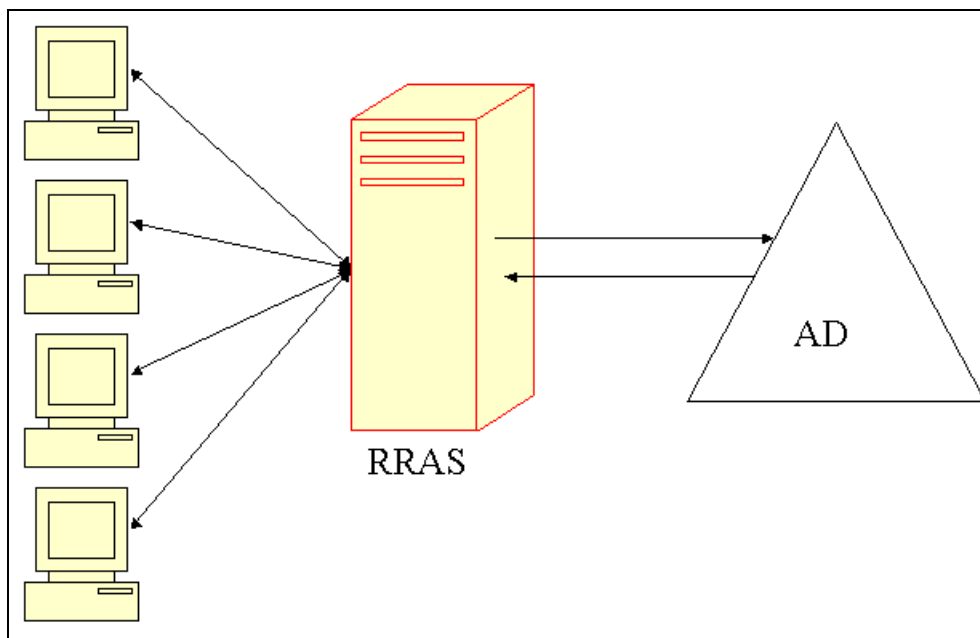


Figure 7.13: RRAS in an AD domain.

In Figure 7.14, the RRAS server acts as the RADIUS client and passes the request to the RADIUS server. The RADIUS server will pass credentials to the domain controller for authentication.

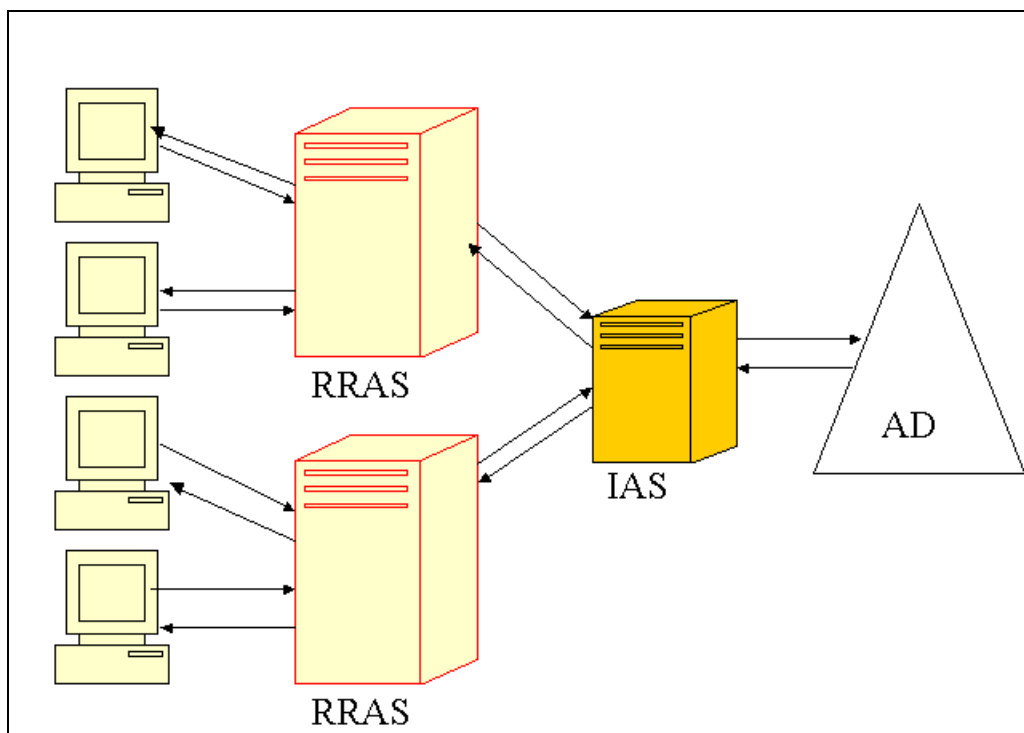


Figure 7.14: RRAS with a RADIUS server.

Whether a workgroup server, member server, or IAS server is used, you will have a choice of authentication methods and the ability to configure remote access policy.

Authentication Methods

Several authentication methods are possible; it is up to you to select and restrict those acceptable for use. Authentication methods must be configured at both the server and client. For authentication to be processed, at least one of the methods available on the client must be available on the server. Although the server has an extensive list of methods available, many clients do not. The primary reason for restricting the authentication methods available is to improve security. However, your ability to restrict authentication methods on the server might be dictated by those available on the clients that are used in your enterprise. Although most Windows clients can utilize several authentication methods, others and many non-Windows clients can only use the older less secure authentication protocols. When making authentication method choices, you must consider both the degree of security desired and the capability of the clients that exist in your environment.

The range of authentication protocols available consists of older protocols, which pass passwords in the clear to sophisticated protocols that require smart cards for authentication. The RRAS server will request the use of protocols in a predefined order from the most to least secure, but it is the client's capability that will dictate the protocol actually used. If the client is not capable of using MS-CHAPv2, for example, but is capable of PAP, and PAP is authorized for use, PAP will be used. If you do not want to allow older, less secure authentication protocols, you must make them inaccessible in the server environment. Doing so might mean that some clients cannot connect. Your written security policy, if implemented to the letter, might then require that clients be replaced.

Server-side authentication protocols are set on the Authentication Methods page of the RRAS properties (see Figure 7.15), and client-side authentication methods are set by editing the profile of the dial-up connection (see Figure 7.16).

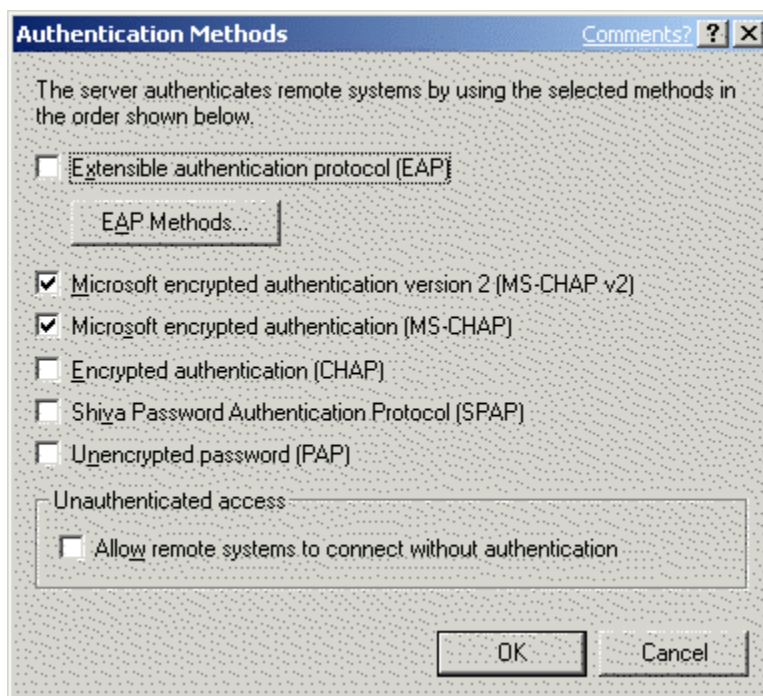


Figure 7.15: RRAS authentication methods.

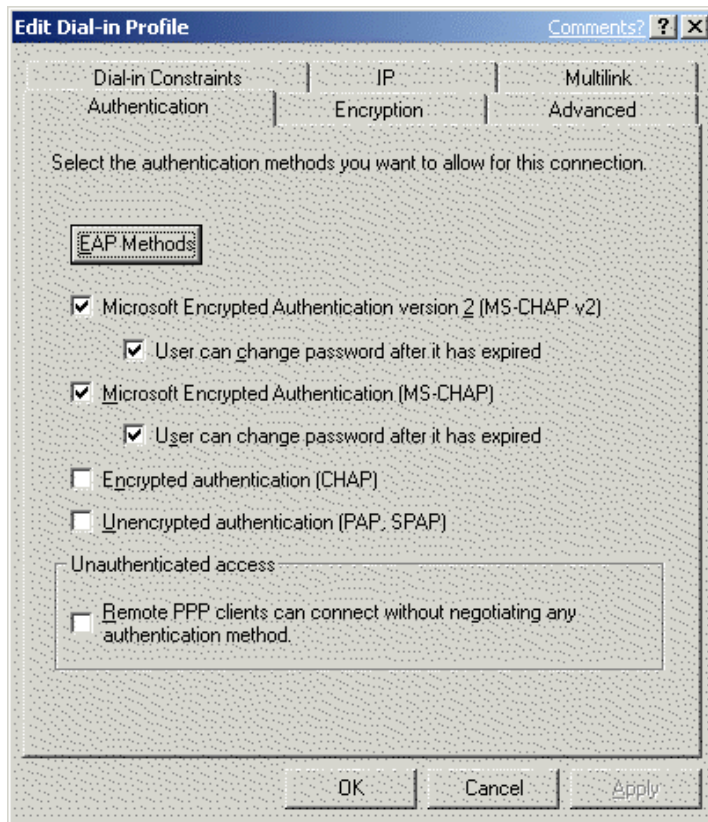


Figure 7.16: Client-side authentication methods.

If the Extensible Authentication Protocol (EAP) is chosen as the authentication method, you must choose an EAP protocol (see Figure 7.17).

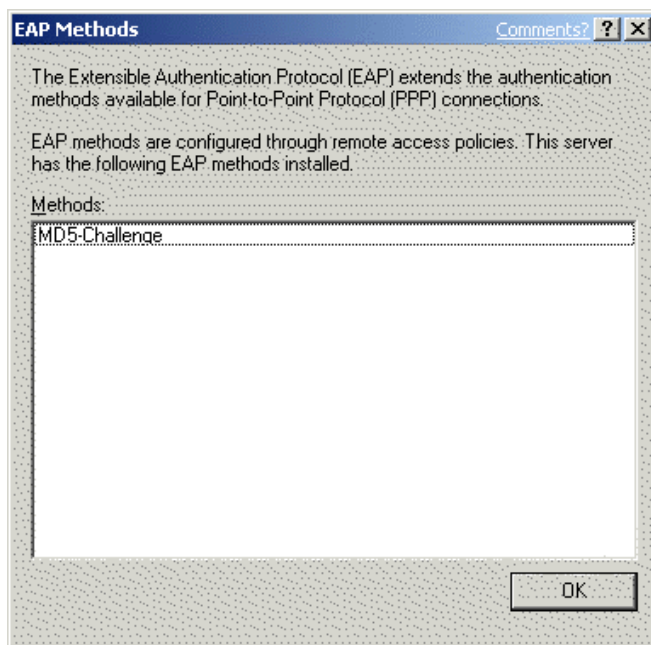


Figure 7.17: If EAP is selected, you must select the EAP Method. Only EAP MD-5 Challenge is available on a standalone server.

I'll provide more information about EAP later in this answer. Table 7.1 describes the protocols that are available regardless of the authentication provider.


Protocol	Data Encryption	Password	Description
MS-CHAP	MPPE	Challenge/response, can be changed by user	Microsoft Challenge Handshake Authentication Protocol
MS-CHAPv2	MPPE	Challenge/response	Microsoft Challenge Handshake Authentication Protocol version 2
SPAP	No	Clear text	Shiva Password Authentication Protocol
PAP	No	Clear text	Password Authentication Protocol
CHAP	No	Challenge/response, the user can remotely complete a password change	Challenge Handshake Authentication Protocol
EAP	Protocol dependent	Protocol dependent	Extensible Authentication Protocol; this protocol is generic and lets several potential authentication processes be used.

Table 7.1: RRAS authentication methods.

MS-CHAP and MS-CHAPv2

For backward compatibility, two versions of MS-CHAP are available. Both versions are nonreversible encrypted password authentication protocols. Using either lets you select Microsoft Point-to-Point Encryption (MPPE) for data encryption. You should always deselect MS-CHAP if possible, as MS-CHAPv2 is a marked improvement over MS-CHAP. The two versions are different in the following ways:

- MS-CHAPv2 does not allow LANMAN-style passwords. LANMAN authentication does not allow capital letters and has other characteristics that make it a cryptographically weak authentication protocol. In addition, the LANMAN password-changing algorithm is weak and is not allowed with MS-CHAPv2.
- MS-CHAP only provides client authentication while MS-CHAPv2 provides mutual authentication. The client authenticates to the server and is able to authenticate the server in return. This activity ensures that rogue servers cannot be used to trick clients into transferring confidential data to non-company servers.
- MS-CHAP uses a 40-bit encryption key based on the user's password. Therefore, each session for which the same password is used will have the same encryption key. This setup gives an attacker more time to determine the key and a longer time to use it to decrypt messages. MS-CHAPv2 uses an encryption key based on the password and an arbitrary challenge string—the encryption key will always be different.
- MS-CHAPv2 uses two encryption keys, one for transmitted data and the other for received data. MS-CHAP uses one encryption key.
- A password longer than 14 characters cannot be used with MS-CHAP.

 A Windows .NET RRAS server doesn't by default support LANMAN (LM) authentication. That is, the authentication mechanisms used by MS-CHAP and MS-CHAP v2 require an NTLM-style password. If you require support for LM style passwords (to support Windows NT 3.5x and Windows 98 clients using MS-CHAP), you can enable it by modifying the registry. You will need to change the AllowLMAuthentication value at HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\RemoteAccess\Policy to 1.

With the MS-CHAPv2 authentication algorithm, the client requests a connection. The server (RRAS or RADIUS) returns an arbitrary challenge string and a session identifier. The client returns a response that consists of a peer challenge string, the user name, and the one-way encryption of the received challenge string, user password, and session identifier. The server compares its version of what the response should be to the client's response. If they match, the server returns an indication of success or failure and a response based on the peer challenge string, the encrypted response of the client, and the user password. (The connection is dropped if the user authentication failed.) The client authenticates the server response and, if correct, uses the connection provided. If the server response fails to authenticate the client, the connection is dropped.

CHAP

Windows passwords are usually stored in the SAM database or in AD using a one-way encryption protocol. Thus, they cannot be decrypted. During authentication, the same algorithm is used to encrypt the password entered by the user, then used to encrypt the challenge string sent from the server. The server is then able to use its copy of the one-way encrypted password to encrypt the challenge string and compare the result with that returned from the client. If, however, the client does not understand the Windows authentication process and cannot use this protocol to encrypt the user-entered password, it cannot participate in this type of authentication process. An older authentication protocol, CHAP, uses the challenge response process but requires that you enable reversible encryption for the storage of the user's password in the SAM or AD. Doing so weakens the ability of the server to protect the password.

SPAP

SPAP also requires that passwords be reversibly encrypted in the database. SPAP is required for use by connections to some remote access devices. The SPAP client can be used to authenticate to a RRAS server if enabled. It is more secure than PAP, but less secure than CHAP and MS-CHAP.

PAP

PAP uses plaintext passwords. Because anyone could capture this information and reuse it to remotely access your network as an authorized user, the use of PAP is discouraged. The only reason to enable its use on the network is for the use of legacy clients. If you need to enable its use, consider the alternative of upgrading the clients so that you will not have to use it.

In addition to specifying the allowed authentication protocols, you have the choice of allowing unauthenticated access. Although doing so defeats the purpose of requiring user identification for access, it's useful when utilizing certain authorization attributes that are available when RADIUS is used:

- Unauthenticated access is necessary when using Dialed Number Identification Service (DNIS), one of the possible RADIUS authorization attributes. This method allows any connection based on the number dialed. DNIS is provided by most telephone companies. To enable its use, you must enable unauthenticated access, provide a remote access policy that specifies DNIS and sets the called station-ID to the phone number, and you must enable unauthenticated access on the remote access policy.
- Automatic Number Identification/Calling Line Identification (ANI/CLI) is based on the phone number of the caller. It is different than caller-ID authorization, which utilizes a user name and password and is configured to only work if the user name and password matches the account that has been configured to match the phone number. In ANI/CLI, no user name and password is sent. This service provides the phone number of the caller to the receiver and is available from most phone companies. To use this method, unauthenticated access must be enabled on the RRAS server and in the remote access policy for ANI/CLI. A user account is created for each telephone number that will be used. The user ID becomes the phone number. For example, if the phone number used will be 444-4444, the account user ID will be 4444444. In addition, the user identity attribute value of the
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Remote
Access\Policy key must be set to 31. Other registry settings can be made to further refine the use of this method.
- If you chose to allow guest access, you can use unauthenticated access, then modify the registry to include the account name used as a value in the key—
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Remote
Access\Policy\defaultuser identity.

EAP's name implies its usage. It was designed to be extensible by not specifying exactly the authentication method used. Therefore, it conceivably can be modified to use any authentication protocol. Two choices currently exist in Windows .NET, EAP-TLS and EAP-MD5. A third type, EAP-RADIUS, is not really an authentication type at all:

- EAP-TLS—This authentication type is a certificate-based protocol similar to but not the same as Secure Sockets Layer (SSL). It is currently only available in a Win2K or Windows .NET domain environment and is used to allow smart card authentication via remote access servers.
- EAP-MD5—This authentication type operates in a similar fashion to CHAP, except it uses EAP. It is often required for user name and password security system devices. It is a good way to test EAP functionality before attempting EAP-TLS and smart cards because it eliminates the smart card piece of the process to verify that EAP is working.
- EAP-RADIUS—EAP-RADIUS is not an EAP type; instead it centralizes EAP authentication via the RADIUS server. The EAP type passed to the RRAS server is available to pass the EAP authentication to the RADIUS server.

Data Encryption

Authentication protocols encrypt, obscure, or pass the password in the clear depending on the protocol chosen. Data encryption is optional. Two options are available: First, for remote connections, data encryption is available by selection from the Encryption tab if MS-CHAP, MS-CHAPv2, or EAP-TLS is selected. Second, if a VPN is configured, data encryption depends on the type of VPN (PPTP or IPSec). Data encryption for remote dial-up access is accomplished via MPPE.

Data encryption levels that the server will accept are configured within the dial-in profile for the remote access policy (see Figure 7.18). Each remote access policy can specify its data encryption choices, including *No encryption*. The policy can specify, therefore, that only the strongest encryption be used; however, the client must be capable of performing the encryption and an appropriate authentication method (MS-CHAP, MS-CHAPv2, or EAP) must be selected.

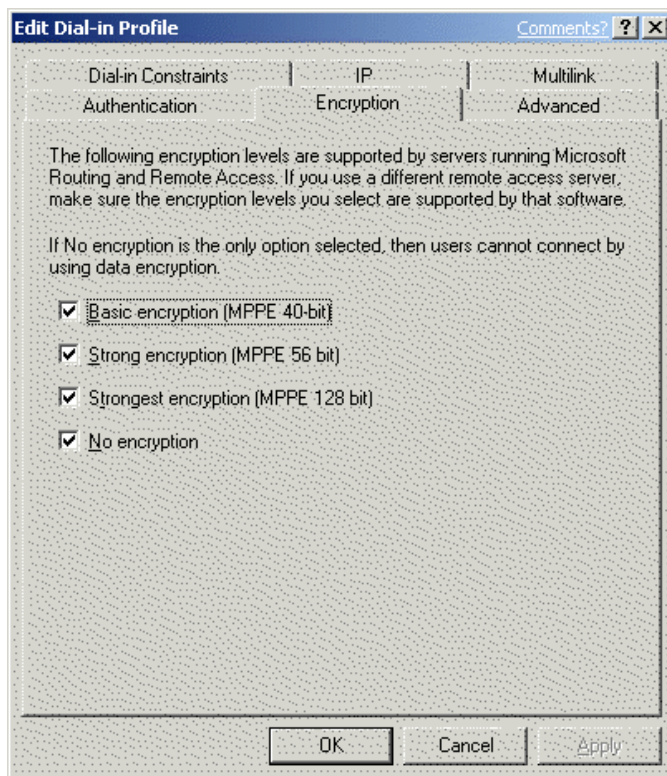


Figure 7.18: Configuring data encryption for dial-up connections.

Client encryption is configured by editing the client profile of the client connection (see Figure 7.19). Your choices are

- Optional encryption (connect even if no encryption)—This setting is the default setting
- Require encryption (disconnect if server declines)—A good client-side control
- Maximum strength encryption (disconnect if server declines)—If the remote access policy on the server does not specify, or if the server is not capable, the connection is refused by the client

- No encryption allowed (server will disconnect if it requires encryption)—For maximum efficiency, minimum security

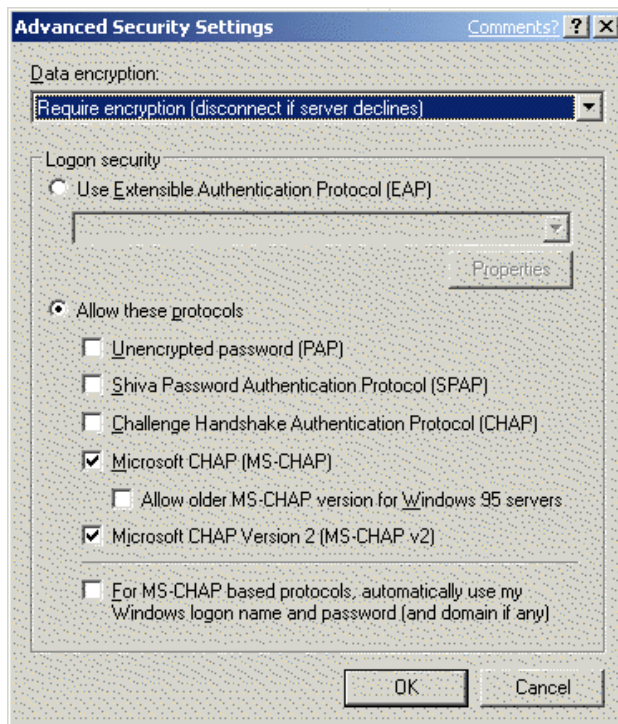


Figure 7.19: Client-side encryption configuration

Remote Access Policy

One of the strengths of Windows .NET and Win2K RRAS is the ability to granularly control remote access authorization. Previous remote access implementations had a simple policy—either you were granted dial-up access or you were not. In addition, the configuration settings of the remote access server were true for every connection. There was no way to specify different authentication, encryption, or authorization attributes.

On Windows .NET and Win2K server, RRAS extends these meager offerings and allows an almost-endless variety of remote access policies to meet sophisticated needs. By default, client access to dial-up connectivity is set on the property pages of the client account to be *Control Access through Remote Access Policy*, as Figure 7.20 shows. When a new RRAS server is enabled, the default remote access policy is, you guessed it, *Allow Access if Dial-in Permission is enabled*. Thus, no access is configured for the new RRAS server.

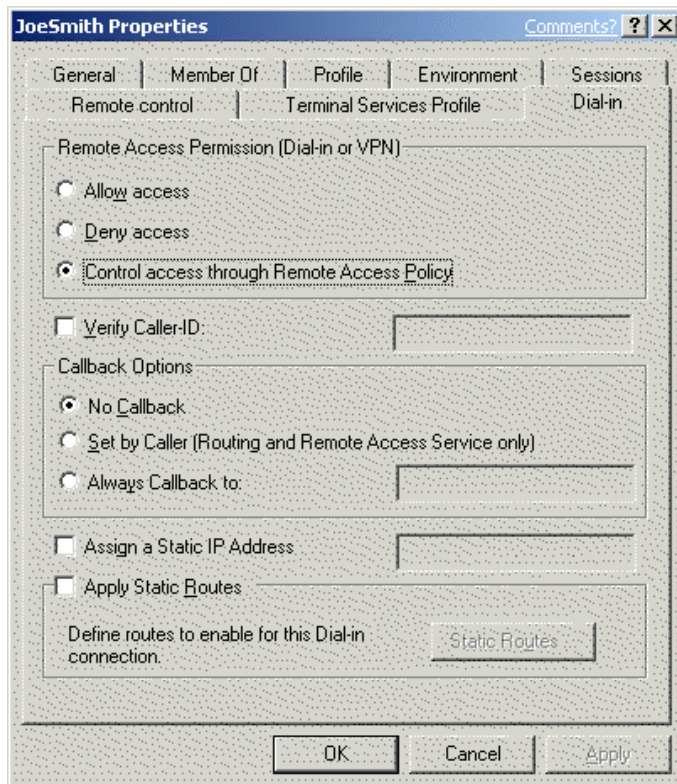


Figure 7.20: Default remote access policy settings.

To provide remote access to your network, you must either set Allow Access on the dial-in properties page of each client account that is authorized for remote access, or you must configure remote access policies. To do so, you create a policy and set the restrictions and conditions under which it will allow access and configure connectivity. Within each policy, you can select the authentication methods and data encryption specifics identified earlier. You also must set the conditions or authorization attributes that determine whether a connection can occur.

Policy Attributes

Within the remote access policy, you configure the acceptable types of connections. A broad range of policy conditions or types are available. Do not confuse these parameters with the authentication process described earlier; these are conditions of access. The authentication protocols selected determine how an individual proves who he or she is. The policy attributes offer a wide range of conditions to restrict authenticated access.

First, although individual user accounts may be selected and given authorization for remote access by selecting Allow Access on their account dial-in property page, remote access policy enables authorization via Windows group membership. Furthermore, a remote access policy can let you granularly authorized access by specifying conditions that must be met for access. A client may be authenticated and authorized for remote access, but if the connection does not meet these conditions, the call will be disconnected. A few of the conditions are available only if RADIUS is being used. Authentication types are configured during the construction of a remote access policy by adding attributes (see Figure 7.21). Table 7.2 lists and describes them.

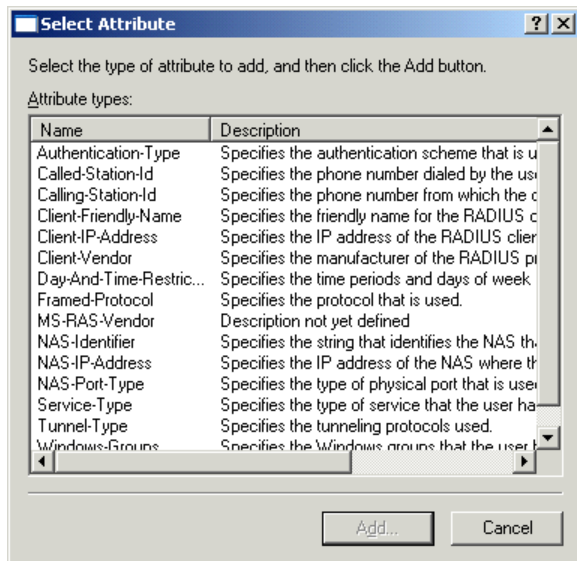


Figure 7.21: RRAS policy conditions.

Type	Windows Authentication	RADIUS	Description
Authentication Types	X	X	Only the authentication method listed is acceptable.
Called Station ID	X	X	Which phone number is dialed?
Calling Station ID	X	X	Which phone number placed the call, or potentially in a VPN, the IP address of the caller (cannot do caller ID and callback together)?
Client Friendly Name		X	The friendly name of the RADIUS client. (The RADIUS client is the RRAS server or other server, not the client computer.)
Client IP		X	IP address of the RADIUS client.
Client Vendor		X	Manufacturer of the RADIUS proxy
Day and Time Restrictions	X	X	A range of acceptable times for connection.
Framed Protocol	X	X	Only the protocols listed are allowed (choices such as AppleTalk, PPP, X.25, and so on).
MS-RAS Vendor			Not yet defined.
NAS identifier		X	A Network Access Server (NAS) vendor string.
NAS- IP- ADDRESS		X	The IP address of the NAS.
NAS Port Type		x	The type of port used by the NAS (port choices include FDDI, 802.11 wireless, ISDN, DSL).
Service Type	x	x	Type of service user has requested (such as callback logon).
Tunnel Type	x	x	Tunnel protocol used (such as PPTP or L2TP).
Windows Group	X	X	Membership in a Windows group.

Table 7.2: RRAS authentication types.

Account Lockout

An additional element of the operations policy for remote access is whether to implement account lockout. In a local network environment, account lockout is configured via the account policy of the domain or standalone server. For remote access, an additional step can be taken. The goal, of course, is to prevent someone from using a password cracker to remotely attack accounts. When set, the remote access account lockout will lockout an account after the designated number of invalid attempts. Account lockout must be enabled via a registry modification if using Windows Authentication.

Installation and Configuration

The RRAS service is pre-installed but disabled on every Windows .NET and Win2K server. To start the service and continue with its configuration, you must enable it. Prior to starting the wizard, determine which options you need. Select Administrative Tools, Routing and Remote Access Service to open the console. Right-click the server, and choose *Configure and enable Routing and Remote Access Service*. On the Welcome page, select Next. From the menu, select *Remote access (dial-up or VPN)*, and click Next. On the Remote Access page, select both Dial-up and VPN, and click Next. Select the use of DHCP if you already have a DHCP server. Provide RRAS with a range of addresses to use for clients and/or allow clients to request an address. In the address dialog box, click Add, then enter the address range. On the *Manage Multiple Routing and Remote Access Servers* page, select *No, I don't want to set up this server to use RADIUS*, then click Next, click Next again, then click OK. After the RRAS service starts, you will need to configure authentication methods, data encryption requirements, and remote access policies.

Chapter 8: Security Tools, Mechanisms, and Emerging Issues

Q. 8.4: I'm always a little leery of using secedit to apply a security template. What if the system is not working the way I want it to after I apply the template?


A: It is certainly possible to make mistakes when configuring security settings. That is why you must thoroughly test changes made in a security template by implementing it in a test environment. However, no matter how carefully a template has been tested, it is possible that some change will wreak havoc on a particular machine. After all, it is impossible to test every possible scenario. The GenerateRollback switch of the secedit command can help you recover from this situation. If this switch is used prior to a security template application, a rollback template is created that you can use to return the system to its former state. Although doing so will have no effect on changes to the permissions settings made in the File and Registry nodes of the security template, all other settings can be returned to the pre-rollback template state.

Here's how it works: When you use `GenerateRollback`, it walks the new template looking for configured settings. When it finds one, it analyzes the current computer's database settings and places the configured setting in the new rollback template. So, for example, if the new template `secure.inf` changes the password length to 8 characters, `secdit` reads the old password length of 0, and places this value in the new rollback template, `rollbacksecure.inf`. No settings are applied during the `secdit` and `GenerateRollback` process. (Your new template must be applied in a separate action, for example, by using a `secdit cfg` command or the Security Configuration and Analysis console.)

After the new template is applied, its changes can be rolled back by using the generated rollback template. However, if other templates have been applied since the original change, the rollback template will not be able to reverse those change made. If you think of the rollback template as a sort of backup, you will not get confused as to what it can do.

To use the `secdit GenerateRollback` command, use the following syntax. The switches are defined in Table 8.6.

```
secdit /GenerateRollback /CFG filename.inf /RBK
SecurityTemplatefilename.inf [/log Rollbackfilename.inf] [/quiet]
```

 `GenerateRollback` does not track registry and file permissions settings. Thus, any changes made using a security template cannot be rolled back using the template produced by the `secdit /GenerateRollback` command. You should always use caution when changing permissions settings and document thoroughly in case you later discover you have made an error.

Switch	Function	Comments
<code>/GenerateRollback</code>	Generate a template that can be used to roll back to the previous state	A good practice is to always generate a rollback template just before applying the template. Doing so ensures that the rollback template was created just prior to the template's application.
<code>/CFG</code>	The name of the template of which to create a rollback template	This is the template you will be applying. (the template must already exist). When the command is run, it will create a reverse of this template and save it with the name entered after the <code>/RBK</code> switch.
<code>/RBK</code>	The name of the rollback template	This is the template that will be generated and can be used to rollback the security settings to the previous state. You can give it any name you like, though I would suggest a meaningful name such as the template name with the word <code>rollback</code> appended. Be sure to specify the path where you want the template stored.
<code>/log</code>	The path where a log of the activity should be located	If no log directory is specified the <code>%windir%\security\logs</code> folder will be used. The file produced is <code>scesrv.log</code> .
<code>/quiet</code>	Activity should not produce comments	This is a useful switch when you want to include this command in a batch file that will run in the background.

Table 8.6 Secedit switches for GenerateRollback.

To produce a simple rollback template and test its function, open a new Microsoft Management Console (MMC) console, and add the Security Configuration and Analysis and Security Templates snap-ins. Create a new security template by right-clicking the Security Templates\%windir%\security\templates node, selecting *New template*, entering a name similar to sample.inf, and clicking OK. Doing so creates a shell template in which nothing is configured. Open sample.inf, and set the password length (Windows Settings, Security Settings, Account Policies, Password Policy, Minimum password length) to 10 characters. Right-click the sample.inf template, and select Save. Right-click the Security Configuration and Analysis node, and select *Open database*. Enter a name for the database, and click OK. Save the MMC console. Open a command prompt, and enter the following command (see Figure 8.13)

```
secedit /GenerateRollback /CFG
%windir%\security\templates\sample.inf /RBK rollbacksample.inf
```

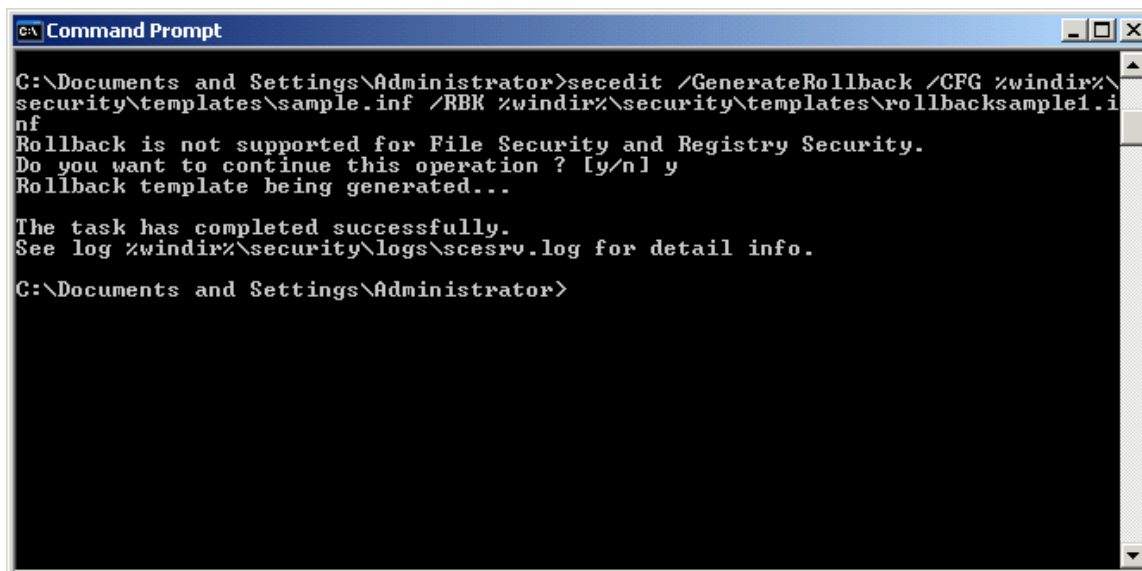
Press Enter to execute the command. When asked if you want to continue, press Y. When the command has completed, check the log at %windir%\security\logs\secsrv.log. At the command-line, enter the following command to apply the sample template

```
secedit /configure /db %windir%\security\templates\sample.sdb
/cfg %windir%\security\templates\sample1.inf /overwrite
```

Use the local security policy to inspect the password policy to determine whether the change has been made. Use the following command to apply the rollback template

```
secedit /configure /db %windir%\security\templates\sample1.sdb
/cfg %windir%\security\templates\rollbacksample1.inf /overwrite
```

Use the local security policy to inspect the password policy to determine whether the change has been made. The password length requirement should be returned to 0 password length.



```

C:\Documents and Settings\Administrator>secedit /GenerateRollback /CFG %windir%\
security\templates\sample.inf /RBK %windir%\security\templates\rollbacksample1.i
nf
Rollback is not supported for File Security and Registry Security.
Do you want to continue this operation ? [y/n] y
Rollback template being generated...

The task has completed successfully.
See log %windir%\security\logs\secsrv.log for detail info.
C:\Documents and Settings\Administrator>

```

Figure 8.13: Note the answer to the file and registry permissions question and the normal, successful completion of the command.