Report Number: C4-50R-02

Guide to the Secure Configuration and Administration of Microsoft® SQL Server® 2000

Network Applications Team

Of the

Systems and Network Attack Center (SNAC)

Authors:
Sheila Christman
James Hayes, Maj USAF, CISSP

Dated: 26 August 2003
Version 1.5

National Security Agency
9800 Savage Rd. Suite 6704
Ft. Meade, MD 20755-6704

W2KGuides@nsa.gov

REVISIONS:

Version 1.1, Releasable version

Version 1.2, 13 December 2002, Made changes to the recommended permissions on registry keys for the SQL Server 2000 service accounts based on recommendations contained in the CIS-sponsored SQL Server 2000 Benchmark document, version 0.5.

Version 1.3, 15 January 2003, Added information concerning firewall configuration for SQL Server 2000 port assignments.

Version 1.4 28 January 2003, Clarified port blocking.

Version 1.5 26 August 2003, Updated Table 4 to remove “Create Subkey” and ”Set Value” permissions previously recommended for the account running SQL Server service.

Warnings

- **Do not attempt to implement any of the settings in this guide without first testing in a non-operational environment.**
- This document is only a guide containing recommended security settings. It is not meant to replace well-structured policy or sound judgment. Furthermore this guide does not address site-specific configuration issues. Care must be taken when implementing this guide to address local operational and policy concerns.
- The security changes described in this document only apply to Microsoft Windows 2000 systems and should not be applied to any other Windows versions or operating systems.
- This document may contain recommended settings for the system Registry. SQL Server 2000 can be severely impaired or disabled with incorrect changes or accidental deletions when using a Registry editor (Regedt32.exe or Regedit.exe) to change the system configuration.
- Currently, there is no **undo** command for deletions within the Registry. Registry editor prompts the user to confirm the deletions if **Confirm on Delete** is selected from the options menu. When a key is deleted, the message does not include the name of the key being deleting. Therefore, check selection carefully before proceeding.
- SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE EXPRESSLY DISCLAIMED. IN NO EVENT SHALL THE CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
- This document is current as of 1 October 2002. See Microsoft's web page for the latest changes or modifications to the Windows 2000 operating system and SQL Server 2000.

Acknowledgements

Some parts of this document were drawn from Microsoft copyright materials with their permission. Thank you to Maj. James Hayes, USAF, for his input into the Network Security section of this document.

Trademark Information

This Page Intentionally Left Blank

Table of Contents

Table of Figures

This Page Intentionally Left Blank

Table of Tables

This Page Intentionally Left Blank

Introduction

SQL Server 2000 is a robust relational database server that can be installed on a Windows NT or 2000 operating system. This document describes how to securely install, configure, and administer SQL Server 2000 on a Windows 2000 domain member server. The focus of this document is security-relevant information pertaining to the installation and day-to-day administration of SQL Server 2000.

This document is intended for the reader (DBA or administrator) who is already familiar with SQL Server 2000, but desires a further understanding of how to install, configure, and administer the database server in a more secure manner. The information presented here is written in a direct and concise manner in deference to this intended audience.

Some SQL Server 2000 security issues and corresponding configuration and administrative actions are very specific to the way the application is being implemented. For this reason, it is difficult in some areas to recommend specific, concrete actions. Instead, a summary is offered which describes the concerns and recommends solutions that the DBA must tailor to his/her own environment.

It is also important to realize that many organizations have developed policies regarding the structure and administration of database servers. Given the wide audience intended for this document, those specific policies could not be considered. It is up to the reader to apply these recommendations in light of their site's local security policies.

PLEASE NOTE THAT THIS DOCUMENT ASSUMES THAT THE READER IS A KNOWLEDGEABLE WINDOWS 2000 ADMINISTRATOR. A knowledgeable Windows 2000 administrator is defined as someone who can create and manage accounts and groups, understands how Windows 2000 performs access control, understands how to set account policies and user rights, is familiar with how to set up auditing and read audit logs, etc. This document does not provide step-by-step instructions on how to perform these basic Windows 2000 administrative functions. It is assumed that the reader is capable of implementing basic instructions regarding Windows 2000 administration without the need for detailed instructions. Information on how to secure a Windows 2000 environment can be found in NSA's Windows 2000 Security guides located on the NSA website at http://www.nsa.gov.

WARNING: This guide does not address security issues for the Microsoft Windows 2000 operating system that are not specifically related to the Microsoft SQL Server 2000 and its implementation.

This document is intended for Windows 2000 network administrators and SQL Server 2000 administrators, but may be read by anyone involved or interested in Windows 2000, database, or network security.

Getting the Most from this Guide

The following contains suggestions for successfully and securely configuring and administering SQL Server 2000 according to this guide:

WARNING: This list does not address site-specific issues and every setting in this book should be tested on a non-operational network.

- Read the guide in its entirety. Omitting or deleting steps can potentially lead to an unstable system and/or network that will require reconfiguration and reinstallation of software.
- Perform pre-configuration recommendations:
  - Perform a complete backup of your system before implementing any of the recommendations in this guide
- Follow the security settings that are appropriate for your environment.

Commonly Used Names

Throughout this guide the domain name "APSTESTING" will be used in the examples, screenshots, and listings.

WARNING: It is extremely important to replace "APSTESTING" with the appropriate network name for the networks being secured. This name is not a real network and has been used for demonstration purposes only.

About the Guide to the Secure Configuration and Administration of SQL Server 2000

This document consists of the following chapters:

Chapter 1 – Operating System Security

Chapter 2 – Installation

Chapter 3 – Network Security

Chapter 4 – Database Security

Chapter 5 – Database Maintenance

Chapter 6 – Additional Security Issues

Additional Information – SQL Server Resources

Chapter

1

Operating System Security for SQL 2000

Prior to installing SQL Server 2000, invoke the recommended settings for securing a Windows 2000 environment (found in the set of NSA Windows 2000 security guides available at http://www.nsa.gov) that are appropriate for your site, based on the local security policy (i.e., W2KServer.inf). Prior to applying any template .inf file, ensure that it is modified to address the specific security needs of your environment, i.e., modify the .inf template to include necessary rights for the account used to run SQL Server services. Table 1 lists the rights required for the Windows 2000 account(s) used to run SQL Server services. (Deny logon locally is recommended to prevent someone from logging on to the database server using the SQL Server service account.) File permissions, registry settings, password usage, user rights, and other issues associated with Windows 2000 security have a direct impact on SQL Server 2000 security.

Table 1 Rights Assignment for SQL Server Service Accounts

Act as part of the operating system	Lock pages in memory
Bypass traverse checking	Log on as a service
Increase quotas	Replace a process level token
Deny logon locally	

Administrators should always check for hotfixes/patches and install them as directed. These hotfixes/patches can be found at the Microsoft Download Center at http://www.microsoft.com/downloads. At the time of this writing, Microsoft had released two service packs, a security patch and a hotfix for SQL Server 2000. In multiple instance environments, it is required to apply the updates separately to each SQL Server 2000 instance on the server.

General Guidelines

When installing SQL Server 2000, the following guidelines are recommended:

- Place the SQL Server 2000 machine where it will be physically secure; i.e., behind a locked door where only authorized personnel can gain physical access to it.
- If the domain where SQL Server 2000 is to be installed requires trust links to other domains, ensure the access granted to domain resources (including the SQL Server instance(s)) through this trust is what was intended. It is important to document domain trust relationships and the resource access/permissions associated with the trust. If the server will be accessed from the Internet, consider configuring a DMZ with a web server and a database server. If possible, only store information that is meant for public dissemination on this database server. Also, do not configure trust links back to the internal domain from the

DMZ. Sensitive or critical information should be kept on a separate database server within the local domain. Several sources are available on the Internet describing DMZ architectures. The “Microsoft Windows 2000 Architecture Guide” is a good resource for this information and can be found on the www.nsa.gov web site.

- Install SQL Server 2000 on a server that is not required to support any other services. If the database is to be accessed by a Web Server, do not install these two services on the same machine. Neither application software nor development tools should be installed on the production database server.
- Install SQL Server 2000 development servers in a separate environment from the production servers.
- For intranet servers, install SQL Server 2000 on a domain member server, where possible. Do NOT install on a domain controller. If SQL Server 2000 is installed on a domain controller and the server is attacked, the entire domain and sensitive domain information may be at risk. Also, the added overhead of being a domain controller will slow down the server’s ability to provide services efficiently.
- Partition the SQL Server 2000 computer so that when the database server is installed, SQL Server 2000 program files and data files are located on a different partition or disk from the OS.
- If access to the SQL Server 2000 computer is required from the Internet, install it initially with all incoming traffic blocked at the router or firewall. Once the configuration of the SQL Server 2000 machine is complete, allow incoming traffic. This is recommended to prevent connections to active SQL Server 2000 services prior to the implementation of security configurations.

Following are several recommended steps to take to secure the operating system in preparation for securing the SQL Server 2000 database server:

- Install Windows 2000 on its own drive or partition and apply the latest W2K service packs.
- Apply NSA’s W2K Server .inf file (available for download from www.nsa.gov website) after modifying it appropriately to reflect the site’s security policy.
- The following listed services are not required for most installations of SQL Server 2000 database domain servers. This list is not inclusive of all possible services and will be dependant upon the administrator’s implementation and server environment. The following unnecessary services should be disabled on a SQL Server 2000 database server:

Table 2 Services to be Disabled

Service	Important Notes
Alerter	
Clipbook Server	
Computer Browser	
DHCP Client	
Distributed File System	
Fax Service	
Internet Connection Sharing	
IPSEC policy agent	Disable unless IPSec policies will be used
License Logging Service	
Logical Disk Manager Administrator Service	
Messenger	
NetMeeting Remote Desktop Sharing	
Network DDE	
Network DDE DSDM	
Print Spooler	
Remote Access Auto Connection Manager	
Remote Access Connection Manager	
Remote Registry Service	Disable unless running HFNETCHK or other network management software requiring remote registry access
Removable Storage	If required to support backups, set to manual start
RunAS Service	
Smart Card	
Smart Card Helper	
Task Scheduler	Disable unless batch jobs will be run from within SQL Server 2000 or if scheduling tasks on the server is required
Telephony	
Telnet	
Windows Installer	

- Remove all unnecessary protocol stacks. (Do not remove TCP/IP.)
- Rename the local computer's Administrator account.
- Create a Windows local user account or domain user account to run SQL Server 2000 services. Create a name that does not easily identify it with SQL Server. This account should only be used to run the SQL Server 2000 services and no other services. A local user account is recommended, unless access to network resources, for such operations as replication, is required. In that environment, use a low-privileged domain user account. It is not required, nor is it recommended, to place the account used to run the SQL Server 2000 services in any Windows 2000 administrators group. In our testing, with Service Pack 2 for W2K and Service Pack 2 for SQL Server 2000, a local user account without W2K administrator privileges was able to auto-start and execute SQL Server 2000 while ensuring that the server ran under minimal privileges to mitigate the impact of any compromises.
- On the database server, deny logon locally rights to the account(s) used to run SQL Server services, as shown in Figure 1. (SQLUser is used in the example to clarify which account is being denied. Do not give the service account a name that is easily identifiable as an account used with a SQL server). This will prevent someone from trying to use the account to login to the database server. If a domain account is used, configure the account so that login can only occur on the database server. Still, deny the logon locally right to the domain account on the database server. Configuring the account to only logon to the database server (as shown in Figure 2) will prevent someone from trying to use the account to gain access to other machines in the domain.

Figure 1 Deny Logon Locally

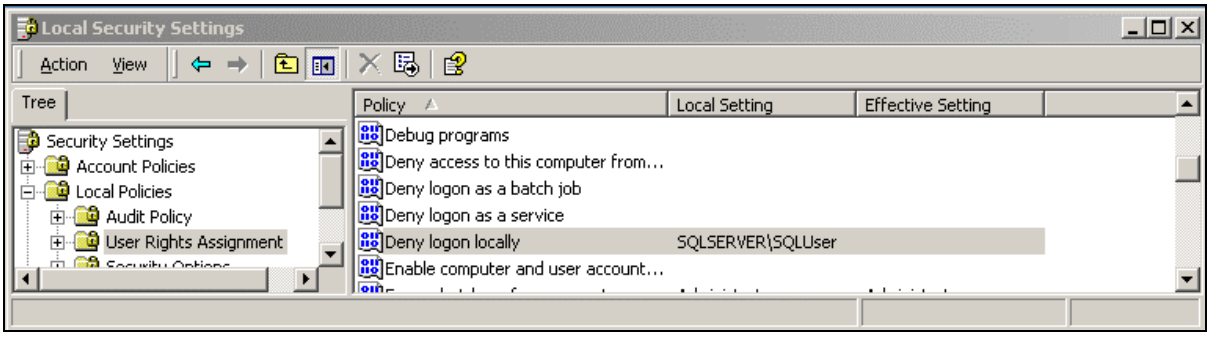
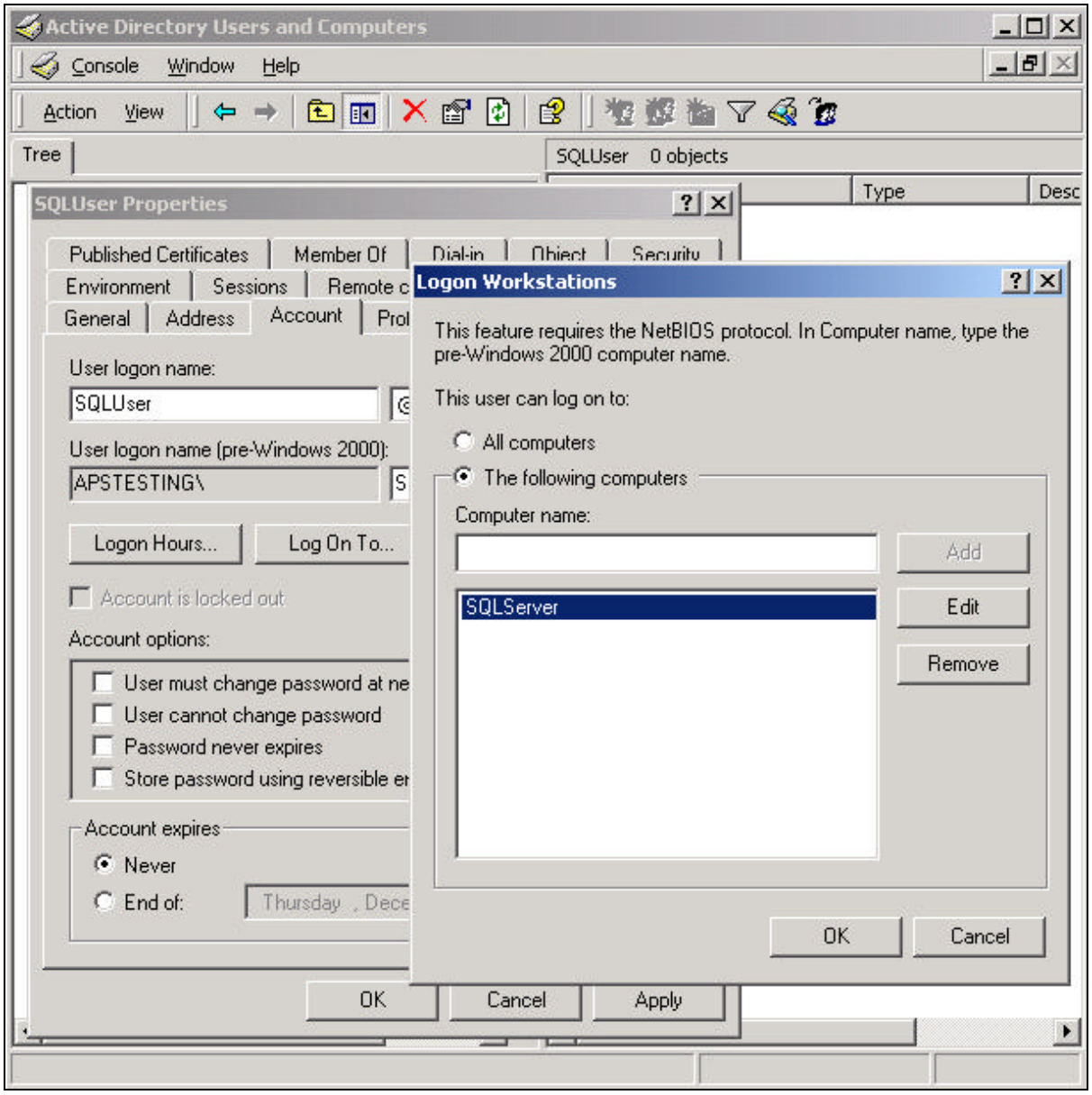


Figure 2 Define SQL Server Service Account Computer Access



- Create user groups at the domain level and consolidate these global groups from the domain(s) into local groups on the SQL Server 2000 machine. This is done so that management of groups accessing SQL Server can be performed at the database server. This is especially important if sites are granting access to similar groups from multiple domains. These similar groups can be combined into one group at the local database server prior to being mapped to the SQL Server database roles.
    - Place DBAs, developers, security users, database users, etc., into separate groups. These groups should then be added to local groups on the actual SQL Server 2000 database server.
    - Determine if DBAs need to be placed in an Administrative group. W2K administrator privileges are only required if the DBA is expected to perform W2K administrative tasks. If this is the case, remember to assign the least amount of privileges required to perform the assigned tasks:
        - Consider whether or not the DBA should be an ordinary user with appropriate privileges; or
        - Should only have Windows local administrative privileges; or
        - Should have domain-wide Windows administrative privileges.
    - Create a global group for the SQL Server service account(s). After adding the account(s) to the group, change the Primary group for the account to this restricted group. The default Primary group for users is set to Domain Users. Remove the account(s) from the Domain Users group. This is done to prevent the SQL Server service account(s) from gaining privileges and permissions granted to domain users.
- Remove the “Everyone” group from the SQL Server installation drive or partition. Give the SQL Server 2000 service account, System and Administrators Full Control over the installation drive or partition. Although **not recommended**, if SQL Server 2000 is installed on the same drive or partition as the OS, **do not** give the SQL Server 2000 service account Full Control over the OS drive or partition. In this case, the account need only have Full Control over the Microsoft SQL Server installation directory. It is also recommended that the installation directory be changed to something other than the default.
- Set screen saver with password protection of 15 minutes.

Chapter

2

Installation

Pre-Installation Considerations

Prior to installing SQL Server 2000, a well thought out security policy should be developed. Several resources are available on the Internet relating to good security policy development and, therefore, will not be reiterated in this guide. However, below is a list of some things to consider during the development of a database server security plan:

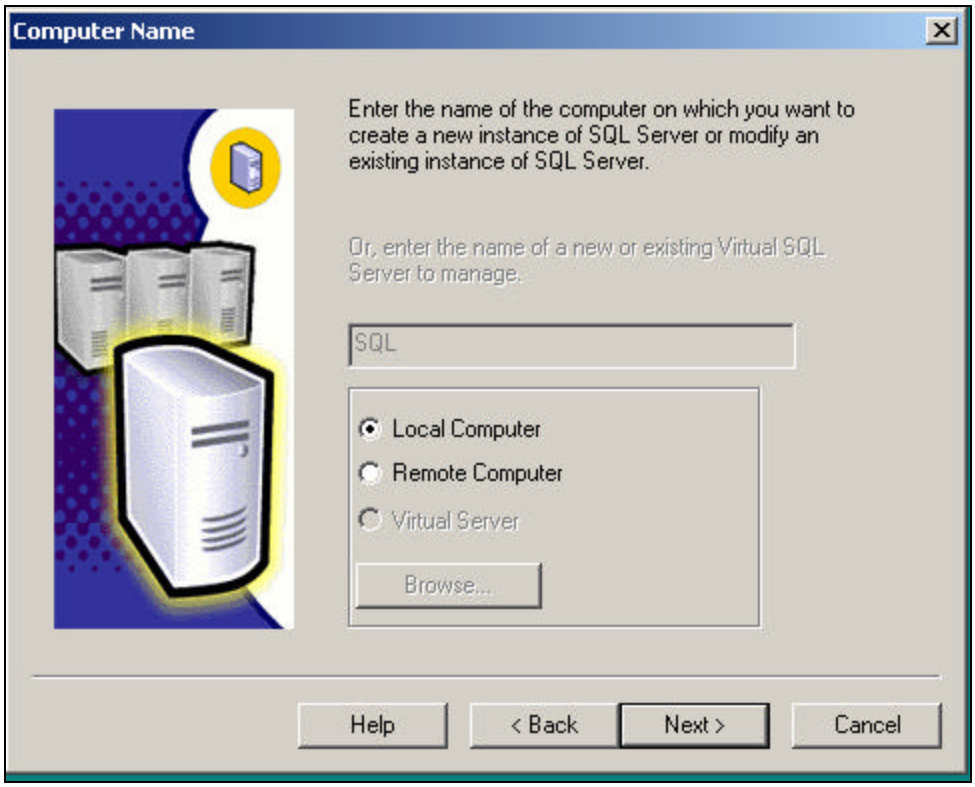
- SQL Server 2000 Architecture – How many databases are required? Can the development and functional testing servers be isolated from the production server?
- The types of accounts requiring access to the database server (e.g., Administrator, Security Administrator, DBA, Developers, Users, etc.)
- Who will be the approving authority for account management (creating, deleting, determining required permissions, etc.)?
- Methods of connection for accessing data in the database(s)
- The security requirements related to the databases/instances associated with the database server
- Authentication mechanism required for access to each database associated with the database server
- Formal Account Request forms
- Standards for usernames and passwords
- Naming conventions for database objects (e.g., tables, views, etc.)
- Standards for creating and removing user accounts
- Audit criteria (to include how often audits are to be performed)
- Frequency and type of backups to be performed on the database

Installation Process

SQL Server 2000 editions include the Enterprise Edition, Standard Edition, Personal Edition, Developer Edition and the Evaluation Edition. This guide covers the installation of the Standard Edition, which supports up to four processors and 2GB of RAM. The Enterprise Edition differs from the Standard Edition in that it can support up to 32 processors and 64GB of RAM. The security recommendations outlined in this guide apply to both editions. However, features of the Enterprise Edition that are not included in the Standard Edition, such as fail-over clustering, are beyond the scope of this guide.

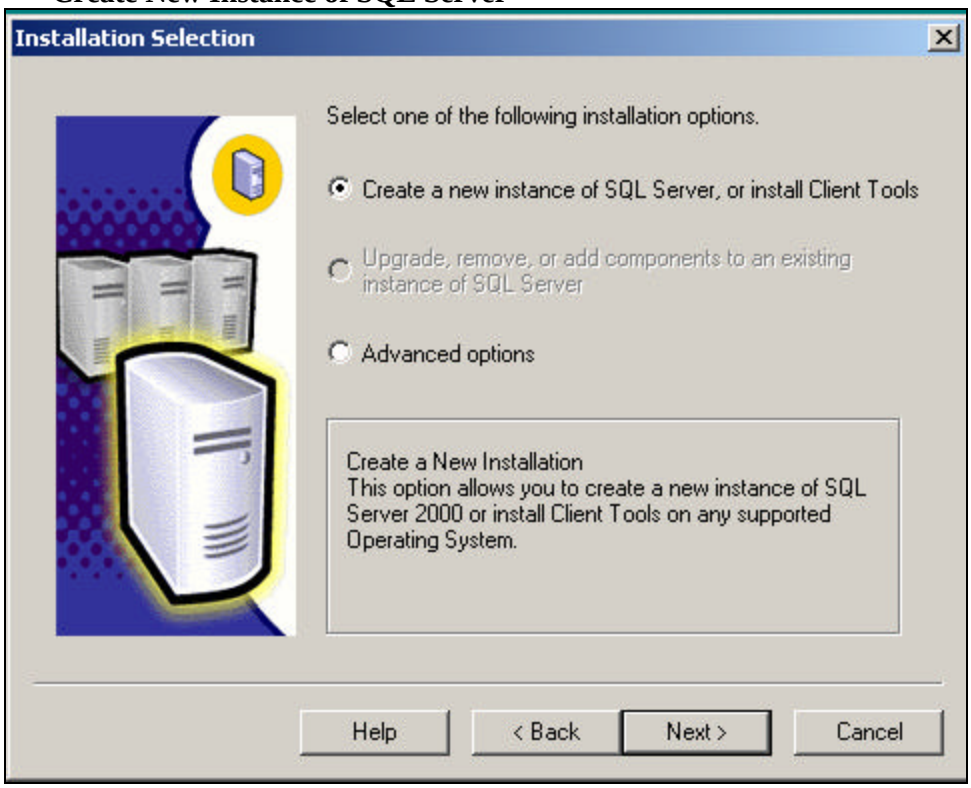
To start the SQL Server 2000 installation, login as the user with administrative privileges. Insert the installation CD. An installation wizard takes you through all of the necessary steps to install an instance of SQL Server 2000. The figures below show the dialog boxes displayed by the install wizard. Chapter 5 of the Microsoft SQL Server 2000 Introduction manual, which accompanies the software, describes the installation options in detail.

Figure 3 Select Local or Remote Installation



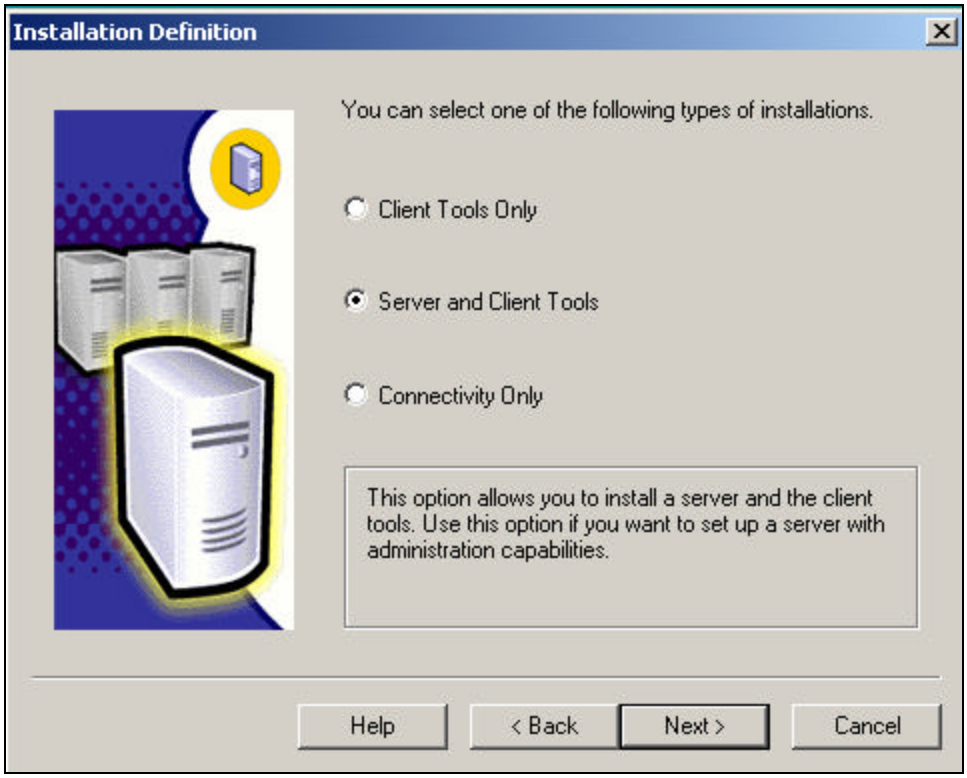
Note: Advanced Options (registry build, unattended installation, and upgrading to a cluster) are not available for a remote installation

Figure 4 Create New Instance of SQL Server



To install SQL Server 2000 in the domain for the first time, choose the option shown in Figure 4. If an instance of SQL Server 2000 is detected on the computer, an option to upgrade, remove or add components will be made available. The Advanced options selection allows for the creation of an initialization file for unattended installations, a registry rebuild in the event of a corrupted installation, and changes to existing clusters.

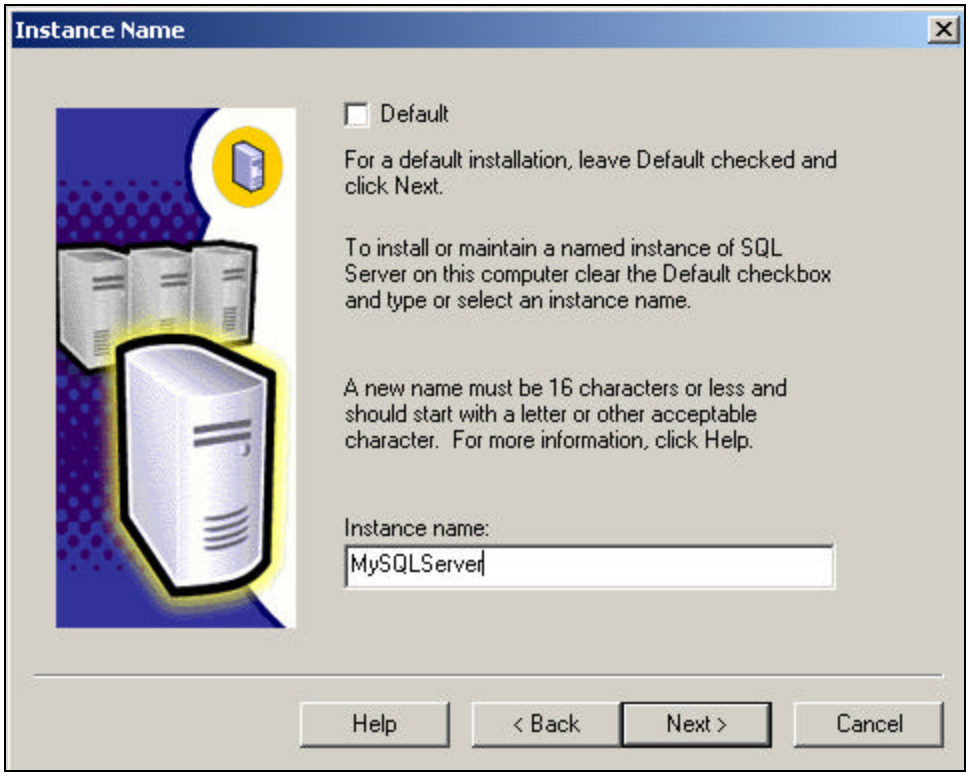
Figure 5 Select Server and Client Tools



The Client Tools Only option installs connectivity components and client relational database management tools. This option is used during the installation of SQL Server Clients.

The Connectivity Only option installs only the components required to make a connection to a SQL Server instance. As with the Server and Client Tools option, the Connectivity Only option includes the Microsoft Data Access Components (MDAC) 2.6 needed to connect to a SQL Server 2000 instance. Ensure the security patch is installed for the MDAC 2.6 components to eliminate a vulnerability that could lead to a compromise of SQL Server. More information on this vulnerability can be found in Microsoft Security Bulletin MS02-040. Although MDAC comes with all versions of Windows, this vulnerability only affects SQL Servers. Another way to eliminate this vulnerability is to configure the HKLM/SOFTWARE/Microsoft/MSSQLServer/Providers/SQLOLEDB DisallowAdhocAccess subkey Registry setting to 1 (default is 0).

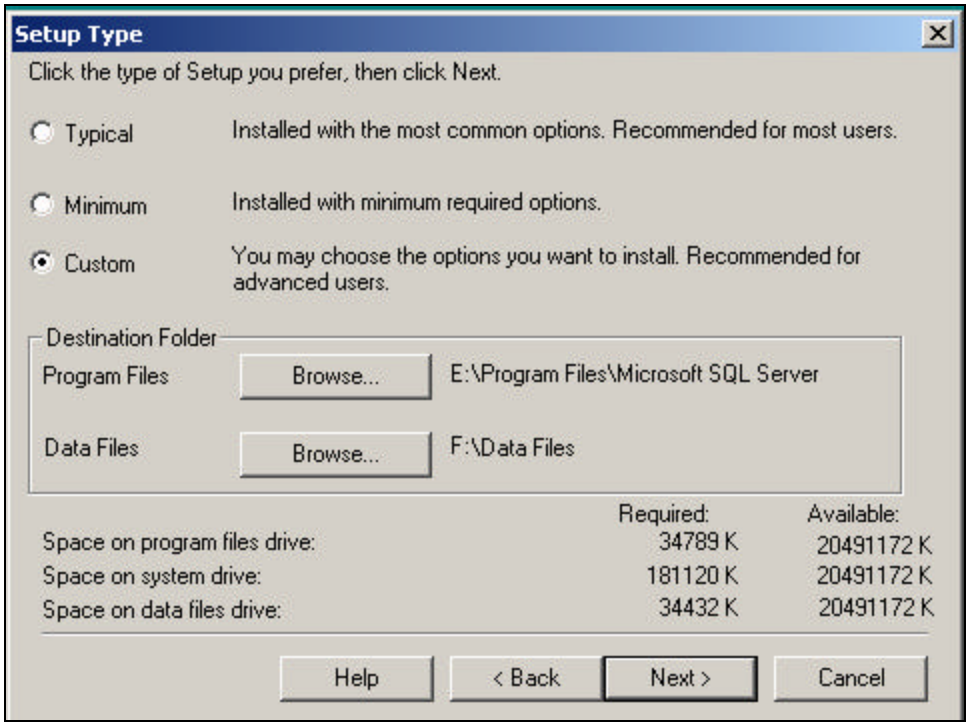
Figure 6 Default or Named Instance



Note: There can only be one default instance of SQL Server per computer. If an instance has been detected, the Default checkbox will be disabled. For initial installations, the Default checkbox will be selected. A default instance is not required. A default instance is required, however, to support applications that use client software from earlier versions of SQL Server. A named instance can be created at initial install instead of selecting the default checkbox. Instance names can be no more than 16 characters long. See page 94 of the Microsoft SQL Server 2000 Introduction manual for more restrictions regarding instance names.

Multiple instances of SQL Server 2000 can be installed on one computer. Each instance operates independent of any other instance. Each instance is installed with a unique directory structure with its own set of services and access permissions. This can be useful in an environment that maintains multiple databases with unique access requirements on one server. Using multiple instances, each database can be completely isolated from the others. This feature can also be used to create a virtual server environment by installing an instance with the same name on multiple computers. As each instance of SQL Server 2000 provides services independent of other instances that may be on the server, security updates, patches, and hotfixes need to be applied to each instance individually.

Figure 7 Installation Type

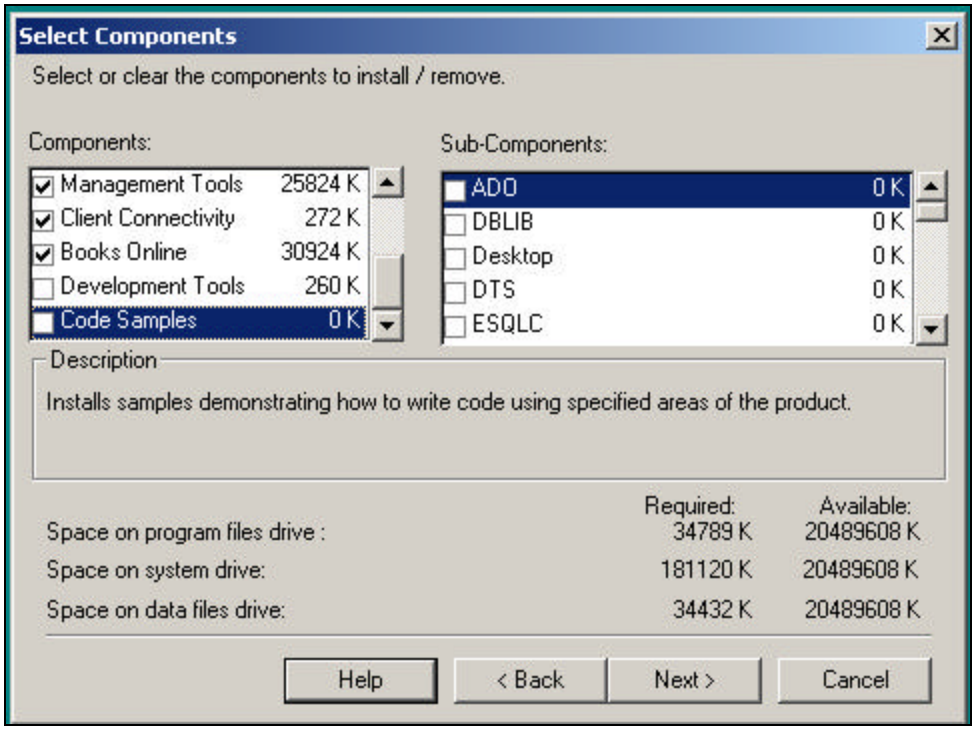


Select the type of installation to perform. Choose the option that best fits your needs based on the information in Table 3. Select a location for the program files and data files. Be sure to select a location that is not on the same partition as the operating system. It is also recommended that the data files be separate from the installation directory and OS partition.

Table 3 Component Selection

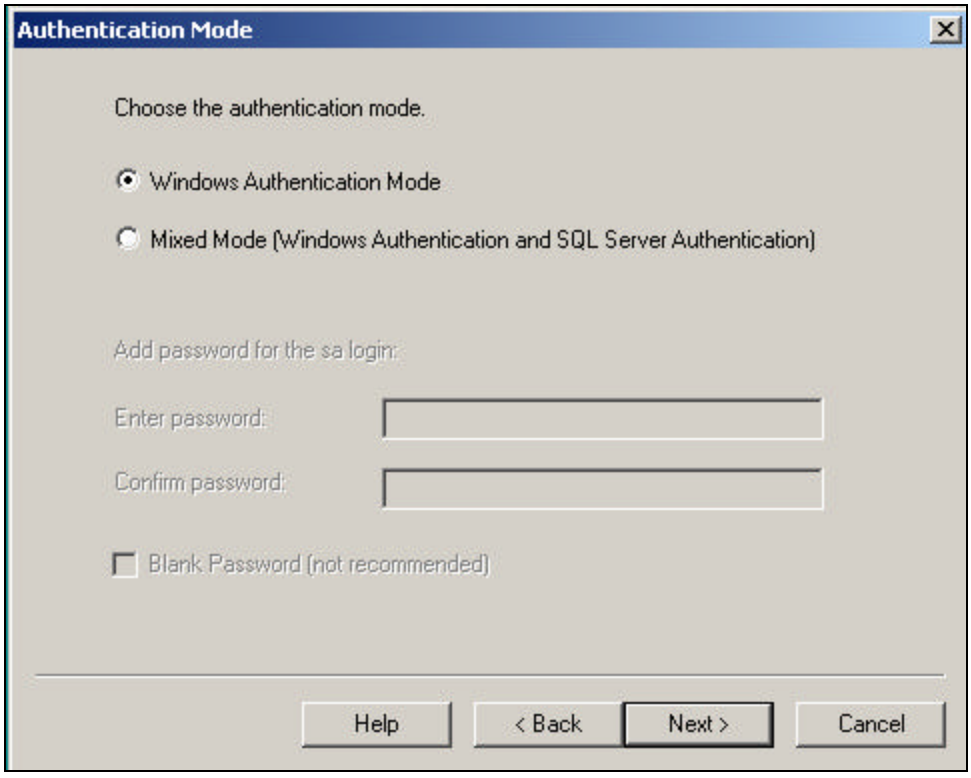
Type of Installation	Components Installed
Typical	Database Server, Upgrade Tools (for default instance only), Replication Support, Full-Text Search, All Client Management Tools, Client Connectivity, Books Online, Debugger, Collation Settings
Minimum	Database Server, Replication, Client Connectivity, Collation Settings
Custom	Allows the installer to choose components to be installed.

Figure 8 Samples



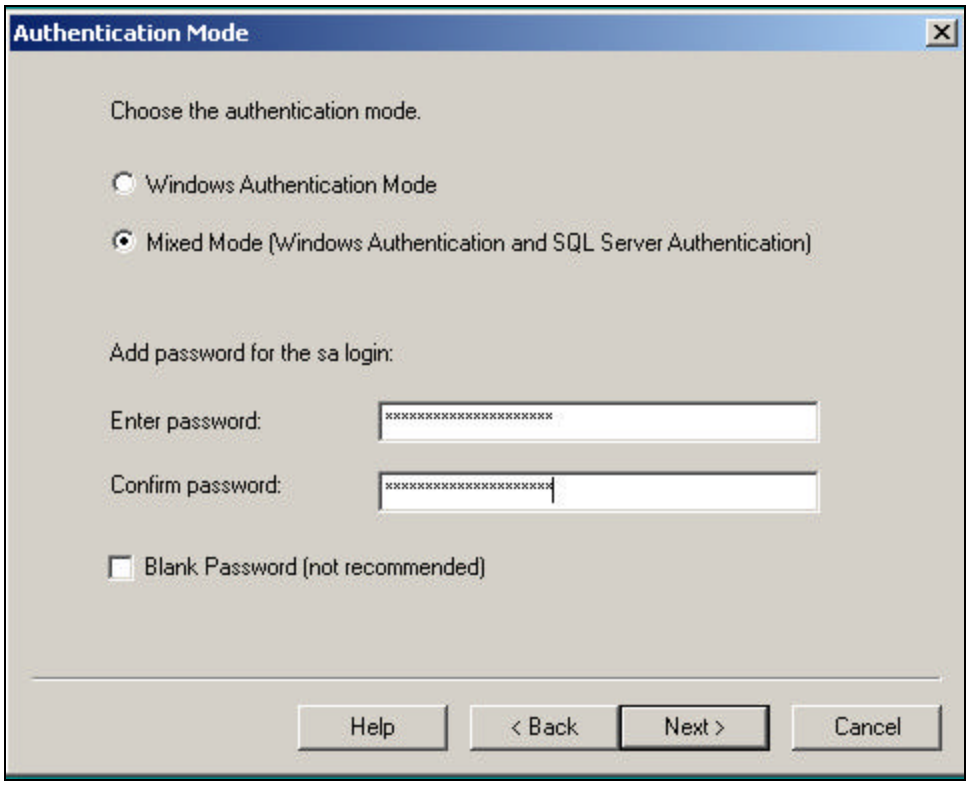
When choosing the components to install from the Select Components dialog box, ensure Code Samples is not selected. Samples can provide an avenue for system compromise and should not be installed in production environments.

Figure 9 Windows Authentication Mode



It is recommended that Windows 2000 Windows Authentication Mode be selected during the installation of SQL Server 2000. This tight coupling for login authentication has the advantage of eliminating excess overhead resulting from additional security and access layer controls while creating a trusted connection. It is understood that this is not always an option; therefore, security concerns pertaining to a Mixed Mode environment will be briefly discussed.

Figure 10 Mixed Mode Authentication



When Mixed Mode authentication is selected, a password is required to be entered in the dialog box for the SQL Server 2000 sa account. Make sure a strong password is entered during initial installation, and if it is ever necessary to switch authentication from Windows to Mixed Mode (not recommended). Also, the Mixed Mode authentication option permits the use of Blank Passwords by selecting the checkbox. **Never select this option**.

With Mixed Mode authentication, there is no lockout policy for failed logon attempts using SQL Server accounts. This opens up the door for brute force attacks against accounts, particularly the “sa” account.

Figures 11 and 12 show examples of the dialog box displayed by the install wizard for defining the account(s) used to run the two SQL Server services. It is not recommended that the Local System account be used to run SQL Server 2000 (unless SSL is implemented, which is explained in Chapter 3). Aside from limiting operations to the local server, a database compromise could lead to the execution of code in the security context of Local System, possibly compromising the entire host. This could potentially lead to a compromise of the entire domain. The account(s) used to run the SQL Server services should have the least amount of privileges required to meet the needs of the architecture, as defined in the environment's security plan. Configuring the service account(s) with Local System or administrator access grants more permissions and directory/file access than required to run the database server services. To define a local user account, enter the local machine name as the Domain (as shown in Figure 11). If access to network resources is required, use a domain user account (a local user account will not be able to access network resources to perform network operations, such as replication).

> **Note:** If SQL Server 2000 is installed with a local or domain user account, and an unmodified W2KServer.inf is applied after SQL Server 2000 installation, all necessary rights for running SQL Server 2000 will be taken away (see Table 1 for service account(s) access rights). If this happens, remove the server from the network; configure SQL Server 2000 to run as LocalSystem, and then reconfigure again to run using the designated user account. This will reset the user rights and permissions needed to run SQL Server 2000 to the designated user. A reboot is not necessary during the reassignment process. Once completed, reconnect the server to the network.

Figure 11 Services Accounts Selection I

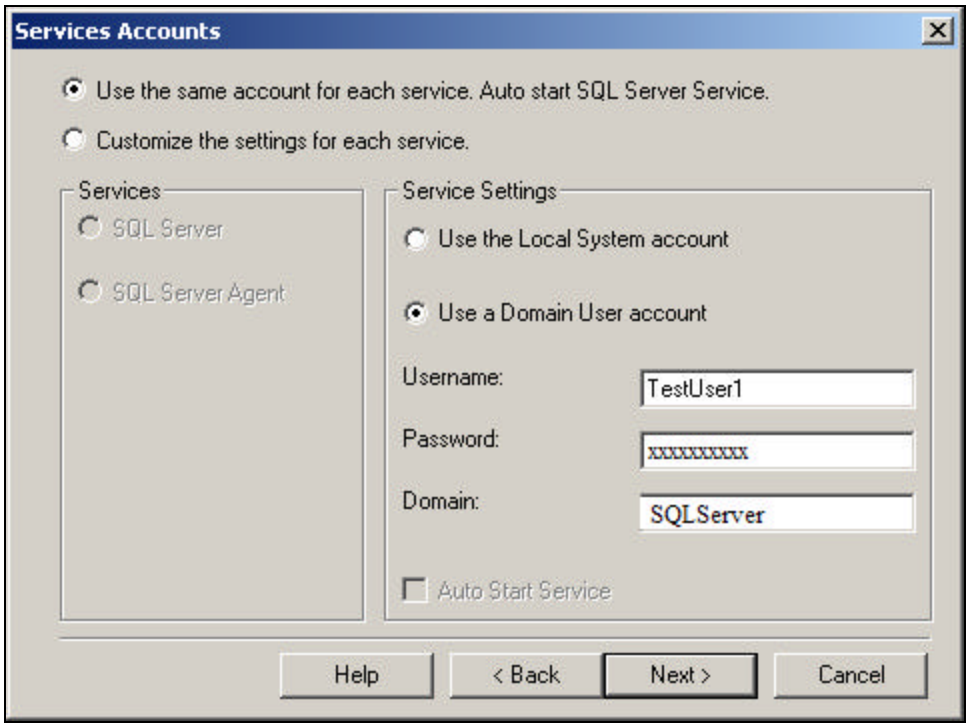
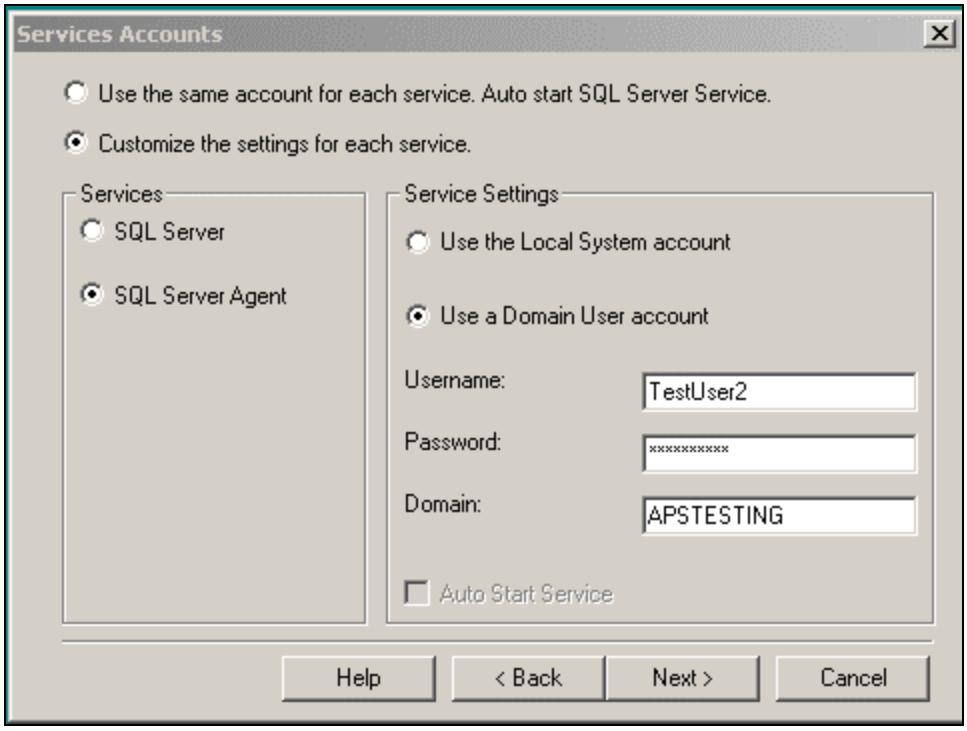
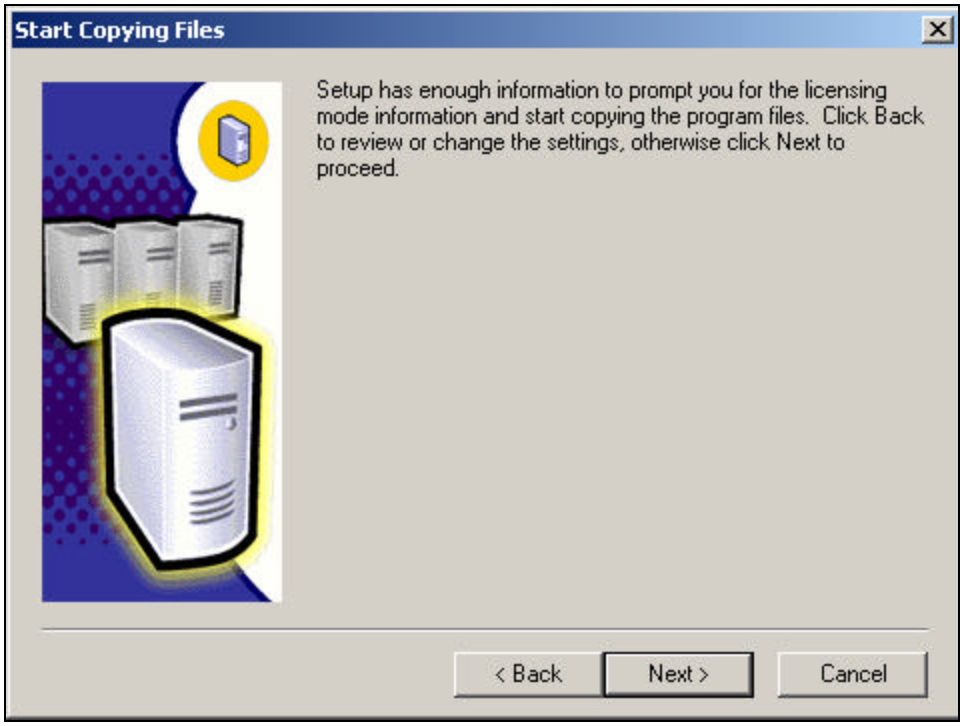


Figure 12 Services Accounts Selection II



The function of the SQL Server Agent is to schedule tasks. **It is not required that the SQL Server service and the SQL Server Agent service use the same account, but it is recommended**. If separate accounts are needed, select the "Customize the settings for each service" option and configure each account separately, as shown in Figure 12 for SQL Server Agent.

Figure 13 Start Copying Files



- Remove any sample databases or scripts that may have been installed during the installation process (i.e., Northwind, Pubs). Samples can provide an avenue for compromise. These databases are used for training purposes and should not be installed on production servers.
- If the service account is changed after installation, ensure the old account is either deleted or only has required permissions on registry keys, directories and files. In addition, remove any unnecessary privileges that the account inherited when configured for SQL Server 2000 operations.
- After SQL Server 2000 is installed, remove read permission from the Everyone group on the HKEY_LOCAL_MACHINE\Software\Microsoft\MSSQLServer registry key. Grant appropriate permissions over the SQL Server registry keys listed in Table 4 to the account that will run SQL Server 2000.
- Remove the Users group from the [Drive]:\Program Files\Microsoft SQL Server directory and grant full control to the SQL Server 2000 service account.
- Only allow Administrators to have full control over .exe files, such as explorer.exe, regedit.ext, cmd.exe, etc. The account used to run SQL Server 2000 services should not be granted execute permissions to these files. See *The 60 Minute Network Security Guide* available on the NSA website for a list of other tools requiring execute restrictions.

An important note about operating system security

It is very important to keep track of permissions on SQL Server 2000 install directories and registry keys. Change the default settings on these directories and keys to reflect the following (ensure the subdirectories and files are configured to inherit permissions from the parent):

Table 4 Permissions on SQL Server 2000 Directories and Registry Keys

Directory/RegKey	User/Group	Permissions
Program Files Directory, e.g., D:\Program Files\Microsoft SQL Server	Administrators	Full Control
	System	Full Control
	Local account used to run SQL Server, i.e., SQLUser (a domain user account must be used if access to remote resources is required by the database server, or if clustering is implemented)	Full Control
Data Files Directory, e.g., D:\SQLData	Administrators	Full Control
	System	Full Control
	Local (or domain) account used to run SQL Server	Full Control
C:\WINNT\system32	Administrators	Full Control
	System	Full Control
	Authenticated Users	Read&Execute

Directory/RegKey	User/Group	Permissions
HKEY_LOCAL_MACHINE\Software\Microsoft\MSSQLServer	System Account used to run SQL Server	Full Control Query Value, Set Value, Create Subkey, Enumerate Subkeys, Notify, Read Control
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Perflib	System Account used to run SQL Server	Full Control Query Value, Set Value, Create Subkey, Enumerate Subkeys, Notify, Read Control
HKEY_LOCAL_MACHINE\Software\Microsoft\Microsoft SQL Server\$InstanceName (for a named instance)	System Account used to run SQL Server	Full Control Query Value, Enumerate Subkeys, Notify, Read Control
HKEY_LOCAL_MACHINE\System\CurrentControlset\Services\SQLSERVERAGENT	System Account used to run SQL Server	Full Control Query Value, Enumerate Subkeys, Notify, Read Control
HKEY_LOCAL_MACHINE\System\CurrentControlset\Services\MSSQLServer	System Account used to run SQL Server	Full Control Query Value, Enumerate Subkeys, Notify, Read Control
HKEY_LOCAL_MACHINE\System\CurrentControlset\Services\MSSQL$InstanceName (for a named instance)	System Account used to run SQL Server	Full Control Query Value, Enumerate Subkeys, Notify, Read Control

Think carefully before granting others access to these directories and registry keys. The more access given, the more likely there could be a compromise.

File permissions, registry settings, password usage, user rights, and other issues associated with Windows 2000 security have a direct impact on SQL Server 2000 security. The recommended source of information for how to securely configure the Windows 2000 server and workstation is NSA's Windows 2000 security guide set. This guide set is comprised of a series of documents covering various aspects of Windows 2000 security, and can be found on NSA's website.

Known Event Issues

The following event failures may occur after applying NSA's W2KServer.inf file, disabling/deleting services as recommended, and installing SQL Server 2000:

Table 5 Event Log Issues

Log	Type	Event ID	Explanation
Security	Failure	560	Object access failure (example: RasPbFile)
Security	Failure	562	File and Object auditing is turned on
Application	Error	19011	Running SQL Server 2000 with an account that does not have permissions to register a Service Principle Name
Application	Error	2001	Telephony and Remote Access services are turned off

These errors occur when the SQL Server 2000 services run with the recommended restricted user account. If these failures cause problems with your implementation, consider using the local administrator account. If access to domain resources is required or clustering is implemented, consider using a low-privileged domain user account.

This Page Intentionally Left Blank

Chapter

3

Network Security for SQL 2000

Network security must, at a minimum, address user authentication and network connections from unauthorized workstations. In addition, some network environments may require protection of SQL Server 2000 network traffic through means of data integrity and confidentiality methods. Windows 2000 supports Internet Protocol Security (IPSec) that will allow administrators to provide data integrity and confidentiality. In addition, SQL Server 2000 can provide the same protection through Secure Socket Layer 3.0 (hereafter referred to as SSL). If your organization already has a network IPSec policy in place, the organization may wish to consider adding a security policy for confidentiality of SQL Server 2000 traffic by way of IPSec's Encapsulating Security Payload (ESP) format for IPSec packets. These topics are addressed in the following sections.

SQL Server 2000 Authentication Options

Two methods are available for user authentication – Windows authentication and a mixed environment where either Windows authentication or SQL Server 2000 authentication can be used (Mixed Mode). Windows Authentication mode is the default and preferred method of user authentication. Mixed Mode authentication is not as secure and its use is strongly discouraged unless a situation requires the use of SQL Server 2000 authentication, i.e., to support legacy programs or non-Windows 2000 clients.

SQL Server 2000 and Database Traffic Protection

In addition to authenticating users, other aspects of network security must be addressed so that database traffic and host availability are protected. Protection mechanisms, such as router access control lists (ACLs), firewalls and network encryption (provided by SQL Server 2000 and Windows 2000) are possibilities that can help achieve this goal.

Applying Security with Firewalls and Routers

Defense in Depth is a strategy for achieving information assurance. One aspect of this strategy is layered defenses. Although authentication may stop access to the database and possible attacks against the database, it may not stop attacks against the host. Although securing the operating system is one mechanism to help prevent attacks against the host, other mechanisms can be used to further protect the host from attack.

The default SQL Server 2000 ports (1433 or 2433 (Hide Server)) should not be used. This adds one more hurdle that an attacker has to overcome, i.e., scanning ports to determine which ports are open and then having to determine the service running on a particular port. This in and of itself is not enough; mechanisms must be put in place to help prevent connections from unauthorized clients. Firewalls and router ACLs can be

used to help prevent connection and port scanning from unauthorized clients. In terms of SQL Server 2000, refrain from using the default ports and ensure the configured port as well as port 1434 are blocked on firewalls and routers. Reference NSA's *Router Security Configuration Guide* for other ports that should be blocked.

Applying IPSec in Windows 2000 Intranet

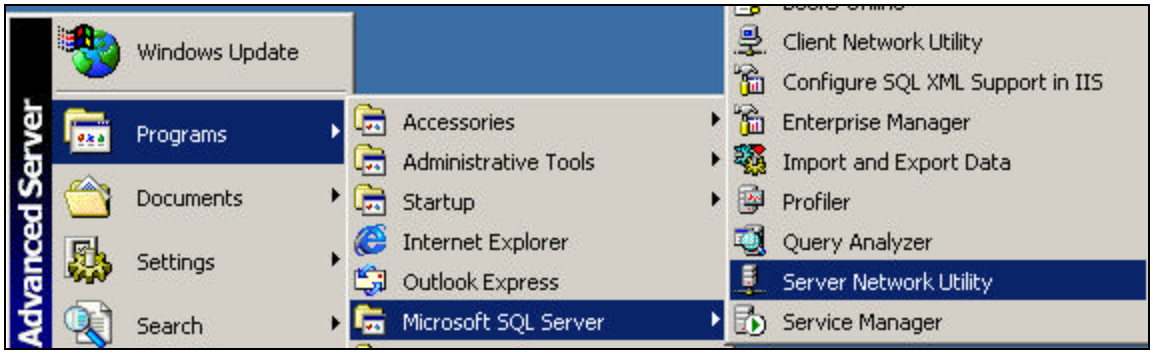
If the clients that will be served are all Windows 2000 clients, then it is strongly encouraged that IPSec be implemented. If the database traffic is sensitive, then Windows 2000's IPSec Encapsulating Security Payload (ESP) capability should be implemented. Even though IPSec can be applied to specific ports and IPs, such as a SQL Server port, implementing IPSec should be done in the context of the total security requirements of an intranet. Guidance can be found in NSA's *Microsoft Windows 2000 IPSec Guide*. Even though confidentiality may not be an issue, and considering the layered approach to defense, IPSec should still be considered for its ability to provide an organization with a trusted computing environment.

IPSec can enable an organization to have a trusted computing environment through its Authentication Header service. This service is especially useful to organizations that do not have a requirement for encryption of network data. The main benefit is that only domain computers (trusted) will be able to communicate with one another. For more background on setting up a trusted computing environment, refer to Microsoft's *Ask Us About…Security* – December 15, 2001 article.

Applying Security with SQL Network Utility

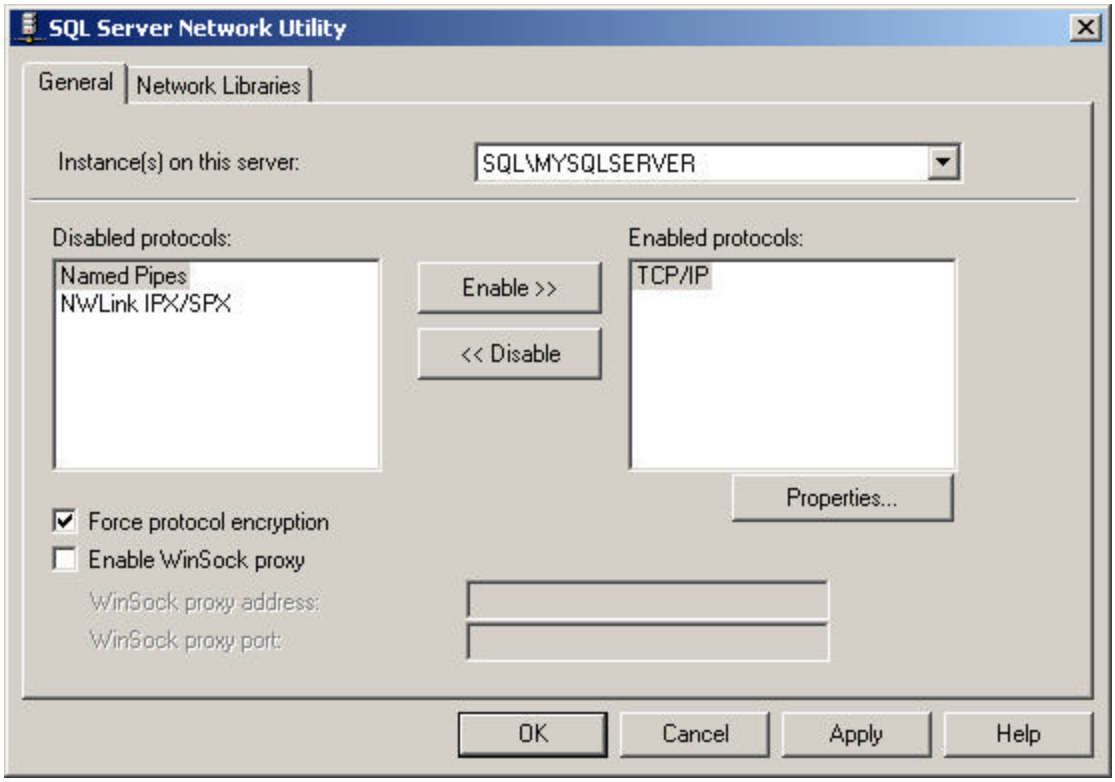
Three options in the SQL Server Network Utility can help in further securing a SQL Server 2000 database server. Access the utility by selecting **Server Network Utility** from the **Microsoft SQL Server** pull down of the **Start** menu on the toolbar.

Figure 14 Access Server Network Utility



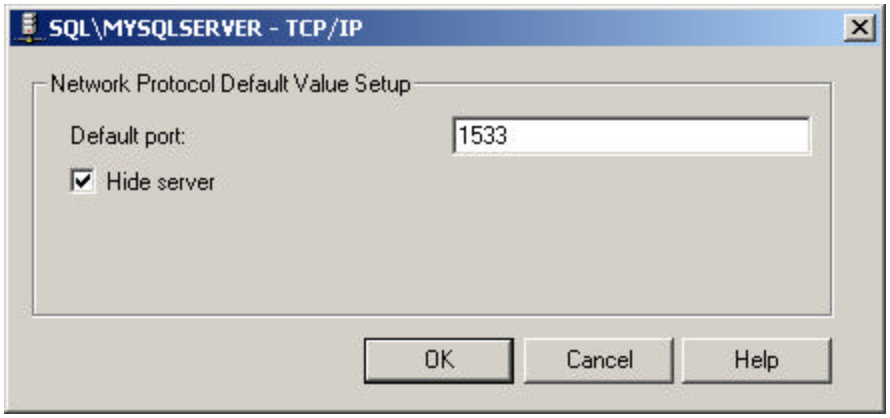
- *Enabled Protocols* – only those protocols required should be enabled, i.e., if Named Pipes are not required, then use only TCP/IP.
- *Force Protocol Encryption* – this option activates SSL; however, the local computer must have its own certificate and private key before this option can be implemented. If an enterprise Certificate Authority (CA) is available, then a certificate can be obtained by using the Certificates snap-in for the computer account. In addition, the service account running SQL Server 2000 must either be the LocalSystem or an administrator account. Since LocalSystem has less permission on directories and files than administrators, it is recommended that the LocalSystem account be used for SSL implementation.

Figure 15 Protocol Encryption



- *TCP/IP Properties* – An administrator has the ability to change the default port and hide the server from clients through the properties tab.
  - The default port is 1433. If possible, administrators should change this to some other unused port.
  - Hiding the server prevents clients from using tools, such as Query Analyzer, to view SQL Server 2000 servers that are online in the domain. Clients are still able to connect to a SQL Server instance by providing the server name. **Note:** When hiding the server, SQL Server software will change the default port to 2433. Even if the port is changed, as in Figure 16, once the instance is restarted, the port number will change back to 2433. Also, only one instance can be hidden per database machine. More information can be found in Microsoft's Knowledge Base Article 308091.

Figure 16 Port Declaration/Hide Server



IPSec vs. SSL 3.0

Internet Protocol Security in Windows 2000 offers authentication, integrity, confidentiality and anti-replay services. The question then becomes “What is the difference between using these services in SSL and IPSec?” The primary difference is that in SQL Server’s configuration, all services will be provided if SSL is turned on. IPSec, as implemented in Windows 2000, can allow these services to be configured on selected computers and ports. Other important notes:

- Authentication, in this sense, is referring to computer authentication, not user authentication. Authentication used in IPSec is used to authenticate connections between pairs of computers. In the case of SQL Server’s configuration of SSL, authentication is not mutual. Only the server provides authentication.
- Integrity in IPSec is determined by the configuration of the services selected for IPSec. IPSec offers integrity services through both the IP AH and ESP header. In either case, packet data integrity can be achieved for all traffic between computers or selected computers and ports. In the case of SQL Server’s configuration of SSL, integrity is provided for database data traffic only. Using SSL means that all services will be used, even if the intention is to only provide integrity services.
- Confidentiality used in IPSec is achieved by using the ESP header option. It can be configured using encryption algorithms, such as DES and 3DES. It is strongly recommended that 3DES be the chosen encryption algorithm. Most implementations of IPSec use ESP services since it provides both integrity and confidentiality. The cryptographic algorithms used in SQL Server’s implementation of SSL are not configurable.

Database Traffic Encryption and Mixed Mode Authentication

If Mixed Mode authentication is used, users who authenticate themselves using SQL Server authentication should use SSL or IPSec to protect user IDs and passwords. It is suggested that Windows 2000 clients that must use SQL Server authentication use IPSec ESP. This does not mean that the server needs to use the Force protocol encryption option; rather, client workstations, where practical, can be forced to use encryption using the Client Network Utility (access this utility as shown in Figure 17) option to Force protocol encryption. This method still requires that the server have a certificate. Administrators, who choose this method of securing SQL Server authentications, should be aware of the possibility that users may attempt to authenticate themselves from clients that have not been properly configured. Forcing encryption at the server end would cause unnecessary overhead in network environments where database traffic does not need to be protected and where other users authenticate themselves using Windows authentication, but it would ensure that all SQL Server authentications are protected.

Figure 17 Access SQL Server Client Network Utility

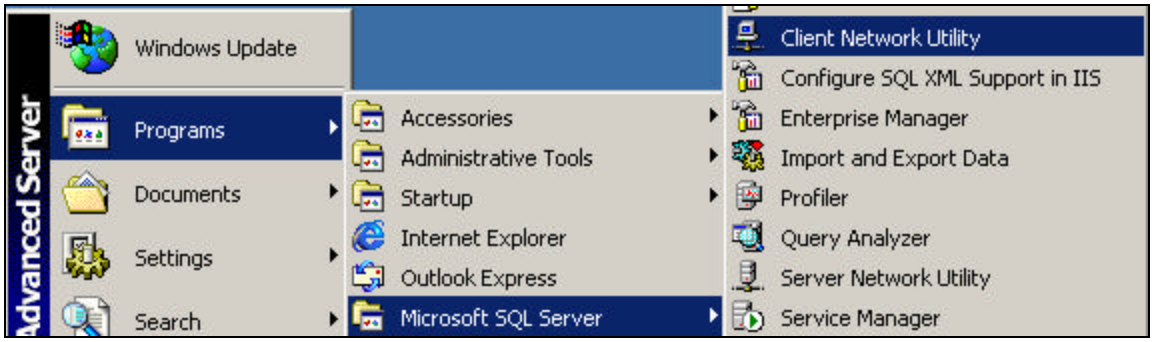
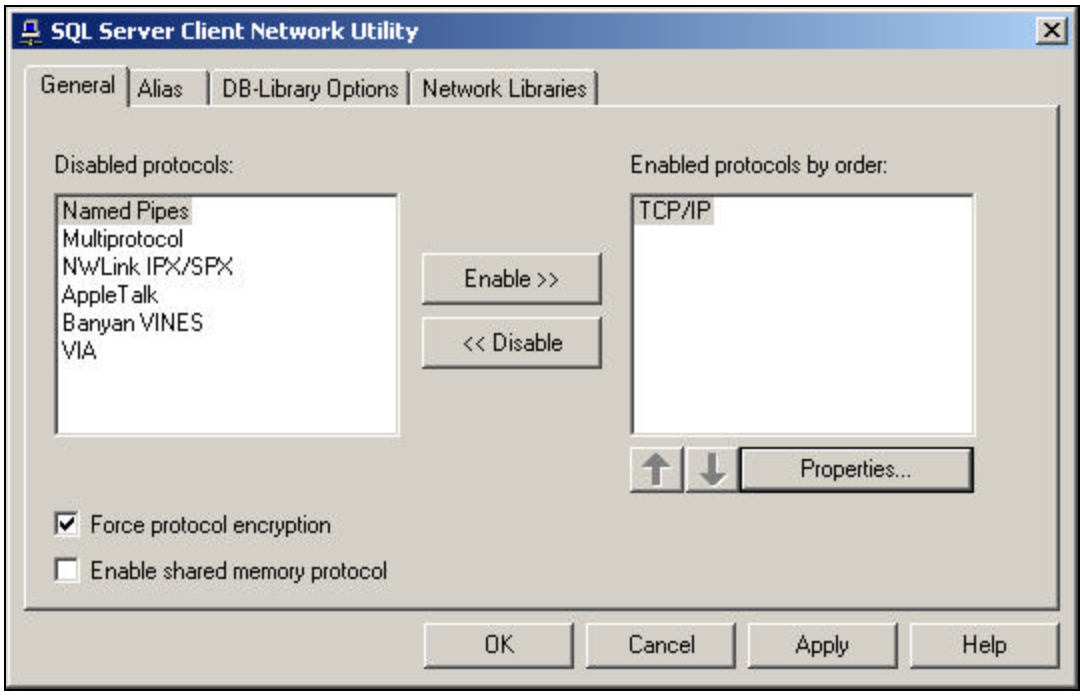


Figure 18 SQL Server Client Network Utility



A similar technique of protection can be used with IPSec. Windows 2000 IPSec will allow ESP services to be used for specific ports and IPs. This will allow administrators, if needed, to protect SQL Server authentication from specific clients.

This Page Intentionally Left Blank

Chapter

4

Database Security

Access Permissions

In order to access information in any database stored within SQL Server 2000, a user must have a valid SQL Server login AND the login must map to a user who has been granted access permissions on the target database. Logins provide the means for a user to connect to a SQL Server 2000 database server, but not access data in a database on the server. During installation, two options are available for SQL Server logins, Windows Authentication and Mixed Mode Authentication. As stated previously, it is recommended that the default Windows Authentication mode be used, if possible, to take advantage of the security features inherent in the Windows 2000 authentication process. This also has the added bonus of providing the user with a single logon that takes advantage of the domain-level account security policies. If Mixed Mode is required, do not create SQL Server logins with the same name as e-mail accounts or operating system logon accounts.

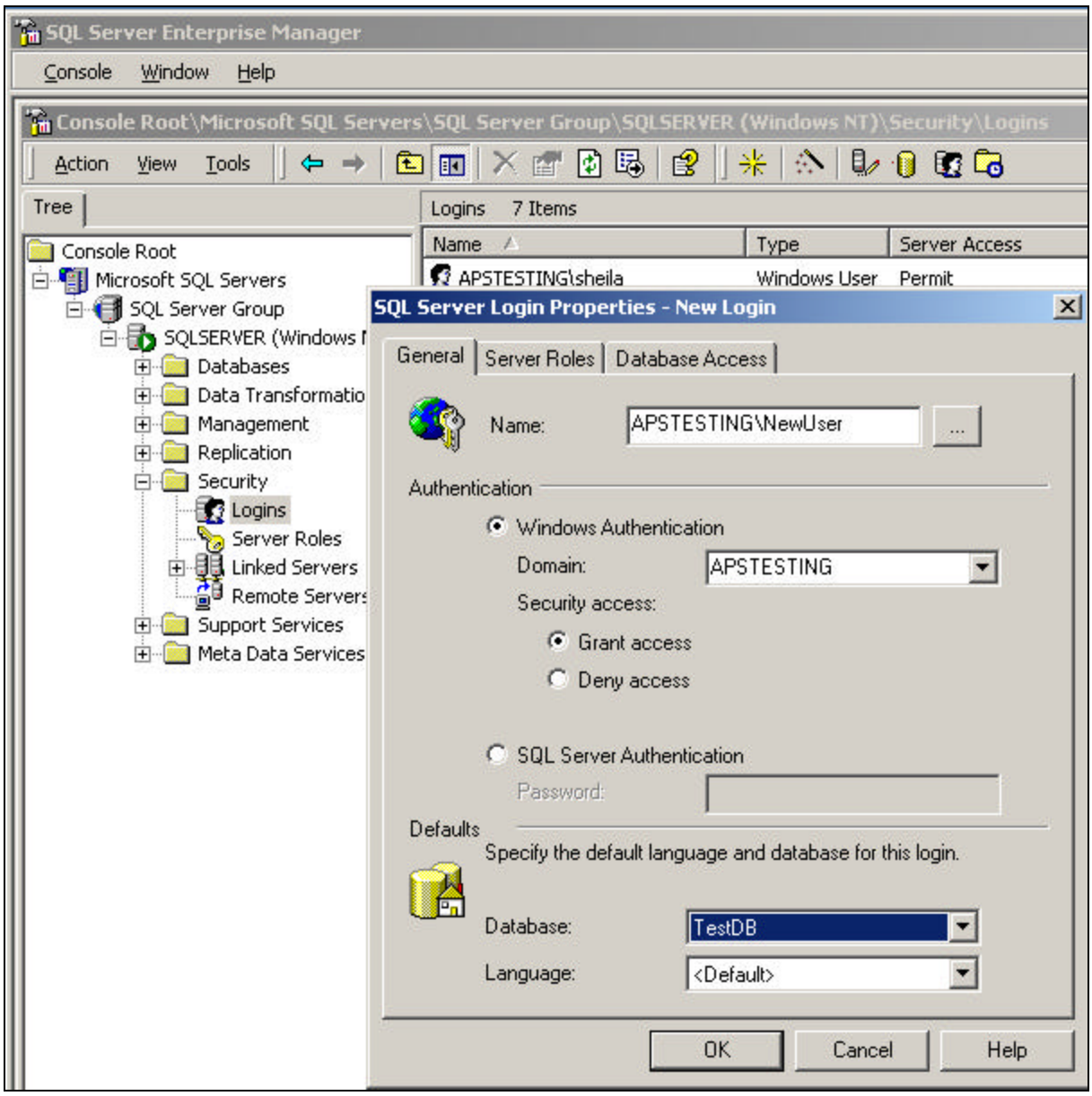
Several types of users require access to SQL Server 2000, i.e., database administrators, database programmers, and users requesting information, each requiring varying levels of access to perform their assigned tasks. It is recommended that each of these user types be placed in global groups within the domain, and then these global groups placed in local groups on the database server. These local groups can then be added to SQL Server 2000 database logins, which map to database user accounts with appropriate levels of access to the data.

NOTE: The special user “guest” can be created in any database. If the guest user exists within a database, any login that does not map to a user in the database will be granted access to objects within the database using the permissions of the guest user. By default, this user is granted the same permissions as the **public** database role. It is recommended that the guest user be deleted from all databases, except the master and tempdb databases. Users do not require unlimited access to master and tempdb databases; therefore, the guest account is configured with a limited set of permissions and privileges within these special databases by default. Since users require limited access to these databases, the guest account cannot be removed. For all other databases, ensure all logins requiring access map to a valid user within a given database. This ensures accountability for all actions performed within the database environment.

In order for users to access data stored within SQL Server databases, access to the server must be granted using a valid login. Perform the following steps to create a login and grant access to a SQL Server instance using Enterprise Manager:

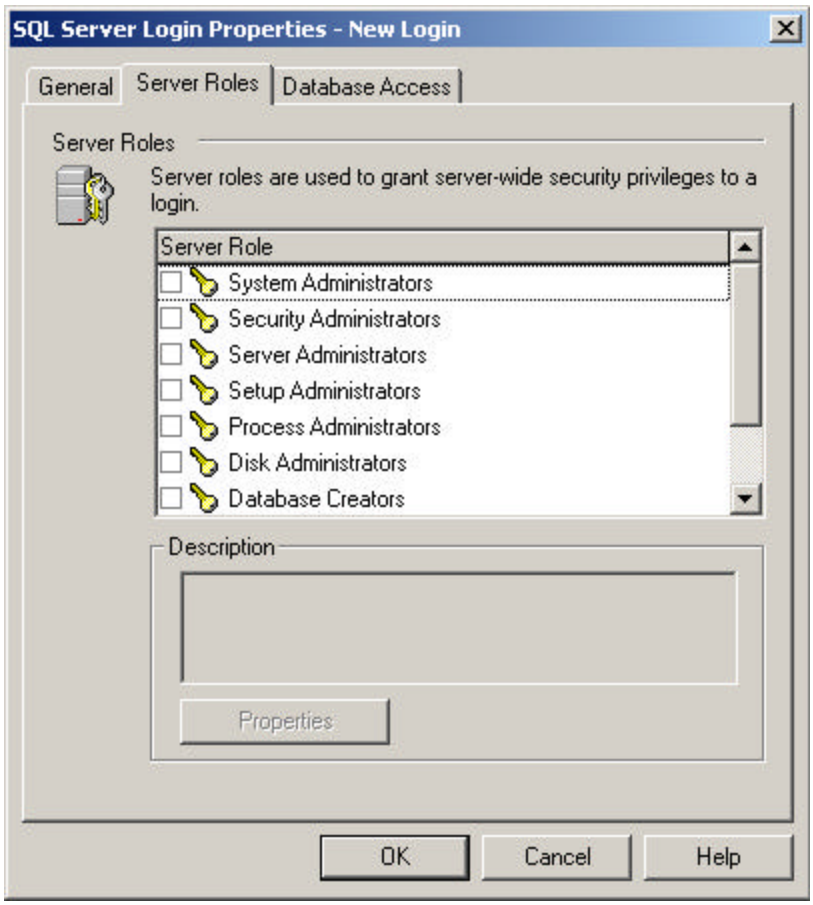
1. Create a new login by right clicking Logins under the Security folder beneath the desired SQL Server Instance. Select **New Login** from the pull-down menu. The **SQL Server Login Properties – New Login** window is displayed.
2. On the General tab, enter a name for this login or click the button to the right of the **Name** box to select a Windows user from a domain or the local server. By default, the Windows Authentication option is selected. If the new login was selected using the button next to the **Name** box, the Domain will be filled in automatically. If not, select the Domain or local server from the pull down options. It is recommended that Windows Authentication be used for all SQL logins; however, some situations may require SQL Server Authentication logins (i.e., in support of older applications that require SQL Server login accounts). If SQL Server Authentication is required for the login, enter a complex password and implement encryption between the server and the client machine.
3. The **Grant access** security access option is selected by default. The **Deny access** option can be used later to temporarily deny access to the login, if the need arises.
4. Select from the pull-down options the database that is to be active when this login is authenticated. The default active database is the master database. It is recommended that this be changed to the database most likely to be accessed by the login.

Figure 19 Create New Login



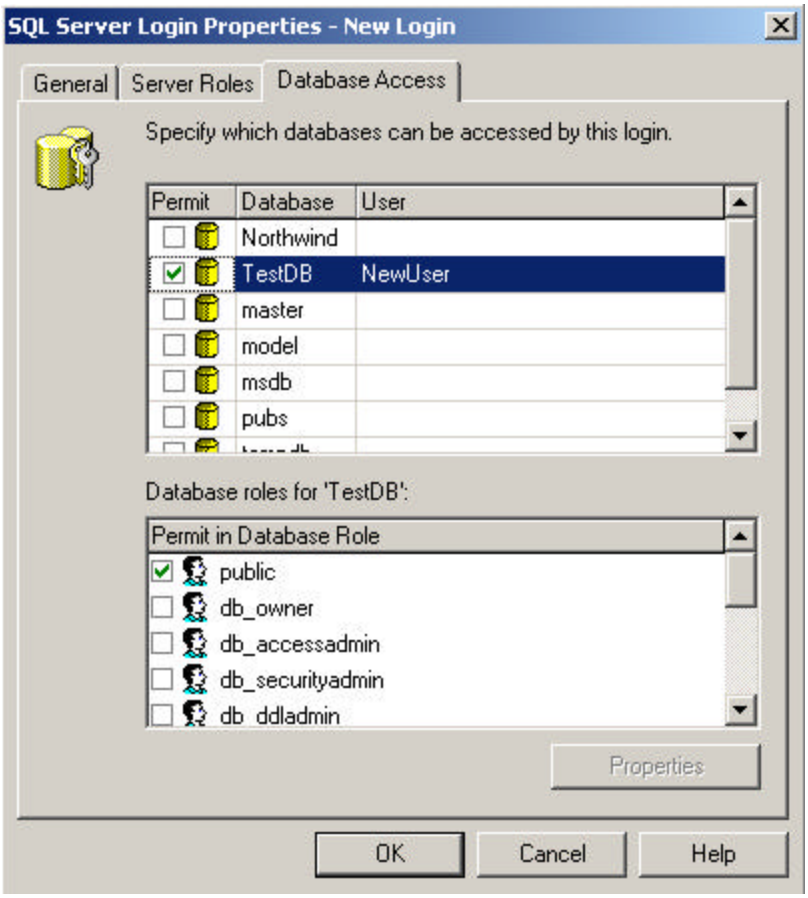
5. On the **Server Roles Tab**, select a server role for this login, if required. Careful consideration should be given prior to granting logins special server privileges through these Server Roles. Most database users do not require a Server Role assignment (Roles will be discussed later in this section).

Figure 20 **New Login Server Roles Tab**



6. On the **Database Access Tab**, select the databases this login is required to access. For each selected database, grant database access permissions through Database Role assignment.

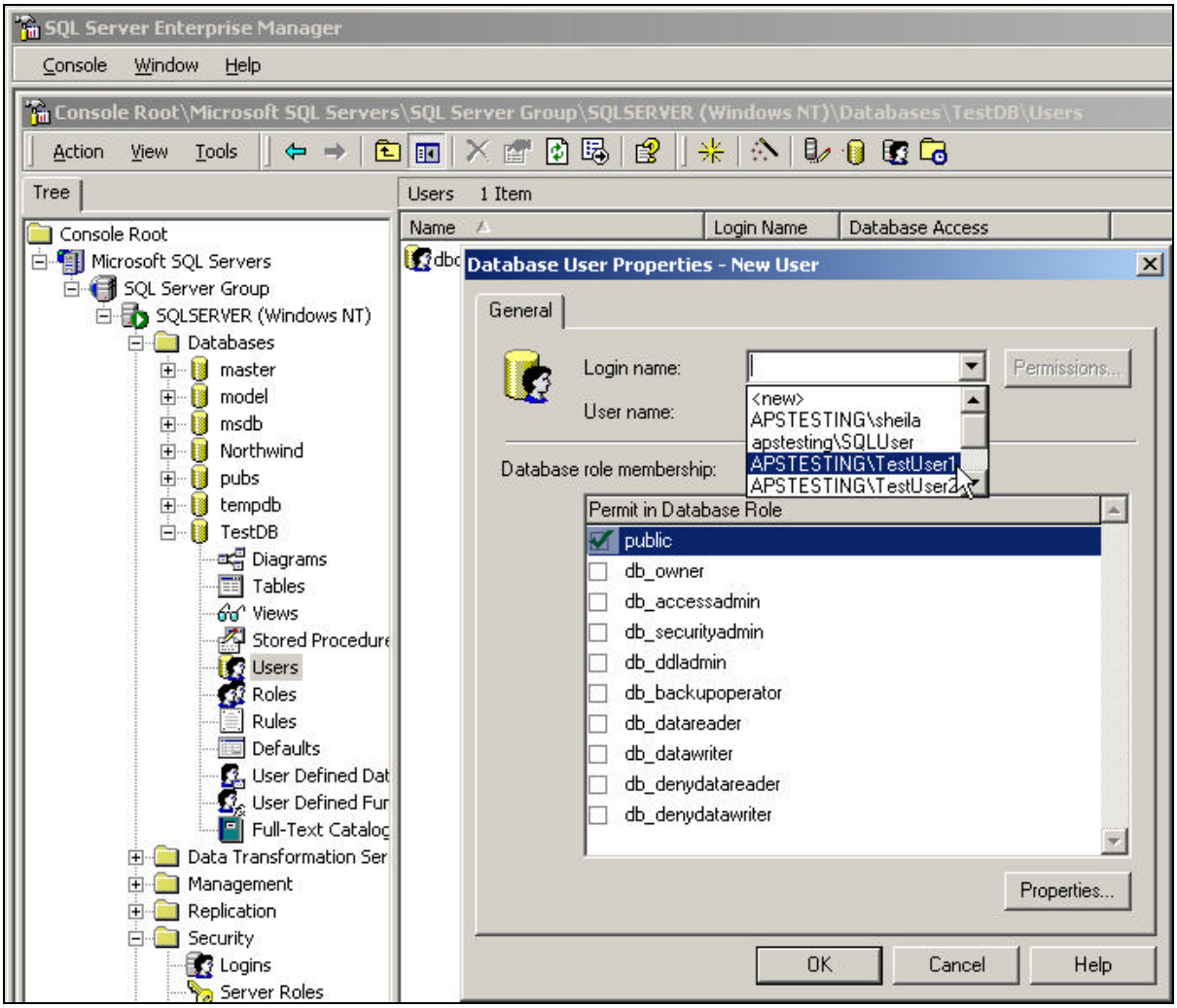
Figure 21 New Login Database Access Tab



Each SQL Server login can be mapped to a user in a database on the server. Mappings are defined during login creation (through default database selection and selecting other databases on the **Database Access Tab** of the **SQL Server Login Properties – New Login** window) and by adding the user manually to the desired database. It is possible to map a single login to users in multiple databases on the server. To add a user to a database:

1. Right click the **Users** folder and select **New Database User** from the pull-down menu. The **Database User Properties – New User** window is displayed.
2. Select a login name from the pull-down. If the **<new>** option is selected from the pull-down menu, the Create New Login wizard will start. This is the same wizard used to create logins described above.
3. Enter a database username to associate with the login. By default, the username and login will be the same.
4. Grant access permissions by selecting a Database Role.

Figure 22 New Database User



Two types of permissions exist in the SQL Server 2000 environment, statement permissions and object permissions. These permissions are assigned as Grant, Deny or Revoke. Permissions are cumulative; therefore, users may inherit permissions through group and role memberships. To ensure a user, group, or role is not granted access through other memberships, specifically set Deny permission on an object or statement for that user. A Deny permission set for any group or role to which a user belongs will override any Grant permission. Revoke reverses the set permission. If Deny permission were set for a user on an object, Revoke would undo the Deny. However, do not assume that the user would automatically be assigned Grant permission as a result of a Revoke. The Grant permission would have to be specifically assigned to the user or it could be assigned as a result of the user's group or role memberships. Be careful if considering invoking Deny permission on the public role. Since the public role includes all users of the database, the Deny permission would prevent any user from accessing the affected object, including the issuer of the Deny statement.

Statement permissions (or Data Definition Language (DDL) commands) include CREATE DATABASE, CREATE TABLE, ALTER TABLE, etc. Statement permissions are applied to the statement itself, not to a specific object in a database. These statement permissions are used to create and manage database objects. Statement permissions can only be granted by members of the sysadmin, db_owner, and db_securityadmin database roles. These permissions should be reserved for highly trusted users who are responsible for administering the database server objects. Generally, DBAs of production servers will not need to grant these permissions to additional users. However, it is common in a development environment for developers to be granted one or all of these statement permissions.

Table 6 Statement Permissions

Permission	Action
CREATE DATABASE	Login can create new databases. If this privilege is granted to a login that is not a member of the sysadmin server role, the login will become the dbo of the database and gain implicit ownership permissions as a result (can perform any action on the object it owns).
CREATE DEFAULT	Database user can create a default value for a table column.
CREATE FUNCTION	Database user can create user-defined functions.
CREATE PROCEDURE	Database user can create a stored procedure.
CREATE RULE	Database user can create a table column rule.
CREATE TABLE	Database user can create a table.
CREATE VIEW	Database user can create a view.
BACKUP DATABASE	Database user can create a backup of the database.
BACKUP LOG	Database user can create a backup log.

Object permissions (Data Manipulation Language (DML) permissions) include SELECT, INSERT, UPDATE, DELETE, EXECUTE and REFERENCES. These permissions are assigned to users in a database and are used to manage access to data within a database. Actions that can be performed on database objects are shown in Table 7.

Table 7 Object Permissions

Object Type	Permissions
Columns of a table or view	SELECT and UPDATE
Row	INSERT and DELETE (since a row affects multiple columns, these permissions can only be applied to a table or view, not to individual columns)
Stored procedures and functions	EXECUTE
User-defined functions	SELECT

These permissions can be assigned to individual server logins and database users. However, unless the SQL Server environment is small, with a minimal number of user accounts, it is recommended that roles be used to manage the delegation of permissions. Periodically review assigned permissions to ensure appropriate authorization is configured for database objects and statements.

Stored Procedures and Views

An excellent way to provide users with access to data in a database, without revealing the underlying database structure, is to limit user access through user-defined stored procedures and views. Security is enhanced on the underlying tables of a database with the use of stored procedures and views. This is because users only require permissions on the views and stored procedures, not the tables. As long as the owner of the table(s) used to create the view is the same as the owner of the view, permissions need only be granted on the view itself. Any action performed by the user is limited to the permissions granted to the user by a particular view or stored procedure. Views and stored procedures can be used together to control modifications to database objects. For example, a view can be created with SELECT permission and CREATE, UPDATE, and/or DELETE permissions can be controlled by predefined stored procedures that only permit users to perform authorized modifications to the database objects.

Stored Procedures

A stored procedure is made up of several SQL statements that are combined to form an executable object in the database. User-defined stored procedures can be created to restrict user access to database tables. Users are granted execute permission on the stored procedure, but have no access permissions on the tables themselves. Using stored procedures to modify data eliminates the possibility of incorrect ad hoc data modification in tables. Stored procedures are also a good countermeasure for SQL injection attacks because the only way access to data can be gained is through the stored procedures.

Although stored procedures are a good idea for protecting data in databases, they can be used to compromise a database server if there are flaws in implementation. A mistake sometimes made by developers is to execute stored procedures that send a string of commands, built from user input, directly to SQL Server. This technique allows for the possibility of users injecting code into the command string. Instead, execute stored procedures using ADO Command objects so each parameter can be properly populated.

SQL Server comes with several extended stored procedures (XPs). These XPs use external libraries to extend the functionality of SQL Server and are designed to increase the efficiency of administration. xp_cmdshell is a common XP used to gain access to the operating system of a SQL Server. The damage done using xp_cmdshell is limited by having the service accounts run as low-privileged users. xp_cmdshell is only one of several powerful XPs that can be used to compromise a SQL Server. Carefully review the need for each of these XPs and delete those that are not absolutely necessary for operations. Below is a list of stored procedures that should be deleted if not needed on a production server. It is not recommended that these stored procedures be deleted from a development server (except to test for functionality loss prior to deleting on a production server). The deletion of several of these stored procedures will result in the loss of some Enterprise Manager features. Prior to deleting stored procedures on a production server, test the affects on a development server for loss of functionality as a result. Execute sp_dropextendedproc to delete stored procedures. If deletion of these stored procedures is not an option in your environment, consider disabling them by denying execute permissions for database users and applications not requiring their use.

Table 8 Stored Procedures to Consider for Deletion

Procedures that affect Enterprise Manager		
sp_OACreate	sp_OADestroy	sp_OAGetErrorInfo
sp_OAGetProperty	sp_OAMethod	sp_OASetProperty
sp_OAStop		
Registry Access Procedures		
Xp_regaddmultistring	Xp_regdeletekey	Xp_regdeletevalue
Xp_regenumvalues	Xp_regremovemultistring	
Other Procedures to Consider Deleting		
sp_bindsession	sp_cursor	sp_cursorclose
sp_cursorfetch	sp_cursoropen	sp_cursoroption
sp_getbindtoken	sp_GetMBCSCharLen	sp_IsMBCSLeadByte
sp_replcmds	sp_replcounters	sp_repldone
sp_replflush	sp_replstatus	sp_repltrans
sp_sdidebug	xp_availablemedia	xp_cmdshell
xp_deletemail	xp_dirtree	xp_dropwebtask
xp_dsninfo	xp_enumdsn	xp_enumerrorlogs
xp_enumgroups	xp_enumqueuedtasks	xp_eventlog
xp_findnextmsg	xp_fixeddrives	xp_getfiledetails
xp_getnetname	xp_grantlogin	xp_logevent
xp_loginconfig	xp_logininfo	xp_makewebtask
xp_msver	xp_perfend	xp_perfmonitor
xp_perfsample	xp_perfstart	xp_readerrorlog
xp_readmail	xp_revokelogin	xp_runwebtask
xp_schedulersignal	xp_sendmail	xp_servicecontrol
xp_snmp_getstate	xp_snmp_raisetrap	xp_sprintf
xp_sqlinventory	xp_sqlregister	xp_sqltrace
xp_sscanf	xp-startmail	xp_stopmail
xp_subdirs	xp_unc_to_drive	

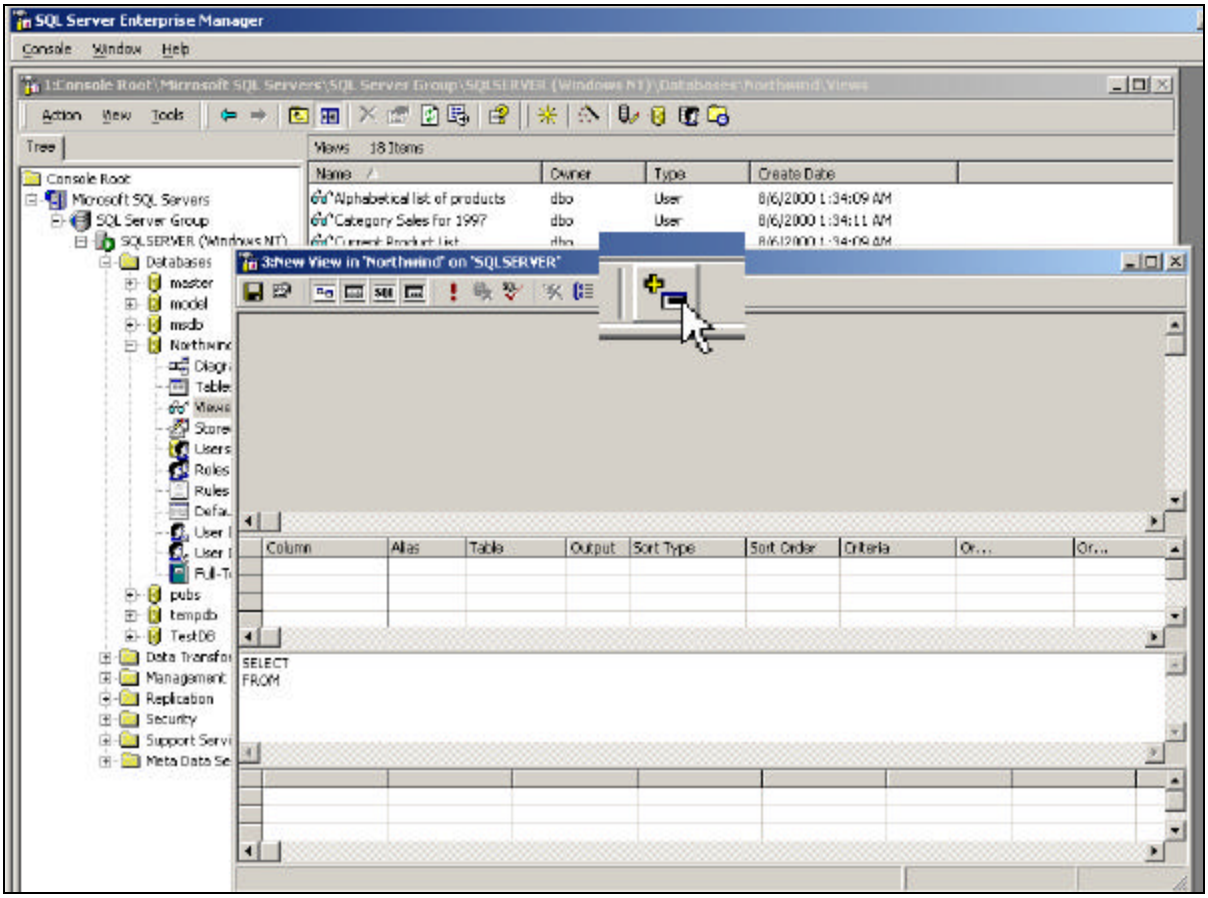
Views

During the creation of views, an encryption parameter can be applied to encrypt the definition of the view. View definitions are stored in the text column of the syscomments table unencrypted by default. If it is necessary to hide the definition of a view, use the WITH ENCRYPT option to store the definition encrypted. Keep in mind, however, views with encrypted definitions do not get published with replication. It is also a good idea to keep a copy of the original script in case the view needs to be recreated or altered at some point in time. Do not attempt to remove the definition from the syscomments table in an attempt to conceal the definition of a view. This will render the view unusable.

Views can be created using Transact-SQL or through the Create View Wizard of the Enterprise Manager. The CREATE VIEW page of the Books Online provides a detailed description of the Transact-SQL syntax and available options for view creation. Following is an example of creating a view through Enterprise Manager.

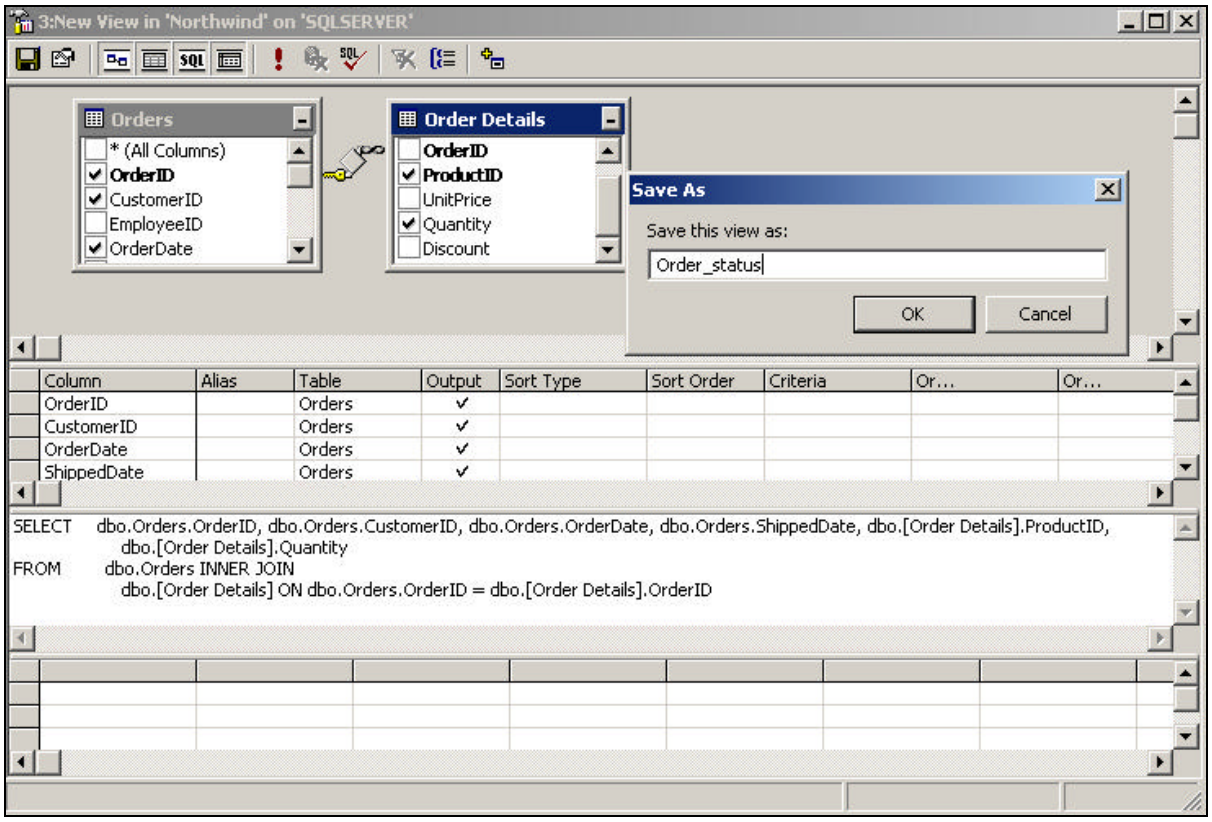
- Start the Create View Wizard by right clicking the Views icon in the target database and selecting New View. Views can be created from tables, other views, and/or user-defined functions that exist in the target database. Select the right-most icon on the toolbar to begin adding tables, other views, and user-defined functions to the query.

Figure 23 Create View, Add Table Button



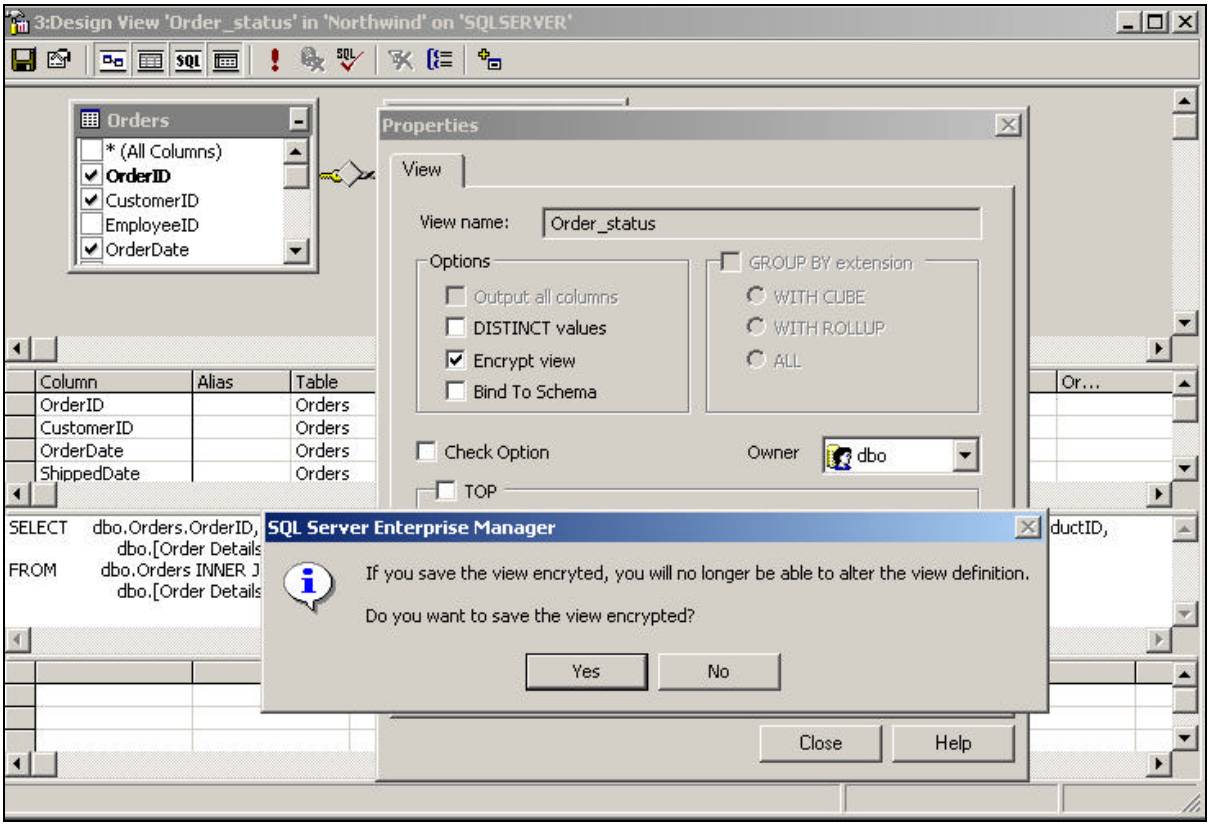
- Once tables, etc, have been added to the view, check the desired columns to add to the SELECT statement of the view definition. Click the floppy icon on the task bar to name and save the view.

Figure 24 Create View, Select Columns



- Once saved, select the Properties icon on the task bar (second from left) to select options, such as **Encrypt view**, to apply to the view. If the **Encrypt view** option is selected, a message box is displayed warning that the view cannot be altered once encrypted. If the options need to be changed after the view is saved, right-click the view and select **Design view** from the pull-down menu. If **Design view** is selected on a view that has been encrypted, an error message will be displayed. It is a good idea to select the **Bind To Schema** option to ensure changes to a view's referenced objects doesn't affect the view without the view being modified or dropped first.

Figure 25 Create View, Options



Roles

Roles are used in SQL Server 2000 to group logins and users for the purpose of assigning a set of permissions for access to objects and statements. Using roles eases the administration of object and statement permissions. When members are added to a role, they inherit the permissions assigned to that particular role. This eliminates the need to assign permissions individually to server logins and database users. Fixed roles are created during SQL Server installation and database creation. These roles cannot be deleted and are assigned a predefined set of statement permissions that cannot be altered. There is one exception. The **public** fixed database role can have default permissions altered. Fixed roles manage the distribution of statement permissions within SQL Server 2000 instances and databases. Three types of roles exist in SQL Server 2000; server roles, database roles, and application roles.

A note about the public role: The public role encompasses all users of a database; therefore, any changes to this role will affect all users of the database. Unless absolutely required for a particular environment, it is recommended that public NOT be used to assign permissions to users in a database. If the access is specifically denied to the public role in an attempt to prevent unauthorized users from gaining access, all users of the database will inherit the deny placed on the role. If access is granted to the public role in an attempt to allow authorized database users to perform a specified action, unauthorized users will enjoy the same privilege. All permissions should be carefully thought out and documented to ensure access is not provided unknowingly. Therefore, permissions should be placed on individual database users and database roles. The public role is assigned a default set of permissions during install that affects all databases on the server. Carefully review these permissions and remove access permission from this role. It is critical that the public role be removed from all system stored procedures. Several of these stored procedures contain buffer overflows that can be used by unauthorized users (via the default access granted to the public role) to gain access to the database server and/or to elevate user permissions.

Server Roles

During the installation of SQL Server 2000, several server roles are created by default. Server roles are fixed and are used to manage SQL Server administrative privileges by adding logins as members to specific fixed server roles. Logins can be assigned to various server roles to distribute administrative responsibility without granting full administrative server privileges. During the installation of SQL Server 2000, the BUILTIN\Administrators group (which is the Windows Administrators group) is added to the System Administrator SQL server role. Carefully consider whether this is required. Windows administrators are not required to manage databases and, therefore, should be removed from this role. Ensure the account created to run the SQL Server services has been added to the SQL Server System Administrator role prior to removing the BUILTIN\Administrators from the role. If not, SQL Server will have problems starting. The DBA group should also be added to this role. Table 9 lists the fixed server roles taken from the SQL Server Books Online, along with a description of the actions permitted for each role. To obtain a list of specific permissions for each role, execute the system stored procedure **sp_srvrolepermission**. Alternatively, select the role from the SQL Server Login Properties/**Server Roles Tab** and click the **properties** button. The Server Roles Property window for the selected role displays members of the role. Members can be added or deleted from here. The **Permissions Tab** of the window displays all of the permissions associated with the role.

Table 9 Fixed Server Roles

Role	Description
sysadmin System Administrators	Can perform any activity in SQL Server – It is recommended that only the DBAs and the account running SQL Server services be members of this role.
serveradmin Server Administrators	Can set server-wide configuration options and shut down the server.
setupadmin Setup Administrators	Can manage linked servers and startup procedures.
securityadmin Security Administrators	Can manage logins, manage CREATE DATABASE permissions, read error logs, and change passwords.
processadmin Process Administrators	Can manage processes running in SQL Server
dbcreator Database Creators	Can create, alter, and drop databases.
diskadmin Disk Administrators	Can manage disk files.
bulkadmin Bulk Administrators	Can execute BULK INSERT statements
public	All default permissions – all users belong this database role. This is the only fixed database role that can accept additional permissions. Permissions assigned to this role affect all users in the database. Users not specifically granted permission to an object, either explicitly or through role membership, use the permissions assigned to this role.

Database Roles

Fixed database roles are created by default for each database to delegate permissions at the database level. The scope of a database role is only within a particular database. If a user exists in multiple databases, adding the user to a role in one database has no affect on the role memberships of other databases in which the user belongs. As defined in the SQL Server Books Online, Table 10 lists the fixed database roles generated for each new database created, along with a description of the actions permitted for each role. To obtain a list of specific permissions for each role, execute the system stored procedure **sp_dbfixedrolepermission**.

Table 10 Fixed Database Roles

Role	Description
db_owner	Has all permissions in the database
db_accessadmin	Can add or remove users and roles
db_securityadmin	Can manage all permissions, object ownerships, roles, and role memberships
db_ddladmin	Can issue ALL DDL (can add, drop or modify database objects), but cannot issue GRANT, REVOKE, or DENY statements
db_backupoperator	Can issue DBCC, CHECKPOINT, and BACKUP statements
db_datareader	Can select (read) all data from any user table in the database
db_datawriter	Can insert, update, delete (modify) data in any user table in the database
db_denydatareader	Cannot select any data from an user table in the database
db_denydatawriter	Cannot modify any data in any user table in the database

User-defined database roles can and should be created to efficiently manage permissions on database objects when a fixed database role does not satisfy the permission requirements for a set of users. When assigning members to a role (fixed or user-defined), ensure the members (be it a group, users, or other roles) already exist within the target database. **Note:** Although existing roles can be added to new roles, keep in mind that deep levels of nested roles can negatively affect performance.

Application Roles

An ideal way to limit access to data within databases is to only permit access through application roles. Application roles contain no members and are an excellent way to provide users with data without having them actually access the data directly. Application roles are database specific. If an application has a need to perform actions on a different database, that database must have a guest user account enabled. During the creation of an application role, a password is specified. To establish a connection to a SQL Server instance with the permissions of the application role, the application must set the role and provide a password. The stored procedure used to set the role (sp_setapprole) and the password are not given to the application users. Instead, they are imbedded in the application connection string. Users connect to a SQL Server instance through execution of the application and gain the permissions assigned to the application role. In this way, users can be limited to accessing the database server through the application only. Even if application users have assigned permissions in the database, these permissions will be ignored as long as the user's connection through the application is active. This is an excellent solution for providing web servers controlled access to database resources.

It is recommended that applications be used to provide access to data within a database. However, application programmers need to ensure encryption is used to protect the role name and password in the connection string. There are several ways of protecting the role name and password of an application. One way is to set a parameter of the sp_setapprole that allows for the encryption of the password prior to sending it over the network, i.e., EXEC sp_setapprole “AppRole”, {ENCRYPT N “password”}. Unfortunately, the password is still stored in cleartext on the server. Store the encrypted password in the registry or file and provide the application with the key to decrypt the password. When the application launches, it decrypts the password and uses it in the sp_setapprole with the ENCRYPT option.

Several stored procedures are available to manage application roles, as follows (see Microsoft’s SQL Server Books Online for the proper syntax to use with these procedures):

- sp_addapprole – adds an application role to the current database
- sp_dropapprole – removes an application role from the current database
- sp_setapprole – sets the role and supplies the password to obtain permissions associated with an application role in the current database
- sp_approlepassword – changes the password of an application role in the current database

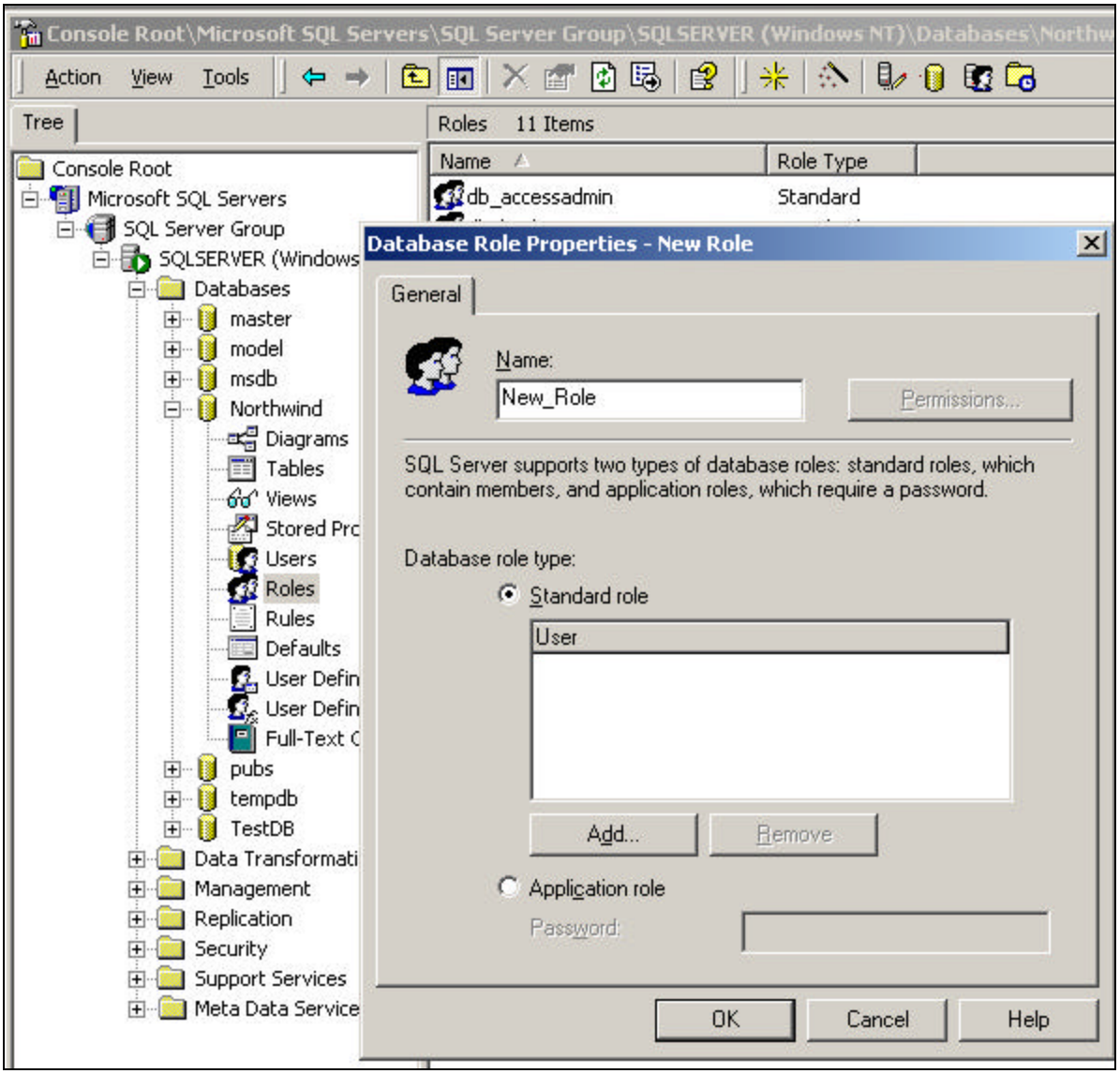
There exists a possibility for applications to be used in an attack on a SQL Server database. To help mitigate this possibility, grant the application role the least amount of permissions required for accessing the database. For example, if the application is not required to make modifications to data, grant SELECT permission on the desired tables/columns and deny all other permissions for the role. Prevent applications from performing actions (using statements such as SELECT, INSERT, UPDATE, and DELETE) directly on the database. Instead, incorporate these statements within stored procedures that are then called by the application. Using this approach helps to hide the internal database structure from the application.

Applications are also commonly used to perform SQL injection attacks. These attacks can be successful if the applications are poorly programmed. Minimize this risk by performing user input validation and do not permit users to input SQL commands that get executed directly against the database. Applications that use dynamic SQL statements require explicit access to the database’s tables, defeating the purpose of using stored procedures and views. It is very important to test stored procedures and applications to prevent SQL injection attacks. Use SQL profiler in a test environment, prior to porting an application to a production server, to monitor the server while injecting an exploit string into fields in your application to ensure an application behaves as expected. An application should validate numeric data (using the ISNUMERIC function and making sure the data entered is within a valid range), change single quotes to two single quotes, and use stored procedures that do not generate a string of commands to send to the SQL server for execution. Since SQL Server does not perform authentication on the user accessing the application, program the application with built-in checks to ensure permissions are not misused.

Perform the following to create roles within a database:

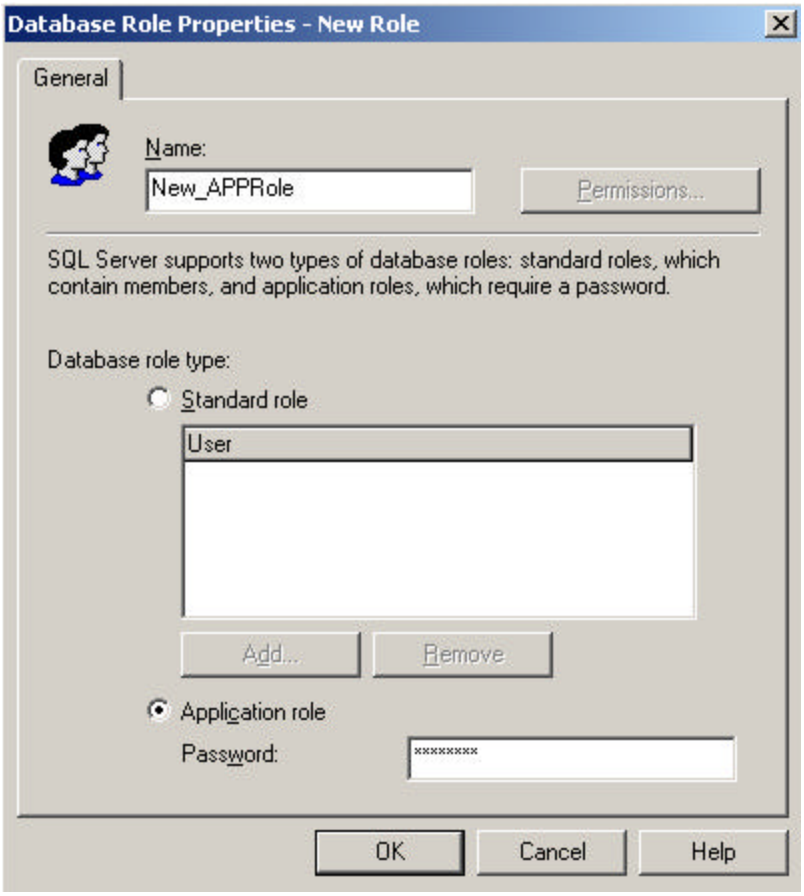
- Right-click the Roles icon under the desired database in the Enterprise Manager and select New Database Role from the pull-down menu. The Database Role Properties - New Role window appears.
- Enter the new role in the Name box (for standard role type). Click the Add button to add database users to the role.

Figure 26 Add New Role



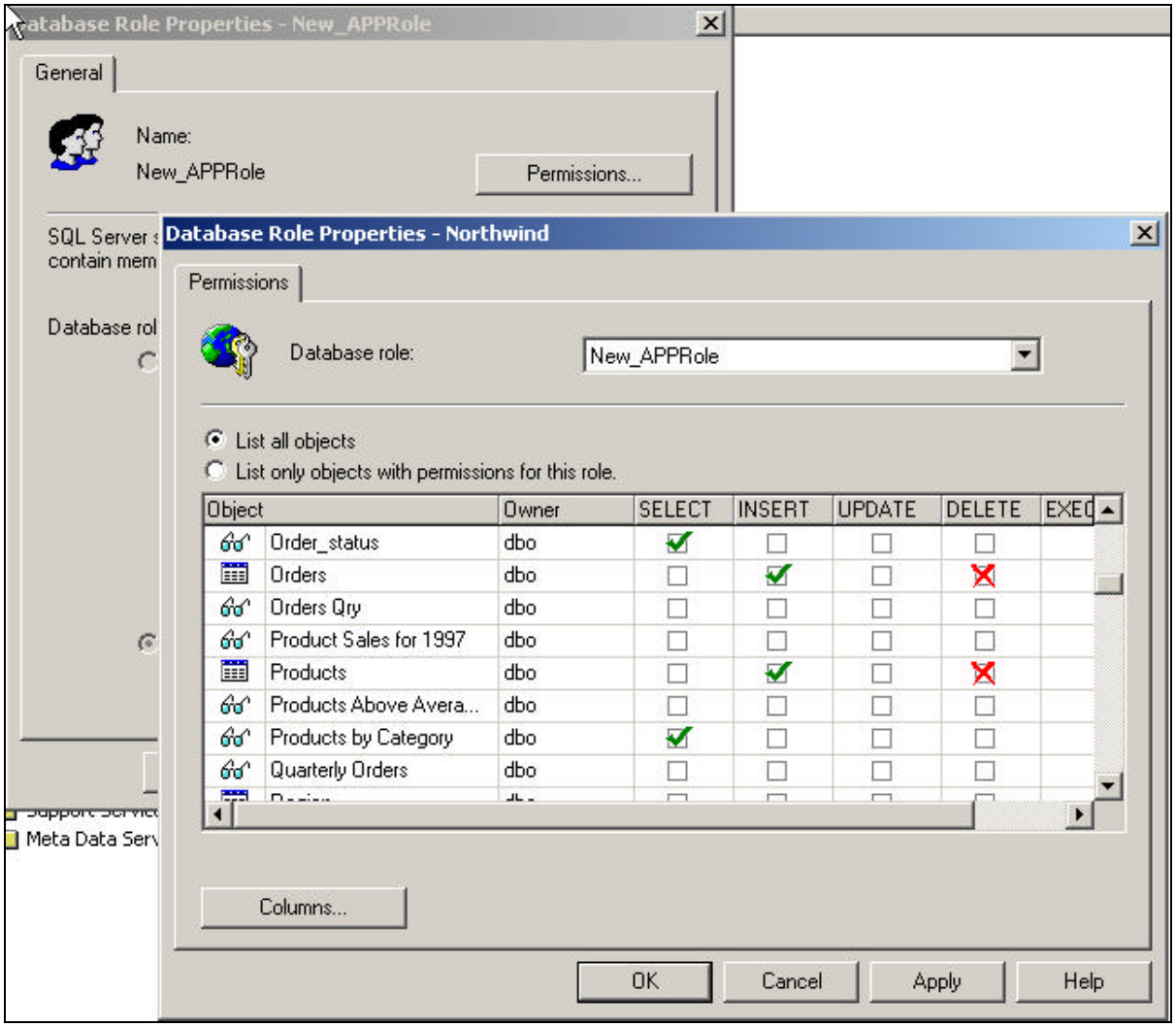
- If the role to be added is an application role, select this option and enter a complex password.

Figure 27 Add New Application Role



- Once the role(s) has been created. Right-click the role and select Properties to access the Permissions tab for setting role permissions in the database.

Figure 28 Setting Role Permissions



It is crucial to the security of the server, instances, and databases that careful planning takes place prior to assigning membership to roles. Granting permissions in excess of those required to perform assigned tasks can lead to a compromise of the database, an instance, or the entire SQL Server 2000 architecture.

In Figure 28, a check explicitly grants the permission, an X explicitly denies the permission, and a blank means the permission has not been set for this role. A blank would allow the permission to be granted to the role as a result of membership in other roles. An X would override any grant permission obtained as a result of other role memberships, resulting in a deny being set for the permission for that role.

This Page Intentionally Left Blank

Chapter

5

Database Maintenance

Backups

It is very important to include a disaster recovery policy in your site's security plan. There are several ways to backup the data on your server. Automatic backups, such as disk mirroring or disk duplexing, where there is a complete copy of the server's hard drive that can go online in the event the primary drive goes down, and manual backups. It is recommended not to rely on disk mirroring or duplexing exclusively. This strategy only protects against a single drive failure. In the event of a multiple disk failure, you must have other backups from which to recover. Here are some things to consider when implementing your backup strategy:

- How often does the database content change?
- Do all databases require regularly scheduled backups?
- How long can your site go without providing services?
- How much downtime is tolerable?
- Members of the Backup Operators group should have special logon accounts when performing backups. Backup privileges should not be assigned to regular user accounts.
- Create a backup plan and make sure it is documented.
- Consider keeping a set of backups offsite in the event of a natural disaster.
- Make a set of backups before and after any maintenance to the server. This includes any software or hardware changes to the system.
- It is very important that you make and TEST your backups regularly. Remember to include a strategy for backing up the Registry in your backup plan.
- Make sure that NTFS permissions are intact when a restore is done from a backup.

Backing Up SQL Server 2000 Databases

SQL Server 2000 backup options provide the capability to recover from data loss due to media failure, user errors, and permanent loss of a server. Four types of backups exist in SQL Server 2000: Full Database Backup, Differential Database Backup, File and Filegroup Backup, and Transaction Log Backup. The type of backup or combination of backups configured will depend on the developed backup plan and the chosen recovery model. It is also important to include the system databases in your backup plan. Microsoft's SQL Server Books Online describes each of these backup types in detail. The Database Maintenance Plan Wizard can assist in configuring the database backup plan. Keep in mind that backup plans can vary among databases and recovery models can be switched from their defaults to meet changing needs.

Recovery Models

Recovery models are used to determine how data is backed up and can be applied differently to each database. Three options are available:

(1) The Simple recovery model allows the data to be recovered to the most recent backup. Differential backups can be used with this recovery model to restore data to the last differential backup. Changes made after the last database or differential backup would be lost.

(2) The Full recovery model allows the data to be recovered to the point of failure. Transaction logs are used with database backups to restore a database to a specific point in time. The Full recovery model is the default recovery model for newly created databases, as the recovery model is inherited from the model database. If the recovery model is changed for the model database, all databases created after the change will inherit the new recovery model.

(3) Bulk-Logged recovery model allows bulk-logged operations. It is similar to the Full recovery model, but logging of bulk copy operations is significantly minimized to enhance performance. Point in time recovery is not supported.

This table (source: SQL Server Books Online) provides an overview of the benefits and implications of the three recovery models.

Table 11 Recovery Models

Recovery Model	Benefits	Work Loss Exposure	Recover To Point In Time?
Simple	Permits high-performance bulk copy operations. Reclaims log space to keep space requirements small.	Changes since the most recent database or differential backup must be redone. Not recommended for production environments.	Can recover to the end of any backup. Changes made after that must be redone.
Full	No work is lost due to a lost or damaged data file. Can recover to an arbitrary point in time (for example, prior to application or user error).	Normally none. If the log is damaged, changes since the most recent log backup must be redone.	Can recover to any point in time.
Bulk-Logged	Permits high-performance bulk copy operations. Minimal log space is used by bulk operations.	If the log is damaged, or bulk operations occurred since the most recent log backup, changes since that last backup must be redone. Otherwise, no work is lost.	Can recover to the end of any backup. Then changes must be redone.

Backup Types

(See SQL Server Books Online for more details)

Full Database Backups – Makes a complete copy of the database. This type of backup may take a long time and may be best combined with differential backups and transaction log backups in a backup plan to restore the database to a particular point in time.

Differential Backups – Copies only the data that has changed since the last database backup. This type of backup is used in conjunction with full database backups to enable the restore of a database to the point at which the differential backup was completed. This type of backup is smaller and runs faster than full database backups. This means they can be run more often. Differential backups can be used with the Simple recovery model when it is not desired to run full backups often.

Transaction Log Backups - The transaction log is a serial record of all the transactions that have been performed against the database since the transaction log was last backed up. With transaction log backups, you can recover the database to a specific point in time or to the point of failure. Transactional log backups are generally used with full database backups and differential backups.

File and Filegroup Backups – Specified files and filegroups are backed up instead of backing up the entire database. Transaction log backups are required with this type of backup. File and filegroups can be restored from full backups as well. This should be considered prior to performing a full restore, as a full recovery may be accomplished quicker by restoring only those files that were lost.

System Databases Backups – The system databases include the master, msdb, and model. Although the tempdb is a system database, it does not include any permanent data and, therefore, backups are not required. It is important to include the other system databases in your backup plan. The default recovery model for the master and msdb databases is the Simple recovery model. Only full database backups of the master database can be created. It is not necessary to change from the Simple recovery model for these databases since they tend to be small in size.

The master database is updated any time a database is created or deleted. It also records all login accounts and server-wide or database configuration settings. The msdb database contains all SQL Server Agent operations. These include alerts, jobs, and scheduling information. Backup and restore history information is also recorded in this database and is used by SQL Server Enterprise Manager to propose a plan for restoring databases and applying any transaction log backups. These databases are a critical part of a database environment and must be backed up regularly to avoid significant downtime in the event a recovery becomes necessary. If the master database becomes unusable, it can be recreated using the rebuildm.exe command. This will restore the master database, as well as the msdb and model databases, to their default states. Backups must then be applied to restore these databases to the desired state.

Newly created databases are based on the model database. All objects in the model database are copied to the new database, including all configuration settings. For this reason, the default recovery model is Full. Regular backups of the model database are not necessary. However, since the tempdb is created each time SQL Server starts, the model database must exist. Therefore, it is important to backup the model database whenever modifications are made to it.

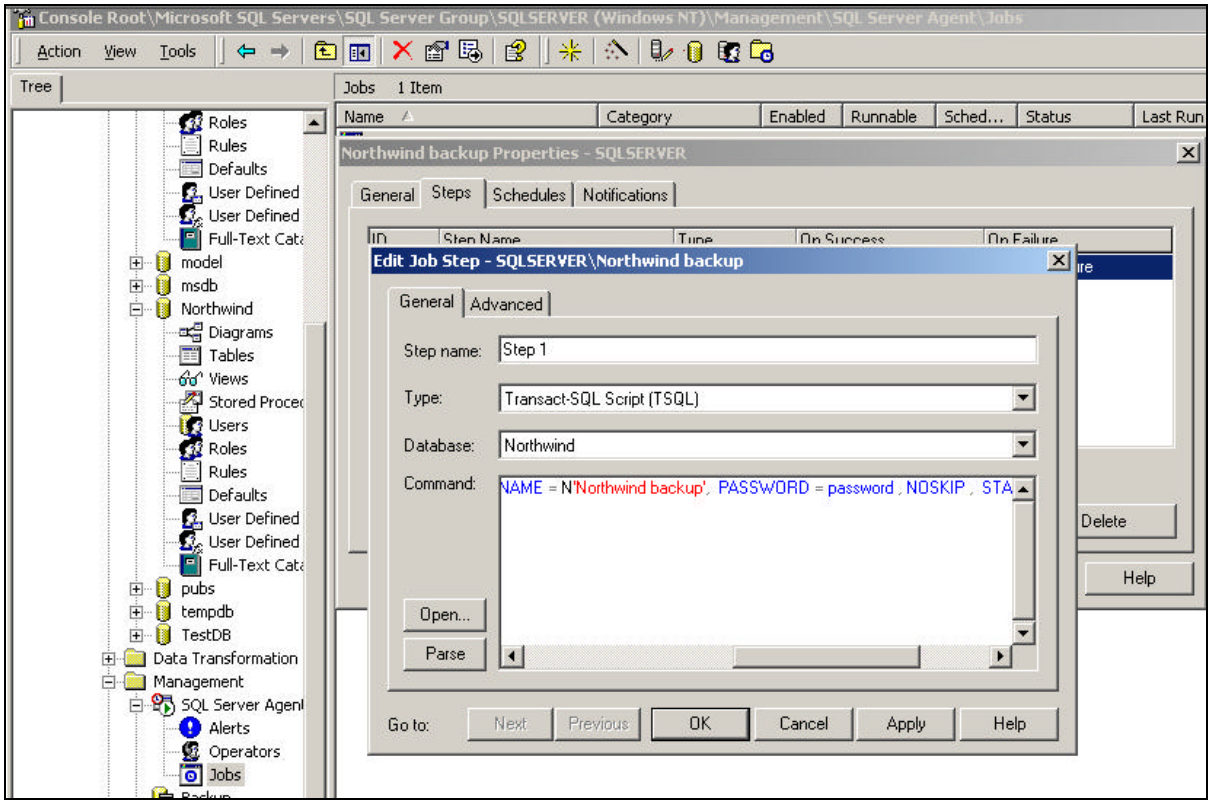
Backup Passwords

An option exists in SQL Server 2000 to provide a password to protect backup media. The PASSWORD and MEDIAPASSWORD options of the BACKUP command are used to set a password on backups. If a password is provided during backups, it will be required to perform a restore or append to an existing media set. Password protecting the backup media ensures the backups will not accidentally be overwritten (although deleting the backup is still possible if the media is reformatted), the media will not be added to, and unauthorized restores are prevented.

Passwords can be applied to a media set or backup set. Media set passwords protect all data stored on the media set (entire database backup). Backup set passwords are applied to each backup set (backup files or filegroups) on the media. Each backup set has its own password that is set when the backup set is written to the media.

It is recommended that passwords be used to protect backup media. A password option is not available through the Enterprise Manager. To set a password after scheduling a database backup, select the jobs icon under the Management/SQL Server Agent in the Enterprise Manager. Double click the scheduled backup to display its properties. Click the **Steps** tab and click **Edit** to display the generated SQL for the backup job. Modify the SQL to include the PASSWORD or MEDIAPASSWORD parameter. Figure 29 shows an example of setting a password on a backup. It is important to note that although it is recommended that passwords be used to protect backups, it does not protect the data if the media is not protected from theft. Backup media must be protected commensurate with the security level of the data stored on it.

Figure 29 Set Password on Backup Set



Note: In many environments, users that do not require access to the databases on a server are required to perform the database backups. For this scenario, create a “backup operators” group and assign the group to a SQL Server 2000 login that is only granted the BACKUP DATABASE permission.

This Page Intentionally Left Blank

Chapter

6

Additional Security Issues

Antivirus and Intrusion Detection Programs

There are numerous public sector sources for information on antivirus and intrusion detection products. A suggested starting point for a review of the various available antivirus software packages is the Stroud's CWSApps web site at http://cws.internet.com/virus.html. This Web page contains the most popular virus scanners along with useful information to help in the selection of a virus software package. Database servers are critical elements of any network environment. These servers, as well as other network servers, require protection from unauthorized access. Implementing an intrusion detection system can provide a layer of security by presenting alerts when known attacks are attempted against the network. Some systems are capable of not only detecting, but also performing defined actions when attacks are detected.

Implement a robust antivirus program and intrusion detection system as part of the security policy for your entire site.

Audits

There are several ways to implement auditing for SQL Server 2000. The best solution will most likely be a combination of the available options, based on the auditing requirements specified in the site's security policy. In this document, several auditing options are discussed, along with some security-related actions that can be captured. The most important thing about audit information is to ensure it gets reviewed regularly. It does a site no good to collect audit data if it just takes up space on the hard drive. Keep in mind that auditing can have a severe impact on performance.

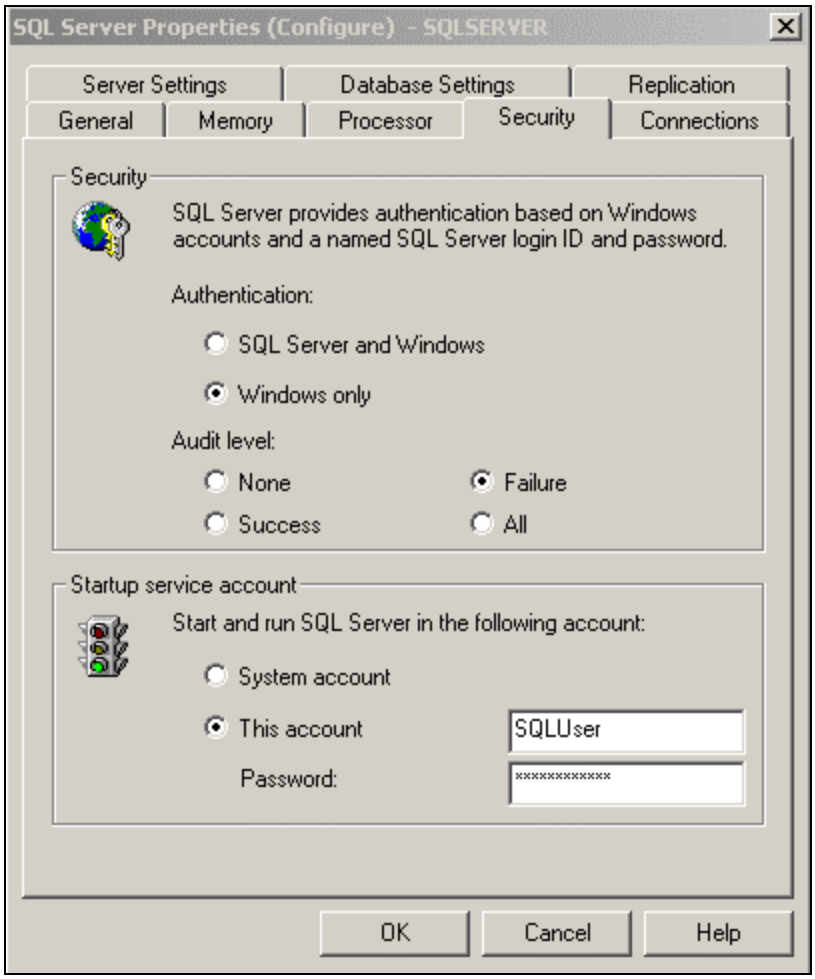
By default, SQL Server has authentication logging disabled. It is recommended that authentication logging be enabled. Available options include success, failure, and all. The selected option will depend on the requirements defined in the site's security policy. Minimally, it is recommended that auditing for failed logins be enabled. Through Enterprise Manager, select the audit option from the Security tab of SQL Server Properties (shown in Figure 30).

The following SQL command can also be issued to the server to configure authentication auditing:

Master..xp_instance_regwrite N'HKEY_LOCAL_MACHINE', N'SOFTWARE\Microsoft\MSSQLServer\MSSQLServer' ,N'AuditLevel' , REG_DWORD, **3**

(Note: 3=all, 2=failure, 1=success, 0=none)

Figure 30 Set Audit Level



Log records for these events appear in the Windows application log, the SQL Server error log, or both, depending on how you configure logging for SQL Server. This is even more crucial in a mixed mode environment. The logs should be monitored regularly to detect brute force attacks against SQL Server login accounts.

SQL Profiler is a tool that can be used to generate and manage audit trails. Events related to security can be captured using the SQL Profiler. If too many events are defined for capture in a trace, review of the trace can become difficult. Consider creating more than one trace to capture required SQL Server activity. Table 12 lists the security-related events that can be captured using the SQL Profiler (as defined in SQL Server Books Online). See "Security Audit Data Columns" in SQL Server Books Online for a detailed description of the data collected for each of these events.

Table 12 Security Audit Events

Event	Description
Audit Add DB User	Records the addition and deletion of database users
Audit Add Login to Server Role	Records the addition or removal of logins to or from fixed server roles
Audit Add Member to DB Role	Records the addition and removal of members to and from a database role (fixed or user-defined)
Audit Add Role	Records the addition or deletion of database roles
Audit Addlogin	Records the addition and deletion of SQL Server logins
Audit App Role Change Password	Records password changes on application roles
Audit Backup/Restore	Records BACKUP and RESTORE actions
Audit Change Audit	Records AUDIT modifications
Audit DBCC	Records issued DBCC commands
Audit Login	Records all new connection events since the trace was started
Audit Login Change Password	Records password changes of SQL Server logins (Passwords are not recorded)
Audit Login Change Property	Records modifications to login properties, other than passwords
Audit Login Failed	Records failed login attempts
Audit Login GDR	Records grant, revoke, and deny actions on Windows account login rights
Audit Logout	Records all new disconnect events since the trace was started
Audit Object Derived Permissions	Records when a CREATE, ALTER, or DROP command is issued for the specified object
Audit Object GDR	Records permissions events for GRANT, DENY, REVOKE objects
Audit Object permission	Records the successful or unsuccessful use of object permissions
Audit Server Starts and Stops	Records shutdown, start, and pause activities for services
Audit Statement GDR	Records permission events for GRANT, DENY, REVOKE statements
Audit Statement Permission	Records the use of statement permissions

It is possible to audit the following aspects of SQL Server through SQL Profiler:

- Date and time of event.
- User who caused the event to occur.
- Type of event.
- Success or failure of the event.
- The origin of the request (for example, the Microsoft Windows 2000 computer name).
- The name of the object accessed.
- Text of the SQL statement (passwords replaced with ****).

Create a new Trace by selecting **New->Trace** from the SQL Profiler **File** pull-down menu. Select the SQL Server instance to audit and select **OK** to display the Trace Properties window. From this property sheet, a new trace can be defined using the Blank template, as shown in Figure 29. Other templates are available from the drop-down menu, which can be modified to suit a site's particular audit requirements. Trace output options are defined on the General tab as well as the option to define a start and stop time for the trace.

Figure 31 Trace Properties – General Tab

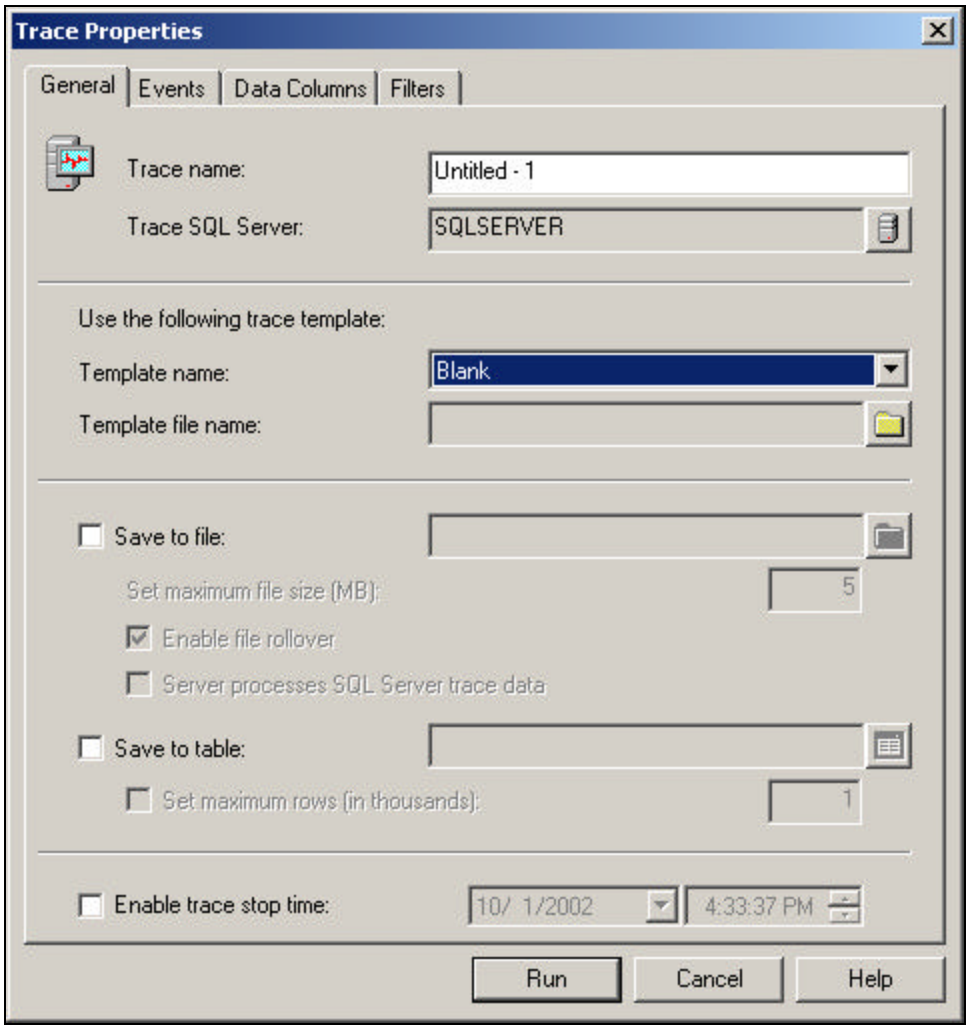
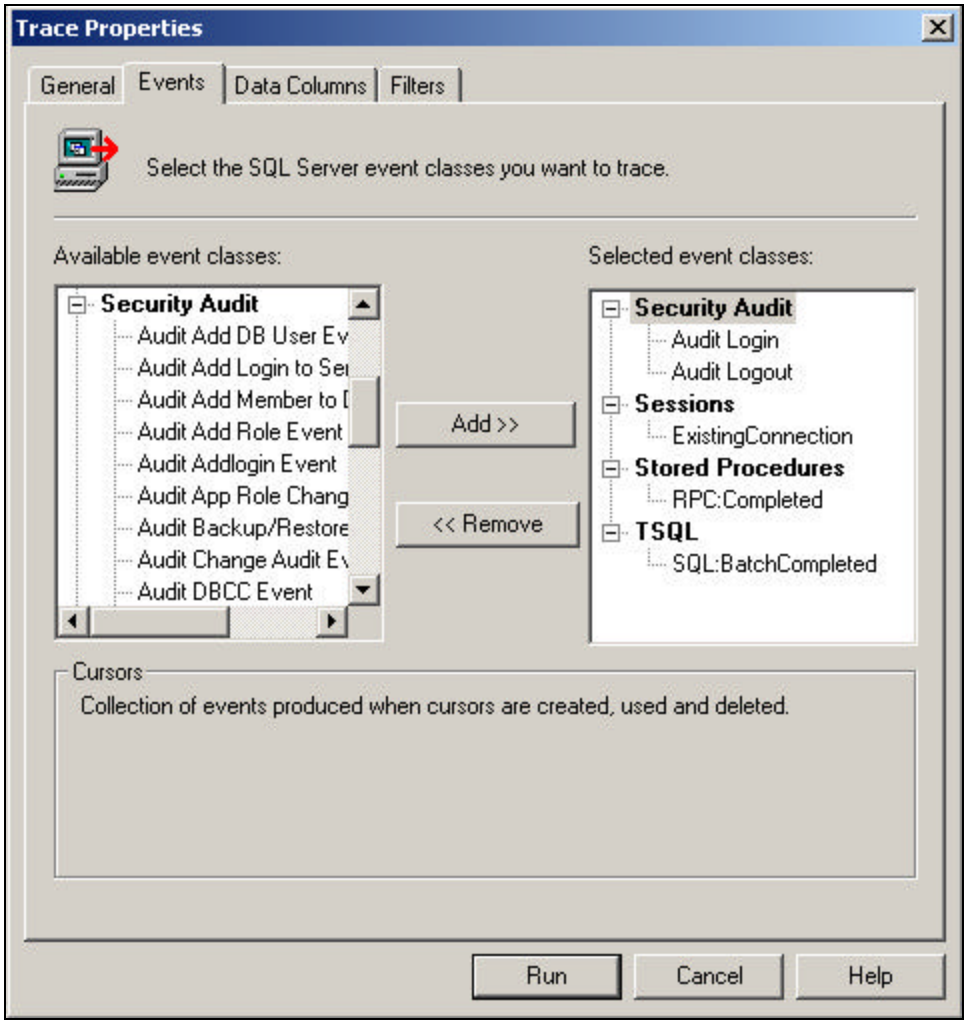
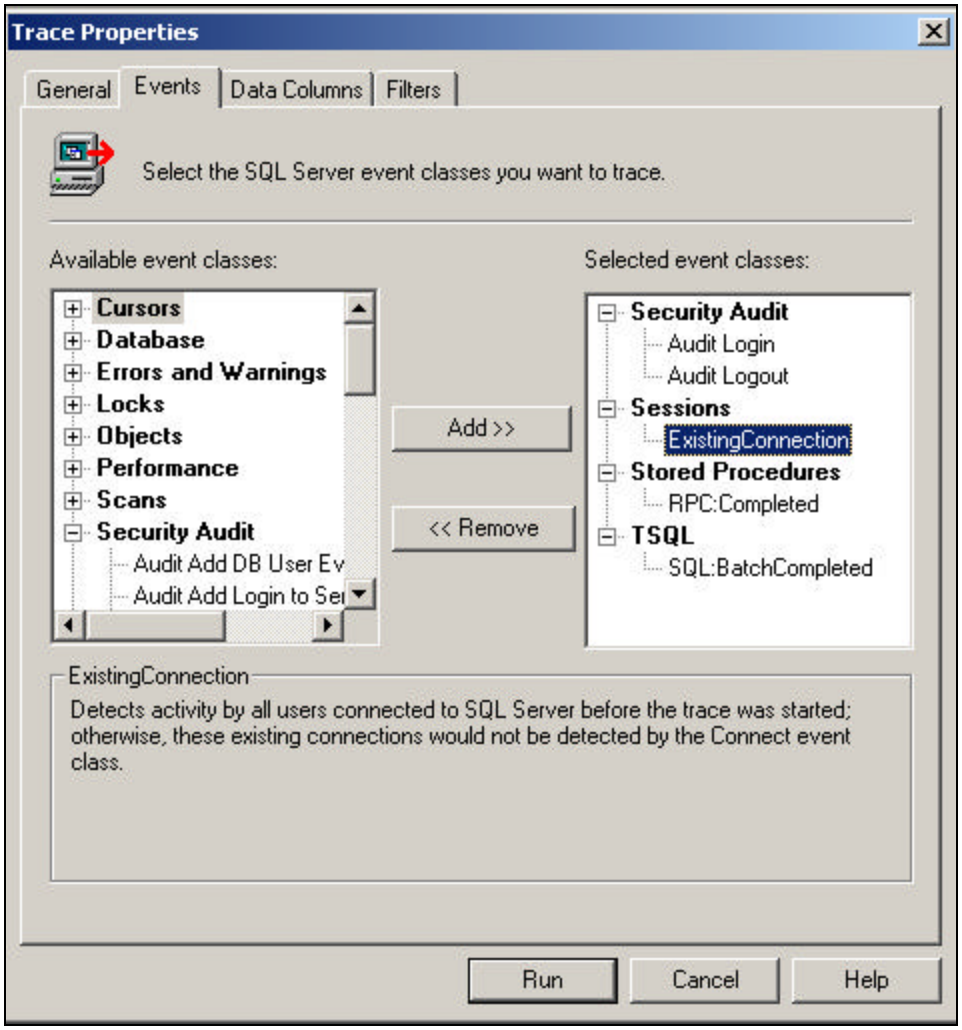


Figure 32 Trace Properties – Security Events



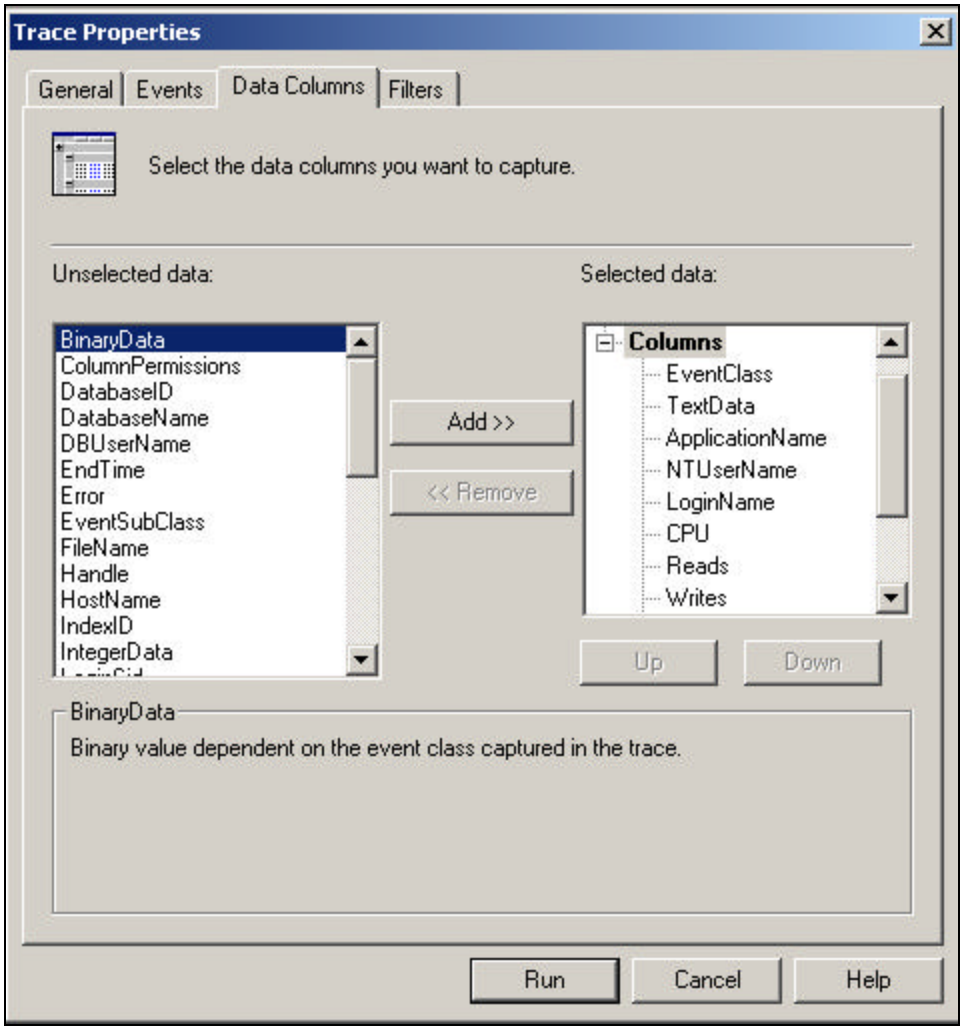
The Events tab lists categories of events that can be added to the trace. Security Audit is a category that lists several security-related events that can be captured in a trace. Select events that are consistent with the audit requirements defined in the site's security policy. A description of each of these security events is defined in Table 12.

Figure 33 Trace Properties – Sessions Event Class



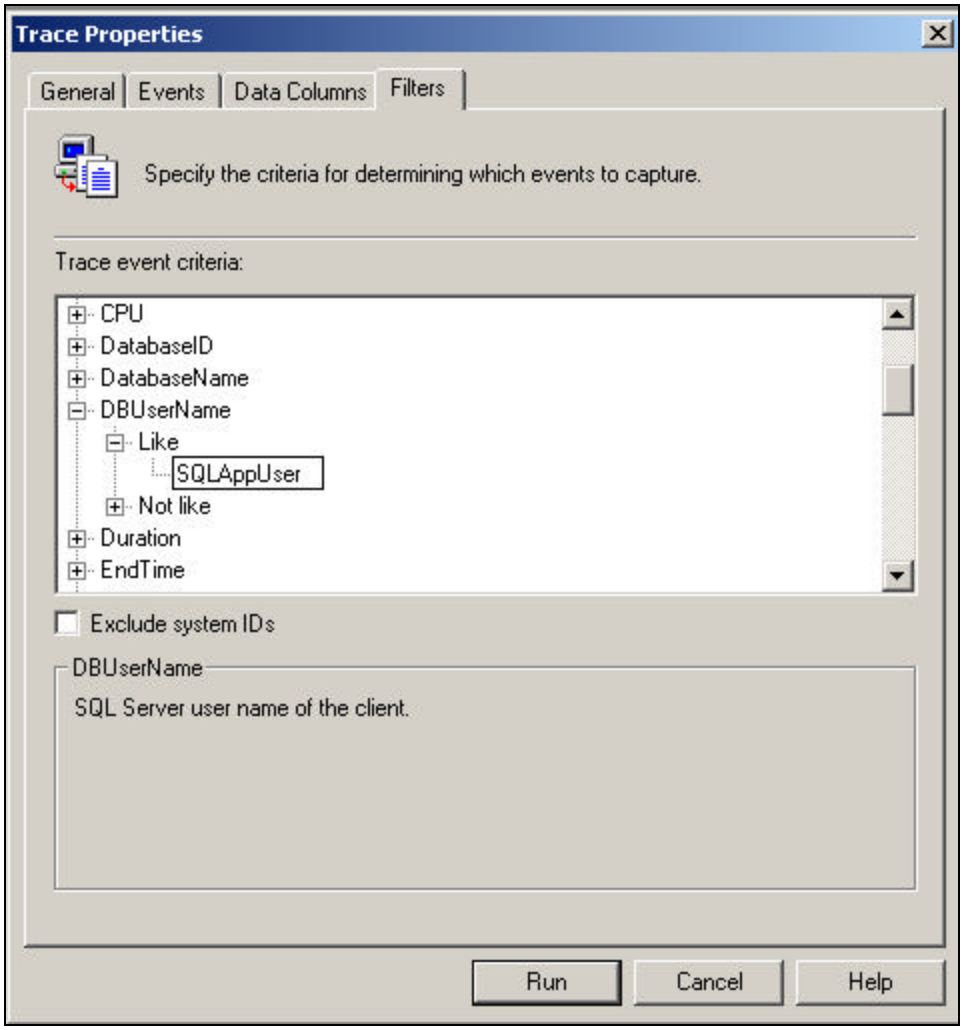
The event classes shown in Figure 32 above are those defined in the SQLProfilerStandard template. As mentioned in the comment box of the Events tab, it is important to include ExistingConnection in the event classes definition to ensure all connections are included in the defined audit.

Figure 34 Trace Properties – Data Columns Tab



The Data Columns tab allows for the fine-tuning of the data to be captured. As generated audit data can be quite large, limit captured data to only what is required. Another option is to create several traces, where each is designed to target a specific security requirement.

Figure 35 Trace Properties – Filters Tab



Filters allow for the capture of data based on defined criteria. Click on all of the available filters to display a description of the data that can be captured based on the particular filter.

C2 auditing is available for environments with C2 certified systems. Be aware that C2 auditing can generate an enormous amount of data. Refer to the *C2 Administrator's and User's Security Guide* from Microsoft for more information regarding C2 certified systems and auditing.

Appendix

A

Further Information (Resources)

McLean, Ian, “Windows 2000 Security Little Black Book”, www.coriolis.com

Rankins, R., Jensen, P., Bertucci, P., “Microsoft SQL Server 2000 Unleashed” USA, 2002, Sams Publishing. IBN# 0-672-31997-7

Microsoft’s SQL Server 2000 Help pages

Microsoft SQL Server Books Online

Microsoft SQL Server 2000 Introduction Manual (accompanies the SQL Server 2000 software)

C2 Administrator’s and User’s Security Guide from Microsoft

Downin, K., LaFountain, S., “Microsoft Windows 2000 IPsec Guide”, C4-045R-01 http://nsa1.conxion.com/win2k/guides/w2k-20.pdf, August 13, 2001

Waymire, R., Thomas, B., “Microsoft SQL Server 2000 Security” http://www.microsoft.com/partner/businessresources/salesresources/whitepapers/Security_SQLServer2000.doc, 2000, Microsoft

“Guidelines for Selecting a Service Logon Account,” http://msdn.microsoft.com/library/default.asp?url=/library/en-us/netdir/ad/guidelines_for_selecting_a_service_logon_account.asp, Microsoft, November 2001

Riley, S., “Ask Us About … Security – December 15, 2001,” http://www.microsoft.com/technet/treeview/default.asp?url=/TechNet/columns/security/askus/aus1201.asp, Microsoft

“Security Operations Guide,” http://www.microsoft.com/technet/treeview/default.asp?url=/TechNet/security/prodtech/windows/windows2000/staysecure/default.asp, Microsoft

“Setting Up Windows Services Accounts,” http://www.microsoft.com/library/default.asp?url=/library/en-us/instsql/in_overview_6k1f.asp, September 2001, Microsoft

“Troubleshooting MSSQLServer or SQLServerAgent Services User Accounts,” http://msdn.microsoft.com/library/default.asp?url=/library/en-us/trblsql/tr_servdatabse_3c37.asp, Microsoft

www.microsoft.com/windows2000/library/howitworks

www.sqlservercentral.com

www.sqlsecurity.com