



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 1.6: The financial managers group would like to be able to encrypt meeting minutes and share them. How do I set this up for them?.....	1
Best Practices and Caveats.....	1
How the Sharing of Encrypted Files Works	3
Step-by-Step Process to Secure Shared Encrypted Files	6
Simple EFS Sharing.....	6
EFS Sharing on a Share	8
Chapter 2: Securing Web Services and Web Servers—the Administrative Perspective.....	9
Q 2.6: How are configuration changes made to IIS, and how can I protect the IIS configuration?9	
Ensuring Survival of the Metabase	10
Stopping and Starting IIS.....	11
Backup	11
Modifying the Metabase	12
Enabling the Edit-While-Running Feature	14
Securing the Metabase	15
Chapter 3: Understanding Active Directory Foundations	16
Q 3.6: What steps should I take to secure Active Directory?	16
System Configuration Including Access Control	16
GPO Permissions	17
Certificate Template Permissions	18
DNS Permissions	19
Administrative Practice.....	20

Chapter 4: Fulfilling the Promises of Group Policy	22
Q 4.6: I want a kiosk on the shop floor to have the same user settings no matter who logs on. What can I do?	22
How Loopback Processing Mode Works	22
Configuring Loopback Processing Mode	23
Chapter 5: Administrative Authority	24
Q 5.6: How can I support delegation of services for an <i>n</i> -tier application without creating a huge security hole?	24
Constrained Delegation in Windows .NET	25
Chapter 6: Triple A's—Authentication, Authorization, and Audit	28
Q 6.6: What impact do the Kerberos policy settings have?	28
The Kerberos Process	28
Authentication Processing—the Ticket Granting Service	29
Service Ticket Generation—the Authentication Service	29
Authorization	30
The Kerberos Policy	30
Chapter 7: Remote Access	32
Q 7.6: How can I securely and remotely administer systems?	32
How It Works	32
Applying Maximum Security	35
Using Group Policy to Secure Remote Desktop for Administration	39
Remote Desktop Web Connection	41
Chapter 8: Security Tools, Mechanisms, and Emerging Issues	41
Q 8.6: How can I secure communications between two computers on the LAN?	41
Create the Policy	43
Testing the Policy	47

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 1.6: The financial managers group would like to be able to encrypt meeting minutes and share them. How do I set this up for them?

A: Unlike Windows 2000 (Win2K), Windows XP and Windows .NET can be configured to allow users to share encrypted files. The process is not complex, but requires you to complete a number of steps and employ several best practices to ensure the security of the files as well as provide for data recovery should encryption keys be damaged or lost.

Best Practices and Caveats

A number of security and practical issues should be addressed before implementing a shared environment for encrypted files:

- Education—Administrators and users should be educated in the specifics of file encryption. Although a folder can be set to automatically encrypt files saved or moved to its location, thus making the actual encryption and decryption of data files transparent to the user, several actions might render the supposed encrypted file to be decrypted and therefore available to those who should not have access. A notice is not always given when an action will decrypt a file. Every participant should be instructed about these areas, including:
 - Copying an encrypted file to a drive formatted as FAT (including floppy disks) will decrypt the file (see Figure 1.30).

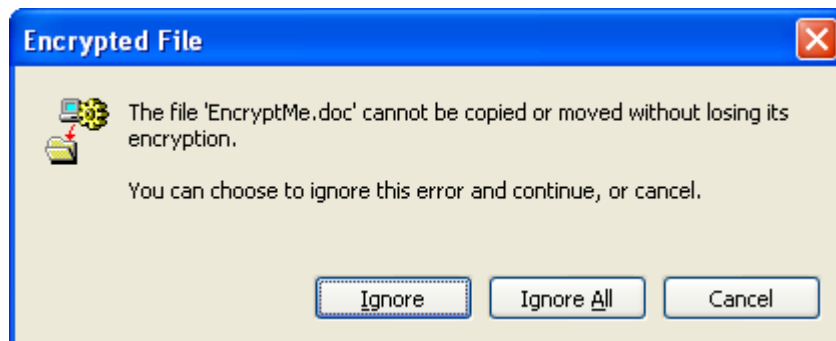


Figure 1.30: Warning that a file will be decrypted if copied or moved to a FAT-formatted drive.

- If an encrypted file is copied over the network to a shared folder, it is first decrypted locally, then re-encrypted at the server (if the server has been configured to store encrypted files). Thus, the file traverses the network in clear text. If the data is captured, it can be read.
- If an unencrypted file is placed in a folder not marked for encryption, the file will not be encrypted.
- If a folder's encryption setting is removed, new files placed in the folder will not be encrypted, but the files already encrypted will not be decrypted.

- Recovery—Windows XP and Windows .NET do not require that a recovery agent be present for files to be encrypted. Self-signed certificates will be issued to any user the first time the user encrypts a file. Thus, unless registry settings (or a corresponding policy) have been changed to disable this feature, any user can encrypt files and the potential for data loss is great because a recovery agent isn't required.
- Recovery agent keys can be created and placed into action. Files encrypted before they are configured will not be recoverable.
- A recovery policy and procedure needs to be in place before files are encrypted. Who will be able to recover files? How should they do so? How should recovered files be handled?
- Key backup—Encryption keys should always be backed up. There have been many cases in which user and recovery agent keys were both destroyed, leaving data unrecoverable.
- Data location—If users are going to share encrypted files, where will the data be located? If the data will be on a network share, precautions for securing the data as it traverses the network should be made.
- In the case of sharing encrypted files, users and administrators need to realize that once a user has shared the ability to encrypt and decrypt the file with another, that user has the ability to share the ability to encrypt and decrypt the file with others as well. File permissions should be used in conjunction with file encryption to ensure that data is not provided to unauthorized individuals.
- File auditing should be established to record file access and determine whether attempts are made by unauthorized individuals. Files should be audited for failure and success: Audited for failure to determine whether any attempts were made, and audited for success to if the attempts were successful.
- The physical location for the files is important, and the server or workstation where they are located should be physically protected. (The specific computers involved should be secured.)
- The users involved should be required to use strong passwords. Remember, if someone can deduce the password and log on as that individual, that person can decrypt the file.
- Recovery agent(s) should be created before the first file is encrypted. A recovery agent is a role represented by a user account and a recovery certificate and its matching private key. The recovery agent role does not have to belong to a specific individual; however, the policy for its management and use should be determined. The recovery agent private key should be archived, removed from the system, and kept in a safe place. The private key is not necessary except to recover files.

It is my opinion that most organizations should disable EFS until and unless they are willing to invest in the establishment and maintenance of a Public Key Infrastructure (PKI). Only with a well-designed PKI can you establish data (or, in the case of .NET, key) recovery considerations. The past years of data loss due to improper management of file encryption keys is testimony to this fact. However, I'm willing to agree that it is possible to safely utilize EFS without an investment in a PKI. Such can occur in a situation in which small, manageable groups require sensitive data management; they are willing to invest in the procedural safeguards necessary to ensure data recovery; they are willing to provide the educational process to prevent unnecessary or inadvertent exposure of data files; and backup of encryption keys is followed religiously.

How the Sharing of Encrypted Files Works

Before setting up secure shared encrypted files, you should know how and why this feature works. This information will help you to trouble shoot problems, and prevent you from inadvertently exposing files. First, make sure you understand the basic file encryption algorithm. When a file is encrypted, a symmetric key is created and used to encrypt the file. It is not reused. Symmetric key encryption uses a single key to encrypt and decrypt. Figure 1.31 illustrates this concept. In EFS, the key used to encrypt the file is called a File Encryption Key (FEK).

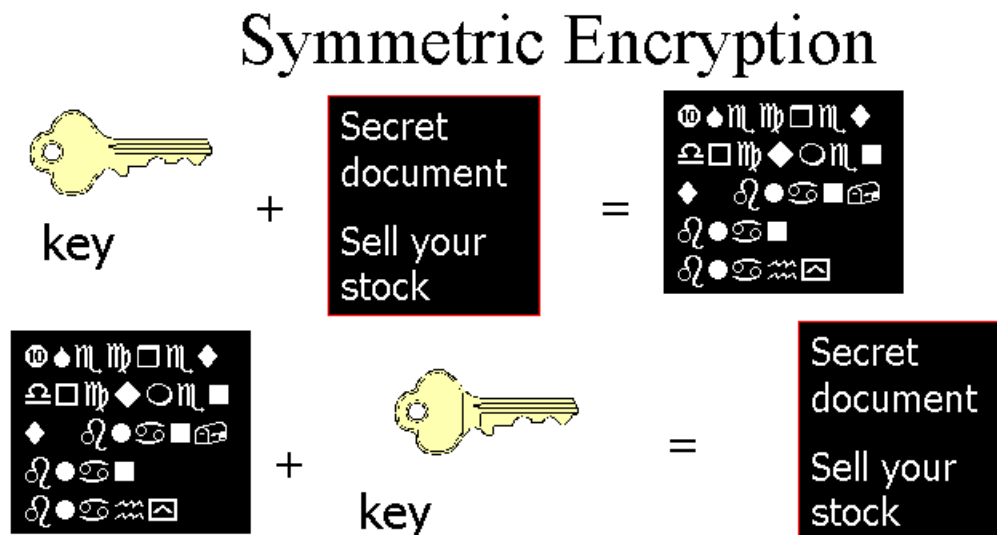


Figure 1.31: Illustration of symmetric encryption.

Next, the public key of the user who has requested that the file be encrypted is used to encrypt the FEK used in step 1. Public key encryption, unlike symmetric key encryption, is asymmetric, that is it uses a key pair. One key is used to encrypt, and the other to decrypt. Figure 1.32 illustrates this concept.

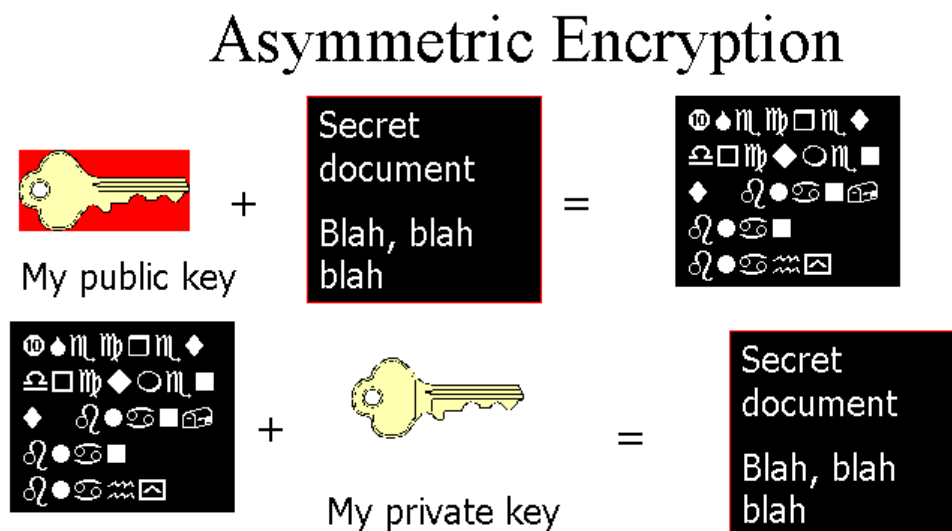


Figure 1.32: Illustration of asymmetric encryption.

The encrypted FEK is placed in a field called the Data Decryption Field (DDF). If, and only if, a recovery agent exists, the public key of the recovery agent is used to encrypt the FEK and the result is placed in the Data Recovery Field (DRF). Figure 1.33 illustrates these processes.

File Encryption

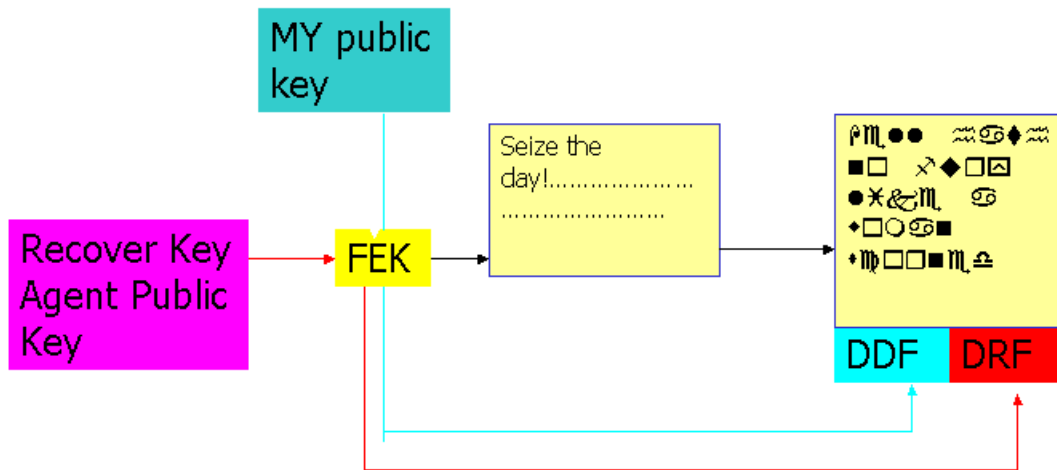


Figure 1.33: Illustration of file encryption.

Now the file is encrypted. To decrypt the file, it is necessary to have the key that is the other part of the key pair for either the user who encrypted the key or, if a recovery agent exists, the recovery agent key. This key is called the private key. The process is as you might expect: The user's private key is used to decrypt the encrypted FEK, which is available in the DDF. The FEK is used to decrypt the file (see Figure 1.34).

File Decryption

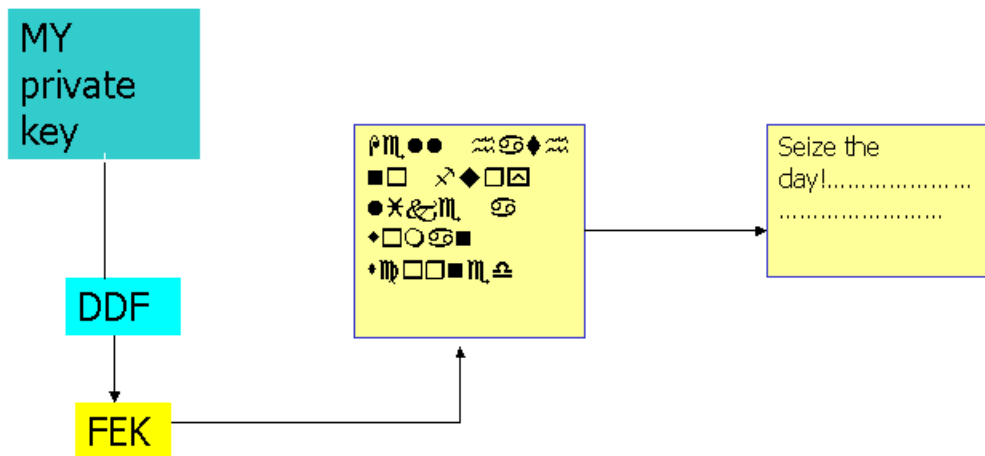


Figure 1.34: Illustration of file decryption.

If the recovery agent's private key is used to decrypt the encrypted FEK, which is available in the DDR, the FEK is used to decrypt the file. Figure 1.35 illustrates the recovery process.

Recovery

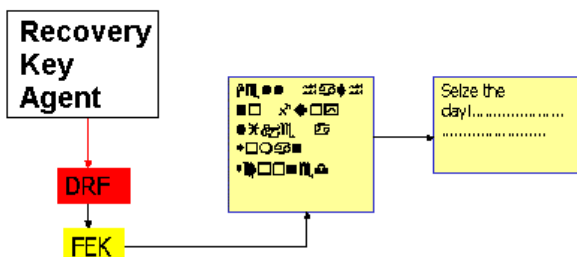


Figure 1.35: Illustration of the recovery process.

Have you deduced a way that multiple people can share encrypted files? You're right if you guessed that you can allow multiple users' public keys to encrypt the FEK, and add the result to the DDF. The ability to share encrypted files in Windows XP Professional and Windows .NET Server is possible because both OSs allow DDFs to contain a FEK encrypted by a different users' public keys.

The only piece of the puzzle left: How does the file system know which users' public keys to use and where to find the keys? Here's how it works: A user encrypts a file. After the file is encrypted, the user can right-click the file, select Properties, click Advanced, then click Details. The encryption property page shows which account was used to encrypt the file as well as which recovery agent account was used (see Figure 1.36).

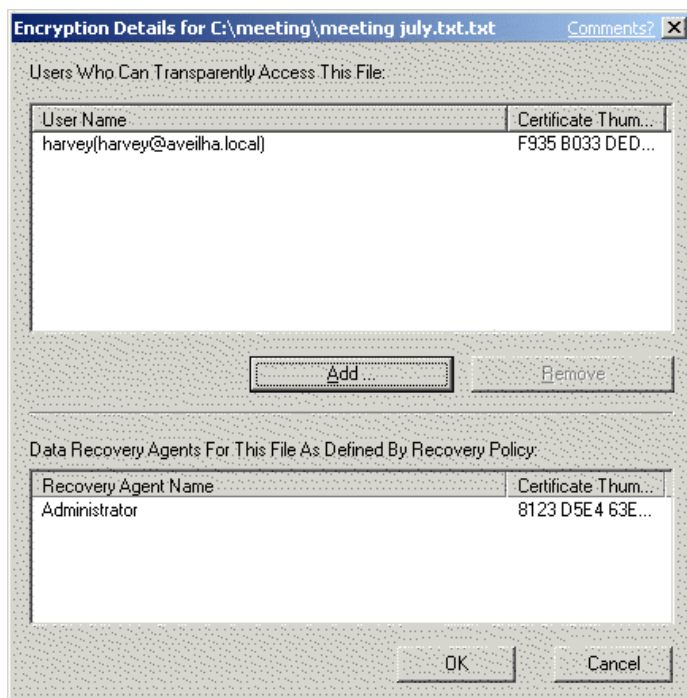


Figure 1.36: Viewing encryption details.



This knowledge is useful, use it! You do not need to be the owner or the one who encrypted the file to see who encrypted it and which recovery agent can be used to recover the file.

By clicking Add, the user can select additional users to whom the user wants to give the ability to decrypt and encrypt the file. A copy of the encrypted FEK is decrypted using the user's private key. (The original encrypted FEK remains in the DDF.) The FEK is encrypted using the public key of the each selected user and placed in a new DDF field.

After the process, information about every user who can decrypt the file is shown on the encryption details page. For this process to work, several things are necessary:

- The user who encrypted the file must select who can encrypt the file.
- The certificates for those users that the original user wants to add must be in that user's Trusted People certificate store.
- The public and private keys of the users who will be encrypting and decrypting the files must be available (typically through the profile).

Step-by-Step Process to Secure Shared Encrypted Files

So, you've educated your administrative staff and users. A policy is written and procedures for use and recovery are in the works. How do you establish shared encryption files for the financial management group? I'm going to detail two scenarios. The first one is simple to set up, though somewhat awkward for users. The second is more convenient for users, but more difficult to set up and protect. (The second scenario involves the use of a network file share, which requires an additional step, securing communication, which is not detailed in this answer. Check out Question 8.6 for more information about this step.) As I previously mentioned, each user and administrator must be educated about the use of EFS as well as potential problem areas.

Simple EFS Sharing

In this scenario, a Windows XP Professional computer, computer Z, is designated as the repository for the encrypted files. The computer is hardened and physically secured. Local accounts will be used. Three folders are designated for use in setting up the process: Folder A is marked as encrypted and will be used during setup. Folder B is not marked as encrypted and will be used during setup and to store a copy of each user's certificate. Folder C will be used to store the shared, encrypted files. Two processes are required, one to set up each user, and the other to share the files. When every user has completed the first four steps and the user in charge of the shared files has completed the rest of the steps, the users will be able to log on and decrypt the file. Each time a new file to be shared is added to Folder C, the user in charge must repeat Step 8 (of the following sequence) before other users will be able to decrypt the file. First, a recovery agent is created. Next, users are set up using the following steps:

1. User logs on to computer Z.
2. User creates a file in Folder A. Because Folder A is marked for encryption, the file will be encrypted and a key pair and self-signed certificate will be created for the user. A self-signed certificate is merely one which is not signed by a Certification Authority (CA).



3. The user exports or archives a copy of his or her certificate, placing the copy in Folder B (instructions for this process follow). It is not necessary to include a copy of the user's private key in this export. This point is also a good time to create an archive of the certificate and private key for backup. This copy should not be stored online; instead it should be placed on a floppy disk or CD-ROM and stored in a safe place.
4. The user logs off.
5. Steps 1 through 4 are repeated for each user who will have access to encrypted files. When a user logs on, a profile for the user is created. A copy of the user's certificate and private key are stored in the profile.
6. The user who will be in charge of the files to be shared logs on and creates a file to be shared in Folder C. All files to be shared will be created in Folder C.
7. The user then opens his or her certificates store, navigates to the Trusted People store, and adds the certificates created in Step 3—one for each user with whom the user will share the encrypted file.
8. The user opens the EFS details page for the encrypted file and adds the users. (Their names will show up as available after their certificates have been placed in the user's Trusted People certificate store.) The user then closes the page. Any designated user can now log on and read the encrypted file.

To export certificates only, the user opens a Microsoft Management Console (MMC), selects Add/Remove Snap-in from the File menu, clicks Add, selects Certificates, clicks Add again, clicks Finish to select the user certificate, then clicks Close. Next the user clicks OK to return to the console, then selects Save As from the File menu to save the certificates console for later use. A good practice on a computer to be used by multiple users is to name the console using the user's initials or other identifying information. The user then right-clicks the user certificate (expand the Personal Store, and select the Certificates folder), selects All Tasks, then selects Export the certificate, and clicks Next on the Welcome page. The user then clicks Next twice to accept the default settings (no private key is necessary for our purposes; however, a private is required for archival purposes, and the user would need to adjust the request accordingly), browses to Folder B, names the file, selects Next, then selects Finish to export the certificate and close the wizard. Finally, the user needs to click OK to close the pop-up message that notifies you that the export has been successful.

To add certificates to the Trusted People store, you open the Certificates console, right-click the Trusted People certificates store, select All Tasks, then select Import, and click Next on the Welcome page. Browse to the location for the certificates stored in Folder B, select a certificate file, then click OK. The certificate is added to the Trusted People store. Repeat as necessary.

To share an encrypted file, navigate to the file in Windows Explorer, right-click the file, select Properties, click Advanced, then click Details. On the Details page, click Add. The users whose certificates have been added to this user's Trusted People store will be displayed (see Figure 1.37).

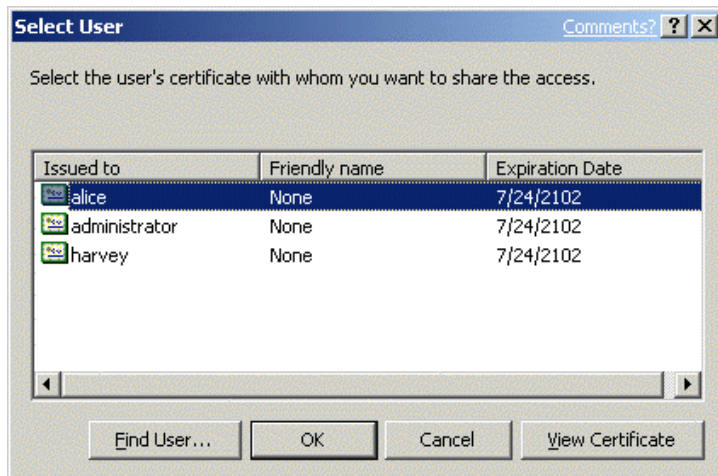


Figure 1.37: Viewing the users' certificates with whom you want to share access to the encrypted file.

On the Add user page, select the user to share the file and click Add. The user is added and his or her certificate is used to encrypt the FEK. The details page now displays the users who can encrypt and decrypt the file (see Figure 1.38).

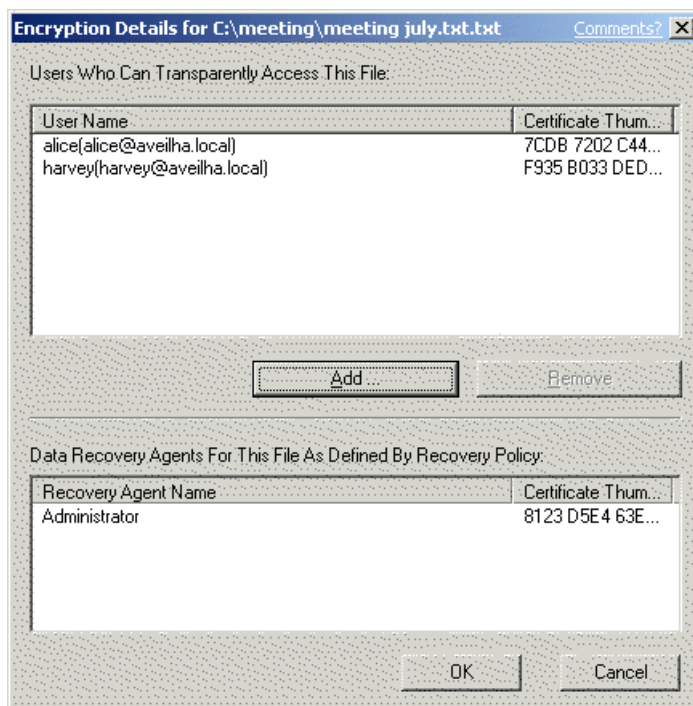


Figure 1.38: Viewing which users can transparently access this encrypted file.

EFS Sharing on a Share

It might be more convenient to store encrypted files on a folder share. However, additional steps are necessary to make this setup work and to ensure the security of the file. First, if the files are available via the network, additional vulnerabilities exist, and care must be taken to ensure the security of the file server, its file shares, and traffic to and from the server. Second, each user will probably be using his or her desktop to access the files. The precautions you should take depend

on the nature of the files, the location of the file server, users, and so on. At a minimum, the shared folder permissions should be restricted to those users who need to have access to the files, file permissions should ensure that files are not deleted or accessible to those other than intended, and traffic between users' computers and the file server should be protected. Communication protection could take the form of a virtual private network (VPN) or IPSec.


It should be clear from the previous discussion that both user certificates and private keys must be available on the file server if users are to connect and encrypt and decrypt files. After that is accomplished, a process similar to that already discussed must be followed (that is the user in charge of the files must include the certificate for each of the group's users in his or her Trusted People store, and the user must add each user to the encryption properties details page of each file the users will be allowed to encrypt and decrypt).

The considerations that result from users' certificates and private keys being available can be resolved by using roaming profiles for these users and by having the roaming profile location be on the file server. The shared folder could include three folders set up like Folders A, B, and C from the previous example. In such a situation, each user would do the following: Log on to the domain. This step creates the user's profile on the file server in the profile folder designated in the user's account. Next, the user would need to connect to the share, then create a new file in the encrypted Folder A. Doing so will create a certificate and private key for the user. Finally, the user needs to export a copy of his or her certificate, and place it in non-encrypted Folder B, then connect to the share and encrypt a file to obtain his or her certificate. The user in charge of the files can then access each certificate and put a copy in his or her Trusted People store, create the files to be shared in encrypted Folder C, and add each user to the encrypted details properties of the new file.

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

Q 2.6: How are configuration changes made to IIS, and how can I protect the IIS configuration?

A: The IIS metabase is the hierarchical store of configuration information and schema for the IIS Web server. Although the IIS console exposes some administrative settings, other items can only be managed by modifying the metabase. If the metabase is missing or corrupt, there can be no Web server. In Windows NT and Windows 2000, the metabase was stored in a proprietary format binary file. This file was difficult to modify and understand. It was not directly readable or editable. It was subject to corruption and was not automatically backed up. Windows .NET replaces the binary file with two XML files. When the Web service is running, the metabase consists of these two files and the in-memory metabase. This setup makes it easier to understand and modify and a revision history is automatically kept. In addition, troubleshooting the metabase and discovering corruption is easier because the plain text file can be easily examined and compared with backup copies written to the history folder.

 Because the metabase is now an XML file, it can be manually modified using text editors such as Notepad as well as with scripts or programs. You should remember that its text is case sensitive.

Ensuring Survival of the Metabase

Unlike IIS 5.0, IIS 6.0 attempts to ensure survival of the metabase, and thus IIS, by creating automatic backups of MetaBase.xml (configuration) and MBSchema.xml (identifies the properties that can be written to keys in the metabase). During operation, a copy of the metabase is in memory. If the edit-while-running feature is enabled, the MetaBase.xml file can be edited while IIS is running. After a predetermined number of changes within 5 minutes, a copy is saved to disk. The file can also be manually edited and saved to disk if the IIS service is stopped, the file is modified and saved programmatically using Active Directory Service Interfaces (ADSI).

A save event is also triggered, regardless of changes made, at 4-hour intervals. If changes have been made, IIS uses temporary files to ensure the availability of the metabase files. The procedure is as follows: IIS locks the in-memory metabase, then increments by one the value of the HistoryMajorVersionNumber property in the in-memory metabase. This number is checked against existing files in the history folder. If the number is already in use, the value of HistoryMajorVersionNumber is increased by one. This process continues until IIS arrives at a unique number. A temporary metabase file copy named metabase.bak is created and given the HistoryMajorVersionNumber. IIS then unlocks the MetaBase.xml file. (New writes to the file are now possible.) If the file is write-locked or read-only, an error message is posted to the event logs and processing proceeds. The metabase.bak file is written to the history folder and renamed Metabase_HistoryMajorVersionNumber_minor version number.xml. If the MetaBase.xml file is newer than the metabase.bak file, an error is posted to the event logs and the process is stopped. If the MetaBase.xml file is not newer than the metabase.bak file, the MetaBase.xml file is overwritten by the .bak file, and the .bak file is deleted. Checks are made to ensure that the metabase is ready for future modification and that the copy being made is current. (The metabase file can be write-locked by an application and no changes can then be made to it. The metabase file is not normally read-only.)



The MetaBase.xml file can become newer than the metabase.bak file if the administrator makes a change to the MetaBase.xml file about the same time that the in-memory file is written to disk.

You can manually modify the MetaBase.xml file, but not the MBSchema.xml file. However, you can make changes to the MBSchema.xml file programmatically. Because the MetaBase.xml file and the MBSchema.xml file are a pair, a current copy of the MBSchema.xml file must be saved to the history folder with the same major and minor version numbers as its MetaBase.xml file. Before this is done, the MBSchema.xml file is checked for pending changes. If no changes are pending, a copy of the MBSchema.xml file is made and copied to the history folder and renamed. If changes are pending, the MBSchema.xml file is renamed to a temporary schema file. The in-memory schema (the one with the changes made in it) is written to MBSchema.xml. The temporary schema file is renamed with the version numbers and saved to the history file.


Stopping and Starting IIS

When IIS is started, the stored copy of the metabase is written to memory. If the file cannot be parsed (missing XML tags, misspelled property name, corruption, or other errors), a copy is written to the history file, an error is placed in the event logs, and IIS will not start.

When IIS is stopped, the configuration and schema files are checked for changes that might have been made since a copy was last written to disk. If changes have been made, a new copy is written to disk and a new history file is created. Otherwise, only a history file is created; no new copy is written to disk. (This configuration reduces the time taken to stop IIS.)

Backup

In addition to the automatic creation of history files, regular backups should be made. When the configuration backup/restore feature is used, backups of the metabase can be restored to the same IIS server or to another installation of IIS on a Windows .NET Server.

 This backup does not back up your IIS server content or your SSL certificate, if one is used. You should also back up your content files and, of course, a copy of the SSL certificate should always be maintained.

Two types of backup are possible:

- Secure backup—An administrator-provided password is used by IIS to encrypt the secure properties and a copy of the password. The rest of the data is in plain text.
- Nonsecure backup—Data is encrypted with a blank password and therefore can be restored by any member of the Administrators group.

To back up, in the Start menu's Run text box, type

```
inetmgr
```

and press Enter to open the IIS Manager. In IIS Manager, click the computer listed under Internet Information Services. From the Action menu, select All Tasks, click Backup/Restore Configuration, then click Create Backup. In the *Configuration backup name* text box, enter a name for the backup file. To make the backup secure, select the *Encrypt backup using password* check box, and enter a password in the Password box and Confirm Password box (see Figure 2.17). Click OK. A set of two files, *namex.mdx* and *namex.scr*, will be created (where *name* is the name you entered and *x* is the version number). If the file name does not exist in the backup location, *x* will equal 0; otherwise, it is incremented each time the same file name is used.

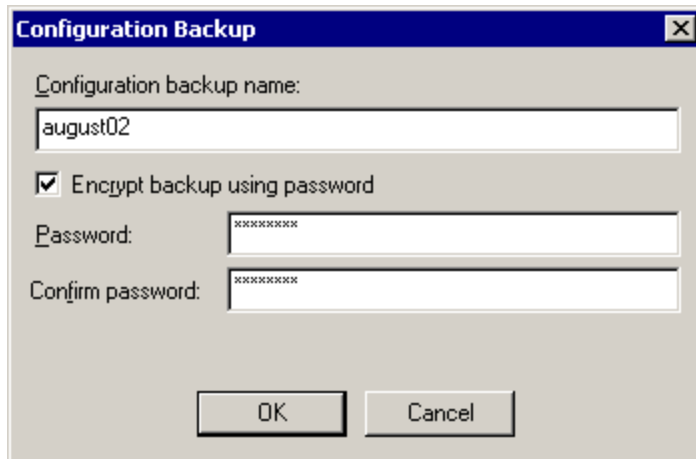


Figure 2.17: Naming and protecting the backup file.

To restore a backup, in the Start menu's Run text box, type

```
inetmgr
```

and press Enter to open the IIS Manager. In IIS Manager, click the computer listed under Internet Information Services. From the Action menu, select All Tasks, click Backup/Restore Configuration, and under Previous Backups, click a file name in the list, then click Restore. If prompted for a password, enter the password. If you use an SSL certificate on your Web server and you restore to a different Windows .NET server, you will need to install the same SSL certificate to the IIS server in order for IIS to start.

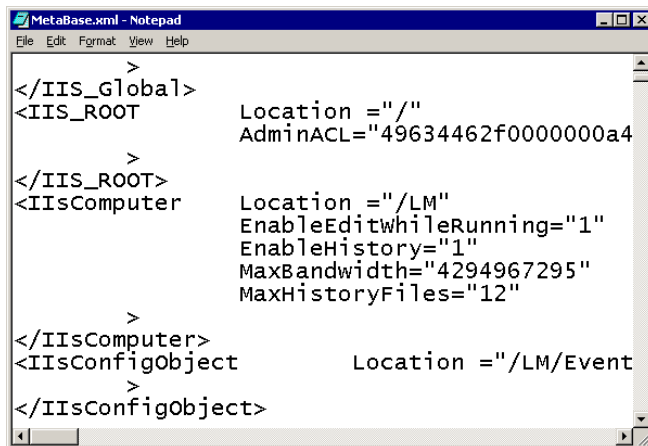
Modifying the Metabase

Modifications are made to the metabase via IIS Manager, ADSI, Windows Management Instrumentation (WMI), COMM DLLs, or executable programs. This process uses the Admin Base Objects layer. As I previously mentioned, while IIS is running you can make changes to the MetaBase.xml in-memory file. This method is a very efficient way to make changes, especially when doing so remotely.

To do so, you must first enable the edit-while-running feature. To safely edit the file, you should understand both XML and the metabase schema. Incorrect edits and improperly formatted edits can result in catastrophic damage to the metabase and prevent IIS from running. The basics of XML and the metabase schema are provided in the Help files:

- Metabase Configuration Files—which defines XML terminology and metabase terminology
- Metabase Property Reference—which details the metabase properties and attributes.

The following instructions detail how to make a simple change, such as editing a common property. To change the property value for the number of maximum history files that are kept, make sure that the edit-while-running feature has been enabled. Open the MetaBase.xml file in Notepad. Navigate to the section of the file at which the MaxHistoryFiles value is located, as Figure 2.18 shows. (You might need to use the find feature in Notepad). Change the value listed to the number you want. (The default setting is 10.) Close and save the MetaBase.xml file.



```

MetaBase.xml - Notepad
File Edit Format View Help
</IIS_Global>
<IIS_ROOT      Location = "/"
                AdminACL="49634462f0000000a4"
>
</IIS_ROOT>
<IisComputer   Location = "/LM"
                EnableEditwhileRunning="1"
                EnableHistory="1"
                MaxBandwidth="4294967295"
                MaxHistoryFiles="12"
>
</IisComputer>
<IisConfigObject Location = "/LM/Event"
>
</IisConfigObject>

```

Figure 2.18: Manually editing the MetaBase.xml file.

When the file is saved, IIS uses the file change notification feature of the file system to trigger changes to the in-memory version. The following process is activated: IIS reads the `HistoryMajorVersionNumber` property value in `MetaBase.xml`. The corresponding history file (the one with the highest minor version number and same major version number in the history folder) is searched for. If the file is found, the `MetaBase.xml` file is parsed, looking for fatal XML errors. If no errors are found, `MetaBase.xml` is compared with the history file and changes are identified. If changes were made (the key to which changes are made is also in the in-memory metabase through the Admin Base Objects layer—see the illustration in Figure 2.19), the changes do not violate the schema, and the metabase is not write-locked, the changes are written to the in-memory metabase. (The metabase might be write-locked if changes are being made at this time programmatically.) A new history file is then written to the history folder. If errors occur during this process, they are written to the event logs and changes are not recorded.

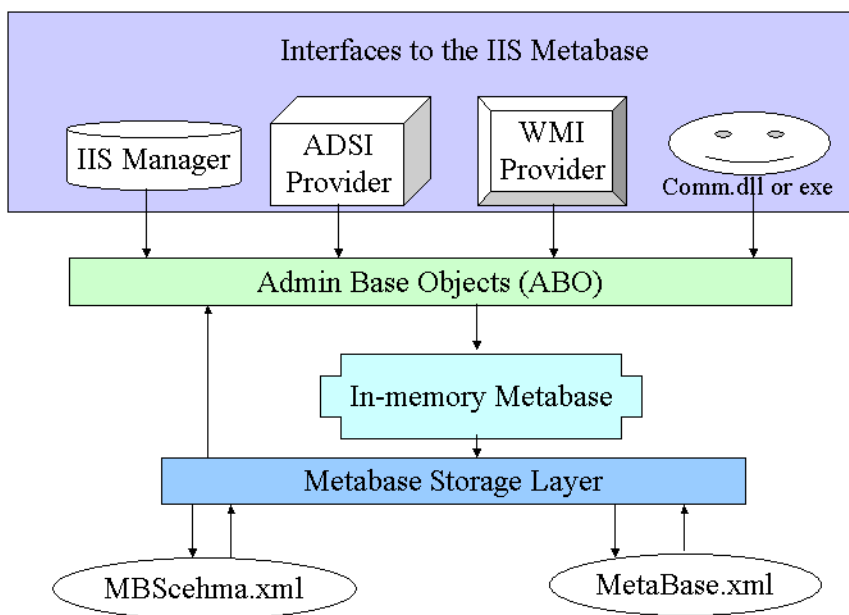



Figure 2.19: Illustration of the process triggered when the modified MetaBase.xml file is saved.

Enabling the Edit-While-Running Feature

To allow modifications to be made to the metabase while IIS is running, you must enable the edit-while-running feature. You can do so from the IIS Manager (IIS will not need to be stopped and started), or by manually editing the MetaBase.xml file and restarting IIS. Metabase history must be enabled to do so (enable history = 1).

To enable the edit-while-running feature using IIS Manager, right-click the local computer in IIS Manager, and select Properties. Select Enable Direct Metabase Edit (see Figure 2.20), then click OK.

 Enabling the edit-while-running feature depends on metabase history to track setting modifications. If metabase history is not enabled (it is by default), you cannot enable the edit-while-running feature. Each MetaBase.xml file is marked with a unique version number and saved in the history folder (%windir%\system32\inetsrv\history) along with MBSchema.xml. A change history is therefore available as well as a backup. The default number of metabase file pairs is 10. (If the number of file pairs has reached 10 and a new file pair needs to be saved, the oldest file pair is deleted.)

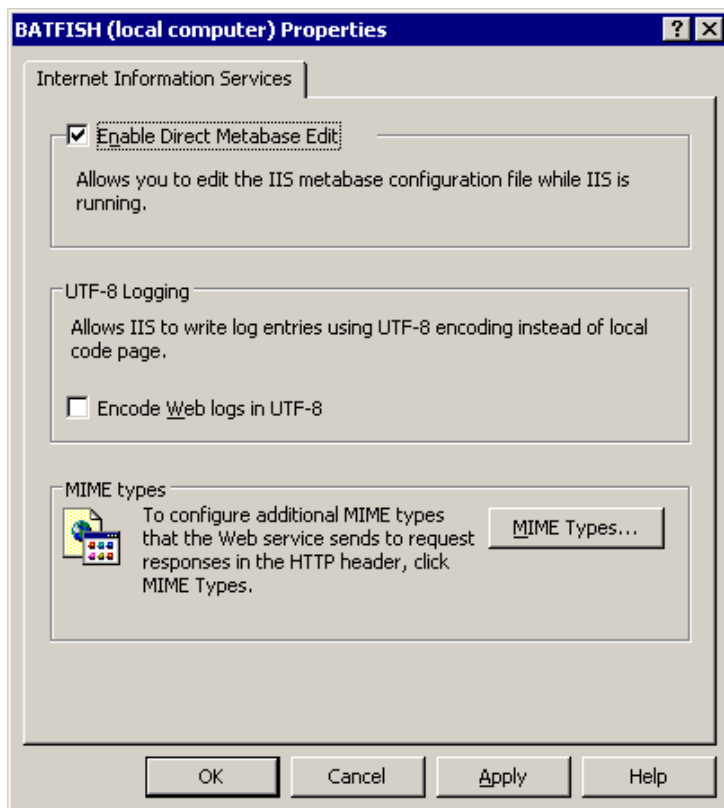


Figure 2.20: Enabling the edit-while-running feature.

To enable the edit-while-running feature by directly editing the MetaBase.xml file, open MetaBase.xml with Notepad from the %windir%\system32\inetsrv directory. Navigate to the IISComputer node, and change the value of EnableEditWhileRunning to 1. Save changes, close the file, and restart IIS.

Securing the Metabase

The metabase survival is critical to the operation of IIS. Thus, you do not want unauthorized changes to be made. To secure the metabase, use sound security practices and include file-level security and encrypted properties. The security recommendations that Microsoft suggests for this purpose are paraphrased in Table 2.6.

Recommendation	Comments
Use complex passwords	Don't write them down
Maintain access control on metabase-related files	For more information about these files, see Table 2.7
Use the concept of least privilege	Only give the minimal and necessary permissions and privileges to users
Do not use EFS to encrypt the metabase	Doing so slows IIS and might cause errors
Copy the MetaBase.xml file before manually editing it	You can restore a good copy of MetaBase.xml by copying the good version of a corrupted MetaBase.xml
Create backups using the backup/restore configuration tool	Do so frequently and whenever changes are made to the metabase
Do <i>not</i> use import and export to backup the metabase file	Sensitive data (such as passwords) is not exported; import and export is meant for transferring sections of the metabase file to another computer
Backup content	Metabase backup does not backup content files
Prevent simultaneous metabase changes	Can cause corruption
Monitor event logs for related IIS event messages	Learn what the error messages mean and how to correct the situation
Set up file auditing on the metabase files	You can identify who has been mucking with, and potentially corrupting, the metabase files; only manual edits of the file are recorded giving the user name; set auditing on tool executables to determine who is using them


Table 2.6: Microsoft security recommendations.

All metabase-related files and folders are secured by the default permission settings, which grant full control to NT AUTHORITY\SYSTEM and BUILTIN\Administrators. To restrict these permissions further, you might want to create a select group of administrators and give them Full Control while removing the local Administrators Group. Related files and folders are identified in Table 2.7.

File/Folder (all are located at %systempath%\System32)	Comments
\Inetsrv\MetaBase.xml	The configuration storage database
\Inetsrv\MBSchema.xml	The schema for the configuration file
\Inetsrv\History\historyfile	The metabase history files
\Inetsrv\MetaBack\backupfile	The metabase backup files

Table 2.7: Metabase-related files.

Because the MetaBase.xml file is written in plain text, several sensitive metabase properties (such as the anonymous user password) are encrypted in the MetaBase.xml file so that they cannot be viewed by unauthorized users. These encrypted properties should not be edited directly. A table of encrypted properties can be found in the Encrypted Properties Help file. You cannot modify the encrypted property of an existing schema property; however, you can use ADSI to add new properties to the schema with the encrypted property set.

 Do not attempt to edit directly the encrypted properties, as there is no way to encrypt your entries before saving the xml file. Encrypted properties should only be modified using WMI, ADSI, or Admin Base Objects.

Chapter 3: Understanding Active Directory Foundations

Q 3.6: What steps should I take to secure Active Directory?

A: Active Directory (AD) provides a security configuration assignment mechanism for the forest. Authentication is controlled there, as are security settings (through Group Policy) for all users and computers in each domain. Security for AD is set and maintained in a number of ways. You should develop a comprehensive, formal security policy that is approved by management. AD security practices can be divided into four areas: system configuration, including secure, default settings on AD objects; administrative practice; physical security; and authentication and authorization.

System Configuration Including Access Control

System requirements for authentication and authorization ensure a secure directory environment for the AD database. Users must authenticate in and have appropriate authorization to access and modify AD objects. Network, as well as local, access must be authenticated. After this authentication is accomplished, each access to an AD object is checked by comparing the privileges and permissions granted to a user and the groups to which the user is a member with the access control lists (ACLs) assigned to AD objects.

Like files and folders, AD objects can be assigned common access properties such as Read, Write, and Delete. However, AD objects also have specific permission settings that are applicable for the type of object they represent. User and computer accounts have a password property, and each user automatically has the Change Password DACL on his or her account (see Figure 3.17). The Reset Password property on all user accounts is set for Administrators, and can be set for other groups or users.

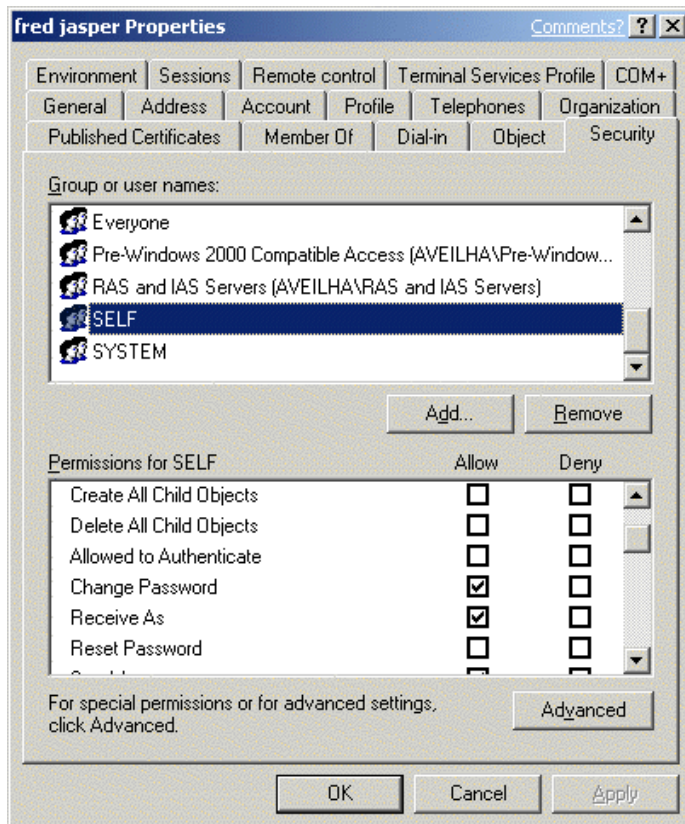


Figure 3.17: Observing Group Policy permissions.

Many AD objects are managed by assigning permissions that have far-reaching effects on the security of AD and other resources within your forest. For example, setting permissions on Group Policy Objects (GPOs) determines who can edit the GPO and the accounts to which the policy applies. In addition, setting permissions on certificate templates determines which users and computers are allowed to request that type of certificate. And permissions on DNS entries determine which computer has the right to change the IP address listed.

GPO Permissions

Unlike simple access to the Security tab of a user's properties page, security on GPOs must be set by accessing the GPO through the property pages of the site, domain, or organizational unit (OU) to which the policy is linked. To access a GPO's properties, right-click the container, select Properties, select the Group Policy page, select the GPO link, and click Properties. Select the Security tab. To specify a security group or user who may modify this group policy link, assign the group Full Control. To specify a security group or user to whom the GPO is applied, assign the group Read and Apply Group Policy permissions, as Figure 3.18 shows.

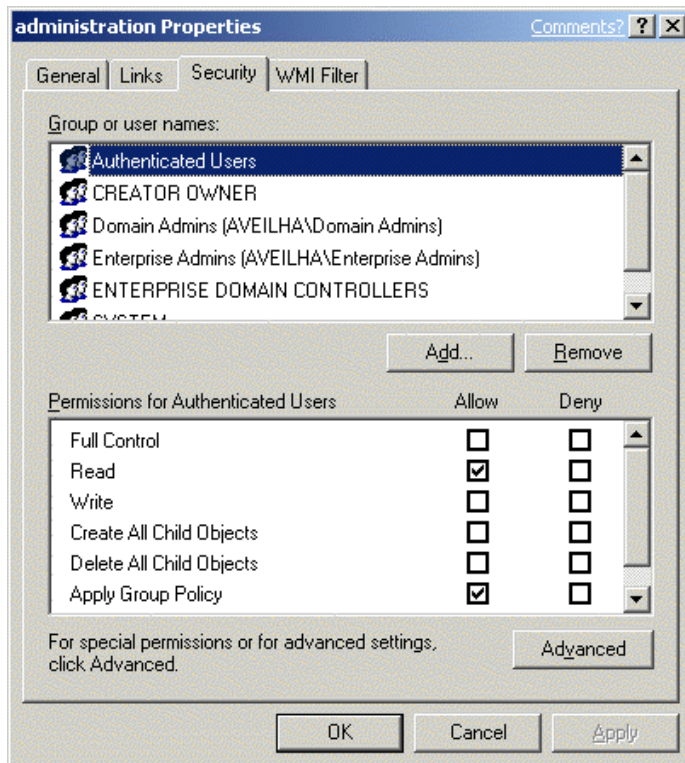


Figure 3.18: Setting a GPO's security settings.

Note that by default, the Authenticated Users group has the Read and Apply Group Policy permissions on each GPO. Thus, any user account that is located within the container to which the GPO is linked will have the policy applied. Should you want to filter policy application, you can substitute the group or groups who should have the policy applied. Be sure to note that other groups require permissions on the GPO.

Certificate Template Permissions

Certificate template permissions are set on the certificate templates, which can be located in the Certificates Template console. The Certificates Template console can be added to a Microsoft Management Console. As Figure 3.19 shows, permissions include Enroll, which means that a user can request and obtain a certificate; Autoenroll, which means an account may autoenroll; Read, Write, and Full Control.

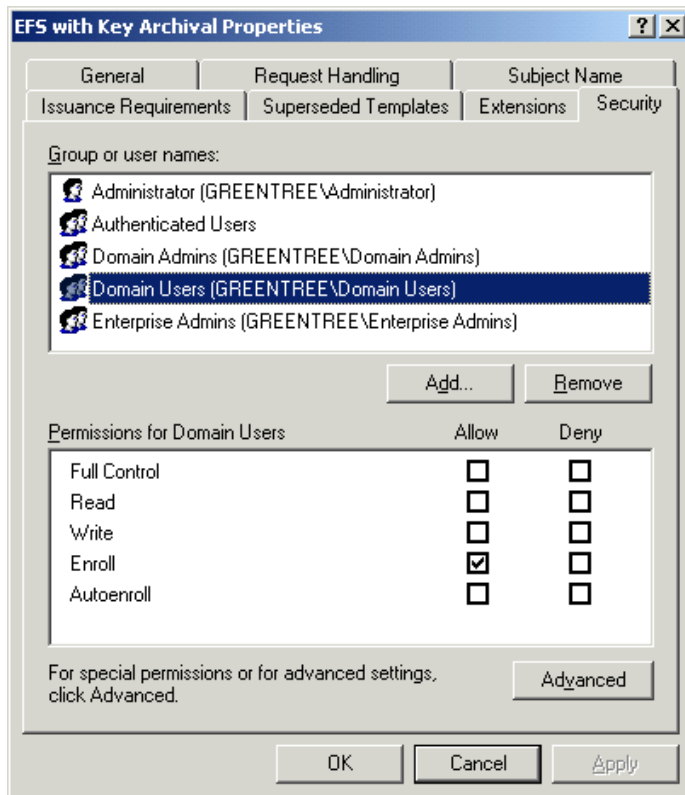


Figure 3.19: Certificate template permissions.

DNS Permissions

When DNS zones are stored in AD, and the zones are set to allow only secure dynamic updates, permissions set on each resource record can control who is allowed to modify the IP address for each record. To evaluate or modify the permission settings, open the DNS console, expand the zone, right-click the resource record, and select Properties. Select the Security tab, and examine the default settings. A wide view of the permissions can be examined by clicking Advanced (see Figure 3.20). To modify the settings, use the Write Permission to identify who can modify the resource record.

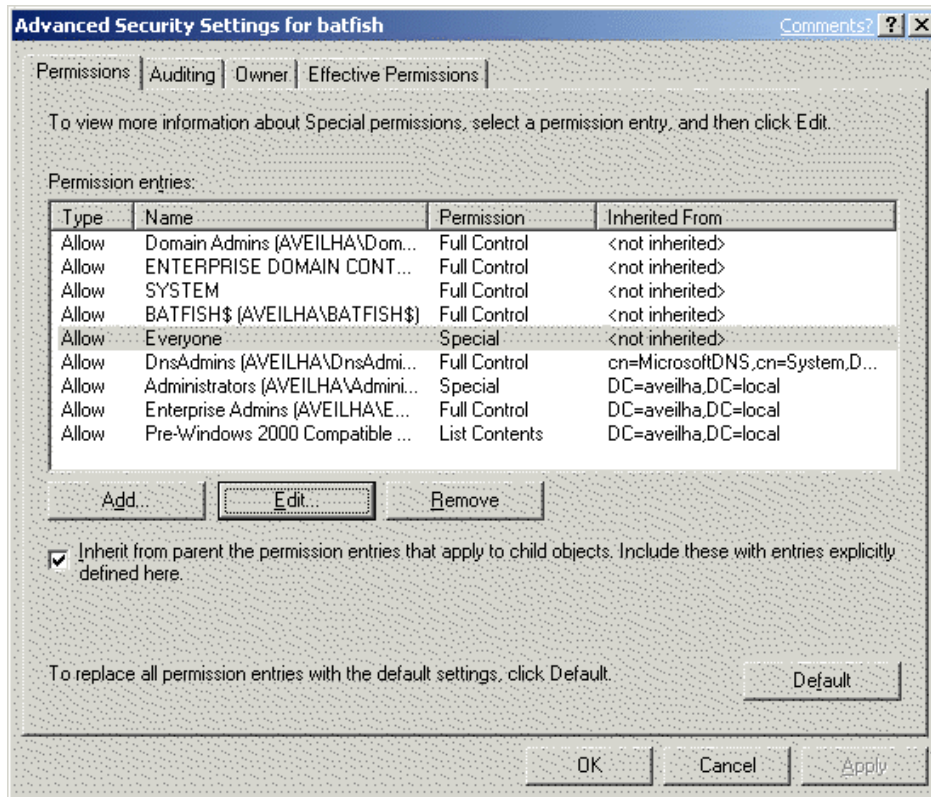


Figure 3.20: Examining resource record permissions. Note that the computer batfish has full control over the setting of its resource record.

Administrative Practice

Microsoft has prepared a best practices list for AD security. In Table 3.3, I've incorporated this list into my best practices list and added comments.

Practice	Comments
Do not log on using your administrative account. Instead use a normal user account and use the Run as utility to run administrative tools.	Using an ordinary account mitigates the risk that inadvertent damage is done either through accident, malicious activity, or the accidental running of malicious code.
Physically secure computers in a locked room.	Pay special attention to domain controllers at remote offices. Special computer rooms are not often available and disregard for physical security is often the rule.
Rename or disable the default Administrator account.	If not disabled, do not use for ordinary administrative work. Each administrator should have an administrative account for accountability.
Manage security relationship between forests.	Simplify by using forest trusts where many trust relationships between domains are necessary. Use external domain trusts to isolate access from forest to forest.
Remove all users from the Schema Admin group. Add a user account only when changes to the schema must be made.	An empty schema admin group can assist in making sure schema changes are approved. Require approval before an account can be placed in the group. That approval should only be granted after a review of the required schema changes.

Restrict user, group, and computer access to shared resources.	Shared resources include files, folders, printers, devices, data, and programs that are made available to network users.
Do not disable the encryption of AD traffic.	By default, all traffic on AD administrative tools is encrypted while in transit. It is also signed. Although you can disable this feature via a registry setting, don't. Audit access to this key and ensure that it is not changed.
Ensure that members of administrative groups are trusted.	Administrative groups are Administrators, Domain Admins, Enterprise Admins, Account Operators, Server Operators, Printer Operators, and Backup Operators. Investigate potential and current members of these groups. Only trustworthy users should be assigned membership and their activity should still be audited.
Establish sites where geographic locations exist that need immediate access to changes to the AD.	AD provides security management information. Establishing separate sites ensures the resources to get the information there quickly.
For each site have at least one domain controller and one Global Catalog (GC) server	If sites must access this information from another site, there may be delays in receiving and applying modified information.
Perform regular backups.	Need I say more?
For each domain, have at least two domain controllers and one GC server.	A single domain controller domain is a risky setup. If the domain controller crashes, how long will it take you to restore from backup? Having an extra domain controller can avoid outages while the other is repaired.
Limit membership in administrative groups.	Use delegation of authority to assign only the privileges to groups that are need by their members.
Audit activity that affects critical AD operations, changes to administrative groups, and security policy.	Auditing provides the accountability you require. You need to know when and who are modifying key files, settings, and group memberships.
Maintain default settings on AD objects unless study by respected researchers proves changes that tighten security.	Windows .NET is a new OS. Microsoft has set the security high on AD objects. Are they perfect? No. And I'm sure time will suggest changes that may tighten security. However, it is unlikely that the ordinary administrator has spent the time with Windows .NET necessary to understand the permissions required for all operations that need to work with AD objects. Modifying these permissions can be a risky undertaking.

Table 3.3: Secure AD best practices.

Chapter 4: Fulfilling the Promises of Group Policy

Q 4.6: I want a kiosk on the shop floor to have the same user settings no matter who logs on. What can I do?

A: Sometimes it is necessary to severely control machines that will operate in public places. The systems are placed there for specific purposes, not for general use. Typical uses for kiosks are in malls, plant floors, and public Internet access systems. In these cases, most users who might step up to the system either use the auto-logged-on user account, which has minimal system access permissions. However, it is possible for such not to be the case; for example, in libraries and computer labs in which each user has an account. In addition, autologon can be bypassed by holding down the Shift key during startup.

In any of these scenarios, it is always possible that a user with elevated privileges on the network might log on to the kiosk, plant floor, library, or lab computer. In this case, the best policy to follow is the one you request—ensure that the user's access to the system from this machine is no different than that of the restricted users. Fortunately, this setup is easily accomplished by using Group Policy and the loopback processing mode.

How Loopback Processing Mode Works

In normal processing mode, Group Policy is applied at the Local, Site, Domain, and organizational unit (OU) level. Settings are merged except when they represent conflicts, in which case the most recently applied settings wins. Computer settings are applied separately from user settings and are the result of settings in Group Policy Objects (GPOs) that are linked to containers within which user and computer accounts exist.

In the typical setting in which loopback processing is required, the computer to be used and most user accounts that may access the systems may reside in the same OU. Settings for computers and users can be strictly set and maintained. However, users with accounts in other OUs will receive the settings made in the policies linked to that OU. This action is not what we want. If loopback processing mode is set, after this user's typical user properties are applied, the local user settings for the Group Policy within which the computer account resides are reapplied. The user's configuration settings now reflect this policy. Two versions of loopback processing mode exist:

- Loopback with merge—User configuration is composed of settings from both the user's usual list and that of the user settings in the GPO that is applied to the computer. In the case of conflict, the computer/user settings win.
- Loopback with replace—The user configuration is replaced in entirety by the GPO list user settings for the computer. Some organizations use the loopback with replace because they want to secure the computer in the same manner for all users.

Understanding loopback processing, although simple in principal, is not all that easy. Figure 4.23 illustrates the concept. In the figure, two scenarios are simply represented. In one, the GPO for the plant OU specifies many user and computer configurations which lock down access to the system as well as the network. The GPO is applied to all computer and user accounts in the OU. Jasper Johnson (JasperJ) has an account at the in the user's container. The GPO in the Plant OU does not ordinarily apply to him. For lack of a more visible illustration, we'll pretend one characteristic of the GPO is red wallpaper. In the A scenario, loopback processing has not been applied. When users with accounts in the Plant OU log on, the computer receives red wallpaper. When Jasper logs on, he does not. Instead, he gets the default blue background and can modify the wallpaper as he wishes. In the B scenario, loopback processing with replace has been applied. Users accounts in the Plant OU receive the red wallpaper. This time, Jasper does as well.

No matter how many configuration settings are applied in the Plant OU, all of them will be applied to Jasper's account if he should log on to a computer with its account in the Plant OU. When Jasper returns to his desktop computer, one that is not in the Plant OU, he receives the settings configured for his account, not the Plant OU GPO settings.

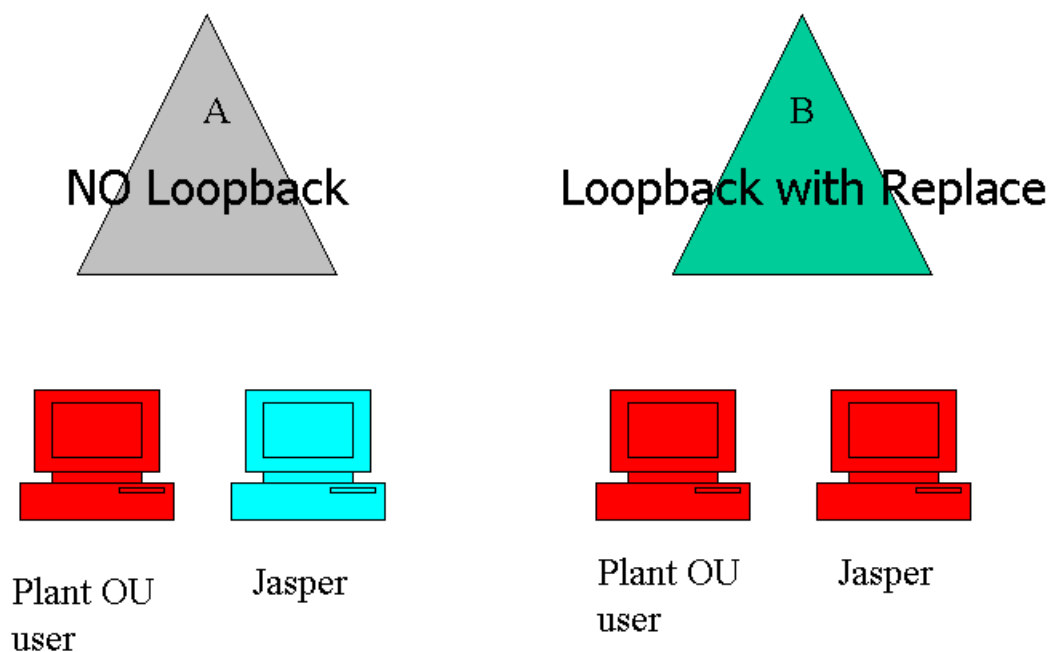


Figure 4.23: Illustration of loopback processing.

Configuring Loopback Processing Mode

Configuring loopback processing is simple: Determine the appropriate restrictive setting for computer and user accounts in the Plant OU. set them in the GPO, and test. Have Jasper log on to see that the settings do not apply to him. Open the GPO, and navigate to Computer Configuration, Administrative Templates, System, Group Policy. Double-click *User Group Policy loopback processing mode*, click Enable, select the mode using the Mode drop-down box (see Figure 4.24). Click OK to complete the changes.

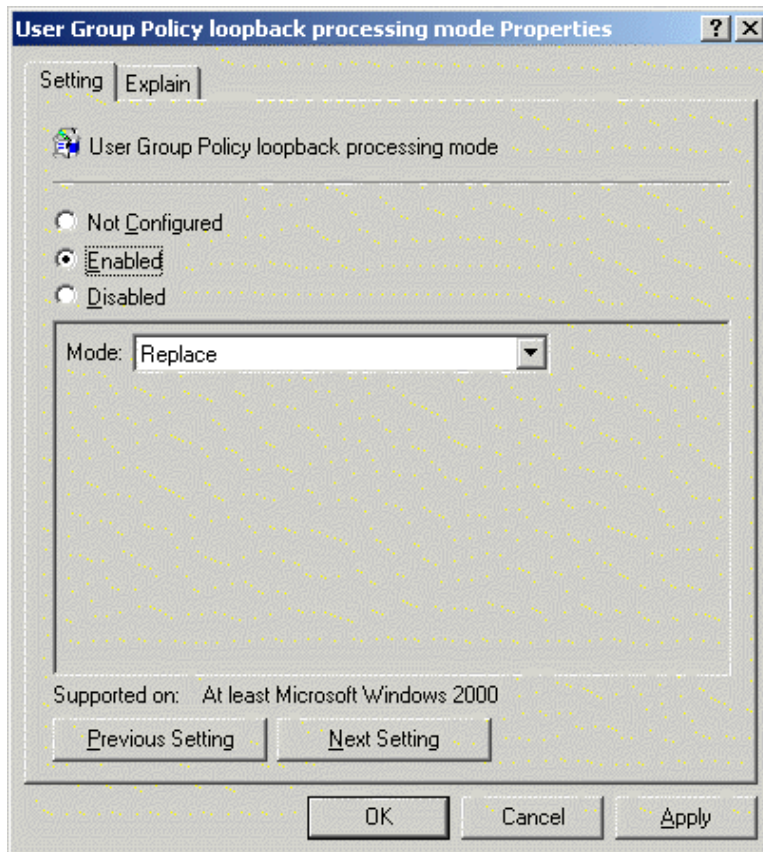


Figure 4.24: Configuring policy loopback processing mode.

Chapter 5: Administrative Authority

Q 5.6: How can I support delegation of services for an n -tier application without creating a huge security hole?

A: First it's important to consider why delegation is necessary in this situation, only then will you be able to select the most secure way to do so. In the typical scenario, an Internet-accessible Web server application requires access to multiple SQL database servers. To do so, the Web server must be able to provide credentials that are acceptable to the databases. No problem, we think, we'll use the domain user ID to assign privileges in the databases. When the user runs the application, the appropriate access will be granted. However, if we try to run such an application, we soon find that this setup will not work. A problem exists because the application on the Web server needs to access the SQL databases, and to do so needs to act as if it is the user. This process, sometimes known as impersonation, is referred to as delegation. (Do not confuse this with delegation of authority, the ability to provide users with administrative responsibility in organizational units—OUs).

Delegation is not possible in Windows NT 4.0, but is, if permissions are set, in Windows 2000 (Win2K) and Windows .NET. The ability to do so in Win2K is based on the Kerberos protocol delegate flag specification. According to the Kerberos standard (as defined in Request for Comments—RFC—1510), a Ticket Granting Ticket (TGT) can be granted with the delegate flag set, which then allows the ability of a third party (someone other than the user who acquired the ticket) to use the ticket to obtain a service ticket for another service. For example, suppose userA uses Kerberos to authenticate to WebServerB. UserA then uses a program on WebServerB, which needs to access data in a database on SQLServerC. UserA does have appropriate permissions for the data that resides in the database. Because the delegate flag is set, WebServerB can impersonate userA and authenticate to SQLServerC. WebServerB obtains data from SQLServerC, and uses the data to complete its processing and return results to userA.

As useful as this feature is, it is often criticized as a security issue in Win2K. The reason is because delegation cannot be restricted. The server that has been trusted for delegation can then act for the user account in any way. Thus, the privileges and access rights that the user holds are available for the server. This feature might be too easy to abuse. Unfortunately, in Win2K, there is no way to restrict this action.

Constrained Delegation in Windows .NET

In Windows .NET, this issue is mitigated by providing the ability to restrict, or constrain, delegation. Delegation can be restricted to certain services. Thus, delegation may only be granted for querying a database. In our example scenario, WebServerB can be trusted for delegation only in accessing SQL Server. If userA authenticates to WebServerB and attempts to access data on SQLServerC, userA will be successful. However, an attempt by an administrator of WebServerB to run a program and impersonate userA to access files on FileServerD to which userA has access, will not.

The choice of whether to trust a computer for delegation to any service is still possible in Windows .NET. However, the ability to constrain this delegation to only certain services is available as well. You can do so by designating a computer or user account that is assigned to a service as trusted for delegation to specific services. To use constrained delegation, the following must be true.

- The computer must be in a Windows .NET domain functionality level
- The computer doing the delegation must be trusted for delegation
- The account that the server is delegating for (userA in our example) must not have the local account policy option setting *Account is sensitive and cannot be delegated* selected.

Implementing constrained delegation is a three-step process. First, determine the computer and/or services to be trusted for delegation. Next, determine sensitive accounts (good candidates are administrator-level accounts) and mark the local account policy option *Account is sensitive and cannot be delegated*. Finally, mark the selected service accounts/computers as trusted for delegation.

To protect sensitive accounts, you can use the following process to mark the local account polity as *Account is sensitive and cannot be delegated*. Right-click the user account in the Microsoft Management Console (MMC) Active Directory Users and Computers snap-in, and select Properties. Select the Account tab. In the *Account options* section, select the *Account is sensitive and cannot be delegated* check box (see Figure 5.9). Click OK to close the property pages. Repeat this process for each account identified as sensitive.

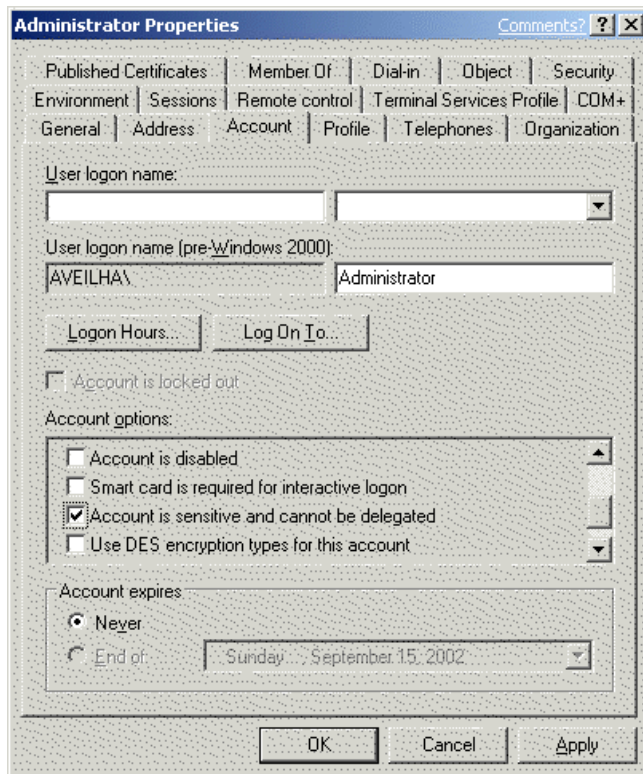


Figure 5.9: Restricting sensitive accounts.

To set a computer/service accounts as trusted for delegation, open Active Directory Users and Computers, click the Computers Container in the console tree, right-click the computer to delegate, and select Properties. This computer is the mid-tier computer, such as the Web server in our example. Select the Delegation tab. Select the *Trust this computer for delegation to specified services only* check box (see Figure 5.10).

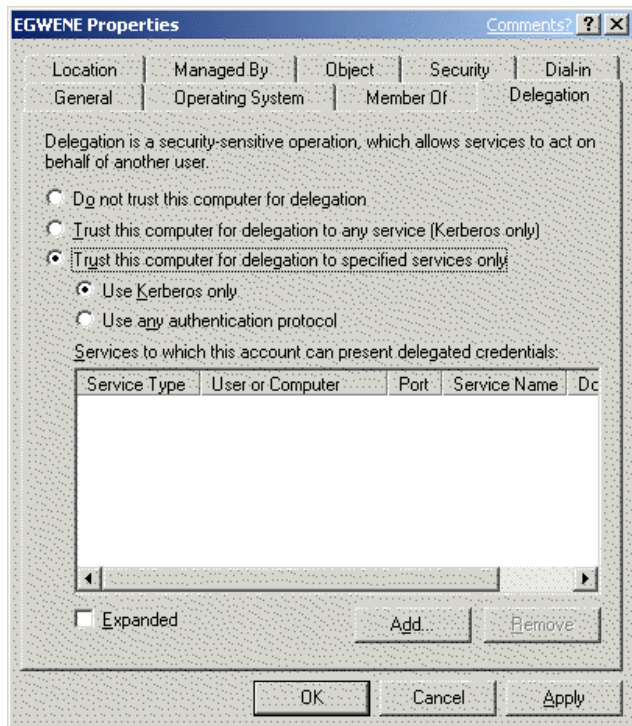


Figure 5.10: Trust the computer for delegation.

Next, select either the *Use Kerberos only* or *Use any authentication protocol* option, and click Add. Under Add Services, click Users and Computers. Under Select Users or Computers, enter the name of the user or computer for which the system will be trusted to delegate. In our example, this would be SQLServerC. In the Available Services section (see Figure 5.11), select the service to be trusted for delegation. (The services available on the computer selected will be displayed.) Click OK, repeat the steps as necessary (multiple computers and services on each computer can be selected), then click OK to close the property page.

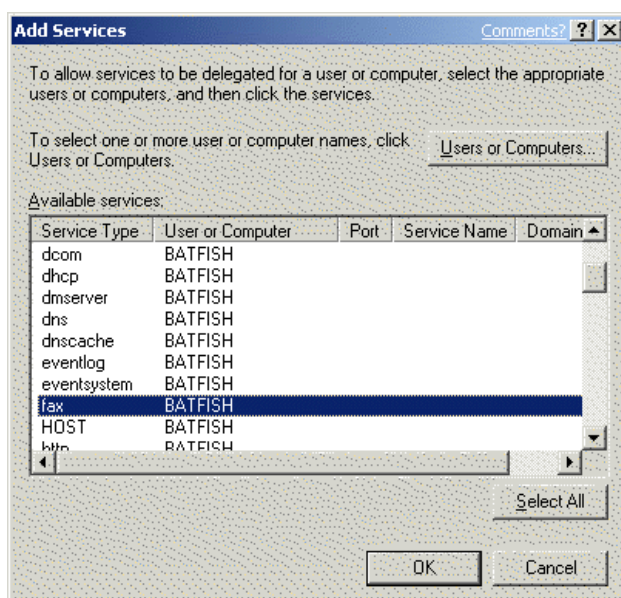


Figure 5.11: Select the service to be trusted for delegation.

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

Q 6.6: What impact do the Kerberos policy settings have?

A: Kerberos policy settings are established in the default Group Policy Object (GPO) for the domain. Current settings can be viewed or modified in the Computer Configuration, Windows Settings, Security Settings, Account Policy, Kerberos Policy container (see Figure 6.17).

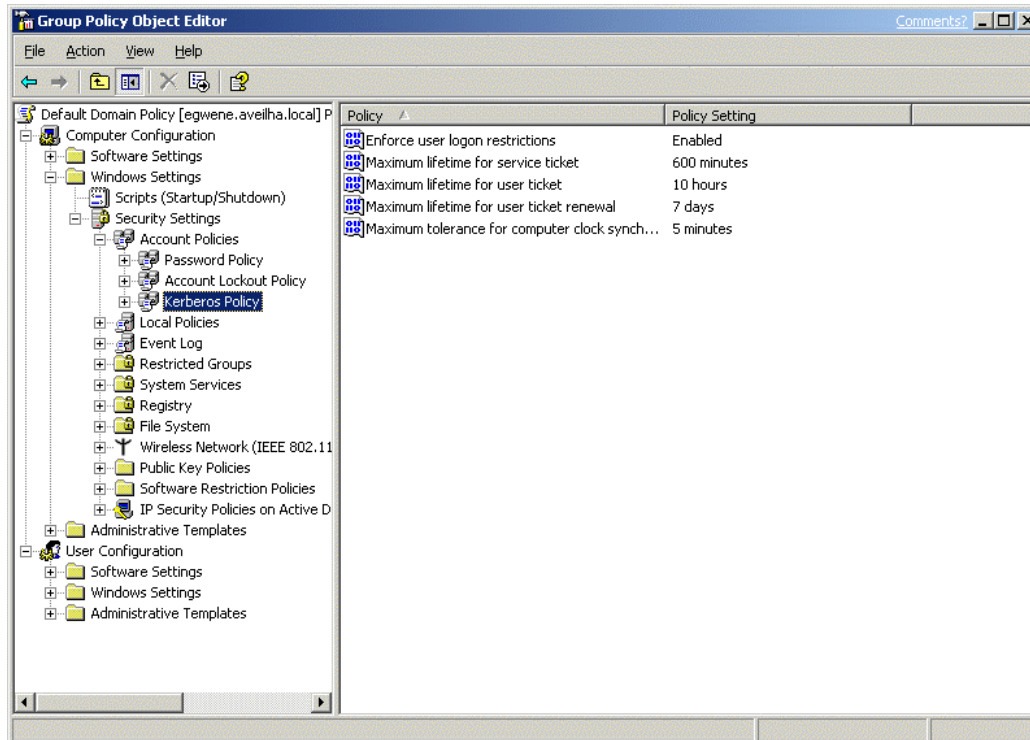


Figure 6.17: Kerberos policy for the domain.

To understand these settings and before considering modifications, it is necessary to understand how the Kerberos protocol is used in Windows 2000 (Win2K) for authentication.

The Kerberos Process

The Kerberos standard is defined in Request for Comments (RFC) 1510 and the Win2K implementation follows the standard. Unlike traditional challenge and response authentication protocols, Kerberos was designed to work in an assumed hostile environment and has several built-in protective mechanisms that work to make it a more secure protocol. Two major sub-protocols form the basis of the algorithm: the Ticket Granting Service and the Authentication Service.

Authentication Processing—the Ticket Granting Service

First the user is authenticated. Credentials are presented and validated against stored data. The successful completion of this phase results in the granting of a Ticket Granting Ticket (TGT). In Win2K, this ticket is often referred to as the user ticket. The follow steps detail this process: The user presses Ctrl+Alt+Delete and is presented the logon screen. The user enters his or her ID and password. The Kerberos client modifies the entered password and uses it to encrypt a timestamp—the current time on the client system. This information, along with a clear-text copy of the timestamp is sent to the Kerberos Key Distribution Center (KDC). In Win2K and Windows .NET, the KDC resides on every domain controller. When a client is booted, it uses DNS to locate an available domain controller.

The KDC examines the clear-text timestamp. If the time varies from the time on the KDC by more than the *Maximum tolerance for computer clock synchronization* (referred to as the time skew), the authentication request is rejected. Otherwise, the KDC uses its copy of the user's password from its account database to encrypt the clear-text timestamp. It then compares its encrypted timestamp with the one provided by the client. If the two match, the client is authenticated.

The KDC prepares the TGT (the user ticket). It includes information about the client and the domain and is signed by the domain controller. Portions of the TGT are encrypted using the KDC's password, and portions are encrypted using the client's password. The TGT is sent to the client, and the client examines the TGT and stores it in its cache.



After this process, often referred to as pre-authentication, the client does not yet have a desktop. Access to any computer service must be obtained by presenting a valid service ticket to the service. After the TGT or user ticket is obtained, this process continues automatically as part of the logon process. The TGT is also used when the user requests access to network services. Use of the TGT to obtain a service ticket is transparent to the user.

Service Ticket Generation—the Authentication Service

After the client receives the TGT, a service ticket must be obtained before the user has access to his or her desktop. To get a service ticket, the following process occurs: The client submits the TGT, an authenticator (a new timestamp encrypted with the user password and a clear-text copy of the timestamp), and a request for a service ticket for the desktop to the KDC. The KDC examines the authenticator, checking against its own time to ensure that any time difference falls within the allowed time skew, and encrypts the timestamp with its own copy of the user password for comparison with the provided client-encrypted timestamp. If everything is OK, the KDC prepares a service ticket. The ticket includes information about the client and the resource (the computer). Some of the information is encrypted using the password of the client, and some using the password of the computer. The service ticket is returned to the client.

Because portions of the ticket are signed using the password of the local computer, the computer can establish its authenticity. The service ticket is also proof to the computer that the client has authenticated to the KDC.





At this point, the client has authenticated to the domain and received a TGT to be used for proving it has authenticated and to request service tickets for access to network resources. The client has requested and received a service ticket that proves to the logon computer that the client has authenticated to the network. However, nothing so far described grants the user authorization to access that resource. The reason is that in order to keep the process description simple, I have left out a portion of the process.

Authorization

Before any resource on a Win2K, Windows XP, Windows NT, or Windows .NET computer can be accessed, appropriate authorization to do so must be obtained. When the NT LAN Manager protocol is used, a list of SIDs identifying the user account and the user's memberships in domain groups and rights is returned to the client during logon. In addition, local group membership and privileges is gathered at this time. This information is gathered into an access token, which can be used by the system to compare with required user or group memberships assigned in resource discretionary access control lists (DACLS). How is this information obtained when the Kerberos protocol is used?

The Kerberos protocol is an authentication protocol. However, the standard identifies an authorization field in the ticket and describes it as the location to be used by the implementer of the protocol to store authorization data. In Win2K and Windows .NET, when the TGT (user) ticket is prepared by the KDC, the authorization information is placed in the authorization field. Thus, the data travels with the ticket back to the client. When a request for a service ticket is made, because the TGT accompanies this request, the authorization data is also available to be included in the authorization field of the service ticket.

So, in our logon scenario, when the service ticket is received by the client system, the user's authorization data for the domain is available. The local group memberships and privileges on the local machine can be retrieved from the local SAM, and an access token can be built. At this point, the normal process can be used to determine the user's privileges on the system. This information is added to the access token and the user obtains his or her desktop. User actions and requests for local resources can be examined in the normal manner using the access token.

The Kerberos Policy

Armed with knowledge of the Kerberos process, we can now examine the Kerberos policy and comment on the effect of modifying it.

- **Enforce User Logon Restrictions (Enabled)**—When enabled, this setting requires that the KDC validates every request for a session ticket against the user rights policy of the target computer. In other words, if the target computer user rights policy denies the user the right to access this computer from the network, and this Kerberos policy is disabled, the check is not made and a service ticket is issued. When this policy is enforced, users must have the right to logon locally if the requested service is located on the local computer, and the right to access this computer from the network if the resource is located on another computer. Administrators may be tempted to disable this policy as this processing takes more time. However, disabling the policy weakens security and should not be done.



- **Maximum lifetime for service ticket (600 minutes)**—The service ticket is issued to the client and stored in the client’s cache. It can be reused if future requests are made for the same network resource. However, there is no reason for the client to store this ticket forever, and doing so would not be a good idea. The longer tickets are stored, the more possibility there is that a malicious user might capture and reuse the ticket. Setting the ticket expiration date is part of the Kerberos specification. By making the default 600 minutes (10 hours), the ticket is valid longer than the typical user session. (After the typical user day of 8 hours, the user should be logging off, which will also delete the tickets). However, should the ticket expire, a new request for a new service ticket is made in the background. The user will not be aware of this request. Changing the number of minutes that this ticket is valid will have no obvious effect that a user can detect; however, making it excessively large serves no purpose and may weaken security. Shortening the time period may gain little in the way of security except in a very hostile environment. If you chose to change the setting, it must be at least 10 minutes and less than or equal to the setting of the maximum lifetime for a user ticket.
- **Maximum lifetime for user ticket (10 hours)**—Likewise, the user ticket must also expire and the time is set at a reasonable level. All arguments and process descriptions given for the service ticket are valid. If the user ticket does expire, the client can generate a request for ticket renewal or for a new ticket without requiring action from the user.
- **Maximum lifetime for user ticket renewal (7 days)**—A TGT may be renewed. However, this setting will limit to a number of days the renewal period. If the renewal timeframe is exceeded, a new request must be made (that is, a ticket that is older than 7 days cannot be renewed, a new ticket must be generated).
- **Maximum tolerance for computer clock synchronization (5 minutes)**—This setting configures is the maximum time difference between the client system and the KDC. This setting assists Kerberos in being resistant to replay attacks. Should an attacker capture the TGT, in addition to decrypting the ticket data and submitting its own request for a service ticket, the attacker would have to create an authenticator. Without a valid password for this specific account, doing so would be impossible. However, the attacker might attempt to reuse the authenticator captured with the ticket. If this is done, the reasoning is that the time difference will now exceed the allowable skew time and the request will be denied.



Ticket expiration date has no effect on the current connection to or use of a service.

Chapter 7: Remote Access

Q 7.6: How can I securely and remotely administer systems?

A: Allowing remote administration risks penetration. Unfortunately, there are few organizations that can insist that all administration take place at the local console for all systems. Perhaps some critical, sensitive systems can be restricted, but for the vast majority, it is necessary to provide a secure way to remotely administer them.

When administering systems on the LAN, Windows .NET encrypts the traffic between the administrator's desktop system and the server when any of multiple administrative tools are used. When remote administration is necessary, more security, as well as the ability to run the server's local administration tools, is required. The local LAN administrator can use administrative tools present on his or her desktop. Kerberos authentication, authorization settings, and encrypted traffic ensure the security of the action. Accessing the server over dial-up lines or the Internet from the other side of the corporate firewall presents different challenges. The protocols present on the LAN will not be available, and the desktop tools will be useless. To correctly and conveniently administer the server, the ability to run the server's local tools is necessary.

With Windows .NET Server, this can be accomplished by enabling Remote Desktop for Administration. Remote Desktop for Administration uses Terminal Services technology but does not install Terminal Server application-sharing, multi-user technology or process scheduling. Nor does Remote Desktop for Administration require special licensing. This lightweight technology is efficient, causing hardly a noticeable effect on a busy server. It can even be utilized by administrators from earlier versions of the Windows OS if Remote Desktop Connection is installed.

How It Works

Remote Desktop for Administration uses the Terminal Services Remote Desktop Protocol (RDP) on port 3389. The user interface (UI) is transmitted to the Remote Desktop Connection, and the mouse movements and keyboard clicks of the Remote Desktop Connection on the client are transmitted to the server. To the administrator, the session looks as if she or he is actually sitting at the remote server's console. While the session is in place, the administrator can also use the local system. Each session, the remote and the local session, operate independently. A second administrator can also connect simultaneously. Each session is also independent, however, the second administrator will see information about the other connection.

Connections can be made via LAN, WAN, virtual private network (VPN), or dial-up connection, and jobs started on the remote server will continue to run even though the administrator has disconnected. Thus, time-consuming jobs—such as backup—can be started by the administrator, who then disconnects. The administrator can later reconnect to see the status of the job. In addition, tasks that normally would not be candidates for remote administration, such as domain controller promotion, can be managed in this manner.

Remote Desktop for Administration is disabled on domain controllers by default, but is easily enabled. To do so, open the System applet from Control Panel, select the Remote tab, and clear the *Allow users to connect remotely to this computer* check box (see Figure 7.29).

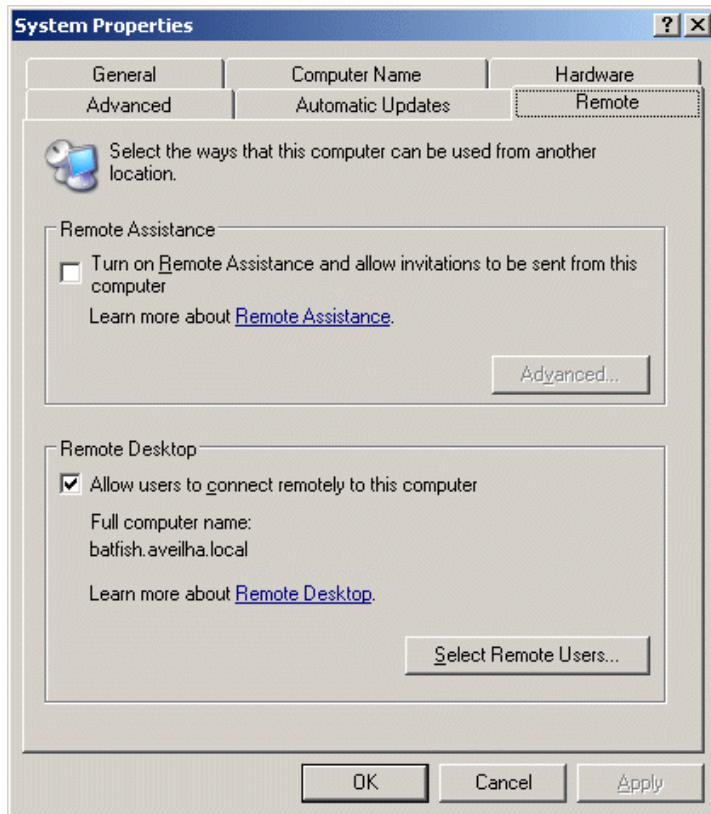


Figure 7.29: Enabling Remote Desktop for Administration.

Click Select Remote Users... Note that while you can add non-administrative users and give them remote access, all members of the Administrators group can connect even if not listed (see Figure 7.30). For this example, we'll click Cancel, then click OK.

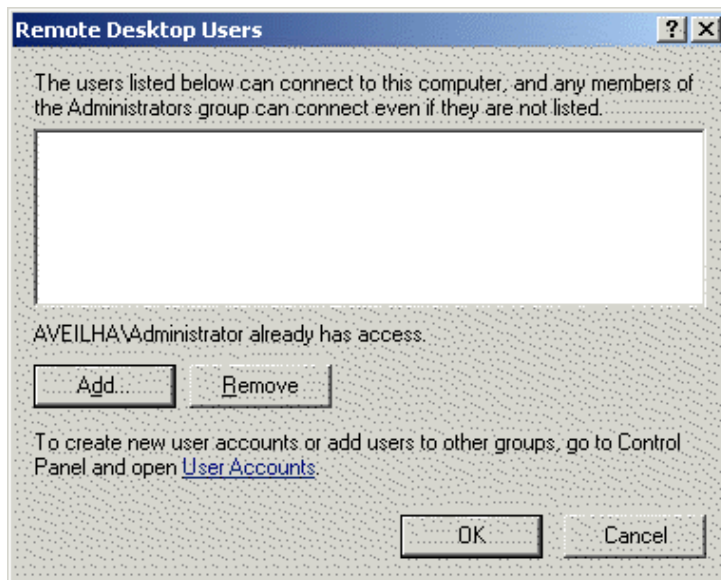


Figure 7.30: Giving users remote access.

To connect and administer the server, use the Windows .NET Remote Desktop Connection by opening the Remote Desktop Connection from the Start menu (click Accessories, Communications, Remote Desktop Connection). Enter the name of the computer or IP address to administer (see Figure 7.31), then click Connect.

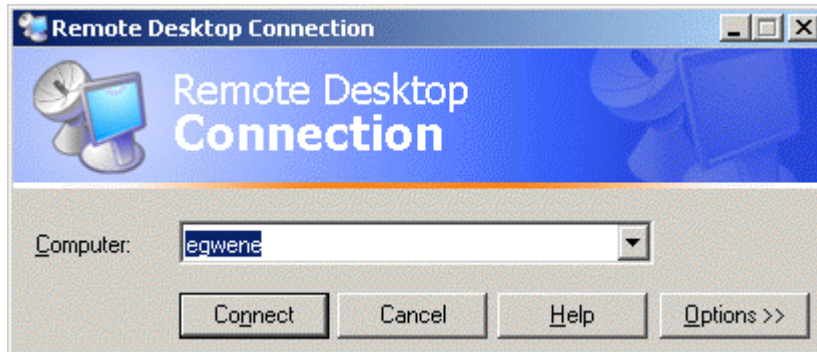


Figure 7.31: Specifying the computer you want to administer.

Log on to the system in the normal manner. A desktop screen that represents the remote server is displayed. You may use any administrative tool. Log off when your administrative session is complete.

If you need to administer multiple computers, instead of using the Remote Desktop Connection, load the Microsoft Management Console (MMC) Remote Desktops snap-in. Doing so will let you store multiple connections and more easily move between remote administrative sessions. To configure the Remote Desktop snap-in, from the Administrative Tools menu, click Remote Desktops, or from a command prompt type

```
tsmmc.msc
```

Right-click Remote Desktops in the console tree, and select *Add new connection*. In the *Server name or IP address* text box of the Add New Connection dialog box, enter the name or TCP/IP address of the server you want to administer. Enter a friendly name in the Connection name box, and clear the *Connect to console* check box if you are just setting up the connection (see Figure 7.32). Optionally enter the user name, password, and domain name, then click OK. Simply repeat these steps to add additional servers to the console.

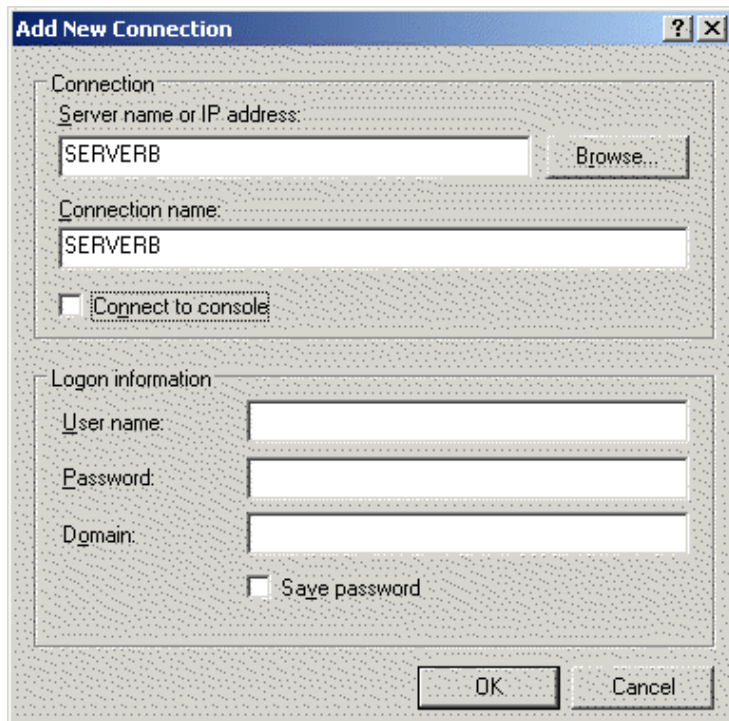


Figure 7.32: Adding a new connection.

Applying Maximum Security

When making the decision to allow remote administration, and when configuring it, you should be aware of the security implications, and make every effort to ensure that security is a primary consideration.

By default all connections to the Remote Desktop Connection are encrypted using the maximum key strength supported by the client. If you want to ensure that only clients capable of the large key size be used to connect, you have the option of changing this setting to High. You do so on the General tab of the RDP-Tcp Properties page (see Figure 7.33). The RDP properties page can be located from the Terminal Services Configuration console by right-clicking the RDP-Tcp icon in the Connections container. Your encryption choices are High, to be used when only 128-bit clients will be used; or Client compatible, which negotiates the encryption strength. Both settings use the RSA RC4 algorithm.

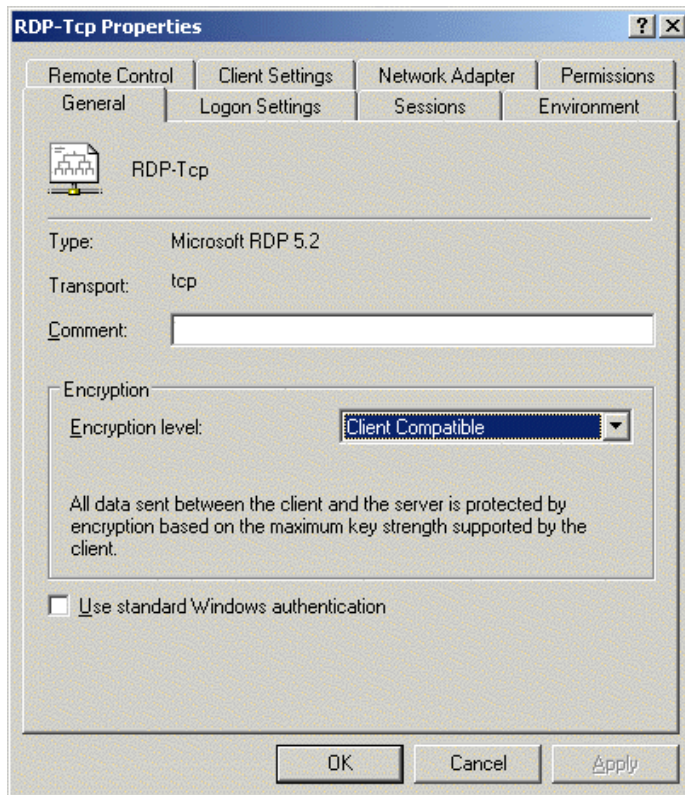


Figure 7.33: Setting the encryption level.

On the Logon Settings tab, which Figure 7.34 shows, you can select either *Use client-provided logon information* or *Always use the following logon information*. If the second option is selected, use the provided spaces to enter user name, domain name, and password. However, selecting this option and providing this information is a very a bad idea. Do not configure these settings! Doing so is the same as allowing anyone to log on who can connect to the server. That individual will have the authority given to the user identified in the logon information and stored here.

Do set the *Always prompt for password* option. This setting prevents the casual interloper who obtains access to the user's desktop from being able to administer servers.

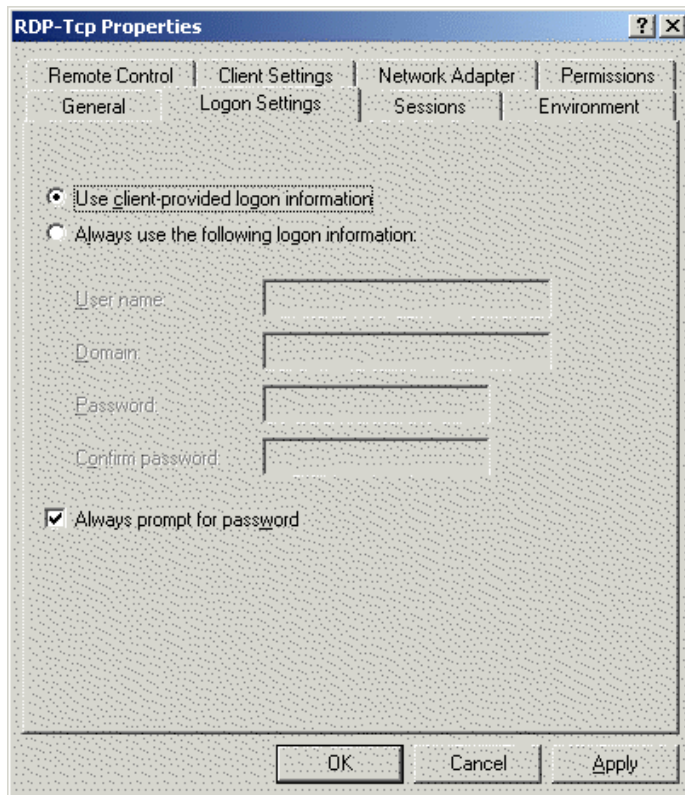


Figure 7.34: Configuring logon settings.

On the Sessions tab, which Figure 7.35 shows, you can control the settings that determine whether disconnected sessions will be entered, whether active sessions have a limit, and what happens if a session connection is broken. In addition, you can control whether a session can be reconnected to only from the previous client or from any client.

Although you might not want to make limitations that restrict administration, where sensitive servers are at risk, it may be that you want to restrict reconnection to occur only from the previous client. Doing so prevents the hijacking of a session, even should an attacker know the username and password of the administrator. Likewise, you might want to limit session time and disconnect idle sessions. However, these settings might interrupt processing if a function is in process that requires an open session and the session disconnects.

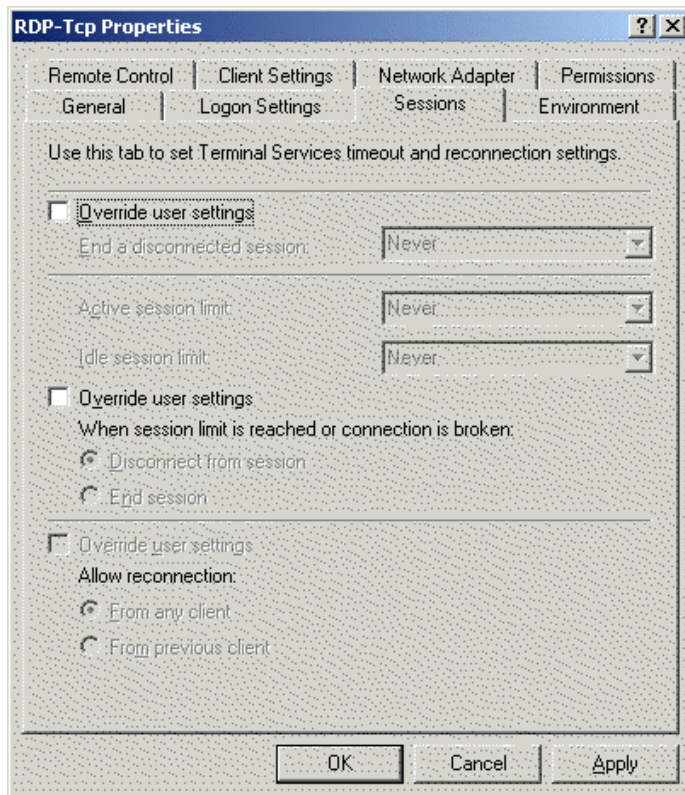


Figure 7.35: Configuration session settings.

From the Remote Control tab, which Figure 7.36 shows, select the *Do not allow remote control* option. Preventing remote control of a user session on a server is important. Should there be a need to provide administrative assistance to another administrator, the local administrator can reset this setting to allow the connection.

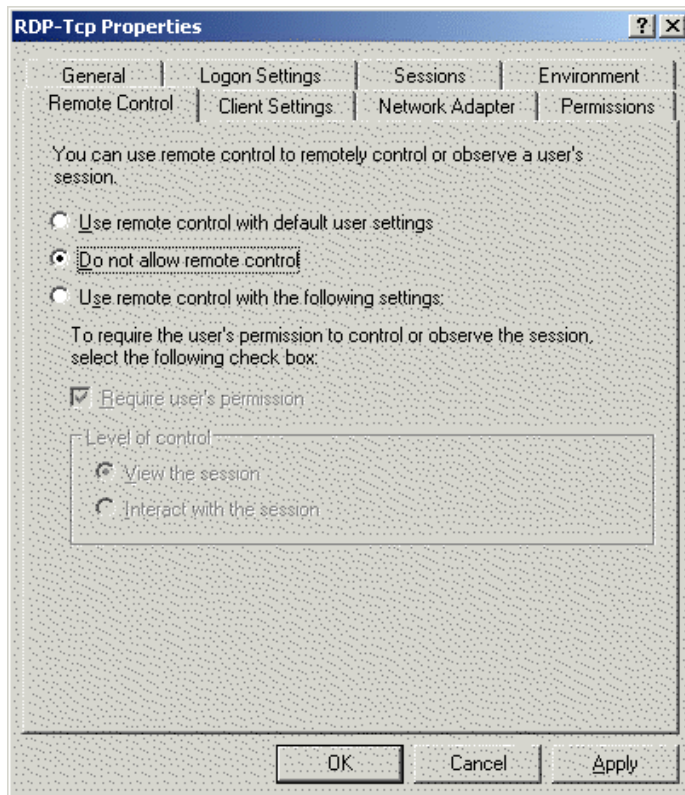


Figure 7.36: Disabling remote control.

Using Group Policy to Secure Remote Desktop for Administration

Configuring multiple servers for remote administration can be a tedious chore. Use Group Policy to reduce the work and to ensure consistent settings across servers. Many of the settings have more to do with Terminal Services for application servers; however, you can use Group Policy for all your terminal server and remote administration settings. You will find that the Group Policy Administrative Templates offer more choices for Terminal Services settings than those available for a standalone system. You can also use Group Policy to apply different settings for different computers and users. Table 7.3 shows some settings that I recommend you change according to your policy.

Setting	Defined	Comments
Client/Server Data Redirection		
Allow Time Zone redirection	By default, session time zone is the server time zone. If this setting is enabled, a capable client (Remote Desktop Connection and Windows CE 5.1) connecting to a Windows .NET terminal server will redirect its time zone information to the server.	Time setting is very important for audit purposes and Kerberos tickets. Time zone redirection should not be allowed for remote administration.

Do not allow clipboard redirection	Allowed by default, enable this feature to prevent users from sharing data between the client computer and the server via the clipboard.	This setting is more relevant for application users. Nevertheless, it should be enabled to prevent using the clipboard during an administrative session.
Do not allow smart card device redirection	Smart card devices are mapped on connection. Enable this setting to prevent this. A remote administrator will not be able to use a smart card to log on to a remote administration session.	Leave this setting alone. It is a good idea to require smart card logon for remote administration.
Allow audio redirection	Allow user to redirect server sound to the client computer.	I know of no security reason not to allow this.
Do not allow COM port redirection	COM port redirection allows redirection of server data to client COM ports	COM port redirection may be necessary for administrative tasks.
Do not allow client printer redirection	If this setting is enabled, it prevents print jobs origination at the server from being redirected to the client computer-attached printer.	Policy dependent.
Do not allow LPT port redirection	If enabled prevents LPT port redirection.	Policy dependent.
Do not allow drive redirection	Prevents mapping of client drives to session. By default, client drives are mapped.	Policy dependent
Do not set default client printer to be default printer in a session	Disables the automatic remapping of the default printers. If not set, it automatically maps local computer printer as the default printer to be used.	Policy dependent.
Encryption and Security		
RPC Security Policy/secure server – (require security)	Requires secure RPC communication, allowing only authenticated encrypted RPC requests. If not set, allows unsecured RPC.	RPC connections are used to configure terminal server services.
Always prompt client for password upon connection	If not set, client may configure his or her system to use a recorded password.	Set to prevent just anyone from using the stored credentials to administer the server.
Set client encryption level	Set as conditions require.	Set at high if possible.
Sessions		
Set time limit for disconnected sessions	More specific for client sessions than remote administration.	Consider effect on long administrative processes before setting.
Set a time limit for active Terminal Services sessions		

Allow reconnection from original client only	Prevents connection to an in-progress session from other than the computer that started the session.	Prevents hijacking of a session.
Terminate session when time limits are reached.	More specific for client sessions than remote administration	Consider effect on long administrative processes before setting.
Client Only		
Start a program on connection	Application mode relevant.	Relevant only for application mode.
Set rules for remote control of Terminal Services user session	Allows remote control if enabled, does not need user's permission.	

Table 7.4: Recommended settings to change according to your company's policies.

Remote Desktop Web Connection

The Remote Desktop Web Connection allows Terminal Services connection from any Web browser. It requires, however, that a special Web page be loaded on the machine to be connected to; in essence running a Web server on the computer. When a connection is made to the Web page, a special client control is downloaded to the client machine if it does not have the Remote Desktop Connection utility. The client still needs to log on. This control could be used for remote administration purposes, but places additional risk on the server because a Web server is now running on the system, and because the user does not require a client.

Chapter 8: Security Tools, Mechanisms, and Emerging Issues

Q 8.6: How can I secure communications between two computers on the LAN?

A: As you know, most computer-to-computer communication can be captured and the data read by anyone who has access to the network. Years ago this was less of a problem—less data of a sensitive nature traversed the LAN, and protocol sniffers were hardware-based and expensive. Now we give no real thought to the nature of data that is available, and sniffers are cheap, even free, and knowledge about how to use them is very widespread. This does not mean that someone is taking the time and trouble to monitor communications on your LAN, but that you should give some consideration to protecting internal communications. You cannot assume that data is safe, or that nothing of value is exposed.

Windows .NET, Windows 2000 (Win2K), and Windows XP Professional can use IPSec to protect communications on the LAN. By creating and assigning IPSec policies on two or more computers, you can ensure that communications between them are secured using encryption, integrity, mutual authentication, and so on. You can specify all communications as being effected, or opt to trigger policy negotiation and thus protection, for only specific protocols or IP address ranges. I'm going to outline a scenario and step you through implementation while defining terms as I go. You can adapt my scenario to yours by adjusting the filters, authentication, encryption, and integrity parameters to those you and your company require.

In my scenario, I've decided to follow through on the example in Question 1.6, in which an administrator needed to setup sharing of encrypted files. The decision to place the files on a file server has been made. The file server has been hardened, the share restricted, and the encryption and sharing process tested on sample user accounts and files. What remains is to put into place encrypted and authenticated communications. We know we'll eventually have multiple users connecting, but to start, we'll build the policy and test it between one user and the file server. After this setup is up and running, it's a simple process to place the IPsec policy on an organization unit (OU) in which computers used for communication will have their accounts.



It's important to remember that communication takes place between two computers. The computers mutually authenticate, not the users.

Before creating the policy, we have some decisions to make. For each decision, I'll give you my recommendation. It's important to make these decisions before creating the policy. Policy parameters must match or communication won't work. (I'll explain where you can find the following policies later in this tip.)

- **Authentication**—Three choices are available: shared key, Kerberos, and certificate. Shared key is primarily for testing purposes. It lets you eliminate Kerberos or certificates as the cause of failure. It's a good practice to test the policy first with shared key, then move to another authentication. If you're comfortable and both computers are members in the forest, Kerberos can be used at startup. Certificates are not difficult to use, but each computer must have one and must trust the root source of the other computer's certificate. Be sure to test your policy with a shared key before going with certificates. The rationale behind this recommendation is that the shared key is visible in the GUI for any administrator to see and visible in clear text when certain troubleshooting tools are used. Using it is sort of like writing down your password on a post-it note and putting it on your monitor.
- **Encryption**—Choices are DES or 3DES. The important consideration here is that for each phase, the policy on each machine matches the other. If 3DES is chosen for the file server policy and the client machine is set at DES, negotiation will fail and no communication will take place. Because default choices create a list that includes both types (negotiation will settle on the highest strength that both computers can do), I'd leave this setting alone. After the policy is tested and working, it can always be tweaked to remove the weaker choice.
- **Integrity** ensures that each packet has not been tampered with and, because it is signed by the sending computer, authenticates that the packet came from the correct source. Choices are SHA1 or MD5. Once again, I recommend leaving the default setting.
- **Diffie-Hellman group**—During Main Mode (the first part of negotiations), a master key is calculated using the Internet Key Exchange (IKE) algorithm using large prime numbers in the calculation. The Diffie-Hellman group specifies the relative size of these prime numbers. The larger the prime, the stronger the key, the longer time encryption takes. Time is not the only consideration here; like authentication, integrity, and encryption, each policy must match, or negotiation will fail. I recommend leaving the default setting.

- **Key regeneration**—A master key is created in Main Mode or the IKE phase. The master key is used to create the session keys—the keys used to encrypt the data. The session keys can be used for the entire session or can be changed every so many packets or minutes. Likewise, the master key can be regenerated over time. The principal is that frequently changing keys keeps more of the data secure. If an attacker should decrypt a key, that key will only be good on a small part of the data. Once again, the default settings are adequate for our purposes and a good place to begin testing.
- **Filter**—The main consideration is to determine when the policy will be brought to bear. Our choices are to trigger the policy based on a specific protocol, specific IP addresses, or both. In many cases, the IPSec policy is developed to secure certain types of communication (such as Telnet or that which is generated from or to a specific computer). In our case, we could establish the policy to ensure encryption between the two computers only for file transfer. For our test here we'll set a single filter based on IP addresses. You can expand the filters and make the encryption more specific if you choose.

Create the Policy

Now we are ready to create the policy. After we do so, we'll test it. To create the policy, open the Administrative Tools menu, select Local Security Policy, right-click the IP Security Policies on the Local Machine, and select Create IP Security Policy. On the Welcome page, click Next, enter a name for the rule, and click Next again. Clear the *Activate the default response rule* check box, then click Next, and click Finish. Select the General Page to examine IKE (or Main Mode) policy settings, then click Advanced. Note that a new master key will be generated every 480 minutes, or every 8 hours. Click Methods, and note that the security methods (encryption, integrity, and Diffie-Hellman group, are arranged from strongest preference to lowest. This arrangement allows negotiation to try several possible strengths. Should the other computer not be able to match any of these, the negotiation will fail. Click Cancel twice, and return to the Rules page.

On the Rules tab, click Add to add a Rule. An IP filter list consists of one or more filters that specify IP addresses and/or protocols for which the rule will fire. On the Welcome page, click Next. On the tunnel endpoint page, leave the default setting: the *This rule does not specify a tunnel* check box selected. IPSec can create a tunnel should you want to tunnel data from one network to another across a third. Because all machines are on the local LAN, this setup is not necessary. Click Next.

On the Network tab, select *All network connections*. We want the rule to fire no matter the location of the client machine. Though the clients should be on the local LAN, what if their IP address is spoofed across a remote connection? Yep, if the policy fires, unless the attacker can match the policy and authenticate to the computer, the connection will fail. Click Next.

Leave the Authentication method as Kerberos. If these computers were not member computers in a Win2K or Windows .NET domain, you would have to select shared key or certificate. Click next. Click Add to add a filter list on IP filter list tab. Enter a name for the filter list, and click Add to add a filter, then click Next. Enter a name the filter, and click Next. Leave the source as *My IP address*, and click Next.

Use the *Destination address* drop-down box to select *A specific IP address*, enter the IP address of the file server (see Figure 8.16), then click Next. You are telling the IPSec Policy agent that

any traffic from this computer addressed to that IP address should negotiate the connection and use IPSec to protect the communication.

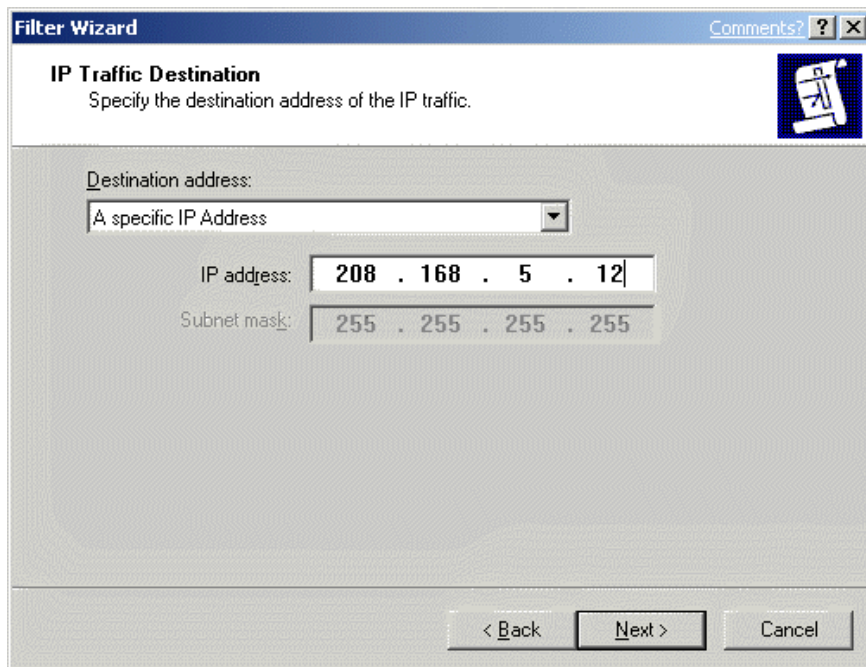


Figure 8.16: Specifying the destination address of the IP traffic.

On the IP Protocol page, leave the setting at Any, click Next, then click Finish. Doing so returns you to the IP Filter List page. Here, you could use the Add button to create additional filters if necessary. Click OK to return to the Filter List Page. On this page, select the filter list you just created (see Figure 8.17), then click Next.

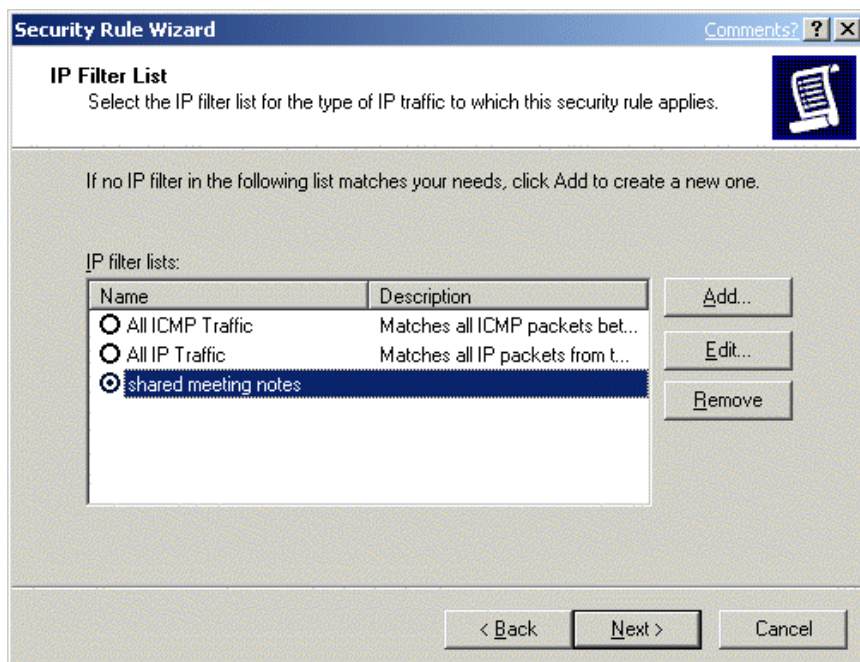


Figure 8.17: Selecting the filter list that you just created.

On the Filter Action page, click Add. Now that you have set a trigger, you must indicate what action will be taken. Enter a name for the filter action, and click Next. Because you want security to be negotiated, click Negotiate security, then click Next. Leave the setting *Do not communicate with computers that do not support IPSec* selected, then click Next. If your filter is triggered, you want the communication to proceed. Communication with other computers won't trigger the policy. Click Custom on the IP Traffic Security Page, then click Settings. Examine the settings for encryption and integrity. This is where settings for the Quick Mode of IPSec are established. Phase two is where the session key is used to encrypt the data for transfer (see Figure 8.18).

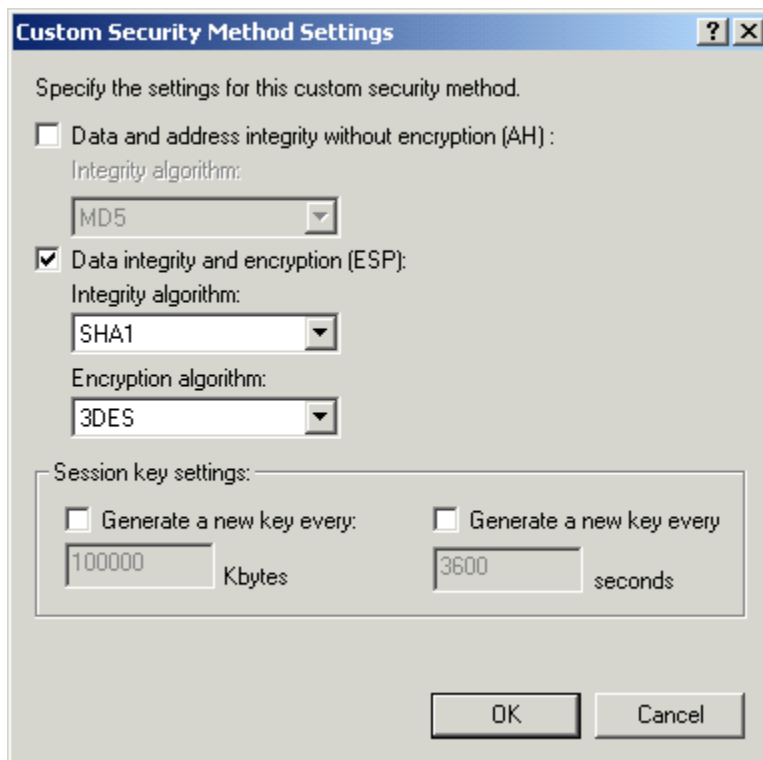


Figure 8.18: Specifying the security method settings.

Examine the Session key settings, which determine how frequently the session key will be changed. Remember to set these the same for each computer; otherwise, communication might commence, then halt after the first key change, because it will no longer match that on the other computer. We'll leave the settings at their defaults to make our job easier. You might determine that they need to be adjusted, just don't forget to make them match on each computer.

Click Next, then click Finish to return to the Filter Action page. Select the filter action you just created (see Figure 8.19), then click Next, and click Finish to return to the New Rule Page.

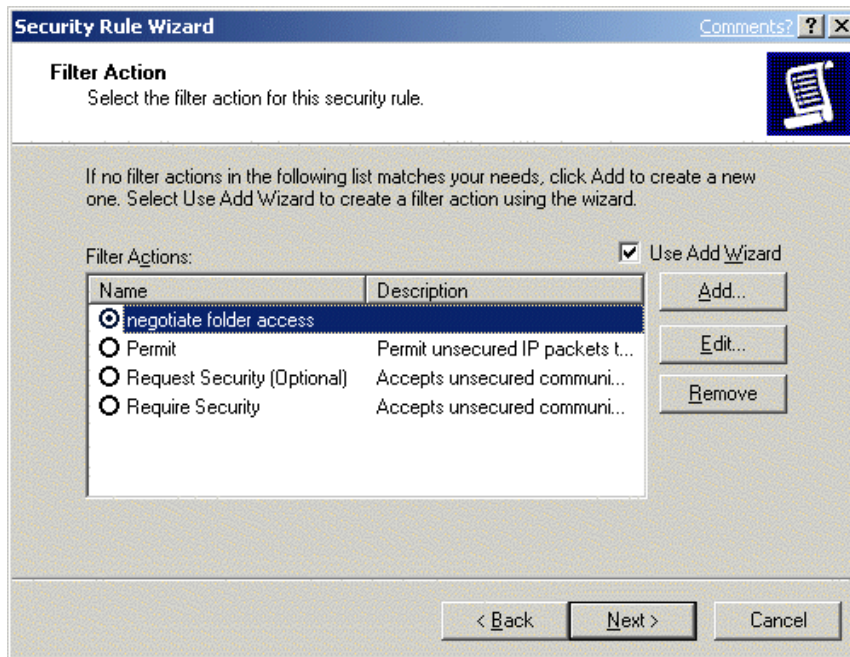


Figure 8.19: Selecting the filter action you just created.

Examine each page of the rule to make sure you have it correctly written, then click OK, then Close to complete the rule.

Before you can assign, or activate, the rule, you must complete its duplicate on the file server. The simplest way is to export a copy of the IPsec policies on this computer, then import it at the file server and make changes. To export the rule, right-click the IP Security Policies on Local Computer, and select All Tasks\Export Policies. Browse to a floppy disk, and save the file. On the file server, open the Local Security Policy, right-click the IP Security Policies, and choose All Tasks\Import policies. Locate the policy file on the floppy, and click Open

Before you can use the policy, you must adjust it for the new computer. Double-click the policy to open it, and click Edit to edit the rule. Double-click the filter list to open it, and click Edit to edit the filter properties. You will change the source and destination address values. The source address will still be My IP Address, remember that now represents the IP address of the file server. However, the destination address should be changed to the client IP address. Later, after the policy is tested, you might need to make a filter for each client machine IP address, or, if they represent a contiguous range of IP addresses, you can modify the filter list to reflect that. Click OK four times to save the changes and close the policy.

Testing the Policy

Now the policy is ready to be tested. You must first assign the policy on both machines. To do so, right-click the policy, and select Assign. The assigned policy will always be evident by a Yes in this column in the interface.

To test the policy, map a drive to the folder share on the file server, and create a text file there. If you are successful and can demonstrate that the communication was encrypted in transit, then you have correctly established IPSec communications. Either you can create a file on the share or not, so that part is easy to demonstrate. Evidence of IPSec activity is also easy to prove. A new, IP Security Monitor Microsoft Management Console (MMC) snap-in documents activity. Be sure to open it before beginning your test. Statistics about both phases of IPSec are documented.

Figure 8.20 shows the Security Association (SA) for Quick Mode, and Figure 8.21 shows clearly that confidentiality (encrypted) packets have been sent and received. An SA represents IPSec secured connection between two computers. Each secured connection will have an IKE SA (created in Main Mode to transfer data used to create the master key) and two in Quick Mode (for data traffic). The Quick Mode SA noted here represents the two made for the connection. Two SAs are created, one for sending and the other for receiving. Because Quick Mode cannot happen without successful completion of Main Mode, the policy is working as expected.

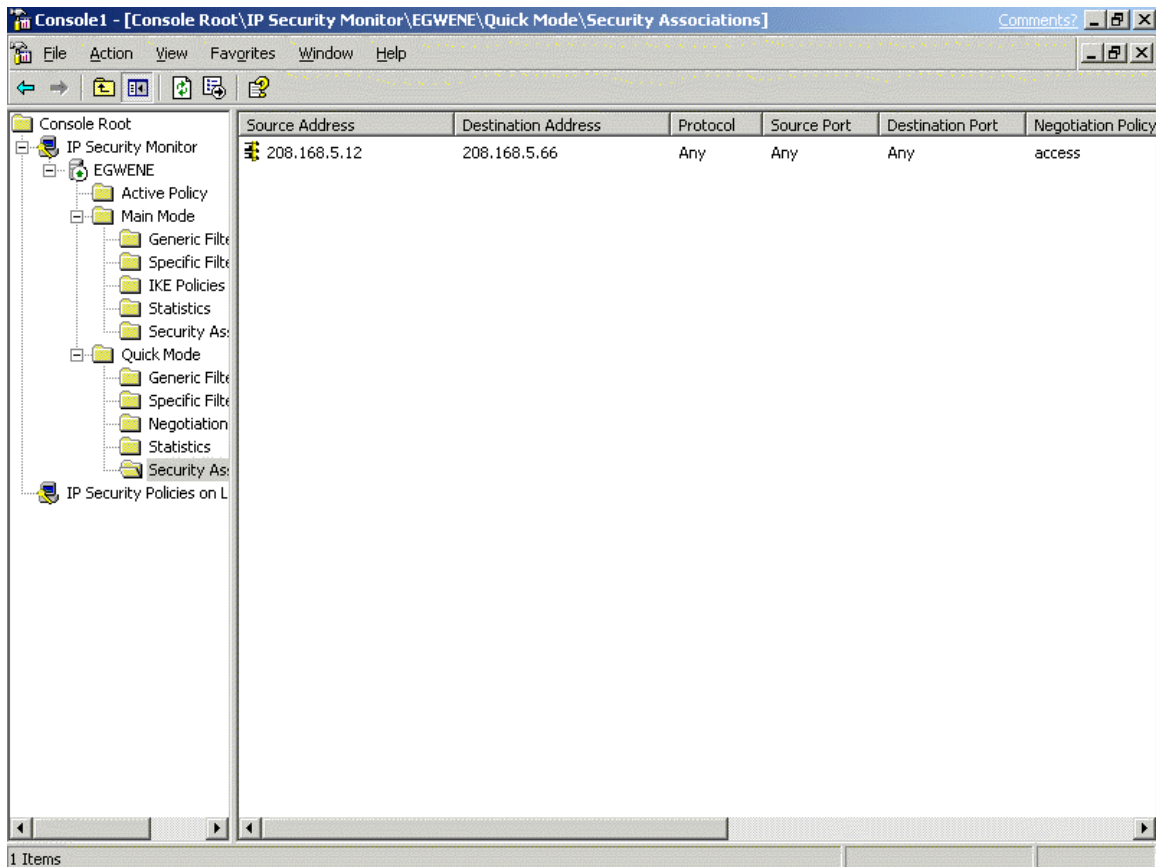


Figure 8.20: The SA for Quick Mode.

The screenshot shows the IP Security Monitor console with a tree view on the left and a statistics table on the right. The tree view is expanded to show the 'Quick Mode' folder under 'EGWENE'. The statistics table lists various parameters and their corresponding values.

Parameters	Statistics
Active Security Associations	1
Offloaded Security Associations	0
Pending Key Operations	0
Key Additions	3
Key Deletions	2
Re-Keys	1
Active Tunnels	0
Bad SPI Packets	0
Packets Not Decrypted	0
Packets Not Authenticated	0
Packets With Replay Detection	0
Confidential Bytes Sent	55277
Confidential Bytes Received	122475
Authenticated Bytes Sent	66016
Authenticated Bytes Received	135592
Transport Bytes Sent	65257
Transport Bytes Received	155624
Bytes Sent In Tunnels	0
Bytes Received In Tunnels	0
Offloaded Bytes Sent	0
Offloaded Bytes Received	0

Figure 8.21: Evidence that confidentiality (encrypted) packets have been sent and received.