Microsoft®

SQL Server 2000 High Availability

Volume 2
Deployment

patterns & practices
proven practices for predictable results

ISBN: 0-7356-1836-4

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft, MS-DOS, Active Directory, and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

© 2002 Microsoft Corporation. All rights reserved.

Version 1.0

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Contents

Preface **vii**
System Requirements . . . viii
Document Conventions . . . ix
Acknowledgments . . . x

Chapter 1
Implementing Failover Clustering **1**
Understanding Failover Clustering . . . 1
Deploying Failover Clustering . . . 3
Understanding the Logical Design . . . 3
Configuring Accounts and Names and Obtaining IP Addresses . . . 5
Configuring the Network . . . 6
Configuring Storage Devices . . . 7
Installing and Configuring MSCS . . . 9
Installing and Configuring SQL Server on the Cluster . . . 13
Applying the Most Recent SQL Server Service Pack . . . 16
Performing Additional Configuration Tasks . . . 17
Failing Over to a Secondary Node . . . 21
Testing Failover . . . 21
Minimizing Failover Time . . . 21
Reconnecting Clients . . . 23
Failing Back to the Original Node . . . 23
Restoring a Node After a Catastrophic Failure . . . 24
References . . . 26

Chapter 2
Implementing Log Shipping **27**
Understanding Log Shipping . . . 27
Deploying Automatic Log Shipping . . . 29
Understanding the Logical Design . . . 30
Creating and Sharing Synchronization Folders . . . 33
Creating Database Folders on Each Standby Server . . . 34
Creating the Log-Shipping Jobs and Initializing the Standby Database . . . 35
Creating and Executing a Login Synchronization Job . . . 39
Scripting and Copying Jobs, Alerts, and Operators . . . 43
Copying DTS Packages . . . 46
Changing the Role of a Standby Server . . . 46

Chapter 3
Implementing Transactional Replication 51
Understanding Transactional Replication . . . 51
Deploying Transactional Replication . . . 53
Understanding the Logical Design . . . 54
Creating an Empty Database on Each Subscriber . . . 56
Creating and Sharing the Snapshot Folder . . . 57
Creating and Sharing the Synchronization Folder . . . 58
Preparing the Schema for Replication . . . 58
Configuring the Distributor . . . 63
Publishing the Production Database . . . 64
Configuring and Initializing Subscribers . . . 67
Creating and Executing a Login Synchronization Job . . . 69
Scripting and Copying Jobs, Alerts, and Operators . . . 73
Copying DTS Packages . . . 76
Changing the Role of a Subscriber . . . 77
Reverting to the Original Publisher . . . 78

Chapter 4
Implementing Network Load Balancing 79
Understanding Network Load Balancing . . . 79
Deploying NLB . . . 80
Understanding the Logical Design . . . 81
Obtaining the Public IP Address, Server Name, and Port . . . 82
Configuring the NLB NIC and Installing and Configuring NLB . . . 83
Enabling the Primary Server Node . . . 86
Changing the Primary Server with NLB . . . 86

Chapter 5
Implementing Remote Mirroring and Stretch Clustering 89
Understanding Remote Mirroring and Stretch Clustering . . . 89
Deploying a Remote Mirror . . . 91
Failing Over to a Remote Mirror . . . 94
Deploying Stretch Clustering . . . 94
Understanding the Logical Design . . . 95
Configuring Accounts and Names and Obtaining IP Addresses . . . 97
Configuring the Network . . . 97
Configuring Storage Devices . . . 99
Installing and Configuring MSCS . . . 100
Installing and Configuring SQL Server on the Cluster . . . 104
Applying the Most Recent SQL Server Service Pack . . . 108
Performing Additional Configuration Tasks . . . 108

Failing Over to a Secondary Node in a Stretch Cluster 112
Testing Failover 112
Minimizing Failover Time 113
Performing a Manual SAN Failover 114
Reconnecting Clients 115
Failing Back to the Original Node 115
Restoring a Node After a Catastrophic Failure 116
References 118

Preface

The Microsoft® SQL Server 2000 High Availability Series helps you plan and deploy a highly available data center that uses Microsoft SQL Server 2000. If you are a consultant, designer, or systems engineer involved in designing and deploying a SQL Server 2000 data center, this series is for you. The series is presented in two pieces that correspond to the design and roll-out process: a Planning Guide, which helps you design a data center to achieve the level of availability necessary for the business environment, and a Solution Guide that helps you implement targeted server redundancy solutions to limit unavailability caused by server failure or planned downtime.

Specifically, the Planning Guide covers the following topics:

- Best Practices to Prevent Downtime
- Using Backups to Recover from Disaster
- Using Hardware Redundancy to Prevent Downtime
- Using Redundancy to Minimize Downtime

The Solution Guide provides information on the following topics:

- Clustering
- Log Shipping
- Replication
- Using Network Load Balancing (NLB) to Switch SQL Servers
- Stretch Clustering

Note that availability of third-party applications and scaling up are beyond the scope of this guidance.

The SQL Server 2000 High Availability Series provides information that is both practical and prescriptive. If you need more in-depth discussions of the concepts behind this material, you should refer to resources such as the Microsoft SQL Server 2000 documentation, the Windows 2000 Resource Kit, and additional documentation and clustering documents available from the Microsoft TechNet Web site.

The guides include information from consultants working in the field and from organizations that have already confronted and solved issues that affect the availability of SQL Server 2000 in enterprise data centers. This series provides guidance that is currently defined as best practice. We hope you enjoy reading this series and that you find the material helpful, informative, and interesting.

System Requirements

Microsoft SQL Server 2000 requires the minimum system configuration shown in Table 0.1. (For more detailed requirements, see your Microsoft SQL Server documentation.)

Table 0.1 Microsoft SQL Server 2000 System Requirements

Component	Requirement
Processor	Intel Pentium or compatible 166-megahertz (MHz)
Operating System	SQL Server 2000 Enterprise Edition and Standard Edition run on Microsoft Windows® 2000 Server, Windows 2000 Advanced Server, Windows 2000 Datacenter Server operating systems, Microsoft Windows NT® Server version 4.0 Service Pack 5 (SP5) or later, and Windows NT Server 4.0 Enterprise Edition with SP5 or later.
Memory	Enterprise Edition: 64 megabytes (MB) RAM; 128 MB recommended. Standard Edition: 64 MB. (Additional memory may be required depending on operating system requirements.)
Hard Disk	95–270 MB free hard-disk space for the server; 250 MB for a typical installation. 50 MB free hard-disk space for a minimum installation of Analysis Services; 130 MB for a typical installation. 80 MB free hard-disk space for English Query (supported on the Windows 2000 operating system but not logo certified). Desktop Engine requires 44 MB of available hard-disk space (Hard disk requirements vary based on your system configuration and the applications and features you choose to install.)
Drive	CD-ROM drive
Display	VGA or higher-resolution monitor
Other Devices	Microsoft Internet Explorer version 5.0 or later. Windows 95, Windows 98, Windows Me, Windows NT 4.0, Windows 2000 and Windows XP have built-in network software. Additional network software is required if you are using Banyan VINES or AppleTalk ADSP. Novell NetWare IPX/SPX client support is provided by the NWLink protocol of Windows-based networking. Client Support: Windows 95, Windows 98, Windows Me, Windows NT Workstation 4.0, Windows 2000 Professional, Windows XP Professional and Windows XP Home Edition are supported. UNIX, Apple Macintosh, and OS/2 require Open Database Connectivity (ODBC) client software from a third-party vendor.

Microsoft Windows 2000 Server requires the minimum system configuration shown in Table 0.2. (For more detailed requirements, see your Windows 2000 Server documentation.)

Table 0.2 Microsoft Windows 2000 Server Requirements

Component	Requirement
Processor	133 MHz or higher Pentium-compatible CPU
Memory	256 MB of RAM recommended (minimum 128 supported; maximum 4 gigabytes [GB] supported)
Hard Disk	2 GB hard disk with a minimum of 1.0 GB free space. Additional free hard disk space is required if you are installing over a network.
CPU Support	Windows 2000 Server supports up to four CPUs on one computer.

Document Conventions

This guidance uses the style conventions and terminology shown in Table 0.3.

Table 0.3. Document Conventions

Element	Meaning
bold font	Characters that you type exactly as shown, including commands and switches. Programming elements, such as methods, functions, data types, and data structures appear in bold font (except when part of a code sample, in which case they appear in monospace font). User interface elements are also bold.
Italic font	Variables for which you supply a specific value. For example, *Filename.ext* could refer to any valid file name for the case in question. New terminology also appears in italic on first use.
`Monospace font`	Code samples.
%SystemRoot%	The folder in which Windows 2000 is installed.
Tip	Alerts you to supplementary information that is not essential to the completion of the task at hand.
Note	Alerts you to supplementary information.
Important	Alerts you to supplementary information that is essential to the completion of a task.
Caution	Alerts you to possible data loss, breaches of security, or other more serious problems.
Warning	Alerts you that failure to take or avoid a specific action might result in physical harm to you or to the hardware.

Acknowledgments

We wish to thank the following people for their valuable contribution to this series:

- Roger Toren
- Carl Rabeler
- Don Vilen
- Manoj Nayar
- Vaqar Pirzada
- Scott Gaskins
- Edward Lafferty
- John Gonzalez (HP)
- Mike Tolliver (HP)
- Kim Vance (SANcastle)

1

Implementing Failover Clustering

This chapter teaches you how to increase the availability of a Microsoft® SQL Server™ data center by using failover clustering. After reading this chapter, you will be able to configure each component in this high-availability solution, including the storage device, network, Microsoft Cluster Service (MSCS), Microsoft Distributed Transaction Coordinator (MS DTC), and Microsoft SQL Server 2000.

The implementation steps include annotation to help you understand the selected configuration choices. The configuration options presented in this chapter have been designed using input from the field, reviewed thoroughly by Microsoft Product Support Services (PSS), and tested by an independent team to ensure accuracy and functionality.

Failover clustering is only part of achieving a highly available data center. To achieve a highly available data center, you must also follow the processes outlined in the accompanying Planning Guide.

Understanding Failover Clustering

Failover clustering is a high-availability solution provided by SQL Server 2000 Enterprise Edition that uses the clustering services provided by either Microsoft Windows® 2000 Advanced Server or Microsoft Windows 2000 Datacenter Server.

Failover clustering is the best solution for simple, quick, and automatic failover. It is the only high-availability solution that provides absolute transactional currency and consistency and makes failover virtually transparent to the client. Consider using failover clustering as your first choice for high availability ahead of log shipping and transactional replication.

Failover clustering maintains at least one standby server in an MSCS cluster in case the primary data center server fails. Windows 2000 Advanced Server supports two-server clusters, and Windows 2000 Datacenter Server supports up to a four-server cluster. When the MSCS detects that the primary server has failed, it automatically starts the cluster resources on a standby server that were running on the failed server. MSCS then redirects all client traffic to the standby server. You can also manually failover to a standby server. With failover clustering, all committed transactions are always available through the standby server after the primary server fails.

Note: Failover clustering does not permit widely separated nodes (stretch clusters) or remote disk mirrors without hardware and software assistance provided by third-party hardware vendors.

To implement a failover clustering solution, you need to have certain things in place:

- You need people in the following roles:
  - Database administrator
  - Shared disk/network storage administrator
  - Network administrator
- You need the following software:
  - Windows 2000 Advanced Server or Windows 2000 Datacenter Server
  - SQL Server 2000 Enterprise Edition
- You need systems and system components that are certified to work together and with the chosen Windows 2000 operating system edition, and are certified for failover clustering. Certification of systems and system components for failover clustering is more stringent than operating system certification. In addition, Windows 2000 Datacenter Server has more stringent certification requirements than Windows 2000 Advanced Server. To find systems and system components that are certified by Microsoft for failover clustering with either operating system, search the "Hardware Compatibility List" on the Microsoft Web site at *http://www.microsoft.com/hcl/search.asp*.

For more detailed information about failover clustering, see "SQL Server 2000 Failover Clustering" on the Microsoft Web site at *http://www.microsoft.com/SQL/techinfo/administration/2000/failoverclustering.doc*.

Deploying Failover Clustering

You will be successful at implementing failover clustering if you avoid shortcuts and parallel installations. If the setup program fails or you cancel the setup program for any reason, completely remove the previous setup before attempting another, and remove any Domain Name System (DNS) entries relating to the failed setup. Also, verify that the service names and IP addresses you plan to use for MSCS and SQL Server are unused immediately before you install MSCS and SQL Server.

This chapter assumes the following with respect to each participating server in the failover clustering solution:

- Windows 2000 Advanced Server or Windows 2000 Datacenter Server is installed on a mirrored volume.
- The most recent operating system service pack is installed.

Note: Restoring disks to a cluster in the event of SAN failure requires at least Windows 2000 Service Pack 3.

- MSCS has not been installed.
- The server is a member of the same Windows domain as each other server.
- A SAN or disk array is connected.
- All disks are formatted for New Technology File System (NTFS) and are not configured as dynamic disks.
- At least two network interface cards (NICs) are installed.
- The server is connected to the corporate network and is connected to each server on a private network.
- SQL Server (any version) is not installed.

The logical design of the failover cluster built in this chapter is described below, followed by the steps required to build this high availability solution.

Understanding the Logical Design

Figure 1.1 on the next page shows the logical design of the failover cluster that is described in this chapter. Although a storage area network (SAN) is not required to implement a failover cluster solution, it is frequently used for shared storage.

Figure 1.1
Logical design diagram of a single instance cluster using a SAN

Each node in the Windows 2000 failover cluster runs Windows 2000 Advanced Server or Windows 2000 Datacenter Server. MSCS maintains a heartbeat between the nodes by using a dedicated network to ensure each node remains alive. One or more SQL Server virtual servers run within the Windows 2000 failover cluster. The active node for the virtual server exclusively accesses and maintains the data and the log files for that virtual server. The SAN is connected to each node using fibre. Within the SAN, the SAN software maintains mirrored copies of the data and log files as well as the temporary and backup files.

In a multiple instance cluster, each virtual SQL Server requires a separate group of disks in the SAN for the data, log, temporary, and backup files. Each virtual SQL Server requires exclusive control of its disks. MSCS assigns control to the active node for a virtual SQL Server.

The generalized steps required to implement this logical design are below, followed by the steps in detail.

► **To implement a failover cluster with SQL Server**

1. Create and configure domain user accounts, determine unique server and instance names, and obtain an allocation of corporate IP addresses.
2. Configure the network.

3. Configure the storage devices.
4. Install and configure MSCS.
5. Upgrade MS DTC.
6. Install and configure a SQL Server 2000 virtual server instance.
7. Apply the most recent SQL Server 2000 service pack.
8. Perform additional configuration tasks.

Configuring Accounts and Names and Obtaining IP Addresses

Before configuring a Windows 2000 failover cluster with SQL Server, you must create and/or configure domain user accounts for SQL Server, configure domain user account permissions for MSCS, obtain server names for MSCS and SQL Server, and allocate IP addresses for the heartbeat and the corporate network NICs. Use the following information to configure accounts and names, and obtain IP addresses.

- **Domain user accounts —** Create and configure the following domain user accounts:
  - **A domain user account to administer the failover cluster —** Make this account a member of the Administrators local group on each node of the failover cluster.
  - **A domain user account for the SQL Server and the SQL Server Agent services —** This account does not need to be a member of the Administrators local group in any node of the failover cluster.
- **Server names —** Have the IT department approve the following names:
  - **A virtual cluster name —** This name must be unique within the domain.
  - **A virtual SQL Server name for each SQL Server instance —** These names must be unique within the domain and on the local subnet if multiple domains reside together on the same subnet.
  - **An instance name for each SQL Server instance —** These names must be unique within the failover cluster.
- **New static IP addresses —** Have the IT department allocate static IP addresses for each of the following:
  - A virtual cluster IP address
  - A virtual IP address for each SQL Server instance
  - An IP address for each corporate network NIC

These addresses must be unique on the corporate network and must be on the same subnet.

Note: In a cluster, each SQL Server instance must have its own virtual IP address, virtual server name, and SQL Server instance name. This enables a SQL Server instance to fail over to a secondary node without affecting other SQL Server instances on the same physical node.

Configuring the Network

Before you install MSCS, you must assign the allocated IP addresses for each corporate network NIC, including the default gateway and preferred DNS server for each NIC. You must also assign static IP addresses to the NICs in each server that will carry the heartbeat and other internal cluster communications among the servers in the failover cluster.

The heartbeat NICs should be on their own network. This reduces the contention between cluster administration and user traffic. You can use any IP addresses you want because there will be no other NICs on this network.

► **To configure a NIC that connects to the corporate network**

1. On the desktop, right-click **My Network Places**, and then click **Properties**.
2. In **Network and Dial-up Connections**, right-click the network connection for the NIC that connects to the corporate network, and then click **Properties**.
3. In **Connection Properties** for this corporate NIC, click **Internet Protocol (TCP/IP)**, and then click **Properties**.
4. In **Internet Protocol (TCP/IP) Properties**, type the IP address allocated to this corporate NIC.

   Change the subnet mask only on the advice of a network specialist if your network uses subnetting.
5. Type the IP address of the default gateway.
6. Click **Use the Following DNS Server Addresses**, and then type the DNS server address.
7. Click **OK** to save the new TCP/IP properties.
8. Click **OK** to save the new connection properties for this NIC.
9. In **Network and Dial-up Connections**, rename the connection to **Corporate** to clearly identify its function.

Note: Repeat this process if you use a second corporate NIC that provides redundant connectivity to the corporate network. Name this redundant corporate connection **Corporate2**.

► **To configure the NIC that connects to the private network**

1. On the desktop, right-click **My Network Places**, and then click **Properties**.

2. In **Network and Dial-up Connections**, right-click the network connection for the NIC that connects to the private network, and then click **Properties**.
3. In **Connection Properties** for this private NIC, click **Internet Protocol (TCP/IP)**, and then click **Properties**.
4. In **Internet Protocol (TCP/IP) Properties**, type an IP address for this private NIC.

   You must use an IP address for each private NIC that is on the same subnet as the IP address you use for the other private NICs in the cluster. Use the automatically completed subnet mask. It is common to use an IP address of the form 10.1.0.*x* for the private segment of the network.
5. Click **OK** to save the new TCP/IP properties.
6. Click **OK** to save the new connection properties for this NIC.
7. In **Network and Dial-up Connections**, rename the connection to **Heartbeat** to clearly identify its function.

Note: Repeat this process if you use a second private NIC that provides a redundant heartbeat. Name this redundant private connection **Heartbeat 2**.

Configuring Storage Devices

Each vendor has different procedures for configuring its storage devices. Follow the procedures provided by the vendor and configure the SAN as follows:

- Use the SAN for SQL Server only, not for Windows 2000.
- Create a mirrored volume for the quorum.
- Create three mirrored volumes for each SQL Server instance — one each for data files, transaction log files, and temporary files.

Before configuring the storage devices for MSCS to use, turn off all but one of the nodes to avoid configuration errors.

Preparing the Drives

To prepare the storage device for use, you must write disk signatures, create partitions, and format drives. Initialize the drives used in a cluster as basic drives.

► **To initialize a drive for use in a cluster**

1. Right-click **My Computer**, and then click **Manage**.
2. Expand **Storage**, and then click **Disk Management**.
3. Right-click a drive that you want to initialize, and then click **Initialize**.
4. In **Initialize Disk**, select the disk(s) to initialize, and then click **OK**.

► **To create a partition and format a drive**

1. In **Computer Management**, right-click the drive that you want to partition, and then click **Create Partition.**
2. On the **Welcome to the Create Partition Wizard** page, click **Next**.
3. On the **Select Partition Type** page, click **Primary Partition**, and then click **Next**.
4. On the Specify Partition Size page, click **Next** to use all of the disk space on the drive for this partition.
5. On the **Assign Drive Letter or Path** page, assign a drive letter, and then click **Next.**
6. On the **Format Partition** page, click **Format Partition**.
7. In **Formatting**, set the formatting options as follows:
   a. In **File System to Use**, specify **NTFS**.
   b. In **Allocation Unit Size**, specify **Default**.
   c. In **Volume Label**, specify a meaningful name

   Do not select the **Enable File and Folder Encryption** check box.
8. Click **Next**.
9. On the **Completing the Create Partition Wizard** page, click **Finish**.

► **To format a partitioned but unformatted drive**

1. Right-click **My Computer**, and then click **Manage**.
2. Expand **Storage**, and then click **Disk Management**.
3. Right-click a drive that you want to format, and then click **Format.**
4. In **Format Disk**, click **OK**.

Note: Repeat these procedures for each drive.

Specifying Drive Letters

Specify drive letters for each drive in the cluster to ensure that they do not change dynamically based on the drives that are online. If a drive letter that SQL Server uses for a data or log file changes after SQL Server is installed, SQL Server cannot attach the database when SQL Server starts.

Note: If you created a new partition in the preceding procedures and assigned a drive letter, you do not need to complete the following procedure.

► **To specify drive letters**

1. Right-click **My Computer**, and then click **Manage**.
2. Expand **Storage**, and then click **Disk Management**.
3. Right-click a drive that the cluster uses, and then click **Change Drive Letter and Paths**.
4. In **Change Drive Letter and Paths**, click **Edit**.
5. In **Edit Drive Letter or Path**, assign a drive letter from the list of available drive letters, and then click **OK**.
6. Click **Yes** to confirm that you want to change the drive letter.
7. Repeat this process for each drive that the cluster uses.

When you add more nodes to the cluster, they inherit these drive letters from the cluster configuration information stored on the quorum drive.

Installing and Configuring MSCS

When you install and configure MSCS, set up only one node at a time. Turn off all other nodes that are connected to the shared storage device while you install MSCS on the first node. You risk disk configuration errors if you attempt to configure MSCS on the first node while other nodes are connected and running. After you install MSCS on the first node and the drives in the shared storage device are allocated, you can install MSCS on additional nodes without turning off nodes on which MSCS is running.

Note: If you are reinstalling MSCS or recovering from a failed MSCS installation, remove any entries in the DNS server that relate to the failed MSCS installation before you reinstall. Stray or duplicate IP addresses or names can prevent a clean installation.

Install MSCS on all nodes in the cluster. The first node of an MSCS installation requires more setup than an additional node in the same cluster.

► **To add MSCS on the first node**

1. On the first server on which you are installing MSCS, click **Start**, point to **Settings**, and then click **Control Panel**.
2. In **Control Panel**, double-click **Add/Remove Programs**.
3. In **Add/Remove Programs**, click **Add/Remove Windows Components**.
4. On the **Windows Components** page, select the **Cluster Service** check box, and then click **Next**.

   The **Internet Information Services** check box is automatically selected with some Internet Information Services components.

5. If Terminal Services is installed, click **Next** on the **Terminal Services Setup** page to run Terminal Services in remote administration mode.
6. When prompted, insert your Windows 2000 Advanced Server or Windows 2000 Datacenter Server CD, and then click **OK**.
7. After the clustering components install, click **Next** on the **Welcome to the Cluster Service Configuration Wizard** page.
8. On the **Hardware Configuration** page, click **I Understand** to acknowledge that only certified configurations listed in the Cluster category of the HCL are supported by MSCS; then click **Next**.

► **To define the cluster**

1. On the **Create or Join a Cluster** page, click **The First Server in the Cluster**, and then click **Next**.

   If the Cluster Service Configuration Wizard does not detect disks that meet the criteria for clustering, you cannot continue.
2. On the **Cluster Name** page, type the name for the new cluster (up to 15 characters), and then click **Next**. Use a name that is unique on the network and within the Windows domain.

► **To specify the setup program administrator account**

- On the **Select an Account** page, type the domain user account, password, and domain name in the appropriate text boxes, and then click **Next**.

  Use the domain user account you previously added to the Administrators local group on this node.

► **To define the cluster storage**

1. On the **Add or Remove Managed Disks** page, specify the disks that you want to belong to this cluster, and then click **Next**.

   By default, all disks that meet the criteria for clustering are selected.
2. Remove any disks that you do not want this cluster to use.

   Do not remove any disks that you plan to use for any SQL Server instance on this cluster.
3. On the **Cluster File Storage** page, click the drive belonging to the cluster that you want to use as the quorum drive.

   Select a mirrored drive with a minimum size of 500 MB. If the quorum drive fails, the cluster fails. Do not use this drive for SQL Server.

► **To define the networks**

In this procedure, you tell Cluster Services which NICs are to be used for internal cluster communication such as the heartbeat, and which NICs are to be used for client access. The **Network Connections** page is displayed once for each NIC.

1. On the **Network Connections** page, select the **Enable This Network for Cluster Use** check box if this NIC is used in the cluster. Select the **Internal cluster communications only (private network)** check box for each heartbeat NIC, or select the **Client access only (public network)** check box for each corporate NIC. Click **Next**.
2. On the **Cluster IP Address** page, type the IP address in the **IP Address** text box, and confirm that the subnet mask is correct for your configuration.

   If your network employs subnetting, change the subnet mask only on the advice of a network specialist.
3. Click the corporate network, and then click **Next**.

► **To finish the installation**

1. Click **Finish** to apply the settings for the cluster, and then start MSCS on this first node.
2. When the message informs you that MSCS has started, click **OK**.
3. Click **Finish** to close the Cluster Configuration Wizard.
4. Close **Add/Remove Program** and Control Panel.

Installing and Configuring MSCS on Additional Nodes in the Cluster

Installing MSCS on additional nodes in a cluster requires less setup than installing MSCS on the original node because additional nodes inherit many settings from the first node.

► **To install MSCS on an additional node in the cluster**

1. On each additional server on which you want to install MSCS, click **Start**, point to **Settings**, and then click **Control Panel**.
2. In **Control Panel**, double-click **Add/Remove Programs**.
3. In **Add/Remove Programs**, click **Add/Remove Windows Components**.
4. On the **Windows Components** page, select the **Cluster Server** check box, and then click **Next**.

   The **Internet Information Services** check box is automatically selected with some Internet Information Services components.

5. If Terminal Services is installed, click **Next** on the **Terminal Services Setup** page to run Terminal Services in remote administration mode.
6. When prompted, insert the Windows 2000 Advanced Server or Windows 2000 Datacenter Server CD, and then click **OK**.
7. After the clustering components install, click **Next** on the **Welcome to the Cluster Service Configuration Wizard** page.
8. On the **Hardware Configuration** page, click **I Understand** to acknowledge that only certified configurations listed in the Cluster category of the HCL are supported by MSCS; then click **Next**.

► **To join an existing cluster**

1. On the **Create or Join a Cluster** page, click **The Second or Next Node in the Cluster**, and then click **Next**.

   If the Cluster Service Configuration Wizard does not detect disks that meet the criteria for clustering, you cannot continue. Disks must be formatted for NTFS and must not be configured as dynamic disks.
2. On the **Cluster Name** page, type the name of the existing cluster that you want this node to join. Leave the **Connect to cluster as** checkbox cleared, and then click **Next**.

   The Cluster Service Configuration Wizard verifies that a heartbeat is detected from the first node and then configures each NIC on the node as either a corporate NIC or a heartbeat NIC.

► **To specify the setup program administrator account**

- On the **Select an Account** page, type the domain user account, password, and domain name in the appropriate text boxes, and then click **Next**.

  Use the domain user account you created earlier that is an administrator in each node.

► **To finish the installation**

1. Click **Finish** to apply the settings for the cluster, and then start MSCS on this additional node.
2. When the message box informs you that MSCS has started, click **OK**.
3. Click **Finish** to close the Cluster Configuration Wizard.
4. **Close Add/Remove Program**.
5. Close the Control Panel.

Upgrading the MS DTC

You must upgrade MS DTC to work in a cluster before installing SQL Server in the cluster. Distributed queries, two-phase commits, and certain types of replication use DTC. Even if you do not currently use these features, upgrade DTC because you may need to use them in the future. Upgrading now prevents the need to fix errors later.

► **To upgrade DTC**

1. At one node in the cluster, open a command prompt.
2. At the command prompt, type **comclust**, and then click **Enter**.
3. Close the command prompt.

Note: Repeat this procedure on all other nodes participating in the SQL Server virtual server.

Installing and Configuring SQL Server on the Cluster

When you install SQL Server on a cluster, the setup program creates a virtual server. A virtual server consists of a virtual name, an IP address, and a port. Clients connect to these virtual resources, and MSCS points to the actual resources on the node in the cluster that is currently active. The SQL Server setup program installs SQL Server program files on each node and SQL Server system databases on a cluster resource disk. To successfully install a virtual server, you must properly configure and start MSCS.

Note: If you are reinstalling SQL Server or recovering from a failed SQL Server installation, remove any entries in the DNS server that relate to the failed MSCS installation before you reinstall. Stray or duplicate IP addresses or names can prevent a clean installation.

Creating a Single Cluster Group

Before you install SQL Server, place the disk resources for the SQL Server instance into a single cluster group. The group should contain a mirrored volume for the data files, a volume for the log files, and a volume for the backup and temporary files. The setup program adds additional resources to this cluster group when the virtual server installs.

Note: Create a separate cluster group for each SQL Server instance you plan to install.

► **To create a single cluster group**

1. Click **Start**, point to **Administrative Tools**, and then click **Cluster Administrator**.
2. In Cluster Administrator, expand **Groups** and locate the disk groups containing the disks that you plan to use for this SQL Server instance.

   Initially, each disk (or set of mirrored disks) belonging to the cluster is in a numbered disk group, such as Disk Group 1, Disk Group 2, and so on. The quorum drive is in the Cluster Group.
3. Right-click one of the disk groups, and then click **Rename**.
4. Type a descriptive name for this resource group (such as **VirtualServer1Group**).
5. Click the **Resource** node to display the additional drives owned by the cluster.
6. Right-click a drive for SQL Server, point to **Change Group**, and then click the newly renamed cluster group.
7. Close the Cluster Administrator.

Note: Repeat this procedure for each additional drive that will be used by this SQL Server instance.

Installing SQL Server

After creating the cluster group, you can begin installing SQL Server. Use the server name and IP address allocated by your IT department when you create the virtual server. Follow these procedures for each instance you install. Before you begin, you may want to create a folder on each clustered drive to easily identify all files belonging to SQL Server.

Note: You must install SQL Server from the server that owns the shared disks. To change the server that owns the shared disks, use Cluster Administrator.

► **To start the installation and define the virtual server name**

1. Insert the SQL Server 2000 Enterprise Edition CD into the CD-ROM drive.
2. On the **SQL Server 2000 Enterprise Edition** page, click **SQL Server Components**, and then click **Install Database Server**.
3. On the **Welcome** page, click **Next**.
4. On the **Computer Name** page, type a virtual name, click **Virtual Server**, and then click **Next**.
5. On the **Product Key** page, enter the product key for SQL Server 2000 Enterprise Edition, and then click **Next**.
6. On the **Software License Agreement** page, click **Yes** to accept the terms of the License Agreement.

7. On the **Virtual Server Information** page, enter the IP address allocated for the virtual server, click the corporate network, click **Add**, and then click **Next**.

► **To define disk resources**

1. On the **Select the Cluster Disk Where the Data Files Will Be Placed** page, select the cluster group (or a drive within that cluster group) for this SQL Server.

   This step defines the cluster group that will contain the resources for the virtual SQL Server. The actual drive locations for executables and system databases are selected later. Do not select the cluster group containing the quorum drive.

2. On the **Cluster Definition** page, specify the nodes supporting the virtual server, and then click **Next**.

   By default, all nodes in the cluster are selected. Remove any nodes that you do not want to support SQL Server.

3. On the **Remote Information** page, type the domain user administrator account, password, and domain name in the appropriate text boxes.

   The setup program uses this user account to install the program files on each node. Use the administrator account you configured to administer the failover cluster. For more information, see "Configuring Accounts and Names and Obtaining IP Addresses" earlier in this chapter.

4. On the **Instance Name** page, click **Next** to install a default instance. To install a named instance, clear the **Default Instance** check box, type the name for the named instance, and then click **Next**.
5. On the **Setup Type** page, click the appropriate setup type for your installation.
6. Specify the location for the program files on each node.

   The default location is the system volume. Change this default if appropriate, and ensure that sufficient space exists for these files. The setup program installs the program files into this location on each node.

7. Specify the location for the data files on the cluster.

   The data files must be on one of the cluster disks and should not be on the quorum disk. The setup program installs the system databases on this drive.

   If you have created a folder for this SQL Server instance on the data drive, browse to this folder, and then click **Next**.

► **To complete the installation**

1. On the **Services Accounts** page, type the domain user account, password, and domain name in the appropriate text boxes, and then click **Next**.

   The SQL Server and SQL Server Agent services use this account. Use the domain user account you configured for the SQL Server and SQL Server Agent services. For more information, see "Configuring Accounts and Names and Obtaining IP Addresses" earlier in this chapter.

2. On the **Authentication Mode** page, click the authentication mode for SQL Server, and then click **Next**.

   Windows Authentication Mode is more secure than Mixed Mode. If you select **Mixed Mode**, you must provide a password for the sa account or agree to use a blank password. Do not use a blank password because it is highly insecure, even in a development environment, and can lead to poor coding practices.
3. On the **Licensing Mode** page, specify the appropriate licensing mode for your environment, and then click **Next**.
4. Click **Finish** to install SQL Server. When installation is complete, restart all nodes in the cluster.

Adding Cluster Disks as SQL Server Dependencies

To ensure that the failover process operates properly, manually add each disk on which SQL Server depends to the list of SQL Server dependencies. If a disk is not included in the SQL Server dependencies, SQL Server cannot use it after a failover. Add the mirrored volumes used for the transaction log files, backup files, and additional data files. This procedure requires all disk resources to be members of a single cluster group.

► **To add additional cluster disks as dependencies**

1. Click **Start**, point to **Programs**, and then click **Cluster Administrator**.
2. In Cluster Administrator, click the cluster group containing the SQL Server virtual server.
3. Right-click **SQL Server**, and then click **Take Offline**.

   SQL Server must be offline to change SQL Server dependencies.
4. Right-click **SQL Server**, and then click **Properties**.
5. On the **Dependencies** tab in **SQL Server Properties**, review the default resource dependencies.

   The **Network Name** resource and at least one disk resource appear.
6. Click **Modify**.
7. In the **Available Resources** list, double-click each additional disk resource on which SQL Server depends, and then click **OK**.
8. Click **OK** to close SQL Server Properties.
9. Right-click the group containing the SQL Server resources, and then click **Bring Online**.

Applying the Most Recent SQL Server Service Pack

Before using SQL Server, install the most recent SQL Server service pack.

► **To install the most recent SQL Server service pack**

1. Log on to one of the nodes in the cluster by using a domain account that is a member of the Administrators local group on that node.
2. Insert the CD containing the service pack into the CD-ROM drive.

   If it does not start automatically, navigate to the **\X86\Setup** directory, and then double-click **setupsql.exe**.
3. On the **Welcome** page, click **Next**.
4. On the **SQL Server Name** page, type the name of the SQL Server virtual server, and then click **Next**.
5. On the **Authentication Mode** page, click **Next** to connect to SQL Server using Windows authentication.
6. On the **Domain Administrator** page, type a domain administrator name, password, and domain name, and then click **Next**.

   The setup program must use a domain administrator account to install the service pack on all nodes of the virtual server.
7. After the service pack installs, click **OK**.
8. Click **Yes**, and then click **Finish** to restart the node.
9. Manually restart each additional node on which the service pack has been installed.
10. Back up the **master** and **msdb** databases.

    They have been modified by the service pack installation.

Note: This procedure is for SQL Service 2000 Service Pack 2. Check the ReadMe file on the CD for later service pack procedures.

Performing Additional Configuration Tasks

After you have installed the virtual server, review these additional configuration options and apply the settings that are appropriate for your environment. These options include the following:

- Set memory allocation between virtual servers
- Set new database properties
- Set the TCP/IP port that SQL Server uses
- Set advanced failover options
- Set SQL Server properties in the cluster

Use the following guidelines when performing these tasks.

Allocating Memory

If you are running multiple instances of SQL Server on separate nodes in the cluster, you must determine the total amount of physical memory required for each server and determine the appropriate allocation of memory for each SQL Server instance. Performance of SQL Server is directly related to the amount of physical memory allocated to it by the operating system. SQL Server provides two modes for memory management: dynamic and fixed. Dynamic memory allocation lets SQL acquire and release memory from the operating system as needed. Dynamic memory allows you to set a maximum amount of physical memory that the operating system can allocate to SQL Server. This allows the remaining memory to be reserved for other purposes, such as the operating system and other SQL Server instances. Fixed memory allows you to set a specific amount of physical memory that SQL Server can use. This amount neither grows nor shrinks.

Configure each server with the same amount of physical memory, and then allocate memory to each SQL Server instance to ensure that sufficient memory is available to support the failover of a SQL Server instance from another node. If a server does not use a lot of memory compared to the available memory on the server, specify dynamic memory and set the maximum amount of memory that the operating system can allocate. By doing so, you leave sufficient memory for the failover to occur quickly and successfully. If an instance uses all of the available physical memory, failover will occur more slowly and can fail if memory is not released quickly enough.

To determine the total amount of memory to install on each server, test your applications by fixing the maximum amount of memory available to SQL Server, and then evaluate the performance. Install enough memory on each server to enable all instances to operate on that server and meet the minimum service level for each instance. Using dynamic memory allows the SQL Server instance on each node to use most of the memory on that node. If you use fixed memory, the additional memory will only be used during a failover.

In a system under continuous heavy load, install sufficient memory to ensure that each SQL Server instance can meet its required minimum allocation. Use a fixed memory allocation to ensure that the SQL Server instance does not expand into memory reserved for the failover of other instances. To determine the appropriate setting, initiate a failover and test the performance level of each virtual server. If the operating system is unstable after a failover, decrease the memory allocated to each instance to increase the memory available to the operating system.

To specify a fixed amount of memory for each instance, use SQL Server Enterprise Manager or the **sp_configure** system stored procedure. When determining the amount of memory to allocate to SQL Server, remember to allow sufficient memory for the operating system. With large databases, allow 2 gigabytes (GB) of memory for the operating system.

If the amount of memory installed on each node is greater than 4 GB, add the **/PAE** switch to the Boot.ini file, and enable Address Windowing Extensions (AWE) in SQL Server. For more information, see article Q268363, "Intel Physical Addressing Extensions (PAE) in Windows 2000," in the Microsoft Knowledge Base at *http://support.microsoft.com/default.aspx?scid=kb;[LN];Q268363* and article Q274750, "Configure Memory for More Than 2 GB in SQL Server," in the Microsoft Knowledge Base at *http://support.microsoft.com/default.aspx?scid=kb;en-us;Q274750*.

Setting New Database Properties

You can specify the database location for all new databases on the cluster disks you created for the data and log files.

► **To set new database properties**

1. On one of the nodes, click **Start**, point to **Programs**, point to **Microsoft SQL Server**, and then click **Enterprise Manager**.
2. Expand **Microsoft SQL Servers**, and then expand **SQL Server Group**.
3. Right-click the SQL Server instance, and then click **Properties**.
4. On the **Database Settings** tab, in the **Default Data Directory** text box, type or browse to the drive and folder you want for the data files.
5. In the **Default Log Directory** text box, type or browse to the drive and folder you want for the log files, and then click **OK**.
6. Close SQL Server Enterprise Manager.

Setting the TCP/IP Port SQL Server Uses

The default instance of SQL Server usually listens on port 1433. If you are configuring a multiple instance cluster, you must use a different port for each additional instance. Although SQL Server can set this port dynamically at startup, you might want to set each named instance to listen on a specific, unused port. Check with a network administrator for a recommendation. Specify a static port if you have to open ports on a firewall and do not want the SQL Server ports to change.

► **To specify the TCP/IP port for an instance**

1. Click **Start**, point to **Programs**, point to **Microsoft SQL Server**, and then click **Server Network Utility**.
2. In **SQL Server Network Utility**, click the instance of SQL Server for which you want to set the port.
3. In the **Enabled Protocols** list, click **TCP/IP**, and then click **Properties**.
4. In the **Default Port** text box, type the appropriate port, and then click **OK**.
5. Click **OK** to close the **SQL Server Network Utility** dialog box.
6. Repeat these steps for each node participating in the SQL Server virtual server.

Setting Advanced Failover Options

Advanced failover options determine the failover behavior of SQL Server in the MSCS cluster.

► **To specify failover behavior**

1. Click **Start**, point to **Programs**, and then click **Cluster Administrator**.
2. In Cluster Administrator, click the cluster group containing SQL Server.
3. Right-click **SQL Server**, and then click **Properties**.
4. On the **Advanced** tab, use the following information to configure the advanced cluster failover properties. The defaults should not be changed without careful consideration and testing:
   - **Do Not Restart/Restart —** Click **Restart** to fail over to a secondary node if a failure is detected. This is the default.
   - **Affect the Group —** Select this check box to have the failover of SQL Server cause a failover of all resources in the cluster group.
   - **Threshold —** Type the number of attempts MSCS should make to restart SQL Server before failing over to a secondary node.
   - **Period —** Type the seconds between retry attempts. This value is ignored if the threshold is set to 0.
   - **Looks Alive Poll Interval —** Click **Use Value from Resource Type** to use the SQL Server default of 5,000 milliseconds (5 seconds).
   - **Is Alive Poll Interval —** Click **Use Value from Resource Type** to use the SQL Server default of 60,000 milliseconds (60 seconds).
   - **Pending Timeout —** Specify the time SQL Server has in either the Offline Pending or the Online Pending state before MSCS places the resource in Offline or Failed status. The default is 180 seconds.
5. When you are finished, click **OK** to close SQL Server Properties in the Cluster Administrator.

Configuring Service Properties in a Cluster

In a cluster environment, you must use the correct management tool to configure SQL Server services, as follows:

- Use Cluster Administrator to start and stop the services.
- Use the Properties dialog box for the SQL Server instance in Enterprise Manager to set the service accounts and passwords for the services.

Failing to use the appropriate tools to configure SQL Server properties can cause SQL Server startup problems. In addition, in a cluster, do not set SQL Server Agent to restart automatically. If SQL Server Agent is set to restart automatically, it can conflict with the cluster services during a failover.

Failing Over to a Secondary Node

Failover is automatic when using MSCS and SQL Server. You can control how long MSCS takes to detect a failure, and you can take steps in your applications to reduce the time to bring up the standby service. Before making any changes to MSCS or to a client application, test the failover cluster to become familiar with how it fails over under different conditions.

Testing Failover

Use Cluster Administrator to initiate a planned failover. This simulates taking a node offline for planned maintenance.

► **To initiate a planned failover**

1. Click **Start**, point to **Programs**, and then click **Cluster Administrator**.
2. Click the cluster group containing the SQL Server virtual server.

   Notice the cluster that owns each resource in this cluster group.
3. Right-click the cluster group containing the SQL Server virtual server, and then click **Move Group**.

Watch each resource in the group switch from online to offline to online pending and then to online in the new group. The entire process takes 1 to 2 minutes for a typical environment. Also notice that the resource owner in this cluster group changes to the secondary cluster.

Use one of the following options to simulate an unplanned failover.

► **To simulate an unplanned failover**

1. Do one of the following:
   - Unplug the corporate network cable from the node that owns SQL Server.
   - Reboot the owning node.
   - Unplug the fibre connected to the SAN (if a SAN is used).
   - Turn off the power on the owning node.
2. Use Cluster Administrator on the secondary node to watch each resource in the group switch from online to offline to online pending and then to online.

   The entire process takes less than 1 minute. Notice that the resource owner in this cluster group changes to the secondary cluster.

Minimizing Failover Time

If the time it takes SQL Server to fail over from one node to another needs to be decreased, use the following information to determine your options to reduce time to availability.

Understanding Time to Availability

The time it takes from the occurrence of a fault to SQL Server being available on the standby node consists of the following:

- The time to detect and confirm the fault, which you can control
- The time to take the resources offline, which is controlled by MSCS and which you cannot control
- The time to bring the resources online, which MSCS controls and which you can control by ensuring that sufficient SQL Server memory and processor resources are allocated
- The time for SQL Server to initialize, which you can control by limiting long transactions

SQL Server is usually available within 1 minute or less.

Controlling the Time Required to Detect and Confirm the Fault

The **LooksAlive** and **IsAlive** poll intervals for SQL Server determine the time that it takes MSCS to detect a failure. Smaller intervals create more overhead and increase the possibility of false failure detections, and therefore result in unnecessary failovers. The default for the SQL Server **IsAlive** interval is 60,000 milliseconds (60 seconds), and **LooksAlive** is 5,000 milliseconds (5 seconds). Change these values on the test platform first, and test them thoroughly under a realistic load. If you set the interval lower, be prepared to measure the impact on the application.

Controlling the Time Required to Bring the Resources Online

If the failover node has insufficient available memory at the time of failover, it will take longer to bring SQL Server online after a failover. Although this condition should not occur in a single-instance cluster, this can be a problem in a multiple-instance cluster. In normal operation, each node runs one instance of SQL Server, and each instance uses memory as needed. If a node does not have sufficient memory to run multiple instances simultaneously during a failover, use a lower maximum memory setting to reserve memory for failover.

If at least 30 MB of unallocated physical memory is available, the instance starts in dynamic memory allocation mode. For monitoring purposes, a low-memory condition creates an entry in the SQL Server event log warning you that the operating system did not fulfill the SQL Server memory request. No entry is made in the Application Log. This does not mean that you should attempt to operate SQL Server with only 30 MB of available physical memory.

Controlling the Time Required to Initialize

At startup, SQL Server must verify the condition of each database, rolling forward completed transactions and rolling back any uncommitted transactions that it finds. If client applications use very long transactions, the incomplete transactions can take a long time to roll back. Use a series of short transactions whenever possible. Determine each application's worst-case transaction size, and test the recovery time when this transaction is rolled back in a failover.

Reconnecting Clients

When the SQL Server service activates on the failover node, the application must reconnect to continue operating. The manner in which clients reconnect depends on whether an application maintains state or not and on the ability of the application to reissue the steps in a transaction. Clients reconnect in the following ways:

- In a Web application, application state is usually not maintained. When a user submits a request to the Web server, the application usually makes a new connection. In this environment, no additional retry logic is required. However, the Web application needs to gracefully handle the error response when the application cannot connect to SQL Server, however.
- In a non-Web application, a connection is often held for a whole session. You must implement logic in the application that detects a broken connection and that attempts to reconnect at appropriate intervals (for example, 30 seconds). The application should be prepared to restart the transaction that was in progress when the connection broke. In some cases, you may have to rely on the application user to retry the submission.

Failing Back to the Original Node

In a single-instance cluster, little reason exists to fail back to the original node. If you have symmetrical servers that are dedicated to the same application, performance is identical on each. Failing back only causes more downtime.

In a multiple-instance configuration, two instances of SQL Server run on the same node after a failover, so performance decreases for both. Choose a failback time to minimize the impact on clients.

You can set MSCS to automatically fail back to the original node; however, doing so incurs another service outage when the service is moved to the other node. In addition, if the failed node is not ready for the service to fail back, additional downtime occurs. Therefore, enabling automatic failback is not necessarily effective in maintaining high availability.

Automatic failback can be set to occur at a time when an additional outage is acceptable.

► **To configure automatic failback in a specific time slot**

1. Click **Start**, point to **Programs**, and then click **Cluster Administrator**.
2. In **Cluster Administrator**, right-click the group containing SQL Server.
3. Choose **Properties**.
4. On the **Failback** tab, set the time of day during which you want to allow failback.

Manual failback enables you to control when to stop the service and move it back to the original node when doing so is least disruptive to your clients.

► **To initiate manual failback**

1. Click **Start**, point to **Programs**, and then click **Cluster Administrator**.
2. In **Cluster Administrator**, right-click the group containing SQL Server.
3. Choose **Move Group**.

Restoring a Node After a Catastrophic Failure

If one of the cluster nodes suffers a catastrophic failure, you might have to rebuild the server. Use the steps in the following procedures to recover from a catastrophic failure.

► **To rebuild the server**

1. On the surviving node, remove the failed node from the virtual server. If you have more than one virtual server on a node, doing so updates all instances.
2. On the surviving node, evict the failed cluster node.
3. Rebuild the failed node.
4. Install SQL Server.

► **To remove a node from a virtual server**

1. Insert the SQL Server 2000 Enterprise Edition CD into the CD-ROM drive.
2. On the **SQL Server 2000 Enterprise Edition** page, click **SQL Server Components**, and then click **Install Database Server**.
3. On the **Welcome** page for the Microsoft SQL Server Installation Wizard, click **Next**.
4. On the **Computer Name** page, type the virtual server name, click **Virtual Server**, and then click **Next**.
5. Leaving **Advanced Options** selected, click **Next**.
6. Leaving **Maintain a Virtual Server for Failover Clustering** selected, click **Next**.

Note: If the server is unavailable, you will receive the following error message: "One or more nodes of Virtual Server are unavailable. Setup cannot modify the virtual server's IP address resource."

7. On this error page, click **OK**.
8. On the **Virtual Server Information** page, click **Next**.
9. On the **Cluster Management** page, click the failed node, and then click **Remove**. Click **Next**.
10. On the **Remote Information** page, type the login name and password of an account that is an administrator for all nodes in the cluster. Click **Next**.
11. In the message box, click **Yes**.
12. Click **Finish.**

► **To evict a node from a cluster**

1. Click **Start**, point to **Programs**, and then click **Cluster Administrator**.
2. In **Cluster Administrator**, right-click the failed node, and then click **Evict Node**. Click **OK**.
3. Close Cluster Administrator.

► **To rebuild a node**

1. Reinstall Windows, join the server to the domain, and apply the most recent Windows service pack on the new hardware.
2. Configure the domain user account to administer the cluster. For more information, see "Configuring Accounts and Names and Obtaining IP Addresses" earlier in this chapter.
3. Configure the corporate and private NICs.
4. Install and configure MSCS as an additional node. For more information, see "Installing and Configuring MSCS on Additional Nodes in the Cluster" earlier in this chapter.

► **To add the rebuilt node to the virtual server**

1. Insert the SQL Server 2000 Enterprise Edition CD into the CD-ROM drive of any node in the cluster.

   You can add a rebuilt node to the virtual server from any node in cluster.
2. On the **SQL Server 2000 Enterprise Edition** page, click **SQL Server Components**, and then click **Install Database Server**.
3. On the **Welcome** page for the Microsoft SQL Server Installation Wizard, click **Next**.
4. On the **Computer Name** page, type the virtual server name, click **Virtual Server**, and then click **Next**.

5. On the **Installation Selection** page, click **Advanced Options**, and then click **Next**.
6. Leaving **Maintain a Virtual Server for Failover Clustering** selected, click **Next**.
7. On the **Virtual Server Information** page, click **Next**.
8. On the **Cluster Management** page, click the new node, click **Add**, and then click **Next**.
9. In the **Username** text box, specify a domain user account that is an administrator on each node participating in the virtual server configuration. Type the password in the **Password** text box, type the domain name in the **Domain** text box, and then click **Next**.
10. On the **Services Accounts** page, type the domain user account, password, and domain name in the appropriate text boxes, and then click **Next** .
11. Click **Finish** when setup completes. Repeat this process for each instance.

Note: After you install SQL Server 2000 and join it to the virtual SQL Server, apply the appropriate SQL Server service pack to the rebuilt node. All nodes run at the same service pack level.

References

For more information about topics covered in this chapter, refer to the following:

- Article Q254321, "Clustered SQL Server Do's, Don'ts, and Basic Warnings," in the Microsoft Knowledge Base at *http://support.microsoft.com /default.aspx?scid=kb;en-us;Q254321*
- Article Q259267, "Microsoft Cluster Service Installation Resources," in the Microsoft Knowledge Base at *http://support.microsoft.com /default.aspx?scid=kb;en-us;Q259267*
- Article Q239885, "How to Change Service Accounts on a SQL Virtual Server," in the Microsoft Knowledge Base at *http://support.microsoft.com /default.aspx?scid=kb;en-us;Q239885*
- Article Q273673, "SQL Virtual Server Client Connections Must Be Controlled by Clients," in the Microsoft Knowledge Base at *http://support.microsoft.com /default.aspx?scid=kb;en-us;Q273673*
- "Microsoft Solutions for Systems Architecture: Internet Data Center" on the Microsoft TechNet Web site at *http://www.microsoft.com/technet /treeview/default.asp?url=/technet/itsolutions/idc/pag/pag.asp*

2

Implementing Log Shipping

This chapter teaches you how to increase the availability of a Microsoft® SQL Server™ data center by using built-in log shipping. After reading this chapter, you will be able to configure each component in this high-availability solution. For information about configuring log shipping using Network Load Balancing (NLB), see Solution Guide Chapter 4, "Implementing Network Load Balancing."

The implementation steps include annotation to help you understand the selected configuration choices. The configuration options presented in this chapter have been designed using input from the field, reviewed thoroughly by Microsoft Product Support Services (PSS), and tested by an independent team to ensure accuracy and functionality.

Log shipping is only part of achieving a highly available data center. To achieve a highly available data center, you must also follow the processes outlined in the accompanying Planning Guide.

Understanding Log Shipping

Built-in log shipping is a high-availability solution provided with SQL Server 2000 Enterprise Edition that can be implemented with any server edition of Microsoft Windows® 2000. If you want to use the client redirection capabilities of NLB, however, you must use either Windows 2000 Advanced Server or Windows 2000 Datacenter Server.

Log shipping is an inexpensive, high-availability solution that uses SQL Server jobs to periodically back up the production database transaction log on the primary server, and then restore these backups to a copy of the production database (the standby database) on one or more secondary servers (standby servers). With log

shipping, proximity is not a limitation; you can place a standby server in a geographically remote location to protect against a catastrophic disaster. Standby servers do not need to be identical to the primary server, although differences in the amount of memory, the number and speed of the processors, and the robustness of the storage system can affect SQL Server performance after a role change.

Built-in log shipping was designed as a high-availability solution. It includes a role change mechanism to promote a standby server to assume the role of the primary server and demote the original server to become a standby server. Built-in log shipping also includes support for a third server (the monitoring server) to monitor log shipping and to notify the database administrator (DBA) team if log shipping fails to synchronize the standby database according to the automated schedule.

You can use the standby database on a standby server in place of the production database if the primary server fails or must be taken offline for planned maintenance. The standby database is always slightly out of synchronization with the production database. You can control the level of synchronicity—it can be as short as several minutes if a high-speed network is used, or as long as you choose. To minimize the transactional latency of the standby database, back up the transaction log on the primary server frequently, and restore it to the standby database with the same frequency.

Because log shipping relies on the SQL Server restore process, user connections to the standby database must terminate each time the transaction log is restored. As a result, if you also want to use a standby database for reporting or maintenance tasks, you must restore the transaction logs less frequently. Restoring the transaction logs less frequently sacrifices the currency of the standby database and increases the time required to bring the standby server online in case the primary server fails or is taken offline. Using multiple standby servers permits you to maintain a standby database with minimum transactional latency and another standby database with greater latency for reporting or maintenance tasks.

Changing the role of a standby server is a manual process that requires the DBA to perform a number of tasks to complete the role change, including the manual redirection of client requests to the promoted standby server. To point clients to the new primary server, use one of the following client redirection options:

- Program an alternative server list into the client application
- Update the client Data Source Name (DSN)
- Update the Domain Name System (DNS)
- Update the COM+ component
- Update the NLB configuration.

If the primary server fails, any transactions that are not backed up and copied to the standby server are lost. During a planned role change, if the production database and the standby database are completely synchronized, no transactions are lost. Because changing server roles is a manual process, changing roles takes longer than changing nodes with failover clustering. Changing roles, however, is significantly faster than restoring a database to a secondary server from a cold backup.

To implement log shipping, you need to have certain things in place:

- You need people in the following roles:
  - Database administrator
  - Network administrator
- You need the following software:
  - Windows 2000 Server, Windows 2000 Advanced Server, or Windows 2000 Datacenter Server
  - SQL Server 2000 Enterprise Edition
- You need systems and system components that are certified to work together and with the chosen edition of Windows 2000 operating system. Windows 2000 Datacenter Server has more stringent certification requirements than Windows 2000 Advanced Server. To find systems and system components that are certified by Microsoft, search the "Hardware Compatibility List" on the Microsoft Web site at *http://www.microsoft.com/hcl/search.asp*.

For more detailed information about log shipping, including setup and troubleshooting, see "How to Set Up Log Shipping" on the Microsoft Support Services Web site at *http://support.microsoft.com/default.aspx?scid=http://support.microsoft.com/support/sql/content/2000papers/LogShippingFinal.asp*.

For information about connectivity between geographically dispersed sites, see "Publishing Data Over the Internet Using VPN" in SQL Server Books Online, and "Virtual private network (VPN) connections overview" on the MSDN Web site at *http://www.microsoft.com/windowsxp/home/using/productdoc/en/default.asp?url=/WINDOWSXP/home/using/productdoc/en/Conn_VPN.asp?frame=true*.

Deploying Automatic Log Shipping

Implementing log shipping is wizard-driven and easy to follow. You cannot install log shipping by using Transact-SQL scripts, and you cannot implement log shipping more than once for a single database.

This chapter assumes the following with respect to each participating server in the log-shipping solution:

- Windows 2000 Server, Windows 2000 Advanced Server, or Windows 2000 Datacenter Server is installed.
- The most recent operating system service pack is installed.
- The server is a member of the same Microsoft Active Directory® service forest as each other server.
- At least one network interface card (NIC) is installed.
- The server is connected to the corporate network.
- SQL Server 2000 Enterprise Edition is installed and configured to use the same authentication mode as each other server.
- The most recent SQL Server 2000 service pack is installed.
- The SQL Server Agent service is configured to use the same domain user account used by the SQL Server Agent service on each other server.
- The SQL Server Agent service is configured to start automatically.
- The server is registered in SQL Server Enterprise Manager.
- If e-mail notifications will be used, SQLAgentMail is configured.

This chapter also assumes that the production database on the primary server is configured to use either the bulk-logged recovery model or the full recovery model. The logical design of the log-shipping solution built in this chapter is described next, followed by the specific steps required to build this high-availability solution.

Understanding the Logical Design

Figure 2.1 shows the logical design of a log-shipping solution. This design illustrates the use of three standby servers. These servers can be geographically distributed, and the data restoration schedules can be different for each standby database.

Figure 2.1
Logical design diagram of a log-shipping solution

After you set up the production database on the primary server, create a log-shipping database maintenance plan for the production database on the primary server. During log-shipping setup, restore a backup of the production database to each standby server. Log shipping setup creates a log-shipping job on the primary server that periodically backs up the production database transaction log. Log shipping setup also creates log shipping jobs on each standby server that copy the transaction log backups from the primary server and restore them to the standby database on the standby server. During restoration, specify either the **STANDBY** or the **NORECOVERY** option. Performing a restoration with either of these options

permits additional transaction logs to be applied. The **STANDBY** option places the database in read-only mode. Finally, specify a monitoring server to monitor these log-shipping jobs and to notify the DBA if the log-shipping jobs fail to keep the standby database synchronized according to schedule. For example, if you configure log shipping to keep each standby database current to within 15 minutes of the production database, you can configure the monitoring server to notify you if a standby database is out of synchronization for more than 45 minutes.

Although no special networking is required, you must ensure that the bandwidth between the primary server and each standby server can handle the network traffic generated by copying the transaction log backup files. The copy of a transaction log backup must complete before the next one is ready to be copied. The traffic depends on the number and size of the transactions generated in the production database. If additional network performance is required, use a private network segment between the primary server and each standby server. On the primary server, ensure that the disks are fault tolerant to minimize the need for a failover because of a disk failure. On each standby server, use fault tolerant disks to ensure that the standby server remains available for a role change and to function as a replacement primary server if necessary.

The steps required to implement this logical design are below, followed by the steps in detail.

► **To implement log shipping with SQL Server**

1. Log on or connect to the primary server with a user account that is a member of the Administrators local group on that computer.
2. Create and share a synchronization folder on the primary server and on each standby server.
3. Create database folders on each standby server.
4. Create log-shipping jobs, and initialize each standby database.
5. Create and execute a login synchronization job.
6. Script and copy jobs, alerts, and operators.
7. Copy Data Transformation Services (DTS) packages.

These steps create a database maintenance plan with log-shipping jobs to log ship a single production database from the primary server to one or more standby servers. To log ship a second production database from the primary server, create a second database maintenance plan.

Note: You can use a standby server for multiple standby databases from multiple servers.

Creating and Sharing Synchronization Folders

On the primary server and on each standby server, create and share a synchronization folder. The synchronization folder on the primary server is used to store transaction log backup files, login information output by the bulk copy program (BCP) utility, the jobs script, the alerts script, and the operators script that you use to synchronize objects between the primary server and each standby server. The synchronization folder on each standby server is used to store transaction log backup files copied from the primary server and login information output by the BCP utility. This synchronization folder also stores the transaction log backup files created on a standby server after a role change. The SQL Server Agent service must have permission to read and write to these synchronization folders. On the primary server and each standby server, use a drive that is not used by the SQL Server data or log files, and mirror the drive for fault tolerance. Be sure only authorized users can access these shared folders.

► **To create a shared folder on the primary server**

1. Log on or connect to the primary server with a user account that is a member of the Administrators local group on that computer.
2. On a drive not used by SQL Server, create a folder for storing transaction log backups.
3. Grant full–control, file-level access to this folder to the domain user account that the SQL Server Agent service uses and to the Administrators local group. Remove the Everyone group from the file level permissions list.
4. Share the new folder.

   Use a share name that has no spaces. If the share name has a space, you have to place double quotation marks around it when you reference it in command prompt utilities.
5. Grant full control share level access through this share to the domain user account that the SQL Server Agent service uses and to the Administrators local group. Remove the Everyone group from the share-level permissions list.

► **To create a shared folder on a standby server**

1. Log on or connect to the standby server with a user account that is a member of the Administrators local group on that computer.
2. On a drive not used by SQL Server, create a folder for storing transaction log backups.
3. Grant full–control, share-level access to this folder to the domain user account that the SQL Server Agent service uses and to the Administrators local group. Remove the Everyone group from the permissions list.

4. Share the new folder.

   Use a share name that has no spaces. If the share name has a space, you have to place quotation marks around it when you reference it in command prompt utilities.
5. Grant full–control, share-level access through this share to the domain user account that the SQL Server Agent service uses and to the Administrators local group. Remove the Everyone group from the permissions list.

Note: Repeat this procedure on each standby server.

Creating Database Folders on Each Standby Server

Log shipping creates a copy of the production database on each standby server. Use an identical path for the data and log files between the primary server and a standby server to avoid having to edit the restoration script on the standby server. Use a mirrored drive to increase the availability of the standby database. In addition, grant full–control, file-level access to the data and log file folders to the domain user account that the SQL Server service uses and to the Administrators local group.

► **To create database folders on a standby server**

1. Log on or connect to the standby server with a user account that is a member of the Administrators local group on that computer.
2. Create a folder for the standby database data file for the copy of the production database.

   Use the same drive letter and path for this data file as is used on the primary server.
3. Grant full–control, file-level access to this folder to the domain user account that the SQL Server Agent service uses and to the Administrators local group. Remove the group Everyone from the permissions list.
4. Create a folder for the standby database log file.

   Use the same drive letter and path for this log file as is used on the primary server.
5. Grant full–control, file-level access to this folder to the domain user account that the SQL Server Agent service uses and to members of the Administrators local group. Remove the Everyone group from the permissions list.

Note: Repeat this procedure on each standby server.

Creating the Log-Shipping Jobs and Initializing the Standby Database

Use the Database Maintenance Plan Wizard in SQL Server Enterprise Manager to create the log-shipping jobs and to initialize the standby database on each standby server. You can run this wizard from any computer that has SQL Server Enterprise Manager installed and to which each participating server is registered; only registered servers are available for selection in the wizard when you are configuring log-shipping. Using the wizard, specify and configure the primary server, specify and configure each standby server, initialize each standby database, configure log-shipping job schedules and alert thresholds, and configure the monitoring server and report properties.

Note: Log shipping can be configured only by using the Database Maintenance Plan Wizard. Setting up log shipping using Transact-SQL scripts is not supported.

► **To specify and configure the primary server**

1. In SQL Server Enterprise Manager, connect to the production database on the primary server, and then expand **Databases**.
2. Right-click the production database, point to **All Tasks**, and then click **Maintenance Plan**.
3. On the **Welcome to the Database Maintenance Plan Wizard** page, click **Next**.
4. On the **Select Databases** page, verify that only the check box for the production database is selected. Then select the **Ship the Transaction Logs to Other SQL Servers (Log Shipping)** check box, and click **Next**.
5. On the **Update Data Optimization Information** page, click **Next**.

   Do not select any of the check boxes.
6. On the **Database Integrity Check** page, click **Next**.

   Do not select any of the check boxes.
7. On the **Specify the Database Backup Plan** page, clear the **Back Up the Database as Part of the Maintenance Plan** check box and then click **Next**.

   Develop a separate backup plan for full and differential database backups.
8. On the **Specify Transaction Log Backup Disk Directory** page, click **Use This Directory**, and then browse to or type the path that you created for the transaction log files.
9. Select the **Remove Files Older Than** check box, specify a retention time for the transaction log backup files according to your business archive policy, and then click **Next**.

   You must ensure that the primary server has sufficient space to hold the transaction log backup files for the retention time specified.

Note: Archive these transaction log backup files that are used for log shipping as part of your regular backup strategy.

10. On the **Specify the Transaction Log Share** page, browse to or type the path for the shared synchronization folder on the primary server, and then click **Next**.

► **To specify and configure standby servers**

1. On the **Specify the Log Shipping Destinations** page, click **Add** to specify a standby server (called a *destination server* in the wizard).
2. In **Add Destination Database**, click a standby server in the **Server Name** list.
3. In the **Directory** text box, browse to or type the path that you created on the standby server for the transaction log files.

   Change the path that the wizard suggests.
4. In the **For Data** text box in **Destination Database**, browse to and type the path that you created for the data file on the standby server.
5. In the **For Log** text box in **Destination Database**, browse to and type the path that you created for the log file on the standby server.

   Notice that the **Database Name** box completes automatically with the same name as the production database. Do not change the database name.

   Note: You can use an already existing database rather than having the wizard create a new one. If you want to use an existing database, you must restore the standby database from backup using either the **NORECOVERY** or the **STANDBY** option. You must also ensure that the standby database is synchronized with the production database before log shipping commences. The backup that is used must be taken when the database is using either the bulk-logged or full-recovery model.

6. In **Download State**, click **No Recovery Mode** unless you plan to use the standby database for reporting and maintenance tasks.

   If you plan to use the standby database for reporting and maintenance tasks, click **Standby Mode**, and select the **Terminate Users in Database** check box.
7. Select the **Allow Database to Assume Primary Role** check box. In the **Directory** text box, browse to or type the path for the synchronization folder on the standby server.

   Specify a path to a folder on the standby server. A common error is to specify a path to a folder on the local server.
8. Click **OK**.

Note: Repeat this procedure for each standby server.

► To initialize the production database copy on the standby servers

1. After you have added all participating standby servers, click **Next** on **Specify the Log Shipping Destinations**.
2. On the **Initialize the Destination Databases** page, click **Perform a Full Database Backup Now**, and then click **Next**.

   If you chose to use an existing database in the preceding procedure, this page will not appear. In this case, do not continue until you ensure that the existing database on the standby server has been initialized and is ready to accept transaction log backups from the production server.

Note: If you have a current backup, click **Use Most Recent Backup File**, and then browse to or type the path and name of a recent backup file. If the backup is not current, either have the wizard create a new backup or manually initialize and synchronize the standby database using the existing database backup and all recent transaction log backups.

► To configure the log-shipping job schedules and alert thresholds

1. To modify the default frequency of transaction log backups on the primary server, on the **Log Shipping Schedules** page, click **Change** in **Backup Schedule**. Change the transaction log backup frequency, and then click **OK** in **Edit Recurring Job Schedule**.

   The default interval is 15 minutes. A shorter interval enables you to reduce the latency between the production database and a standby database but slightly increases the total overhead on the primary server. The smallest interval that you can specify is 1 minute.
2. To modify the default frequency of the copy and load jobs, type or select a frequency in the **Copy/Load Frequency** box.

   The default frequency is 15 minutes. This frequency should match the backup schedule frequency so that each transaction log backup is copied from the primary server to each standby server immediately after it is complete, without delaying the copy process.
3. To specify a delay between copying a transaction log backup and restoring it, type or select a delay in the **Load Delay** box.

   The default delay is 0. Set a delay if you want time to detect corruption to the production database before the corruption affects the standby database. Also set a delay if are using the standby server for reporting or maintenance tasks.

Note: If you are configuring multiple standby servers and want a different load delay for a particular server, edit the database maintenance plan after completing the wizard, and adjust the load frequency for that server.

---

4. In the **File Retention Period** box, type or select a time to keep the transaction log backup files on the standby server.

   Although these transaction log backup files are also archived at the primary site, you might want to archive them in the standby site in case a manual restore is required. Doing so is particularly important if the standby server is geographically remote from the primary server. Ensure that the standby server has sufficient space to hold the transaction log backup files shipped from the primary server for the retention period specified.
5. Click **Next**.
6. On the **Log Shipping Thresholds** page, in the **Backup Alert Threshold** box, click the maximum time allowed between successful transaction log backups on the primary server before an alert is issued.

   The default threshold is three times the backup frequency. Change this threshold according to your business needs. The backup alert is set to error message 14420. You can change which error message the alert uses after the wizard finishes by using the Log Shipping Monitor.
7. In the **Out Of Sync Alert Threshold** box, type or select the maximum allowable time difference between the last transaction log backup of the production database on the primary server and the last restore of the transaction log to the production database copy on the secondary server.

   The default threshold is three times the backup frequency. Change this threshold according to your business needs. The threshold alert is set to error message 14421. You can change which error message the alert uses after the wizard finishes by using the Log Shipping Monitor.
8. Click **Next**.

► **To configure the monitoring server and report properties**

1. On the **Specify the Log Shipping Monitor Server Information** page, in the **SQL Server** list, click the server you want to use to monitor the log-shipping jobs.

   Do not use the primary server as the monitoring server. If the primary server fails, you won't know the current log-shipping status. If you have only two servers, use the standby server; otherwise, choose a third, independent server. A monitoring server can monitor many log-shipping servers.
2. Specify the authentication method for the SQL Server Agent service on the monitoring server to use to connect to the primary server and each secondary server. Then click **Next**.

Use Windows Authentication. If SQL Server Authentication is selected, SQL Server Agent uses the log_shipping_monitor_probe SQL Server login account. You must configure each participating server for mixed mode authentication and create this SQL Server login account on each participating server.

3. On the **Reports to Generate** page, select the **Write Report to a Text File** check box, and then browse to or type the directory to be used to save the reports.
4. Select the **Delete Text Report Files Older Than** check box, and then click a retention period appropriate to your business requirements.
5. Select the **Send E-Mail Report to Operator** check box, and then click an operator from the list, or click **New Operator** to create a new operator. Click **Next**.

   To send e-mail notifications, configure SQLAgentMail.
6. On the **Maintenance Plan History** page, click the maximum number of rows stored in the history table, or clear the **Limit Rows in the Table** check box to remove the limit completely.
7. If you want to use a centralized reporting server, select the **Write History to the Server** check box, and then browse to or type the name of a centralized server.

   For example, you might want to have all log-shipping history centralized to the monitoring server.
8. Click **Next**.
9. On the **Database Maintenance Plan Wizard Summary** page, type a unique descriptive name for this database maintenance plan, and then click **Next**.

   Note: If you are log shipping to an existing database, be sure that you synchronize that database with the production database before you click **Finish**.
10. On the **Completing the Database Maintenance Plan Wizard** page, click **Finish**.

Note: If the Database Maintenance Plan Wizard fails to complete due to an error, some changes are not completely rolled back. You must resolve the error and roll back all changes made by the wizard before running the wizard again. For more information, see article Q298743, "All Changes May Not be Rolled Back when Log Shipping Maintenance Wizard Fails," in the Microsoft Knowledge Base at *http://support.microsoft.com/default.aspx?scid=kb;[LN];Q298743*.

Creating and Executing a Login Synchronization Job

After you have set up log shipping, you must synchronize logins between the primary server and each standby server. The **master** database stores login accounts. Log shipping the production database does not ensure that user accounts in the production database have corresponding login accounts in the **master** database on each standby server.

You must synchronize logins before clients can use the standby database. Synchronizing logins is a manual process requiring you to transfer logins from the primary server to each standby server, export login information from the syslogins table on the primary server to a file, and synchronize logins and user accounts on the standby server. The final synchronization of logins and user accounts occurs when the standby database is recovered during the role change process.

The procedures below create a login synchronization job for each standby server to ensure that the logins remain synchronized between the primary server and the standby server. Run this job when you initially set up each standby server and each time you change logins on the primary server. Server Roles are not synchronized between servers by these procedures, so all users in system roles should be managed manually at each server using a script created by you or directly in Enterprise Manager.

Important: Apply the changes specified in article Q310882, "BUG: sp_resolve_logins Stored Procedure Fails If Executed During Log Shipping Role Change" in the Microsoft Knowledge Base at *http://support.microsoft.com/default.aspx?scid=kb;en-us;Q310882* to each server before proceeding. The article describes how you must edit the stored procedure sp_resolove_logins in the master database.

Locate the following text in the procedure:

```
SELECT   *
INTO     #sysloginstemp
FROM     syslogins
WHERE    sid = 0x00
```

Replace it with the following text:

```
SELECT   *
INTO     #sysloginstemp
FROM     master.dbo.syslogins
WHERE    sid = 0x00
```

► To create a transfer logins DTS package for a standby server

Note: If you have created a login for SQL Server by using a local user account in Windows 2000, this transfer logins task will fail because a local login is not valid on any other server. In a high availability environment, only create logins for domain users, or create SQL Server logins.

1. Using SQL Server Enterprise Manager, connect to the primary server.
2. Right-click **Data Transformation Services**, and click **New Package**.
3. On the **Task** menu, click **Transfer Logins Task**.

4. On the **Source** tab in **Transfer Logins Properties**, browse to or type the name of the primary server in the **Source Server** box, and verify that **Use Windows Authentication** is selected.
5. On the **Destination** tab, browse to or type the name of a standby server in the **Destination Server** box, and then verify that **Use Windows Authentication** is selected.
6. On the **Logins** tab, click **Logins for Selected Databases**.
7. In the **Databases** list, select the check box for the production database, and then click **OK**.
8. On the **Package** menu, click **Save As**.
9. In **Save DTS Package**, type **Transfer_Logins** in the **Package Name** box, and then click **OK**.

   The transfer logins package saves to the **msdb** database on the primary server.
10. Close the transfer logins DTS package.

Note: Create a separate transfer logins package on the primary server for each standby server.

► To create a login synchronization job for a standby server

1. Using SQL Server Enterprise Manager, connect to the primary server.
2. Expand **Management**, expand **SQL Server Agent**, right-click **Jobs**, and then click **New Job**.
3. On the **General** tab in **New Job Properties**, type **Synchronize Logins <*standby server*>** in the **Name** box.
4. In the **Owner** list, select the domain user account that the SQL Server Agent service uses.

► To create a BCP job step for the synchronize logins job

1. On the **Steps** tab, click **New**.
2. On the **General** tab in **New Job Step**, type **BCP Out** in the **Step Name** box.
3. In the **Type** list, select **Operating System Command (CmdExec)**.
4. In the **Command** text box, type the following command as follows:

```
BCP master..syslogins out \\<primary server>\<share name of synchronization
folder>\syslogins.dat /N /S <primary server> /T
```

Note: If the share name of the synchronization folder has a space in it, you must use double quotation marks around the entire path for the output file.

---

5. Click **OK**.

► To create a copy file job step for the synchronize logins job

1. On the **Steps** tab, click **New**.
2. On the **General** tab in New Job Step, type **Copy File** in the Step Name box.
3. In the **Type** list, select **Operating System Command (CmdExec)**.
4. In the **Command** box, type the following command:

```
copy <path to shared synchronization folder>\syslogins.dat \\<standby
server>\<share name for synchronization folder>
```

5. Click **OK**.

Note: The log-shipping role change process uses this file to reconcile SQL Server logins with standby database user accounts.

► To create a transfer logins job step for the synchronize logins job

1. On the **Steps** tab, click **New**.
2. On the **General** tab in New Job Step, type **Transfer Logins** in the Step Name box.
3. In the **Type** list, select **Operating System Command (CmdExec)**.
4. In the **Command** box, type the following command:

```
DTSRun /S<primary server> /E /NTransfer_Logins
```

5. Click **OK**.
6. Click **OK** to close the synchronize logins job.

Note: Create a separate synchronize logins job on the primary server for each standby server.

► To execute the synchronize logins job for a standby server

1. Using SQL Server Enterprise Manager, connect to the primary server.
2. Expand **Management**, expand **SQL Server Agent**, and then expand **Jobs**.
3. Right-click **Synchronize Logins <*standby server*>**, and then click **Start Job**.
4. In the **Start Job on <*standby server*>**, click **Start** to execute all of the steps in the synchronize logins job.

Note: Execute the synchronize logins job for each standby server whenever you change logins on the primary server.

Scripting and Copying Jobs, Alerts, and Operators

The **msdb** database, not the production database, stores SQL Server jobs, alerts, and operators. If you have jobs, alerts, or operators that run on the primary server on which your production database relies, manually copy these jobs, alerts, and operators to each standby server. You cannot simply back up the **msdb** database and restore it to a standby server. Doing so overwrites all jobs, alerts, and operators in the **msdb** database on the standby server, including all the log-shipping jobs. You can copy jobs, alerts, and operators by using SQL Server Enterprise Manager to generate Transact-SQL scripts for them and then applying the scripts to each standby server. You must complete these tasks before users connect to a standby server that is promoted to become the new primary server.

Use the following procedures to copy jobs, alerts, or operators that the production database needs to each standby server.

► **To script all jobs on the primary server**

1. In SQL Server Enterprise Manager, connect to the primary server, expand **Management**, and then expand **SQL Server Agent**.
2. Right-click **Jobs**, point to **All Tasks**, and then click **Generate SQL Script**.
3. In **Generate SQL Script**, browse to or type the path to the synchronization folder on the primary server, and then type a file name for the jobs script using the extension **.sql**.
4. Verify that the **International Text (Unicode)** check box is selected in the **File Format** box and that the **Replace Job If It Exists** check box is selected in the **SQL Generation Options** box.
5. Accept the default for the **TSQL Batch Separator**.
6. Click **OK** to create a Transact-SQL script for all jobs on the primary server.
7. Use SQL Query Analyzer to edit the jobs script to remove all log-shipping jobs.

► **To execute the jobs script on each standby server**

1. Using SQL Query Analyzer, connect to SQL Server on a standby server.
2. On the **File** menu, click **Open**.
3. In **Open Query File**, type or browse to the jobs script in the shared synchronization folder on the primary server, and then open the jobs script.
4. Execute the jobs script, and then close SQL Query Analyzer.
5. In SQL Server Enterprise Manager, connect to the appropriate instance on the standby server, expand **Management**, expand **SQL Server Agent**, and then click **Jobs**.
6. In the details pane, review the jobs that were created. Disable those jobs until they are needed after a role change.

Note: Repeat this procedure on each standby server.

► To script all alerts on the primary server

1. In SQL Server Enterprise Manager, connect to the primary server, expand **Management**, and then expand **SQL Server Agent**.
2. Right-click **Alerts**, point to **All Tasks**, and then click **Generate SQL Script**.
3. In **Generate SQL Script**, browse to or type the path to the synchronization folder on the primary server and then type a file name for the alerts script using the extension **.sql**.
4. Verify that the **International Text (Unicode)** check box is selected in the **File Format** box.
5. Verify that the following check boxes are selected in the **SQL Generation Options** box:
   - **Replace Alert If It Exists**
   - **Include the Name of the Job Executed by the Alert**
6. Clear the Include Notifications Sent by the Alert to the Operator check box.

   Note: Notifications are copied when the operators are copied to ensure that the operators exist prior to creating the notification.

7. Accept the default for the **TSQL Batch Separator**.
8. Click **OK** to create a Transact-SQL script for all alerts on the publisher.

   Note: A bug in the scripting routine exports all Net Send notifications as e-mail notifications. If you use Net Send in any notifications, edit the script file and change the notification method in each **sp_add_notification** statement from **@notification_method =1** to **@notification_method =4** to configure the notification to use Net Send.

   Note: If any alerts use user-defined messages, you must create those messages on the subscriber before applying this script. It is recommended that you always use a script to create user-defined messages so you can recreate the messages on any SQL Server instance.

► To execute the alerts script on each standby server

1. Using SQL Query Analyzer, connect to SQL Server on a standby server.
2. On the **File** menu, click **Open**.
3. In **Open Query File**, type or browse to the alerts script in the shared synchronization folder on the primary server, and then open the alerts script.

4. Execute the alerts script, and then close SQL Query Analyzer.
5. In SQL Server Enterprise Manager, connect to the appropriate instance on the standby server, expand **Management**, expand **SQL Server Agent**, and then click **Alerts**.

   In the details pane, review the alerts that were created.

Note: Repeat this procedure on each standby server.

► To script all operators on the primary server

1. In SQL Server Enterprise Manager, connect to the primary server, expand **Management**, and then expand **SQL Server Agent**.
2. Right-click **Operators**, point to **All Tasks**, and then click **Generate SQL Script**.
3. In **Generate SQL Script**, browse to or type the path to the synchronization folder on the primary server and then type a file name for the operators script using the extension **.sql**.
4. Verify that **International Text (Unicode)** is selected in the **File Format** box.
5. Verify the following check boxes are selected in the **SQL Generation Options** box.
   - Replace Operator If It Exists
   - Include Notifications Sent by Alert to the Operator
6. Accept the default for the **TSQL Batch Separator**.
7. Click **OK** to create a Transact-SQL script for all operators on the primary server.

► To execute the operators script on each standby server

1. Using SQL Query Analyzer, connect to SQL Server on a standby server.
2. On the **File** menu, click **Open**.
3. In **Open Query File**, type or browse to the operators script in the shared synchronization folder on the primary server, and then open the operators script.
4. Execute the operators script, and then close SQL Query Analyzer.
5. In SQL Server Enterprise Manager, connect to the appropriate instance on the standby server, expand **Management**, expand **SQL Server Agent**, and then click **Operators**.
6. In the details pane, review the operators that were created.

Note: Repeat this procedure on each standby server.

Copying DTS Packages

DTS packages can be stored in the **msdb** database or in the file system. If you have DTS packages that run on the primary server on which your production database relies, you must manually copy these DTS packages to each standby server. You cannot simply back up the **msdb** database and restore it to a standby server. Doing so overwrites all jobs, alerts, operators, and DTS packages in the **msdb** database on the standby server. You can open DTS packages saved to the **msdb** database on the primary server and save them to each standby server. Simply copy DTS packages saved to the file system on the primary server to a folder on each standby server. Complete this task before users connect to a standby server that is promoted to become the new primary server.

Use the following procedure to copy DTS packages that the production database needs to each standby server.

► **To copy DTS packages**

1. For each DTS package stored in the file system on the primary server, copy the DTS package to each standby server using Windows Explorer.

   When copying DTS packages to each standby server, use the same drive and path that the primary server uses. Doing so eliminates potential path problems.
2. For each DTS package stored in the **msdb** database on the primary server, use SQL Server Enterprise Manager to open the DTS package, and then save it to the **msdb** database on each standby server.

Note: To ensure DTS packages execute properly on a subscriber, use an alias or the dynamic properties task. For more information, see "Redirecting Client Network Traffic to a Promoted Secondary Server" in Planning Guide Chapter 5, "Minimizing Downtime by Using Redundant Components."

Changing the Role of a Standby Server

To change the role of a standby server when the primary server fails or is taken offline, disable the database restoration job on that server, execute the role change stored procedures, synchronize SQL Server logins and standby database user accounts, and then use the standby database as the new production database. Ensure all logins, jobs, alerts, operations, and DTS packages are synchronized, and then redirect clients to the promoted standby server. For more information about facilitating this client redirection task, see Planning Guide Chapter 5, "Minimizing Downtime by Using Redundant Servers."

In a planned role change, pause all applications that use the production database, and then place the production database in single user mode. Pausing the production database helps ensure that the standby database is transactionally current with the production database. Pausing the production database fails if applications do not disconnect from the database after each transaction. Be sure you don't have open connections in Enterprise Manager, Query Analyzer, or other utilities that would cause the pause procedure to fail.

In an unplanned role change, recent transactions that are not backed up and copied to the standby server before the primary server fails are lost.

Note: Save the log-shipping role change Transact-SQL scripts shown in the procedures later in this chapter, and incorporate them into OSQL batch files to make role changing faster and less error prone.

► To pause applications that use the production database

1. Disconnect applications from the production database if the database is still available.

   The method for doing so varies by application. In a planned failover, allow open transactions to complete, but do not allow new ones to start.

2. If you implemented NLB to direct traffic to a specific server, issue a **WLBS** command to ensure that all transactions have completed:

```
WLBS DrainStop <NLB cluster name>:<primary server> /PASSW <password>
```

Note: For more information about NLB, see Solutions Guide Chapter 4, "Implementing Network Load Balancing."

► To place the production database in single user mode from a command prompt

- Put the following command into a batch file so that it is ready when you need it:

```
OSQL –Q "ALTER DATABASE <production database> SET SINGLE_USER WITH ROLLBACK
IMMEDIATE" –S<primary server> -d master –E
```

Note: Ensure that no users are connected to the production database before continuing, including any connections you have through SQL Server Enterprise Manager.

► To execute the sp_change_primary_role system stored procedure

1. Using SQL Query Analyzer, connect to the primary server by using Windows Authentication.

2. In the query window, type and execute the following Transact-SQL script:

```
Msdb..Sp_change_primary_role
     @db_name=<production database>,
     @backup_log=1,
     @terminate=0,
     @final_state=2,
     @access_level=2
```

Note: You will not be able to run this system stored procedure if the primary server has failed. All transactions not previously backed up are lost and users have to recreate these lost transactions.

When the sp_change_primary_role system stored procedure executes with these parameters, the system stored procedure performs the following actions:

- Removes the production database from the log-shipping database maintenance plan.
- Disables the log-shipping transaction log backup job on the primary server.
- Backs up the tail end of the production database transaction log.
- Terminates and rolls back all pending transactions in the production database.
- Places the production database in single-user mode for the duration of the system stored procedure.
- Sets the recovery state of the production database after the system stored procedure completes to **NO RECOVERY**, which allows the application of transaction log backups from the promoted standby database.
- Sets the access level of the production database after the system stored procedure completes to restricted user mode.

Note: If this system stored procedure fails because SQL Server cannot obtain exclusive access to the production database, rerun this system stored procedure after closing any existing connections to the production database. If the first execution of this system stored procedure fails, the log-shipping transaction log backup job is still disabled.

► To stop the log-shipping restore job on the standby server

1. Using SQL Server Enterprise Manager, connect to SQL Server on the standby server.
2. Expand **Management**, and then click **Jobs**.
3. Right-click **Log Shipping Restore for <*primary server*>.<*production database*>_logshipping**, and then click **Disable Job**.

   Disabling the log-shipping restore job prevents its periodic execution from interfering with the execution of the sp_change_secondary_role system stored procedure.

Note: Ensure that no users are connected to the standby database before continuing, including any connections you have through SQL Server Enterprise Manager.

► To execute the sp_change_secondary_role system stored procedure

1. Using SQL Query Analyzer, connect to the standby server by using Windows Authentication.
2. In the query window, type and execute the following Transact-SQL script:

```
Msdb..sp_change_secondary_role
     @db_name=<standby database>,
     @do_load=1,
     @final_state=1,
     @access_level=1,
     @terminate=0
```

When the sp_change_secondary_role system stored procedure executes with these parameters, this system stored procedure performs the following actions:

- Copies the final production database transaction log backup to the standby server along with any additional transaction log backups that were not previously copied.
- Disables the log-shipping transaction copy job on the standby server.
- Restores all transaction log backups not previously restored to the standby database and initiates the recovery of the standby database.
- Terminates and rolls back all pending transactions.
- Sets the access level of the standby database after completion of the system stored procedure to multiuser mode.
- Creates a log-shipping database maintenance plan on the new primary server for the new production database. The SQL Server Agent job defined in this plan backs up the transaction log of the new production database to the folder location you specified when you set up log shipping.

Note: If this system stored procedure fails because SQL Server cannot obtain exclusive access to the standby database, rerun this system stored procedure after closing any existing connections to the production database. If the first execution of this system stored procedure fails, the log-shipping copy job is still disabled.

► To synchronize SQL Server logins and database users

Note: Be sure you have applied the fix specified in article Q310882, "BUG: sp_resolve_logins Stored Procedure Fails If Executed During Log Shipping Role Change" in the Microsoft Knowledge Base at *http://support.microsoft.com/default.aspx?scid=kb;en-us;Q310882*.

1. Using SQL Query Analyzer, connect to SQL Server on the new primary server by using Windows Authentication.
2. Select the new production database.
3. In the query window, type and execute the following Transact-SQL script:

```
sp_resolve_logins '<database name>', '\\<new primary server>\<share name of
synchronization folder>', 'syslogins.dat'
```

Note: This script will not successfully synchronize mappings between SQL Server logins and database users in the standby database if you do not run the synchronize logins job for each standby server each time you change SQL Server logins or database users on the primary server.

► **To execute the sp_change_monitor_role system stored procedure**

1. Using SQL Query Analyzer, connect to SQL Server on the monitoring server by using Windows Authentication.
2. In the query window, type and execute the following Transact-SQL script:

```
Msdb..Sp_change_monitor_role
     @primary_server=<old primary server>,
     @secondary_server=<promoted standby server>,
     @database=<promoted standby database>,
     @new_source=<transaction log backup folder for promoted standby database>
```

   When the sp_change_monitor_role system stored procedure executes with these parameters, the **msdb** database on the monitoring server updates to reflect the new primary server, the new standby server, and the new transaction log backup folder.

The former standby server is now the primary server. To maintain the database redundancy protection provided by log shipping, edit the database maintenance plan on the promoted standby server to enroll the original primary server as a standby server. If the tail of the transaction log for the production database on the original primary server was backed up and the database left in recovery mode, you can apply transaction logs from the new primary server without reinitializing the original production database. If the original primary server failed, initialize and synchronize the standby database on this server before adding it to the database maintenance plan.

3

Implementing Transactional Replication

This chapter teaches you how to increase the availability of a Microsoft® SQL Server™ data center by implementing transactional replication. After reading this chapter, you will be able to configure each component in this high-availability solution. To configure transactional replication using Network Load Balancing (NLB), read Solution Guide Chapter 4, "Implementing Network Load Balancing."

The implementation steps include annotation to help you understand the selected configuration choices. The configuration options presented in this chapter have been designed by using input from the field, reviewed thoroughly by Microsoft Product Support Services (PSS), and tested by an independent team to ensure accuracy and functionality.

Transactional replication is only part of achieving a highly available data center. To achieve a highly available data center, you must also follow the processes outlined in the accompanying Planning Guide.

Understanding Transactional Replication

Transactional replication is a high-availability solution that is provided with all SQL Server 2000 editions and that can be implemented with any server edition of Windows 2000. If you want to use the client redirection capabilities of the NLB; however, you must use either Microsoft Windows 2000® Advanced Server or Windows 2000 Datacenter Server.

Transactional replication is an inexpensive high-availability solution that uses SQL Server jobs to initialize and then periodically update a copy of the production database (the subscriptions database) on one or more secondary servers (subscribers).

With transactional replication, proximity is not a limitation; you can place a subscriber in a geographically remote location to provide protection against a catastrophic disaster. Subscribers do not need to be identical to the primary server (the publisher), although differences in the amount of memory, number and speed of the processors, and the robustness of the storage system can affect SQL Server performance after a role change.

You can use the subscription database on a subscriber in place of the production database in case the publisher fails. The subscription database is always slightly out of sync with the production database. You can control the level of synchronicity — it can be as short as several seconds (by using a high speed network), or as long as you choose.

With transactional replication, a subscription database can be used for reporting or maintenance tasks without affecting the transactional currency of the subscription database. Transactional replication was not designed as a high-availability solution, however, and does not provide a built-in role change mechanism to change the role of a subscriber. In addition, transactional replication does not replicate most schema changes to the subscription database. You replicate added or dropped columns using special stored procedures.

Changing the role of a subscriber is a manual process that requires the data base administrator (DBA) to perform a number of tasks to complete the role change, including the manual redirection of client requests to the promoted subscriber. To point clients to the new primary server, use one of the following client redirection options:

- Program an alternative server list into the client application
- Update the client Data Source Name (DSN)
- Update the Domain Name System (DNS)
- Update the COM+ component
- Update the NLB configuration.

Any transactions that have not been copied to the distributor before the publisher fails are lost. Because changing server roles is a manual process, changing roles takes longer than changing nodes with failover clustering. Changing roles, however, is significantly faster than restoring a database to a secondary server from a cold backup.

Transactional replication is most useful as an alternative to log shipping when you want to use the subscription database for reporting and maintenance tasks without sacrificing transactional currency.

To implement transactional replication, you need to have certain things in place:

- You need people in the following roles:
  - Database administrator
  - Network administrator
- You need the following software:
  - Windows 2000 Server, Windows 2000 Advanced Server, or Windows 2000 Datacenter Server
  - SQL Server 2000 Standard Edition or SQL Server 2000 Enterprise Edition
- You need systems and system components that are certified to work together and with the chosen Windows 2000 operating system edition. Windows 2000 Datacenter Server has more stringent certification requirements than Windows 2000 Advanced Server. To find systems and system components that are certified by Microsoft, search the "Hardware Compatibility List" on the Microsoft Web site at *http://www.microsoft.com/hcl/search.asp*.

For more information about planning for transactional replication, see "Planning for Transactional Replication" on the MSDN Web site at *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/replsql/replplan_1l4e.asp*.

For more information about tuning transactional replication, see "Transactional Replication Performance Tuning and Optimization" on the MSDN Web site at *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnsql2k/html/sql2k_replperf_tran4.asp*.

For information about using transactional replication with log shipping, see "Transactional Replication and Log Shipping" on the MSDN Web site at *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/replsql/replbackup_3js7.asp*.

For information about connectivity between geographically dispersed sites, see "Publishing Data Over the Internet Using VPN" in SQL Server Books Online, and "Virtual private network (VPN) connections overview" on the MSDN Web site at *http://www.microsoft.com/windowsxp/home/using/productdoc/en/default.asp?url=/WINDOWSXP/home/using/productdoc/en/Conn_VPN.asp?frame=true*.

Deploying Transactional Replication

Implementing transactional replication is wizard-driven and easy to follow. You must make some schema changes before you begin the transactional replication setup wizard, however.

This chapter assumes the following with respect to each participating server in the transactional replication solution:

- Windows 2000 Server, Windows 2000 Advanced Server, or Windows 2000 Datacenter Server is installed.
- The most recent operating system service pack is installed.
- The server is a member of the same Microsoft Active Directory® service forest as each other server.
- At least one network interface card (NIC) is installed.
- The server is connected to the corporate network.
- SQL Server 2000 Standard Edition or SQL Server 2000 Enterprise Edition is installed and configured to use the same authentication mode as each other server.
- The most recent SQL Server 2000 service pack is installed.
- The SQL Server Agent service is configured to use the same domain user account used by each the SQL Server Agent service on each other server.
- The SQL Server Agent service is configured to start automatically.
- The server is registered in SQL Server Enterprise Manager.
- If e-mail notifications will be used, SQLAgentMail is configured.

This chapter also assumes that the production database on the primary server is configured to use either the bulk-logged recovery model or the full recovery model. The logical design of the transactional replication solution built in this chapter is described below, followed by the specific steps required to build this high-availability solution.

Understanding the Logical Design

Figure 3.1 shows the logical design of the transactional replication solution that is described in this chapter. For simplicity, only the subscriber/distributor is shown. Additional subscribers can be added — for instance, at geographically remote locations — to protect against site-level disasters.

Figure 3.1
Logical design diagram of a transactional replication solution

After you set up the production database on the publisher, set up transactional replication. Designate a subscriber as the distributor in a separate site from the publisher. Placing the distributor in a separate site increases the availability of the data center in the event the primary site fails. On the server containing the production database, designate the server as the publisher, and publish the production database. Configure one or more subscribers to receive the publication and maintain a subscription database. The snapshot agent, the distribution agent, and the log reader agent perform the work of maintaining the transactional currency of the subscription database on each subscriber.

The snapshot agent takes an initial snapshot of the production database and stores the snapshot in the snapshot folder on the subscriber/distributor. The distribution agent initializes the subscription database on each subscriber by using the snapshot. The log reader agent reads the production database transaction log, copying all changes since the initial snapshot to the **distribution** database on the subscriber/distributor. The distribution agent periodically retrieves the production database changes stored in the **distribution** database and updates the subscription database on each subscriber.

Although no special networking is required for transactional replication, you must ensure that the bandwidth between the publisher and each subscriber (especially the subscriber/distributor) is sufficient to handle the network traffic generated by the transactional replication agents. These replication agents must be able to keep up with the volume of changes occurring in the production database. The amount of traffic depends on the number and size of the transactions generated in the production database. On the publisher, ensure that all disks are fault tolerant to minimize the need for a role change because of a disk failure. On the subscriber/distributor, use fault tolerant disks to ensure that all copied transactions are replicated to each subscriber and that the subscriber is available to function as a replacement server if needed. On each additional subscriber, use fault tolerant disks to ensure that each subscriber is available to function as a replacement primary server if necessary.

The steps required to implement this logical design are below, followed by the steps in detail.

► **To implement transactional replication with SQL Server**

1. Create an empty database on each subscriber.
2. Create and share the snapshot folder on the distributor.
3. Create and share a synchronization folder on the publisher.
4. Prepare the schema for replication.
5. Configure the distributor.
6. Configure the publisher, and publish the production database.
7. Configure and initialize subscribers.
8. Create and execute a login synchronization job.
9. Script and copy jobs, alerts, and operators.
10. Copy Data Transformation Services (DTS) packages.

Creating an Empty Database on Each Subscriber

If you do not have a current script for the production database, use SQL Server Enterprise Manager to generate a script of the production database. Create an empty database on each subscriber that is the same size as the production database and that uses the same drive letters and paths. If you use a different drive letter or path on a subscriber, edit the script to point to the appropriate drive letter and path before you use it on that subscriber.

► **To create a Transact-SQL script of the production database**

1. In SQL Server Enterprise Manager, connect to the SQL Server instance containing the production database, and then expand **Databases**.
2. Right-click the production database, point to **All Tasks**, and then click **Generate SQL Script**.

3. In **Generate SQL Scripts**, click **Show All** on the **General** tab, and then click the **Options** tab.
4. On the **Options** tab, select the **Script Database** check box, and then click **OK**.
5. In **Save As**, type a file name in the **File Name** box, and then click **Save**.
6. Click **OK** to close the message box.

► **To create the empty database on each subscriber**

1. Use SQL Query Analyzer to connect to SQL Server on a subscriber.
2. In **SQL Query Analyzer**, click **Open** on the **File** menu.
3. In **Open Query File**, browse to and click the Transact-SQL script created in the previous procedure, and then click **Open**.
4. In the Query window, review and edit the script as necessary, and then click **Execute Query** on the toolbar.
5. Verify that the script is executing successfully, and then close SQL Query Analyzer.

Note: Repeat this procedure on each subscriber.

Creating and Sharing the Snapshot Folder

On the subscriber/distributor, create a snapshot folder. The snapshot agent uses the snapshot folder to store the snapshot of the production database. The distribution agent uses this snapshot to populate the subscription database on each subscriber. These replication agents are SQL Server jobs that run in the security context of the SQL Server Agent service. The SQL Server Agent service must have permission to read and write to the snapshot folder. Use a mirrored drive that is not used for data or log files. Be sure only authorized users have access to the snapshot folder.

► **To create the shared snapshot folder on the subscriber/distributor**

1. Log on or connect to the subscriber/distributor with a user account that is a member of the Administrators local group on that computer.
2. On a drive not used for data or log files, create a folder to store the production database snapshot.
3. Grant full-control, file-level access to this folder to the domain user account that the SQL Server Agent service uses and to the Administrators local group. Remove the Everyone group from the file level permissions list.
4. Share the new folder.

   Use a share name that has no spaces. If the share name has a space, you will have to put double quotation marks around it when you reference it in command prompt utilities.

5. Grant full-control, share-level access through this share to the domain user account that the SQL Server Agent service uses and to the Administrators local group. Remove the Everyone group from the share level permissions list.

Creating and Sharing the Synchronization Folder

On the publisher, create and share a synchronization folder. This folder is used to store login information output by the bulk copy program (BCP) utility, the jobs script, the alerts script, and the operators script. You use the BCP output and these Transact-SQL scripts to synchronize objects between the publisher and each subscriber. The SQL Server Agent service must have permission to read and write to the synchronization folder. Use a mirrored drive that is not used for data or log files. Be sure only authorized users have access to the synchronization folder.

► **To create and share the synchronization folder on the publisher**

1. Log on or connect to the publisher by using a user account that is a member of the Administrators local group on that computer.
2. On a drive not used for data or log file, create a folder to store the synchronization information.
3. Grant full-control, file-level access to this folder to the domain user account used by the SQL Server Agent service and members of the Administrators local group. Remove the Everyone group from the file level permissions list.
4. Share this new folder.

   Use a share name without a space. If you create a share name that has a space in its name, you will have to put double quotation marks around the share name when referencing it in command prompt utilities.
5. Grant full–control, share-level access through this share to the domain user account used by the SQL Server Agent service and members of the Administrators local group. Remove the Everyone group from the share level permissions list.

Preparing the Schema for Replication

Before configuring transactional replication, you must prepare the schema for transactional replication. In a database that you want to publish, you must eliminate unresolved references in views and user-defined functions. You must set identity columns and triggers to **Not for Replication** and also ensure that all literals are defined with apostrophes rather than quotation marks. In addition, primary keys must be defined for every table. Finally, you must create Transact-SQL scripts for all tables with identity columns or time stamp data types.

Note: Encrypted stored procedures, views, triggers, and user-defined objects cannot be published because SQL Server transaction replication cannot recreate these objects on a subscriber.

You can only make schema changes to a published table in a database through the Replication Publication Properties dialog box in SQL Server Enterprise Manager or through special replication stored procedures. Changes made using these tools are propagated to each subscriber by the distribution agent. However, if you add a column, you need to reinitialize the subscription database at each subscriber.

Do not make schema changes to published tables using the **ALTER TABLE** statement or by using SQL Server Enterprise Manager. Changes made to the schema of a published table using these tools will not be propagated to subscribers. If you add an existing column to a publication, the subscribers must be re-initialized, which can impact their use as reporting databases. For more information about making schema changes, see *Schema Changes on Publication Databases* in SQL Server Books Online.

Note: If you need to change the schema of a published table, thoroughly test the changes on a non-production server first. This is a very high-risk change for applications in a high-availability environment.

► To eliminate unresolved references in stored procedures, views, and user-defined functions

1. Using SQL Server Enterprise Manager or SQL Query Analyzer, connect to the production database on the publisher and identify all stored procedures, views, and user-defined functions that reference objects that do not exist in the production database.

   The creation of the initial snapshot will fail if these references cannot be resolved.
2. Eliminate all unresolved references by using one or more of the following methods:
   - Create the missing objects.
   - Alter the definition of the views and user-defined functions that reference non-existent objects to point to existing objects.
   - Delete the views and user-defined functions referencing non-existent objects.

Note: If unresolved references are detected during the initialization of the subscription database on a subscriber, the following error message will appear: "Cannot use empty object (or) column names. Use a single space if necessary."

► To set identity columns to Not for Replication

1. Using SQL Query Analyzer, connect to the production database on the publisher and use the following Transact-SQL script to report which columns are identity columns:

```
SELECT O.name, C.name from sysobjects O
  INNER JOIN syscolumns C on O.id = C.id
    WHERE o.type='U' AND objectproperty (o.id, 'TABLEHASIDENTITY') = 1
    AND columnproperty (o.id, c.name, 'IsIdentity') = 1
    ORDER BY O.name, C.name
```

The schema for these tables must be modified before the schema is recreated on a subscriber by using the **Not for Replication** attribute. This attribute causes the value from the production database to be inserted into the subscription database rather than incrementing the identity value in the subscription database.

2. Set the **Not for Replication** attribute for the identity column in each identified table using one of the following methods:
   - In SQL Query Analyzer, use the **ALTER TABLE** statement to add the **Not for Replication** attribute to the identity column in each identified table.
   - In SQL Server Enterprise Manager, right-click each table, and then click **Design Table** to change the identity property to **Yes (Not for Replication)**.

► **To set triggers to Not for Replication**

1. Using SQL Query Analyzer, connect to the production database on the publisher and use the following Transact-SQL script to report which tables have triggers:

```
SELECT O.name from sysobjects O
  WHERE
    o.type='U' AND (objectproperty (o.id, 'TableHasDeleteTrigger') = 1
    OR objectproperty (o.id, 'TableHasInsertTrigger') = 1
    OR objectproperty (o.id, 'TableHasUpdateTrigger') = 1)
    ORDER BY O.name
```

   The schema for these triggers must be modified before the schema is recreated on a subscriber by using the **Not for Replication** attribute. This attribute disables the firing of the trigger on the subscriber when data is inserted into the subscription database by the distribution agent.

2. Set the **Not for Replication** attribute for each trigger in each identified table using one of the following methods:
   - In SQL Query Analyzer, use the **ALTER TRIGGER** statement to add the **Not for Replication** attribute to each trigger in each identified table.
   - In SQL Server Enterprise Manager, right-click each table, point to **All Tasks**, and then click **Manage Triggers** to add the **Not for Replication** attribute to each trigger in each identified table.

► **To change literal definitions to use apostrophes rather than quotation marks**

1. Using SQL Server Enterprise Manager, connect to the production database on the publisher and generate a Transact-SQL script for all stored procedures and views. Then use a text editor to determine the views and stored procedures that define literals with quotation marks rather than apostrophes.

   These views and stored procedures must be changed to use apostrophes or an error will appear that states that there is an invalid column (but with no context for the error) when the view or stored procedure is created on each subscriber.

2. Change the literal definitions from quotation marks to apostrophes in each identified view or stored procedure by using one of the following methods:
   - In SQL Query Analyzer, use the **ALTER VIEW** and **ALTER PROCEDURE** statements to modify the identified views and stored procedures.
   - In SQL Server Enterprise Manager, double-click each identified view and stored procedure, and then modify the literal definitions.

► **To define a primary key for a table**

1. Using SQL Query Analyzer, connect to the production database on the publisher and use the following Transact-SQL script to report which tables do not have primary keys:

```
SELECT O.name from sysobjects O
  WHERE o.type='U' AND objectproperty (o.id, 'TableHasPrimaryKey') = 0
  ORDER BY O.name
```

   You must create a primary key for each of these tables.
2. Create a primary key for each identified table by using one of the following methods:
   - In SQL Query Analyzer, use the **ALTER TABLE** statement to modify the identified table.
   - In SQL Server Enterprise Manager, right-click each identified table, click **Design Table**, and then click **Set Primary Key** on the toolbar.

► **To create Transact-SQL scripts for tables with identity columns or time stamps**

1. Using SQL Query Analyzer, connect to the production database on the publisher and use the following Transact-SQL script to report which tables have identity columns or time stamps:

```
SELECT O.name from sysobjects O
  WHERE o.type='U' AND (objectproperty (o.id, 'TableHasIdentity') = 1
     OR objectproperty (o.id, 'TableHasTimestamp') = 1)
     ORDER BY O.name
```

   You must create a Transact-SQL script for each of these tables. You supply the subscriber/distributor with these scripts so that the distribution agent can use them when creating tables on a subscriber. These scripts are required to ensure that the subscription database will function properly if it is ever used as the new production database.
2. In SQL Server Enterprise Manager, connect to SQL Server on the publisher, and then expand **Databases**.
3. Right-click the production database, point to **All Tasks**, and then click **Generate SQL Script**.

4. In **Generate SQL Scripts**, click **Show All** on the **General** tab.
5. In **Objects to Script**, select the **All Defaults** and the **All User-Defined Data Types** check boxes.
6. In the **Objects on <*production database*>** list, select each identified table, and then click **Add**.

   Each identified table should now appear in the **Objects to Be Scripted** list.
7. On the **Formatting** tab, select the following check boxes.
   - **Generate the CREATE <*object*> Command for Each Object**
   - **Generate the DROP <*object*> Command for Each Object**
8. In the **Table Scripting Options** box on the **Options** tab, clear each of the following check boxes.
   - **Script Indexes**
   - **Script Full-Text Indexes**
   - **Script Triggers**
   - **Script Primary Keys, Foreign Keys, Defaults, and Check Constraints**

   These options are configured later by the snapshot.
9. In the **File Options** box on the **Options** tab, verify that the following options are selected.
   - **International Text (Unicode)**
   - **Create One File**
10. Click **OK**.
11. In **Save As**, browse to the snapshot folder location on the subscriber/distributor, type a name for the Transact-SQL script, and then click **Save**.

    Be sure to save this script to the shared snapshot folder. This script must be accessible to the distribution agent along with the initial snapshot.

► To create Transact-SQL scripts for user-defined data types and defaults

Note: If you make use of user-defined data types, you must create a Transact-SQL script containing their definitions. This step is not necessary if you have followed the steps to create a script for tables with identity columns or timestamps. You will provide this script when configuring the publication properties, which will supply it to the subscriber/distributor so that the distribution agent can use them when creating the schema on a subscriber.

1. In SQL Server Enterprise Manager, connect to SQL Server on the publisher, and then expand **Databases**.

2. Right-click the production database, point to **All Tasks**, and then click **Generate SQL Script**.
3. In **Generate SQL Scripts**, click **Show All** on the **General** tab.
4. In **Objects to Script**, select the **All User-Defined Data Types** check boxes.
5. On the **Formatting** tab, select the following check boxes.
   - **Generate the CREATE <*object*> Command for Each Object**
   - **Generate the DROP <*object*> Command for Each Object**

   These options are configured later by the snapshot.
6. In the **File Options** box on the **Options** tab, verify that the following options are selected:
   - **International Text (Unicode)**
   - **Create One File**
7. Click **OK**.
8. In **Save As**, browse to the snapshot folder location on the subscriber/distributor, type a name for the Transact-SQL script, and then click **Save**.

   Be sure to save this script to the shared snapshot folder. This script must be accessible to the distribution agent along with the initial snapshot.

Configuring the Distributor

Transactional replication requires that a server be designated as the distributor. When you use transactional replication as a high availability solution, use a subscriber as the distributor. This reduces the load on the publisher and, in most environments, will not place too much of a load on the subscriber. Use SQL Server Enterprise Manager to configure the distributor on the selected subscriber. When you configure the distributor, you designate the server, specify a snapshot folder, create the **distribution** database, and enable publishers.

► **To configure the distributor**

1. Using SQL Server Enterprise Manager, connect to the SQL Server instance that will function as the distributor (usually a subscriber), and then click **Databases**.

   You must connect as a member of the System Administrators server role.
2. On the **Tools** menu, point to **Replication**, and then click **Configure Publishing, Subscribers, and Distribution**.
3. On the **Welcome to the Configure Publishing and Distribution Wizard** page, click **Next**.

4. On the **Select Distributor** page, click **Next** to use the server as its own distributor.

   If the SQL Server Agent service on the server uses the local system account, you will be prompted to change this account to a domain user account that is a member of the System Administrators server role. You must use a domain user account. Using the same domain user account for all participating servers makes setup and administration of transactional replication easier.

   If SQL Server Agent is not configured to start automatically on the distributor, you are asked to configure the SQL Server Agent service to start automatically. Configure the SQL Server Agent service to start automatically on all participating servers (unless you are deploying transactional replication on an MSCS cluster).
5. On the **Specify Snapshot Folder** page, type or browse to the share that you created for the snapshot folder, and then click **Next**.

   Specify a Universal Naming Convention (UNC) path for the snapshot folder rather than a drive path. Do not use the default share. This share is accessible only to members of the Administrators local group.
6. On the **Customize the Configuration** page, click **Yes, Let Me Set the Distribution Database Properties**; **Enable Publishers**; **or Set the Publishing Settings**, and then click **Next**.
7. On the **Provide Distribution Database Information** page, type or browse to the folder you want to use for the data file, type or browse to the folder you want to use for the log file, and then click **Next**.
8. On the **Enable Publishers** page, clear the check box for this subscriber, and then select the check box for the publisher in the **Registered Servers** list.

   An informational page appears describing the information you must provide to enable a publisher to use this distributor. Close the page after you have read it.
9. In **Publisher Properties** for the publisher, click **OK** to accept the default properties.
10. On the **Enable Publishers** page, click **Next**.
11. On the **Completing the Configure Publishing and Distribution Wizard** page, click **Finish**.

    The wizard configures the distributor. When complete, close the message. Review the information that appears about the Replication Monitor, and then close the message.

Publishing the Production Database

Transactional replication requires the production database to be published. Use SQL Server Enterprise Manager to configure the publisher and to publish the production database. When you configure the publisher, you designate the distributor for the publisher. When you publish the production database, you specify transactional

replication, the types of subscribers, the articles in the publication, and the name of the publication. After the production database is published, you modify the properties of the publication to use the identity property and time-stamp data type scripts that you created earlier.

► To configure the publisher

1. Using SQL Server Enterprise Manager, connect to the SQL Server instance containing the production database, and then click **Databases**.

   You must connect as a member of the System Administrators server role.
2. On the **Tools** menu, point to **Replication**, and then click **Create and Manage Publications**.
3. In **Create and Manage Publications on <*publisher*>**, click **Create Publication**.
4. On the **Welcome to the Create Publication Wizard** page, click **Next**.

   Do not select the **Show Advanced Options in This Wizard** check box. Advanced options are required only for publications that support updatable and transformable subscriptions. These options are not used for this high-availability solution.
5. On the **Select Distributor** page, click **Use the Following Server (the Selected Server Must Already Be Configured as a Distributor)**, click the server you configured as the distributor, and then click **Next**.

► To begin publishing the production database

1. On the **Choose Publication Database** page, select the production database in the **Databases** list, and then click **Next**.
2. On the **Select Publication Type** page, click **Transactional Publication**, and then click **Next**.
3. On the **Specify Subscriber Types** page, ensure that only the **Servers Running SQL Server 2000** check box is selected, and then click **Next**.

   When transactional replication is used as a high-availability solution, all participating servers should be running the same version of SQL Server.
4. In the **Object Type** list on the **Specify Articles** page, select the **Publish All** check box for **Table**, the **Publish All** check box for **Stored Procedures**, and the **Publish All** check box for **Views**.

► To modify table article defaults

1. Click **Article Defaults**.
2. In **Default Article Type**, click **Table Articles**, and then click **OK**.

3. On the **Snapshot** tab in **Default Table Article Properties**, modify the properties as follows:
   a. Click the **Keep the Existing Table Unchanged** option button.
   b. Select the **Include Declared Referential Integrity** check box.
   c. Select the **Cluster Indexes** check box (the default).
   d. Select the **User Triggers** check box.
   e. Select the **Extended Properties** check box.
   f. Select the **Collation** check box.
   g. Clear the **Convert User-Defined to Base Data Types** check box.
   h. Click **OK**, and then click **Yes** to update all published articles with these new defaults.
4. On the **Specify Articles** page, click **Next**.

► **To finish publishing the production database**

1. On the **Article Issues** page, review the warnings, and then click **Next**.
2. On the **Select Publication Name and Description** page, type a unique descriptive publication name and description, and then click **Next**.
3. On the **Customize the Properties of the Publication** page, click **No, Create the Publication as Specified**, and then click **Next**.
4. On the **Completing the Create Publication Wizard** page, click **Finish**.

   The wizard configures the publisher and publishes the production database.

   If you created custom Transact-SQL scripts for objects with the identity property or time-stamp data types, do not click **Close** when finished.

► **To modify the publication properties to use the custom Transact-SQL scripts for identity property, time-stamp data types, and user-defined data types**

1. After the publication is created, click **Publication Properties** if you created custom Transact-SQL scripts for objects with the identity property or time-stamp data type; otherwise, click **Close**.
2. In **Additional Scripts** on the **Snapshot** tab, in the **Before Applying the Snapshot, Execute This Script** text box, type or browse to the custom script you created and saved earlier to the snapshot folder, and then click **OK**.

   Be sure you use a UNC path to the snapshot folder. If you use a drive path that is local to the subscriber/distributor, the distribution agent, which runs on the subscriber/distributor, will not be able to access and execute this script.
3. In **Create and Manage Publications**, click **Close**.

Configuring and Initializing Subscribers

After the distributor and publisher are configured and the production database is published, you are ready to configure the subscribers. Configure a push subscription with continuously running distribution agents to minimize the transactional latency between the publisher and each subscriber. Use SQL Server Enterprise Manager to connect to the publisher and to configure the push subscriptions. When you configure a subscriber, you specify the publication, the subscription database, and the distribution schedule. You also initialize the subscription database by creating and applying the initial snapshot of the production database. Finally, if you manually created tables to preserve columns with a time-stamp data type, you modify the insert and update stored procedures on the subscriber to properly handle the time-stamp data.

► **To configure a subscriber**

1. Using SQL Server Enterprise Manager, connect to the SQL Server instance containing the production database, and then click **Databases**.

   You must connect as a member of the System Administrators server role.
2. On the **Tools** menu, point to **Replication**, and then click **Push Subscriptions to Others**.
3. In **Create and Manage Publications on <*publisher*>**, expand the production database in the **Databases and Publications** list, select the publication of that database, and then click **Push New Subscription**.
4. On the **Welcome to the Push Subscription Wizard** page, click **Next**.

   Do not select the **Show Advanced Options in This Wizard** check box. Advanced options are required only for updatable and transformable subscriptions, which are not used when transactional replication is implemented as a high-availability solution.
5. On the **Choose Subscribers** page, select one or more subscribers from the **Subscribers** list, and then click **Next**.
6. On the **Choose Destination Database** page, click **Next** to use the same database name as the production database.

   To ensure application compatibility in the event of a role change, the name of the subscription database must be identical to the name of the production database.
7. On the **Set Distribution Agent Schedule** page, click **Continuously**, and then click **Next**.

   This option ensures minimal latency between committed transactions at the publisher and their propagation to the subscriber.

8. On the **Initialize Subscription** page, click **Yes, Initialize the Schema and Data**; select the **Start the Snapshot Agent to Begin the Initialization Process Immediately**; and then click **Next**.

   Doing so locks the production database while the snapshot is being created. If you want to wait to create the snapshot, do not select the check box to start the process immediately. After you finish the wizard, you can schedule the snapshot agent to run at a specific time. Select a time that minimizes the impact on production database users.

9. On the **Start Required Services** page, click **Next**.

   The SQL Server Agent on the server functioning as the distributor must be running for the push subscription to run.

10. On the **Completing the Push Subscription Wizard** page, click **Finish**.

    The wizard configures the subscribers and creates the push subscriptions. Click **Close** when finished, and then close **Create and Manage Publications**.

Note: During the initialization of subscriptions for large publication, you may see the following error in the Replication Monitor: "The agent is suspect. No activity reported within the last 10 minutes." This error occurs due to the length of time required to create the script of the production database, initialize the subscription database, and insert the data into the subscription database on the subscriber. You can ignore this error.

► To modify insert procedures for time-stamp data types

1. Using SQL Query Analyzer, connect to the subscription database on the subscriber and use the following Transact-SQL script to report which tables have time stamps:

```
SELECT O.name FROM sysobjects O
    WHERE o.type='U' AND objectproperty (o.id, 'TableHasTimestamp') = 1
    ORDER BY O.NAME
```

2. Using SQL Enterprise Manager, connect to the subscription database on the subscriber and open each insert stored procedure that needs to be changed.

   The name of each insert stored procedure is sp_MSins_<*tablename*>.

3. Locate the VALUES clause of the **INSERT** statement. Replace the time-stamp parameter with the DEFAULT constant.

   The time-stamp parameter is a variable with a data type of binary (8). The position of this parameter in the **INSERT** statement is different for different tables.

   This change preserves the time-stamp value when the data is inserted into the table on the subscriber. The following Transact-SQL script illustrates this change. The original VALUES clause is commented out and replaced with a modified VALUES clause.

```
CREATE PROCEDURE sp_MSins_SpecialReplText
@c1 int,@c2 uniqueidentifier,@c3 binary(8)
AS
  BEGIN
    INSERT INTO SpecialReplText(ID, UID, TS)
    –VALUES (@c1, @c2, @c3)
    – @c3 is time stamp parameter
    VALUES
    (@c1, @c2, DEFAULT)
    – Removed parameter @c3, substituted DEFAULT
  END
```

4. Save the procedure.
5. Repeat this process for each insert stored procedure that needs to be changed.
6. Using SQL Enterprise Manager, script out all the changed stored procedures for use during a role change. Apply the script file to all subscribers.

Creating and Executing a Login Synchronization Job

Important: Apply the changes specified in article Q310882, "BUG: sp_resolve_logins Stored Procedure Fails If Executed During Log Shipping Role Change " in the Microsoft Knowledge Base at *http://support.microsoft.com/default.aspx?scid=kb;en-us;Q310882* to each server before proceeding. The article discusses how you must edit the stored procedure sp_resolve_logins in the master database. Locate the following text in the procedure:

```
SELECT   *
INTO     #sysloginstemp
FROM     syslogins
WHERE    sid = 0x00
```

Replace it with the following text:

```
SELECT   *
INTO     #sysloginstemp
FROM     master.dbo.syslogins
WHERE    sid = 0x00
```

After you have set up transactional replication, you must synchronize logins between the publisher and each subscriber.

You must synchronize logins before clients can use the subscription database. Synchronizing logins is a manual process requiring you to transfer logins from the publisher to each subscriber, export login information from the syslogins table on the publisher to a file, and synchronize logins and user accounts on the subscriber.

System tables containing logins, passwords, database permissions, users, and user roles do not participate in replication. The procedures below create a login synchronization job for each subscriber to ensure that the logins remain synchronized between the publisher and the subscriber, and SQL Server logins remain

synchronized with user accounts. Run this job when you initially set up each subscriber and each time you change logins on the publisher. Server Roles are not synchronized between servers by these procedures, so all users in system roles should be managed manually at each server using a script created by you or directly in Enterprise Manager.

► **To create a transfer logins task in the transfer logins DTS package**

Note: If you have created a login for SQL Server by using a local user account in Windows 2000, this transfer logins task will fail because a local login is not valid on any other server. In a high-availability environment, only create logins for domain users, or create SQL Server logins.

1. Using SQL Server Enterprise Manager, connect to SQL Server on the publisher.
2. Right-click **Data Transformation Services**, and then click **New Package**.
3. On the **Task** menu, click **Transfer Logins Task**.
4. On the **Source** tab in **Transfer Logins Properties**, browse to or type the name of the publisher in the **Source Server** box, and verify that **Use Windows Authentication** is selected.
5. On the **Destination** tab, browse to or type the name of a subscriber in the **Destination Server** box, and verify that **Use Windows Authentication** is selected.
6. On the **Logins** tab, click **Logins for Selected Databases**.
7. In the **Databases** list, select the check box for the production database, and then click **OK**.

   Do not close the transfer logins DTS package.

► **To create a connection to the subscriber in the transfer logins DTS package**

1. On the **Connection** menu, click **Microsoft OLE DB Provider for SQL Server**….
2. In **Connection Properties**, name the new connection <*subscriber*>.
3. In the server list, click the subscriber to which you are transferring the logins, and verify that **Use Windows Authentication** is selected.
4. In the **Database** list, select the database you are replicating, and then click **OK**.

   Do not close the DTS package.

► **To create a Copy SQL Server Objects task in the transfer logins DTS package**

1. On the **Task** menu, click **Copy SQL Server Objects Task**.
2. On the **Source** tab, type **Copy users** for a description.
3. On the **Source** tab in **Copy SQL Server Objects Task Properties**, in the **Server** list, select the publishing server.
4. On the **Source** tab, verify that **Use Windows Authentication** is selected.
5. On the **Source** tab in **Copy SQL Server Objects Task Properties**, in the **Database** list, select the database being published.

6. On the **Destination** tab in **Copy SQL Server Objects Task Properties**, in the **Server** list, select the subscriber server.
7. On the **Destination** tab, browse to or type the name of a subscriber in the **Destination Server** box, and verify that **Use Windows Authentication** is selected.
8. On the **Destination** tab in **Copy SQL Server Objects Task Properties**, in the **Database** list, select the subscriber database.
9. On the **Copy** tab, select the **Create destination objects** and **Drop destination objects first** check boxes. Clear all other check boxes on the Copy tab.
10. Click **Select objects**.
11. On the **Select Objects** page, click **Uncheck**, and then click **OK**.
12. Click **Options**, and then on the **Options** page, select the **Copy database users and database roles**, **Copy object-level permissions**, and **Use quoted identifiers when copying objects** check boxes. Click **OK** twice.
13. Click the **Transfer Logins Task**, press the **Ctrl** key, and then click the **Copy Users**.
14. On the **Workflow** menu, click **On Success**.

    Do not close the transfer logins DTS package.

► **To create an Execute SQL task in the transfer logins DTS package**

1. On the **Task** menu, click **Execute SQL Task**.
2. In Execute SQL Task Properties, type Resolve Logins in the Description text box.
3. Verify that the **Existing connection** list displays the name of the connection created in the preceding procedure.
4. In the **SQL Statement** box, type the following Transact-SQL script:

```
sp_resolve_logins '<database name>', '\\<publisher>\<share name of
synchronization folder>', 'syslogins.dat'
```

5. Click **OK**.
6. Click **Copy Users Task**, press the **Ctrl** key, and then click **Resolve Logins**.
7. On the **Workflow** menu, click **On Success**.

   A green-and-white striped workflow arrow appears that points toward the Resolve Logins task.

   Do not close the DTS package.

► **To save the transfer logins DTS package**

1. On the **Package** menu, click **Save As**.
2. In **Save DTS Package**, type **Transfer_Logins_<*subscriber*>** in the **Package Name** box, and then click **OK**.

   The transfer logins package is saved to the **msdb** database on the publisher.
3. Close the transfer logins DTS package.

Note: Create a separate transfer logins package on the publisher for each subscriber.

► To create a login synchronization job for a subscriber

1. Using SQL Server Enterprise Manager, connect to the publisher.
2. Expand **Management**, expand **SQL Server Agent**, right-click **Jobs**, and then click **New Job**.
3. On the **General** tab in **New Job Properties**, type **Synchronize Logins <*subscriber*>** in the **Name** box.
4. In the **Owner** list, select the domain user account that the SQL Server Agent service uses.

► To create a BCP job step for the synchronize logins job

1. On the **Steps** tab, click **New**.
2. On the **General** tab in **New Job Step**, type **BCP Out** in the **Step Name** box.
3. In the **Type** list, select **Operating System Command (CmdExec)**.
4. In the **Command** text box, type the following command as follows:

```
BCP master..syslogins out \\<publisher>\<share name of synchronization
folder>\syslogins.dat /N /S <publisher> /T
```

Note: If the share name of the synchronization folder has a space in it, you must put double quotation marks around the entire path for the output file.

5. Click **OK**.

► To create a transfer logins job step for the synchronize logins job

1. On the **Steps** tab, click **New**.
2. On the **General** tab in **New Job Step**, type **Transfer Logins <*subscriber*>** in the **Step Name** box.
3. In the **Type** list, select **Operating System Command (CmdExec)**.
4. In the **Command** box, type the following command:

```
DTSRun /S<publisher> /E /NTransfer_Logins_<subscriber>
```

5. Click **OK**.
6. Click **OK** to close the synchronize logins job.

Note: Create a separate synchronize logins job on the publisher for each subscriber.

► **To execute the synchronize logins job for a subscriber**

1. Using SQL Server Enterprise Manager, connect to the publisher.
2. Expand **Management**, expand **SQL Server Agent**, and then expand **Jobs**.
3. Right-click **Synchronize Logins <*subscriber*>**, and then click **Start Job**.
4. In **Start Job on <*publisher*>**, click **Start** to execute all steps in the synchronize logins job.

Note: Execute the synchronize logins job for each subscriber whenever you change logins on the publisher.

Scripting and Copying Jobs, Alerts, and Operators

The **msdb** database, not the production database, stores SQL Server jobs, alerts, and operators. If you have jobs, alerts, or operators that run on the publisher on which your production database relies, manually copy these jobs, alerts, and operators to each subscriber. You cannot simply back up the **msdb** database and restore it to a subscriber. Doing so overwrites all jobs, alerts, and operators in the **msdb** database on the subscriber, including all of the transactional replication jobs on that subscriber. You can copy jobs, alerts, and operators by using SQL Server Enterprise Manager to generate scripts for them and then applying the scripts to each subscriber. You must complete these tasks before users connect to a subscriber that is promoted to become the new primary server.

Use the following procedures to copy jobs, alerts, or operators that the production database needs to each subscriber.

► **To script all jobs on the publisher**

1. In SQL Server Enterprise Manager, connect to SQL Server on the publisher, expand **Management**, and then expand **SQL Server Agent**.
2. Right-click **Jobs**, point to **All Tasks**, and then click **Generate SQL Script**.
3. In **Generate SQL Script**, browse to or type the path to the synchronization folder on the publisher and then type a file name for the jobs script using the extension **.sql**.
4. Verify that the **International Text (Unicode)** check box is selected in the **File Format** box and that the **Replace Job If It Exists** check box is selected in the **SQL Generation Options** box.
5. Accept the default for the **TSQL Batch Separator**.
6. Click **OK** to create a Transact-SQL script for all jobs on the publisher.
7. Use SQL Query Analyzer to edit the jobs script to remove all transactional replication jobs.

► To execute the jobs script on each subscriber

1. Using SQL Query Analyzer, connect to SQL Server on a subscriber.
2. On the **File** menu, click **Open**.
3. In **Open Query File**, type or browse to the jobs script in the shared synchronization folder on the publisher, and then open the jobs script.
4. Execute the jobs script, and then close SQL Query Analyzer.
5. In SQL Server Enterprise Manager, connect to the appropriate instance on the subscriber, expand **Management**, expand **SQL Server Agent**, and then click **Jobs**.
6. In the details pane, review the jobs that were created. Disable those jobs until they are needed after a role change.

Note: Repeat this procedure on each subscriber.

► To script all alerts on the publisher

1. In SQL Server Enterprise Manager, connect to SQL Server on the publisher, expand **Management**, and then expand **SQL Server Agent**.
2. Right-click **Alerts**, point to **All Tasks**, and then click **Generate SQL Script**.
3. In **Generate SQL Script**, browse to or type the path to the synchronization folder on the publisher and then type a file name for the alerts script using the extension **.sql**.
4. Verify that the **International Text (Unicode)** check box is selected in the **File Format** box
5. Verify that the following check boxes are selected in the **SQL Generation Options** box.
   - **Replace Alert If It Exists**
   - **Include the Name of the Job Executed by the Alert**
6. Clear the **Include Notifications Sent by the Alert to the Operator** check box.

   Note: Notifications are copied when the operators are copied to ensure that the operators will exist prior to creating the notification.

7. Accept the default for the **TSQL Batch Separator**.
8. Click **OK** to create a Transact-SQL script for all alerts on the publisher.

   Note: A bug in the scripting routine exports all Net Send notifications as E-mail notifications. If you use Net Send in any notifications, edit the script file and change the notification method in each **sp_add_notification** statement from **@notification_method =1** to **@notification_method =4** to configure the notification to use Net Send.

Note: If any alerts use user-defined messages, you must create those messages on the subscriber before applying this script. It is recommended that you always use a script to create user-defined messages so you can recreate the messages on any SQL Server instance.

► To execute the alerts script on each subscriber

1. Using SQL Query Analyzer, connect to SQL Server on a subscriber.
2. On the **File** menu, click **Open**.
3. In **Open Query File**, type or browse to the alerts script in the shared synchronization folder on the publisher, and open the alerts script.
4. Execute the alerts script, and then close SQL Query Analyzer.
5. In SQL Server Enterprise Manager, connect to the appropriate instance on the subscriber, expand **Management**, expand **SQL Server Agent**, and then click **Alerts**.

   In the details pane, review the alerts that were created.

Note: Repeat this procedure on each subscriber.

► To script all operators on the publisher

1. In SQL Server Enterprise Manager, connect to SQL Server on the publisher, expand **Management**, and then expand **SQL Server Agent**.
2. Right-click **Operators**, point to **All Tasks**, and then click **Generate SQL Script**.
3. In **Generate SQL Script**, browse to or type the path to the synchronization folder on the publisher and then type a file name for the operators script using the extension **.sql**.
4. Verify that the **International Text (Unicode)** check box is selected in the **File Format** box.
5. Verify that the following check boxes are selected in the **SQL generation options** box.
   - **Replace Operator If It Exists**
   - **Include Notifications Sent by Alert to the Operator**
6. Accept the default for the **TSQL Batch Separator**.
7. Click **OK** to create a Transact-SQL script for all operators on the publisher.

► To execute the operators script on each subscriber

1. Using SQL Query Analyzer, connect to SQL Server on a subscriber.
2. On the **File** menu, click **Open**.

3. In **Open Query File**, type or browse to the operators script in the shared synchronization folder on the publisher, and open the operators script.
4. Execute the operators script, and then close SQL Query Analyzer.
5. In SQL Server Enterprise Manager, connect to the appropriate instance on the subscriber, expand **Management**, expand **SQL Server Agent**, and then click **Operators**.
6. In the details pane, review the operators that were created.

Note: Repeat this procedure on each subscriber.

Copying DTS Packages

DTS packages can be stored in the **msdb** database or in the file system. If you have DTS packages that run on the publisher on which your production database relies, you must manually copy these DTS packages to each subscriber. You cannot simply back up the **msdb** database and restore it to a subscriber. Doing so overwrites all jobs, alerts, operators, and DTS packages in the **msdb** database on the subscriber. You can open DTS packages saved to the **msdb** database on the publisher and save them to each subscriber. For DTS packages saved to the file system on the publisher, simply copy these DTS packages to a folder on each subscriber. This task must be completed before users connect to a subscriber that is promoted to become the new primary server.

Use the following procedure to copy DTS packages that the production database needs to each subscriber.

► **To copy DTS packages**

1. For each DTS package stored in the file system on the publisher, copy the DTS package to each subscriber by using Windows Explorer.

   When copying DTS packages to each subscriber, use the same drive and path as the publisher is using. Doing so will eliminate potential path problems.
2. For each DTS package stored in the **msdb** database on the publisher, use SQL Server Enterprise Manager to open the DTS package, and then save it to the **msdb** database on each subscriber.

Note: To ensure that DTS packages execute properly on a subscriber, use an alias or the dynamic properties task. For more information, see "Redirecting Client Network Traffic to a Promoted Secondary Server" in Planning Guide Chapter 5, "Minimizing Downtime by Using Redundant Components."

Changing the Role of a Subscriber

To change the role of a subscriber when the publisher fails or is taken offline, disable all replication jobs, and then use the subscription database as the new production database. Ensure that all logins, jobs, alerts, operations, and DTS packages are synchronized and then redirect clients to the promoted subscriber. For more information about facilitating this client redirection task, see Planning Guide Chapter 5, "Minimizing Downtime by Using Redundant Servers."

In a planned role change, pause all applications that use the production database, and then place the production database in single user mode. Pausing the production database helps ensure that the subscription database is transactionally current with the production database. Pausing the production database fails if applications do not disconnect from the database after each transaction. Be sure you don't have open connections in Enterprise Manager, Query Analyzer, or other utilities that would cause the pause procedure to fail.

In an unplanned role change, recent transactions that are not replicated before the publisher fails are lost.

► **To pause applications that use the production database**

1. Disconnect applications from the production database if the database is still available.

   The method for doing so varies by application. In a planned failover, allow open transactions to complete, but do not allow new ones to start.
2. If you implemented NLB to direct traffic to a specific server, issue a **WLBS** command to ensure that all transactions have completed:

```
WLBS DrainStop <NLB cluster name>:<publisher> /PASSW <password>
```

Note: For more information about NLB, see Solutions Guide Chapter 4, "Implementing Network Load Balancing."

► **To place the production database in single user mode from a command prompt**

- Put the following command into a batch file so that it is ready when you need it:

```
OSQL -Q "ALTER DATABASE <production database> SET SINGLE_USER WITH ROLLBACK
IMMEDIATE" -S<publisher> -d master -E
```

► **To stop transactional replication jobs on the publisher**

1. Using SQL Server Enterprise Manager, connect to SQL Server on the publisher, right-click **Replication**, and then click **Disable Publishing**.

2. On the **Welcome to the Disable Publishing and Distribution Wizard for** <*server*>page, click **Next**.
3. On the **Disable Publishing** page, click **Yes, Disable Publishing on** <*publisher*>, and then click **Next**.
4. On the **Confirm Dropping of Publications** page, click **Next** to drop all publications.
5. On the **Completing the Disable Publishing and Distribution Wizard** page, click **Finish**.
6. Click **OK** to acknowledge that transactional replication was disabled on the publisher.

► **To stop transactional replication jobs on the subscriber/distributor**

1. Connect to the secondary server instance of SQL Server on the subscriber/distributor, right-click **Replication**, and then click **Disable Publishing and Distribution**.
2. On the **Welcome to the Disable Publishing and Distribution Wizard** page, click **Next**.
3. On the **Disable Publishing and Distribution** page, click **Yes, Disable Distribution (and Publishing) on** <*subscriber/distributor*>, and then click **Next**.
4. On the **Confirm Disabling Remote Publishers** page, click **Next** to disable all publishers.
5. On **Completing the Disable Publishing and Distribution Wizard**, click **Finish**.
6. Click **OK** to acknowledge that transactional replication was disabled on the subscriber/distributor.

Note: The former subscriber is now the primary server. To maintain the database redundancy protection provided by transactional replication, configure the new primary server as a publisher with a subscriber/distributor and additional subscribers as appropriate.

Reverting to the Original Publisher

Reverting to the original publisher is unnecessary if the performance of new publisher equals the performance of the old publisher. It might make sense to leave the system operating the way it is rather than interrupt operations with another role change procedure. Your recovery plan should indicate which option you have selected. If your recovery plan calls for reverting to the original publisher, follow the procedural steps above for a planned role change. Note that you can re-use the Transact-SQL scripts you created to manage tables with identity and time-stamp data types — and the modified replication stored procedures used for replication to tables with time stamps. You do not need to recreate them.

4

Implementing Network Load Balancing

This chapter teaches you how to increase the availability of a data center by creating a Microsoft Network Load Balancing (NLB) cluster. After reading this chapter, you will be able to create an NLB cluster to work with secondary servers maintained by using log shipping, transactional replication, or database backups.

The implementation steps include annotation to help you understand the selected configuration choices. The configuration options presented in this chapter have been designed by using input from the field, reviewed thoroughly by Microsoft Product Support Services (PSS), and tested by an independent team to ensure accuracy and functionality.

Understanding Network Load Balancing

Network Load Balancing is a component of Windows clustering technology that comes with Microsoft® Windows® 2000 Advanced Server and Microsoft Windows 2000 Datacenter Server. An NLB cluster provides a way to provide a single server name and an IP address through which clients connect to one or more servers. You usually use NLB to automatically distribute (load balance) client requests across a number of servers providing the same service to the client, such as Internet Information Services (IIS). With a production database that allows updates however, only one server can be active.

In a high-availability environment, you create an NLB cluster to easily route client requests from the original primary server to a promoted secondary server without having to update each client directly. Although NLB requires you to manually

reconfigure the NLB cluster to point clients to a secondary server, manually modifying the NLBis much faster than manually changing the connection information for each client.

You can use NLB with a standby server that participates in log shipping, with a subscriber that participates in transactional replication, or with a secondary server that contains a database restored from backup. NLB setup is simple and switching to an alternative server is quick; however, NLB requires that each server that participates in an NLB cluster be on the same subnet.

To implement an NLB solution, you need to have certain things in place:

- You need the network administrator.
- You need the Windows 2000 Advanced Server or Windows 2000 Datacenter Server. Microsoft SQL Server™ is not required for NLB, although you must install SQL Server on each server in the NLB cluster as part of the overall high availability solution.
- You need systems and system components that are certified to work together and with the chosen Microsoft Windows 2000 operating system. Windows 2000 Datacenter Server has more stringent certification requirements than Windows 2000 Advanced Server. To find systems and system components that are certified by Microsoft, search the "Hardware Compatibility List" on the Microsoft Web site at *http://www.microsoft.com/hcl/search.asp*.

For more information about NLB, see "Microsoft Clustering Solutions" on the Microsoft TechNet Web site at *http://www.microsoft.com/technet/treeview/default.asp?url=/TechNet/columns/tips/w2kclust.asp*.

Also see "Network Load Balancing Technical Overview" on the Microsoft TechNet Web site at *http://www.microsoft.com/technet/treeview/default.asp?url=/TechNet/prodtechnol/windows2000serv/deploy/confeat/nlbovw.asp*.

Deploying NLB

You will successfully implement an NLB cluster if you avoid shortcuts and parallel installations. If the setup program fails or you cancel the setup program, remove any Domain Name System (DNS) entries that relate to the failed setup before you continue. Also, verify that the server name and IP address you plan to use for NLB are unused immediately before installing NLB. You can check for conflicts by pinging the service name and proposed IP address.

This chapter assumes the following with respect to each participating server in the NLB solution:

- Windows 2000 Advanced Server or Windows 2000 Datacenter Server is installed.
- The most recent operating system service pack is installed.
- NLB has not been installed.
- The server is a member of the same Microsoft Active Directory® service forest as each other server.
- At least one network interface card (NIC) is installed.
- The server is connected to the corporate network and the server may be connected another server on a private network to expedite log shipping or replication traffic.
- The server is on the same subnet as each other server.
- SQL Server 2000 is installed, although NLB does not require its installation.

The logical design of the NLB cluster built in this chapter is described below, followed by the detailed steps required to build this high-availability solution.

Understanding the Logical Design

Figure 4.1 shows the logical design of an NLB cluster. The design shows a secondary server in the local site and a secondary server in a remote site. You need not implement both secondary servers. The remote site provides redundancy in case the local site becomes unusable. The virtual local area network (VLAN) connecting the local and remote sites allows NLB NICS to be on the same subnet, which is a requirement for an NLB operation.

Client traffic and NLB management traffic use the same network. You can use a second network within the local site for log shipping or replication traffic between servers to reduce the load on the public network within the local site. When you deploy a remote site, you must ensure that the connection to the remote site supports sufficient bandwidth for all network traffic between the sites.

Clients use the NLB cluster address to connect to SQL Server. In the configuration shown in Figure 4.1 on the next page, the NLB cluster address is the x.y.z.5 IP address. NLB is configured so that only the current primary server responds to the cluster address. When a secondary server is promoted to become the primary server, use NLB management commands to change the server that responds to client requests directed to the NLB cluster address.

Figure 4.1
Logical design diagram of an NLB cluster

On each server (node) that will be part of the NLB cluster, an instance of Windows 2000 Advanced Server or Windows 2000 Datacenter Server is installed. The steps required to implement this logical design are below, followed by the steps in detail.

► **To implement an NLB cluster**

1. Obtain an allocation of a public IP address and server name for the NLB cluster, and determine the port to be used by SQL Server. The IP address must be on the same subnet as each node in the cluster.
2. Configure the NLB NIC, and then install and configure NLB on each node participating in the NLB cluster.
3. Enable the primary server.

Obtaining the Public IP Address, Server Name, and Port

Before configuring an NLB cluster, obtain a server name, IP address, and port number for the NLB cluster.

- **Server name —** Have the IT department approve the NLB cluster name.
- **IP address —** Have the IT department allocate a static IP address for the NLB cluster and its subnet mask. If you have not yet allocated a static address for the NLB NIC, obtain an allocation for its IP address and subnet mask.
- **Port number —** With the network specialist, determine the port to be used by the SQL Server instances that the NLB cluster will support. A default instance of SQL Server uses port 1433. Each named instance of SQL Server requires a unique number. Each standby instance of SQL Server must use the same port number as its corresponding primary instance in the cluster.

You will use this information when configuring the NLB cluster.

Configuring the NLB NIC and Installing and Configuring NLB

Before you install NLB on each node, you must configure the IP address for the NIC that you plan to use for NLB communication. If you have multiple NICs in the server, choose the NIC connected to the corporate network. You will install NLB on only one NIC in the server. If the NLB cluster is geographically dispersed and you are using a virtual private network (VPN) to connect the sites, you must also configure the default gateway and preferred DNS server for the NIC.

After you configure the NLB NIC, install and configure NLB on the NLB NIC. Use the same cluster parameters for each node in the NLB cluster. Use a unique host IP address for each node, but be sure each node is on the same subnet. Use port rules to determine how traffic will be distributed across the nodes in the cluster. For SQL Server databases that are not read-only, you must configure the port rules so that only one server (the current primary server) will handle all network traffic for SQL Server.

Before you begin, log on by using an account that is a member of the Administrators local group.

► **To configure the NLB NIC**

1. On the desktop, right-click **My Network Places**, and then click **Properties**.
2. In **Network and Dial-up Connections**, right-click the NIC you plan to use for NLB, click **Rename**, and then type **NLB NIC**.
3. In **Network and Dial-up Connections**, right-click **NLB NIC**, and then click **Properties**.
4. In **NLB NIC Properties**, click **Internet Protocol (TCP/IP)**, and then click **Properties**.
5. In **Internet Protocol (TCP/IP) Properties**, type the IP address allocated to the NLB NIC, and then change the subnet mask if your network uses subnetting.
6. Type the IP address of the default gateway. (Your network administrator defines this address.)
7. Click **Use the Following DNS Server Addresses**, and then type the DNS server address. (Your network administrator defines this address.)
8. Click **OK** to save the new TCP/IP properties.
9. Click **OK** to save the new connection properties for the NLB NIC.

► **To install NLB on a node**

1. On the desktop, right-click **My Network Places**, and then click **Properties**.
2. In **Network and Dial-up Connections**, right-click **NLB NIC**, and then click **Properties**.
3. In **NLB NIC Connection Properties**, click **Network Load Balancing**.

4. In **Select Network Component Type**, click **Service**, and then click **Add**.
5. In **Select Network Service**, click **Network Load Balancing**, and then click **OK**.
6. After NLB is installed, click **Close**.

► **To configure the NLB cluster**

1. On the desktop, right-click **My Network Places**, and then click **Properties**.
2. In **Network and Dial-up Connections**, right-click the connection for the NLB NIC, and then click **Properties**.

   If **Properties** is unavailable, select **Network Load Balancing**, and then click **Properties**.
3. In **Network Load Balancing Properties**, use the following information to complete the information requested on the **Cluster Parameters** tab:
   a. **Primary IP address —** This is the cluster address. The cluster address is the same for all the NICs participating in the NLB cluster. Type the allocated address obtained from the IT department for the NLB cluster.
   b. **Subnet mask —** Change the automatically calculated subnet mask if your network uses subnetting.
   c. **Full Internet name —** Use the server name allocated for the NLB cluster plus the full domain name.
   d. **Multicast support —** Select this check box only if you are using a single NIC for all traffic on this server. If you use a second NIC to support log shipping or transactional replication traffic, do not select multicast support.
   e. **Remote password —** Type a password to control NLB on this node from another node.
   f. **Confirm password —** Repeat the remote password.
   g. **Remote control —** Select this check box to allow you to control NLB on this node from another node.
4. Use the following information to complete the information requested on the **Host Parameters** tab:
   a. **Priority (unique host ID) —** Specify a unique number between 1 and 32.

      The node with the lowest number handles all client traffic not assigned by port rules.
   b. **Initial cluster state —** Do not select the **Active** check box for a normal SQL Server scenario. Select the **Active** check box if all SQL Server nodes are read-only.
   c. **Dedicated IP address —** Type the allocated address obtained from the IT department for the node.
   d. **Subnet mask —** Type the appropriate subnet mask for your network.

5. Use the following information to complete the information requested on the **Port Rules** tab:
   a. **Port range —** Specify the port range for the SQL Server instances this NLB cluster will support. If no other services on the server are using NLB, specify a range of 0 through 65,535. If other services on the server are using NLB, configure each service with its own port range and settings.
   b. **Protocols —** Click **Both** to enable both User Datagram Protocol (UDP) and Transmission Control Protocol (TCP).
   c. **Filtering mode —** Click **Single Host** if you use a dedicated NIC for the NLB traffic or click **Multiple Hosts** if you use the NIC for all traffic to the server.
   d. **Affinity —** Click **None** for filtering mode **Single Host**, or **Single for Multiple Hosts**.
6. Click **OK** to close **Network Load Balancing Properties**.
7. Click **OK** to close **Connection Properties**.

► **To add the cluster IP address to the NLB NIC**

1. On the desktop, right-click **My Network Places**, and then click **Properties**.
2. In **Network and Dial-up Connections**, right-click **NLB NIC**, and then click **Properties**.
3. In **Connection Properties**, right-click **Internet Protocol (TCP/IP)**, and then click **Properties**.
4. In **Internet Protocol (TCP/IP) Properties**, click **Advanced**.
5. On the **IP Settings** tab in **Advanced TCP/IP Settings**, in **IP Addresses**, click **Add**.
6. In **TCP/IP Addresses**, type the NLB cluster IP address and subnet mask, and then click **Add**.

   Note: The cluster IP address must come *after* the dedicated IP address in the list.

7. Click **OK** to close **Advanced TCP/IP Settings**.
8. Click **OK** to close **Internet Protocol (TCP/IP) Properties**.
9. Click **OK** to close **Connection Properties**.

Note: Repeat these procedures on each node participating in the NLB cluster.

After you have configured all the nodes, verify with your network administrator that there is a DNS entry for the virtual cluster name. If you are using the cluster IP address instead of the cluster name when connecting to SQL Server, the network administrator should add the cluster IP address and name to the DNS reverse lookup table.

Enabling the Primary Server Node

After you have configured the NLB cluster on all nodes participating in the NLB cluster, enable the node servicing the primary server. Do not enable any nodes servicing secondary servers. When you enable a node, a convergence process occurs that determines the servers that will respond to client traffic directed to the NLB cluster name and IP address. This convergence process evaluates the nodes that are enabled and the NLB configuration settings for the cluster.

► **To enable the primary server node**

1. Open a command prompt.
2. Type **WLBS Start** *clustername*:*primary node name* /PASSW <*password*>.
3. Click **Enter**.
4. Close the command prompt.

Changing the Primary Server with NLB

NLB supports automatic and manual failover modes to change the server that receives traffic directed to the NLB cluster's virtual IP address. The automatic failover mode is designed to be used when NLB is automatically distributing client requests across a number of servers that are providing the same service to the client. When NLB is used as a part of a SQL Server high-availability solution, however, manual failover must be used.

Before changing the NLB cluster configuration to point to a secondary server, you must prepare the secondary server to receive client requests. The steps required to prepare the secondary server vary depending on the high-availability solution used with NLB.

- If you are deploying NLB with log shipping, follow the role change steps in Chapter 2 of the Solution Guides before initiating a manual failover of NLB to a standby server.
- If you are deploying NLB with transactional replication, follow the role change steps in Solution Guide Chapter 3 before initiating a manual failover of NLB to one of the transactional replication subscribers.
- If you are deploying NLB without log shipping or transactional replication, restore and recover the production database to an alternative server before initiating a manual failover of NLB.

After the secondary server is ready to receive client requests, stop all traffic through NLB to the current primary server and then enable traffic to the secondary server.

► To disable client traffic to the primary server

1. Open a command prompt.
2. Type **WLBS DrainStop <*clustername*>:<*primary node name*> /PASSW <*password*>**.
3. Click **Enter**.
4. After waiting a few moments for existing connections to finish draining, test to see whether the primary server is finished by typing **WLBS query <*clustername*> /PASSW <*password*>**.
5. Click **Enter**.
6. After the primary server has finished draining, close the command prompt.

Note: This procedure is required only if you are performing a planned failover. If the primary server has failed, skip this procedure.

► To enable client traffic to the secondary server

1. Open a command prompt.
2. Type **WLBS Start <*clustername*>:<*primary node name*> /PASSW <*password*>**.
3. Click **Enter**.
4. Close the command prompt.

Note: To point clients to the original primary server, prepare the original primary server to receive client requests, and then use the two procedures above to disable client traffic to the current primary server and to point client traffic to the original primary server.

5

Implementing Remote Mirroring and Stretch Clustering

This chapter teaches you how to increase the availability of a Microsoft® SQL Server™ data center by using remote mirroring and stretch clustering with geographically separated nodes. After reading this chapter, you will be able to configure each component in these high availability solutions, including the storage device, network, Microsoft Cluster Service (MSCS), Microsoft Distributed Transaction Coordinator (MS DTC), and Microsoft SQL Server 2000.

The implementation steps are annotated to help you understand the selected configuration choices. The configuration options presented in this chapter have been designed using input from the field, reviewed thoroughly by Microsoft Product Support Services (PSS), and tested by an independent team to ensure accuracy and functionality.

Remote mirroring and stretch clustering are only parts of achieving a highly available data center. To achieve a highly available data center, you must also follow the processes outlined in the Planning Guide.

Understanding Remote Mirroring and Stretch Clustering

Remote mirroring is a solution offered by third-party vendors that allows you to maintain a real-time mirror of your primary storage device at a remote site, and to protect your data on that mirror from site destruction. In a remote mirror, redundant server hardware and a redundant storage system are maintained at the remote site. SQL Server is installed on both the local and remote mirrors. SQL Server transactions affecting one or more production databases are concurrently written to the local storage system and the remote storage system. A remote mirroring solution ensures absolute transactional currency and consistency for

the mirrored production databases. This solution does not have any mechanism for failover for mirroring objects stored in the **master** or **msdb** database. For more information on ensuring these objects are updated from the primary site, see Solution Guide Chapter 3, "Implementing Transactional Replication."

Stretch clustering is a high-availability solution offered by third-party vendors that uses a remote mirror and that extends the capabilities of the failover clustering solution provided by SQL Server 2000 Enterprise Edition and the clustering services provided by either Microsoft Windows® 2000 Advanced Server or Microsoft Windows 2000 Datacenter Server. With stretch clustering, SQL Server transactions are concurrently written to the local storage system and the remote storage system.

Stretch clustering includes most of the benefits of standard failover clustering but also protects the data center from site destruction. Stretch clustering ensures absolute transactional currency and consistency and makes failover virtually transparent to the client. Consider using stretch clustering to protect against site destruction if the risk of site destruction is high, the cost of an unavailable data center is high, and you require transactional currency.

Stretch clustering maintains at least one standby server in an MSCS cluster in case the primary data center server fails. Windows 2000 Advanced Server supports two-server clusters, and Windows 2000 Datacenter Server supports clusters containing up to four servers. With stretch clustering, at least one standby server is in a remote site. If you use Windows 2000 Datacenter Server, you can have an MSCS cluster with two local servers and two remote servers providing server redundancy in each site and failover in the event of site destruction.

When the MSCS detects that the primary server has failed, it automatically starts the cluster resources that were running on the failed primary server on a standby node. MSCS then redirects all client traffic to the standby server. MSCS then redirects all client traffic to the standby server. MSCS does not, however, control the system area network (SAN) and cannot automatically promote the mirror of the primary SAN device. This task must be performed manually. With stretch clustering, committed transactions are always available through the standby server at a remote site after the primary server fails.

To implement a remote mirror or stretch-clustering solution, you need to have certain things in place:

- You need people in the following roles:
  - Database administrator
  - SAN storage administrator
  - Network administrator
- You need the following software:
  - Windows 2000 Server (remote mirroring solution only) or either Windows 2000 Advanced Server or Windows 2000 Datacenter Server (remote mirroring or stretch clustering)

  - Windows 2000 Service Pack 3 or later
  - SQL Server 2000 Standard Edition (remote mirroring only) or SQL Server Enterprise Edition (remote mirroring or stretch clustering)
- You need two identical SAN units capable of synchronous mirroring.
- You need systems and system components that are certified to work together and with the chosen Windows 2000 operating system edition. If you are using failover clustering at either the local site or the remote site, the systems and system components must be certified for failover clustering. Certification of systems and system components for failover clustering is more stringent than operating system certification. In addition, Windows 2000 Datacenter Server has more stringent certification requirements than does Windows 2000 Advanced Server. To find systems and system components that are certified by Microsoft, search the "Hardware Compatibility List" on the Microsoft Web site at *http://www.microsoft.com/hcl/search.asp*.

For more detailed information about failover clustering, see "SQL Server 2000 Failover Clustering" on the Microsoft Web site at *http://www.microsoft.com/SQL /techinfo/administration/2000/failoverclustering.doc*.

Deploying a Remote Mirror

You will succeed at implementing a remote mirroring solution if you avoid shortcuts. This chapter assumes the following with respect to each participating server in the remote mirroring solution:

- Windows 2000 Server, Windows 2000 Advanced Server, or Windows 2000 Datacenter Server is installed on a mirrored volume.
- Windows 2000 Service Pack 3 or later is installed.

Note: Restoring disks to a cluster in the event of SAN failure requires at least Windows 2000 Service Pack 3.

- The server is a member of the same Windows domain as each other server.
- A SAN is connected.
- All disks are formatted for New Technology File System (NTFS) and are not configured as dynamic disks.
- At least one network interface card (NIC) is installed.
- The server is connected to the corporate network and is connected to each participating server on a private network.
- No version of SQL Server is installed.

Figure 5.1 shows the logical design of a remote mirror solution, with a single server in each site. The steps required to build this high-availability solution appear later in this topic.

You could deploy a SQL Server onto an independent MSCS cluster within each site to provide redundancy within the site and to use a remote mirror to protect against site-level disasters.

Figure 5.1
Logical design diagram of a remote mirror solution

Each site in the remote mirror solution runs Windows 2000 Server, Windows 2000 Advanced Server, or Windows 2000 Datacenter Server. If an independent MSCS cluster is deployed within each site, you must use either Windows 2000 Advanced Server or Windows 2000 Datacenter Server.

A SAN is used in each site. Each SAN is connected with fibre (if the sites are separated by less than 100 kilometers (km)) or with conventional Internet Protocol (IP) circuits with fibre channel to IP gateways. At any time, one of the SANs is active, updating its own local store and forwarding updates to the other SAN (the mirror). Only the server (or cluster) in the active site is active.

For distances over 100 km, there are two fundamental problems. First, it is frequently impossible to assemble a complete, dedicated fibre circuit between the sites separated by more than 100 km. Second, because of the speed of light, the signal on the fibre takes too long to travel between sites to be used for control signaling between the SAN devices.

To overcome the lack of fibre connectivity between the two sites, you can install fibre-channel-to-IP (FC/IP) converters at both and use conventional wide area network (WAN) facilities to link the sites.

The delays introduced by using a relatively slow connection (compared to local fibre) require you to operate the SAN mirror in asynchronous mode. This means that the primary site will receive confirmation of a write before the mirror has written — or possibly even received — the data. This creates a very small window of time in which a confirmed transaction may not be committed at the secondary site. If a site failure occurs within this short window, the transaction "caught" in this window will not appear when the secondary site is brought online. SQL Server manages the transaction log well, however, and it is highly unlikely that a torn page will result from a primary SAN failure, even when it is operating in asynchronous mode.

The following procedure describes generalized steps for implementing this logical design.

► **To implement a remote mirror solution with SQL Server**

1. Set up the disk.

   Configure the SAN to allow only the SAN to connect to the server in the primary site. You must limit access to the store to prevent the disk corruption that would occur if both servers connect concurrently. In a stretch cluster, the Windows Clustering Service performs this function. If the sites have substantial latency between them, configure the SAN mirror to execute the mirroring process in asynchronous mode for better performance. Running the SAN mirror in synchronous mode ensures that all transactions are written to the primary SAN as well as to the remote SAN mirror before SQL Server receives notification that a successful write occurred. Running the SAN mirror in asynchronous mode does not ensure that all transactions are written to the remote SAN mirror before SQL Server is notified that a successful write occurred, only that a successful write occurred in the primary SAN. As a result, running in asynchronous mode exposes the data center to a slightly higher risk that a committed transaction that is successfully written to the primary SAN is not successfully written to the remote SAN mirror if the primary site fails.

   Note: For more information about setting up a SAN, see "Configuring Storage Devices" later in this chapter.

2. Install SQL Server.

   In each site, install SQL Server onto a single computer or into an MSCS cluster. Do not install your production database yet.

3. Install the production database in the active site.

   Install your production database or databases onto the drives in the SAN that are being mirrored. The SAN mirror concurrently writes the .mdf and .ldf files for the production database to the SAN in each site.

4. Initiate a SAN role reversal.

   Shut down the servers in each site, initiate a role version between the SANs according to your vendor's procedure, and then start the server or servers in the newly active site. Do not restart the servers in the primary site.
5. Configure the production database in the standby site.

   Using SQL Enterprise Manager in the newly activated site, connect to the SQL Server instance that will manage the production database. Attach the production database on the local SAN, and then shut down the server or servers.
6. Initiate a SAN role reversal.

   Shut down the servers in each site, initiate a role version between the SANs according to your vendor's procedure, and then start the server or servers in both sites.

Failing Over to a Remote Mirror

When the primary site goes down, you must activate the SAN at the alternate site, bring SQL Server online, and redirect client traffic to the new site. You must also ensure that logins, jobs, alerts, operators, and Data Transformation Services (DTS) packages are synchronized between the primary site and the standby site before you bring the standby SQL Server online.

► **To reverse the roles of the sites**

1. Shut down the primary site.
2. Reverse the roles of the SANs according to your vendor's procedure.
3. Start SQL Server in the standby site.
4. Ensure that logins, jobs, alerts, operators, and DTS packages are synchronized between the primary site and the standby site. For more information, see Chapter 3, "Implementing Transactional Replication."
5. Redirect the client network traffic to the standby site. For more information on redirecting client network traffic, see "Redirecting Client Network Traffic to a Promoted Secondary Server," in Chapter 5 of the Planning Guide.

Deploying Stretch Clustering

You will succeed at implementing stretch clustering if you avoid shortcuts and parallel installations. If the setup program fails or you cancel the setup program for any reason, completely remove the previous setup before attempting another. Remove any Domain Name System (DNS) entries relating to the failed setup. Also, verify that the service names and IP addresses you plan to use for MSCS and SQL Server are unused immediately before you install MSCS and SQL Server.

This chapter assumes the following with respect to each participating server in the failover clustering solution:

- Windows 2000 Advanced Server or Windows 2000 Datacenter Server is installed on a mirrored volume.
- Windows 2000 Service Pack 3 or later is installed.

Note: Restoring disks to a cluster in the event of SAN failure requires at least Windows 2000 Service Pack 3.

- MSCS has not been installed.
- The server is a member of the same Windows domain as each other server.
- A SAN is connected.
- All disks are formatted for NTFS and are not configured as dynamic disks.
- At least two network interface cards (NICs) are installed.
- The server is connected to the corporate network and is connected to each participating server on a private network.
- No version of SQL Server is installed.

The logical design of the stretch cluster built in this chapter is described later in this chapter, followed by the steps required to build this high-availability solution.

Understanding the Logical Design

This chapter describes how to build a two-node, three-node, or four-node stretch cluster. Figure 5.2 shows the logical design of a four-node stretch cluster.

Figure 5.2
Logical design diagram of a four-node cluster using a SAN and a SAN mirror

Each node in the Windows 2000 failover cluster runs Windows 2000 Advanced Server or Windows 2000 Datacenter Server. MSCS maintains a heartbeat between the nodes to verify that cluster resources are still alive. A dedicated network is maintained between the sites because the heartbeat NICs must all be on the same subnet. When you have geographically separate sites, you must use a private network or virtual local area network (VLAN) to link the two sites so that you can deploy the same subnet at both sites.

One or more SQL Server virtual servers run within the Windows 2000 failover cluster. The active node for the virtual server exclusively accesses and maintains the data and the log files for that virtual server.

The SAN is connected to each node by using fibre, and the distance between the sites cannot exceed 100 km. Increasing the distance between the sites increases the level of disaster protection, but at the expense of the speed and reliability of the connection.

Within the SAN, the SAN software maintains local mirrored copies of the data and log files as well as the temporary and backup files. The SAN software simultaneously sends new data to the SAN mirror at the remote site. Torn pages in the SQL Server database are a risk when deploying a stretch cluster. To mitigate the risk of a SAN causing torn pages, run the SAN in synchronous mode to ensure that writes complete in both locations before SQL Server receives acknowledgement of a successful write.

In a multiple instance cluster, each virtual SQL Server requires a separate group of disks in the SAN for the data, log, temporary, and backup files. Each virtual SQL Server requires exclusive control of its disks. MSCS assigns control to the active node for a virtual SQL Server.

The generalized steps required to implement this logical design are below, followed by the steps in detail.

► **To implement a failover cluster with SQL Server**

1. Create and configure domain user accounts, determine unique server and instance names, and obtain an allocation of corporate IP addresses.
2. Configure the network.
3. Configure the storage devices.
4. Install and configure MSCS.
5. Upgrade MS DTC.
6. Install and configure a SQL Server 2000 virtual server instance.
7. Apply the most recent SQL Server 2000 service pack.
8. Perform additional configuration tasks.

Configuring Accounts and Names and Obtaining IP Addresses

Before configuring a Windows 2000 failover cluster with SQL Server, you must create and/or configure domain user accounts for SQL Server, configure domain user account permissions for MSCS, obtain server names for MSCS and SQL Server, and allocate IP addresses for the heartbeat and the corporate network NICs. Use the following information to configure accounts and names, and obtain IP addresses.

- **Domain user accounts —** Create and configure the following domain user accounts:
  - **A domain user account to administer the failover cluster —** Make this account a member of the Administrators local group on each node of the failover cluster.
  - **A domain user account for the SQL Server and the SQL Server Agent services—**This account does not need to be a member of the Administrators local group on any node of the failover cluster.
- **Server names—**Have the IT department approve the following names:
  - **A virtual cluster name—**This name must be unique within the domain.
  - **A virtual SQL Server name for each SQL Server instance—**These names must be unique within the domain and on the local subnet if multiple domains reside together on the same subnet.
  - **An instance name for each SQL Server instance—**These names must be unique within the failover cluster.
- **New static IP addresses—**Have the IT department allocate static IP addresses for each of the following:
  - A virtual cluster IP address
  - A virtual IP address for each SQL Server instance
  - An IP address for each corporate network NIC

These addresses must be unique on the corporate network and must be on the same subnet.

Note: In a cluster, each SQL Server instance must have its own virtual IP address, virtual server name, and SQL Server instance name. This enables a SQL Server instance to fail over to a secondary node without affecting other SQL Server instances on the same physical node.

Configuring the Network

Before you install MSCS, you must assign the allocated IP addresses for each corporate network NIC, including the default gateway and preferred DNS server for each NIC. You must also assign static IP addresses to the NICs in each server that will carry the heartbeat among the servers in the failover cluster.

► To configure a NIC that connects to the corporate network

1. On the desktop, right-click **My Network Places**, and then click **Properties**.
2. In **Network and Dial-up Connections**, right-click the network connection for the NIC that connects to the corporate network, and then click **Properties**.
3. In **Connection Properties** for this corporate NIC, click **Internet Protocol (TCP/IP)**, and then click **Properties**.
4. In **Internet Protocol (TCP/IP) Properties**, type the IP address allocated to this corporate NIC.

   If your network uses subnetting, change the subnet mask only on the advice of a network specialist.
5. Type the IP address of the default gateway.
6. Click **Use the Following DNS Server Addresses**, and then type the DNS server address.
7. Click **OK** to save the new TCP/IP properties.
8. Click **OK** to save the new connection properties for this NIC.
9. In **Network and Dial-up Connections**, rename the connection to **Corporate** to clearly identify its function.

Note: Repeat this process if you use a second corporate NIC that provides redundant connectivity to the corporate network. Name this redundant corporate connection **Corporate2**.

► To configure the NIC that connects to the private network

1. On the desktop, right-click **My Network Places**, and then click **Properties**.
2. In **Network and Dial-up Connections**, right-click the network connection for the NIC that connects to the private network, and then click **Properties**.
3. In **Connection Properties** for this private NIC, click **Internet Protocol (TCP/IP)**, and then click **Properties**.
4. In **Internet Protocol (TCP/IP) Properties**, type an IP address for this private NIC.

   You must use an IP address for each private NIC that is on the same subnet as the IP address you use for the other private NICs in the cluster. Use the automatically completed subnet mask. It is common to use an IP address of the form 10.1.0.*x* for the private segment of the network.
5. Click **OK** to save the new TCP/IP properties.
6. Click **OK** to save the new connection properties for this NIC.
7. In **Network and Dial-up Connections**, rename the connection to **Heartbeat** to clearly identify its function.

Note: Repeat this process if you use a second private NIC that provides a redundant heartbeat. Name this redundant private connection **Heartbeat 2**.

Configuring Storage Devices

Each vendor has different procedures for configuring its storage devices. Follow the procedures provided by the vendor and configure the SAN as follows:

- Use the SAN for SQL Server only, not for Windows 2000.
- Create a mirrored volume for the quorum.
- Create three mirrored volumes for each SQL Server instance—one each for data files, transaction log files, and temporary files.
- Configure the SAN in the remote site to mirror the SAN activity in the local site. Figure 5.2 illustrates a typical interconnection configuration for a two-node stretch cluster that provides redundant paths from both servers to both SANs. Because the configuration differs among vendors, discuss the most appropriate configuration for your environment with your storage vendor.

Before configuring the storage devices for MSCS to use, turn off all but one node to avoid configuration errors.

Preparing the Drives

To prepare the storage for use, write disk signatures, create partitions, and format drives. Initialize drives used in a cluster as basic drives.

► **To initialize a drive for use in a cluster**

1. Right-click **My Computer**, and then click **Manage**.
2. Expand **Storage**, and then click **Disk Management**.
3. Right-click a drive that you want to initialize, and then click **Write Signature**.
4. In **Write Signature**, select the disk(s) to initialize, and then click **OK**.

► **To create a partition and format a drive**

1. In **Computer Management**, right-click the drive that you want to partition, and then click **Create Partition**.
2. On the **Welcome to the Create Partition Wizard** page, click **Next**.
3. On the **Select Partition Type** page, click **Primary Partition**, and then click **Next**.
4. On the **Specify Partition Size** page, click **Next** to use all of the disk space on the drive for this partition.
5. On the **Assign Drive Letter or Path** page, assign a drive letter, and then click **Next**.
6. On the **Format Partition** page, click **Format Partition**.
7. In **Formatting**, set the formatting options as follows:
   - In **File System to Use**, specify **NTFS**.
   - In **Allocation Unit Size**, specify **Default**.
   - In **Volume Label**, specify a meaningful name.

   Do not select the **Enable File** and **Folder Encryption** check boxes.

8. Click **Next**.
9. On the **Completing the Create Partition Wizard** page, click **Finish**.

► **To format a partitioned but unformatted drive**

1. Right-click **My Computer**, and then click **Manage**.
2. Expand **Storage**, and then click **Disk Management**.
3. Right-click a drive that you want to format, and then click **Format**.
4. In **Format Disk**, click **OK**.

Note: Repeat these procedures for each drive.

Specifying Drive Letters

Specify drive letters for each drive in the cluster to ensure that they do not change dynamically based on the drives that are online. If a drive letter that SQL Server uses for a data or a log file changes after SQL Server is installed, SQL Server cannot attach adatabase when SQL Server starts.

Note: If you created a new partition in the preceding procedures and assigned a drive letter, you do not need to complete the following procedure.

► **To specify drive letters**

1. Right-click **My Computer**, and then click **Manage**.
2. Expand **Storage**, and then click **Disk Management**.
3. Right-click a drive that the cluster uses, and then click **Change Drive Letter and Paths**.
4. In **Change Drive Letter and Paths**, click **Edit**.
5. In **Edit Drive Letter or Path**, assign a drive letter from the list of available drive letters, and then click **OK**.
6. Click **Yes** to confirm that you want to change the drive letter.
7. Repeat this process for each drive that the cluster uses.

When you add more nodes to the cluster, they inherit these drive letters from the cluster configuration information stored on the quorum drive.

Installing and Configuring MSCS

When you install and configure MSCS, set up only one node at a time. Turn off all other nodes that are connected to the shared storage device while you install MSCS on the first node. You risk disk configuration errors if you attempt to configure MSCS on the first node while other nodes are connected and running. After you

install MSCS on the first node and the drives in the shared storage device are allocated, you can install MSCS on additional nodes without turning off nodes on which MSCS is running.

Note: If you are reinstalling MSCS or recovering from a failed MSCS installation, remove any entries in the DNS server that relate to the failed MSCS installation before you reinstall. Stray or duplicate IP addresses or names can prevent a clean installation.

Install MSCS on all nodes in the cluster. The first node of an MSCS installation requires more setup than an additional node in the same cluster.

► To add MSCS on the first node

1. On the first server on which you are installing MSCS, click **Start**, point to **Settings**, and then click **Control Panel**.
2. In **Control Panel**, double-click **Add/Remove Programs**.
3. In **Add/Remove Programs**, click **Add/Remove Windows Components**.
4. On the **Windows Components** page, select the **Cluster Service** check box, and then click **Next**.

   The **Internet Information Services** check box is automatically selected with some Internet Information Services (IIS) components.
5. If Terminal Services is installed, click **Next** on the **Terminal Services Setup** page to run Terminal Services in remote administration mode.

   If Terminal Services is not installed, this page will not appear.
6. When prompted, insert your Windows 2000 Advanced Server or Windows 2000 Datacenter Server CD, and then click **OK**.
7. After the clustering components install, click **Next** on the **Welcome to the Cluster Service Configuration Wizard** page.
8. On the **Hardware Configuration** page, click **I Understand** to acknowledge that only certified configurations listed in the Cluster category of the HCL are supported by MSCS; then click **Next**.

► To define the cluster

1. On the **Create or Join a Cluster** page, click **The First Server in the Cluster**, and then click **Next**.

   If the Cluster Service Configuration Wizard does not detect disks that meet the criteria for clustering, you cannot continue.
2. On the **Cluster Name** page, type the name for the new cluster (up to 15 characters), and then click **Next**. Use a name that is unique on the network and within the Windows domain.

► To specify the setup program administrator account

- On the **Select an Account** page, type the domain user account, password, and domain name in the appropriate text boxes, and then click **Next**.

  Use the domain user account you previously added to the Administrators local group on this node.

► To define the cluster storage

1. On the **Add or Remove Managed Disks** page, specify the disks that you want to belong to this cluster, and then click **Next**.

   By default, all disks that meet the criteria for clustering are selected.
2. Remove any disks that you do not want this cluster to use.

   Do not remove any disks that you plan to use for any SQL Server instance on this cluster.
3. On the **Cluster File Storage** page, click the drive belonging to the cluster that you want to use as the quorum drive.

   Select a mirrored drive with a minimum size of 500 megabytes (MB). If the quorum drive fails, the cluster fails. Do not use this drive for SQL Server.

► To define the networks

In this step, you tell Cluster Services which NICs are to be used for internal cluster communication, such as the heartbeat, and which NICs are to be used for client access. The **Network Connections** page displays once for each NIC.

1. On the **Network Connections** page, select the **Enable This Network for Cluster Use** check box if this NIC is used in the cluster. Select the **Internal Cluster Communications Only (Private Network)** check box for each heartbeat NIC, or select the **Client Access Only (Public Network)** check box for each corporate NIC. Then click **Next**.
2. On the **Cluster IP Address** page, type the IP address in the **IP Address** text box, and confirm that the subnet mask is correct for your configuration.

   If your network employs subnetting, change the subnet mask only on the advice of a network specialist.
3. Click the corporate network, and then click **Next**.

► To finish the installation

1. Click **Finish** to apply the settings for the cluster, and start MSCS on this first node.
2. When the message informs you that MSCS has started, click **OK**.
3. Click **Finish** to close the Cluster Configuration Wizard.
4. Close **Add/Remove Program** and Control Panel.

Installing and Configuring MSCS on Additional Nodes in the Cluster

Installing MSCS on additional nodes in a cluster requires less setup than installing MSCS on the original node because additional nodes inherit many settings from the first node.

► **To install MSCS on an additional node in the cluster**

1. On each additional server on which you want to install MSCS, click **Start**, point to **Settings**, and then click **Control Panel**.
2. In **Control Panel**, double-click **Add/Remove Programs**.
3. In **Add/Remove Programs**, click **Add/Remove Windows Components**.
4. On the **Windows Components** page, select the **Cluster Server** check box, and then click **Next**.

   The **Internet Information Services** check box is automatically selected with some IIS components.
5. If Terminal Services is installed, click **Next** on the **Terminal Services Setup** page to run Terminal Services in remote administration mode.

   If Terminal Services is not installed, this page will not appear.
6. When prompted, insert the Windows 2000 Advanced Server or Windows 2000 Datacenter Server CD, and then click **OK**.
7. After the clustering components install, click **Next** on the **Welcome to the Cluster Service Configuration Wizard** page.
8. On the **Hardware Configuration** page, click **I Understand** to acknowledge that only certified configurations listed in the Cluster category of the HCL are supported by MSCS; then click **Next**.

► **To join an existing cluster**

1. On the **Create or Join a Cluster** page, click **The Second or Next Node in the Cluster**, and then click **Next**.

   If the Cluster Service Configuration Wizard does not detect disks that meet the criteria for clustering, you cannot continue. Disks must be formatted for NTFS and must not be configured as dynamic disks.
2. On the **Cluster Name** page, type the name of the existing cluster that you want this node to join. Leave the **Connect to cluster as** check box cleared, and then click **Next**.

   The Cluster Service Configuration Wizard verifies that a heartbeat is detected from the first node and then configures each NIC on the node as either a corporate NIC or a heartbeat NIC.

► **To specify the setup program administrator account**

- On the **Select an Account** page, type the domain user account, password, and domain name in the appropriate text boxes, and then click **Next**.

  Use the domain user account you created earlier that is an administrator on each node.

► **To finish the installation**

1. Click **Finish** to apply the settings for the cluster, and then start MSCS on this additional node.
2. When the message box informs you that MSCS has started, click **OK**.
3. Click **Finish** to close the Cluster Configuration Wizard.
4. Close **Add/Remove Program** and Control Panel.

Upgrading the MS DTC

You must upgrade MS DTC to work in a cluster before installing SQL Server in the cluster. Distributed queries, two-phase commits, and certain types of replication use DTC. Even if you do not currently use these features, upgrade DTC because you may need to use the features in the future. Upgrading now avoids errors later.

► **To upgrade DTC**

1. At one node in the cluster, open a command prompt.
2. At the command prompt, type **comclust**, and then click **Enter**.
3. Close the command prompt.

Note: Repeat this procedure on all other nodes participating in the SQL Server virtual server.

Installing and Configuring SQL Server on the Cluster

When you install SQL Server on a cluster, the setup program creates a virtual server. A virtual server consists of a virtual name, an IP address, and a port. Clients connect to these virtual resources, and MSCS points to the actual resources on the node in the cluster that is currently active. The SQL Server setup program installs SQL Server program files on each node and SQL Server system databases on a cluster resource disk. To successfully install a virtual server, you must properly configure and start MSCS.

Note: If you are reinstalling SQL Server or recovering from a failed SQL Server installation, remove any entries in the DNS server that relate to the failed MSCS installation before you reinstall. Stray or duplicate IP addresses or names can prevent a clean installation.

Creating a Single Cluster Group

Before you install SQL Server, place the disk resources for the SQL Server instance into a single cluster group. The group should contain a mirrored volume for the data files, a volume for the log files, and a volume for the backup and temporary files. The setup program adds additional resources to this cluster group when the virtual server installs.

Note: Create a separate cluster group for each SQL Server instance you plan to install.

► **To create a single cluster group**

1. Click **Start**, point to **Administrative Tools**, and then click **Cluster Administrator**.
2. In **Cluster Administrator**, expand **Groups**, and then locate the disk groups containing the disks that you plan to use for this SQL Server instance.

   Initially, each disk (or set of mirrored disks) belonging to the cluster is in a numbered disk group, such as Disk Group 1, Disk Group 2, and so on. The quorum drive is in the Cluster Group.
3. Right-click one of the disk groups, and then click **Rename**.
4. Type a descriptive name for this resource group (such as **VirtualServer1Group**).
5. Click the **Resource** node to display the additional drives owned by the cluster.
6. Right-click a drive for SQL Server, point to **Change Group**, and then click the newly renamed cluster group.
7. Close Cluster Administrator.

Note: Repeat this procedure for each additional drive that this SQL Server instance will use.

Installing SQL Server

After creating the cluster group, you can begin installing SQL Server. Use the server name and IP address allocated by your IT department when you create the virtual server. Follow these procedures for each instance you install. Before you begin, create a folder on each clustered drive to easily identify all files belonging to SQL Server.

Note: You must install SQL Server from the server that owns the shared disks. To change the server that owns the shared disks, use Cluster Administrator.

► **To start the installation and define the virtual server name**

1. Insert the SQL Server 2000 Enterprise Edition CD into the CD-ROM drive.
2. On the SQL Server 2000 Enterprise Edition page, click **SQL Server Components**, and then click **Install Database Server**.

3. On the **Welcome** page, click **Next**.
4. On the **Computer Name** page, type a virtual name, click **Virtual Server**, and then click **Next**.
5. On the **Product Key** page, enter the product key for SQL Server 2000 Enterprise Edition, and then click **Next**.
6. On the **Software License Agreement** page, click **Yes** to accept the terms of the License Agreement.
7. On the **Virtual Server Information** page, enter the IP address allocated for the virtual server, click the corporate network, click **Add**, and then click **Next**.

► **To define disk resources**

1. On the **Select the Cluster Disk Where the Data Files Will Be Placed** page, select the cluster group (or a drive within that cluster group) for this SQL Server.

   This step defines the cluster group that will contain the resources for the virtual SQL Server. The actual drive locations for executables and system databases are selected later. Do not select the cluster group containing the quorum drive.
2. On the **Cluster Definition** page, specify the nodes supporting the virtual server, and then click **Next**.

   By default, all nodes in the cluster are selected. Remove any nodes that you do not want to support SQL Server.
3. On the **Remote Information** page, type the domain user administrator account, password, and domain name in the appropriate text boxes.

   The setup program uses this user account to install the program files on each node. Use the administrator account you configured to administer the failover cluster. For more information, see "Configuring Accounts and Names and Obtaining IP Addresses" earlier in this chapter.
4. On the **Instance Name** page, click **Next** to install a default instance. To install a named instance, clear the **Default Instance** check box, type the name for the named instance, and then click **Next**.
5. On the **Setup Type** page, click the appropriate setup type for your installation.
6. Specify the location for the program files on each node.

   The default location is the system volume. Change this default if appropriate, and ensure that sufficient space exists for these files. The setup program installs the program files into this location on each node.
7. Specify the location for the data files on the cluster.

   The data files must be on one of the cluster disks and should not be on the quorum disk. The setup program installs the system databases on this drive.

   If you have created a folder for this SQL Server instance on the data drive, browse to this folder, and then click **Next**.

► **To complete the installation**

1. On the **Services Accounts** page, type the domain user account, password, and domain name in the appropriate text boxes, and then click **Next**.

   The SQL Server and SQL Server Agent services use this account. Use the domain user account you configured for the SQL Server and SQL Server Agent services. For more information, see "Configuring Accounts and Names and Obtaining IP Addresses" earlier in this chapter.
2. On the **Authentication Mode** page, click the authentication mode for SQL Server, and then click **Next**.

   Windows Authentication Mode is more secure than Mixed Mode. If you select **Mixed Mode**, you must provide a password for the **sa** account or agree to use a blank password. Do not use a blank password because it is highly insecure, even in a development environment, and can lead to poor coding practices.
3. On the **Licensing Mode** page, specify the appropriate licensing mode for your environment, and then click **Next**.
4. Click **Finish** to install SQL Server. When installation is complete, restart all nodes in the cluster.

Adding Cluster Disks as SQL Server Dependencies

To ensure that the failover process operates properly, manually add each disk on which SQL Server depends to the list of SQL Server dependencies. If a disk is not included in the SQL Server dependencies, SQL Server cannot use it after a failover. Add the mirrored volumes used for the transaction log files, backup files, and additional data files. This procedure requires all disk resources to be members of a single cluster group.

► **To add additional cluster disks as dependencies**

1. Click **Start**, point to **Programs**, and then click **Cluster Administrator**.
2. In **Cluster Administrator**, click the cluster group containing the SQL Server virtual server.
3. Right-click **SQL Server**, and then click **Take Offline**.

   SQL Server must be offline to change SQL Server dependencies.
4. Right-click **SQL Server**, and then click **Properties**.
5. On the **Dependencies** tab in **SQL Server Properties**, review the default resource dependencies.

   The **Network Name** resource and at least one disk resource appear.
6. Click **Modify**.
7. In the **Available Resources** list, double-click each additional disk resource on which SQL Server depends, and then click **OK**.

8. Click **OK** to close SQL Server Properties.
9. Right-click group containing the SQL Server resources, and then click **Bring Online**.

Applying the Most Recent SQL Server Service Pack

Before using SQL Server, install the most recent SQL Server service pack.

► **To install the most recent SQL Server service pack**

1. Log on to one of the nodes in the cluster by using a domain account that is a member of the Administrators local group on that node.
2. Insert the CD containing the service pack into the CD-ROM drive.

   If it does not start automatically, navigate to the **\X86\Setup** directory, and double-click **setupsql.exe**.
3. On the **Welcome** page, click **Next**.
4. On the **SQL Server Name** page, type the name of the SQL Server virtual server, and then click **Next**.
5. On the **Authentication Mode** page, click **Next** to connect to SQL Server using Windows authentication.
6. On the **Domain Administrator** page, type a domain administrator name, password, and domain name, and then click **Next**.

   The setup program must use a domain administrator account to install the service pack on all nodes of the virtual server.
7. After the service pack installs, click **OK**.
8. Click **Yes**, and then click **Finish** to restart the node.
9. Manually restart each additional node on which the service pack has been installed.
10. Back up the **master** and **msdb** databases.

    They have been modified by the service pack installation.

Note: This procedure is for SQL Service 2000 Service Pack 2. Check the ReadMe file on the CD for later service pack procedures.

Performing Additional Configuration Tasks

After you have installed the virtual server, review these additional configuration options, and apply the settings that are appropriate for your environment. These options include the following:

- Setting memory allocation between virtual servers
- Setting new database properties

- Setting the TCP/IP port that SQL Server uses
- Setting advanced failover options
- Setting SQL Server properties in the cluster

Use the following guidelines when performing these tasks.

Allocating Memory

If you are running multiple instances of SQL Server on separate nodes in the cluster, you must determine the total amount of physical memory required for each server, and determine the appropriate allocation of memory for each SQL Server instance. Performance of SQL Server is directly related to the amount of physical memory allocated to it by the operating system. SQL Server provides two modes for memory management: dynamic and fixed. *Dynamic memory* allocation lets SQL Server acquire and release memory from the operating system as needed. Dynamic memory allows you to set a maximum amount of physical memory that the operating system can allocate to SQL Server. This allows the remaining memory to be reserved for other purposes, such as the operating system and other SQL Server instances. *Fixed memory* allows you to set the amount of physical memory that SQL Server can use. This amount neither grows nor shrinks.

Configure each server with the same amount of physical memory, and then allocate memory to each SQL Server instance to ensure that sufficient memory is available to support the failover of an instance of SQL Server from another node. If a server does not use a lot of memory compared to the available memory on the server, specify dynamic memory and set the maximum amount of memory that the operating system can allocate. By doing so, you leave sufficient memory for the failover to occur quickly and successfully. If an instance uses all of the available physical memory, failover occurs more slowly and can fail if memory is not released quickly enough.

To determine the total amount of memory to install on each server, test your applications by setting the maximum amount of memory available to SQL Server, and evaluate the performance. Install enough memory on each server to enable all instances to operate on that server and to meet the minimum service level for each instance. Using dynamic memory allows the SQL Server instance on each node to use most of the memory on that node. If you use fixed memory, the additional memory is used only during a failover.

In a system under continuous heavy load, install sufficient memory to ensure that each SQL Server instance can meet its required minimum allocation. Use a fixed memory allocation to ensure that the SQL Server instance will not expand into memory reserved for the failover of other instances. To determine the appropriate setting, initiate a failover, and test the performance level of each virtual server. If the operating system is unstable after a failover, decrease the memory allocated to each instance to increase the memory available to the operating system.

To specify a fixed amount of memory for each instance, use SQL Server Enterprise Manager or the **sp_configure** system stored procedure. When determining the amount of memory to allocate to SQL Server, remember to allow sufficient memory for the operating system. With large databases, allow 2 gigabytes (GB) of memory for the operating system.

If the amount of memory installed on each node is greater than 4 GB, add the **/PAE** switch to the Boot.ini file, and enable Address Windowing Extensions (AWE) in SQL Server. For more information, see article Q268363, "Intel Physical Addressing Extensions (PAE) in Windows 2000," in the Microsoft Knowledge Base at *http://support.microsoft.com/default.aspx?scid=kb;[LN];Q268363*. Also see article Q274750, "HOW TO: Configure Memory for More Than 2 GB in SQL Server," in the Microsoft Knowledge Base at *http://support.microsoft.com/default.aspx?scid=kb;en-us;Q274750*.

Setting New Database Properties

You can specify the database location for all new databases on the cluster disks you created for the data and log files.

► **To set new database properties**

1. On one of the nodes, click **Start**, point to **Programs**, point to **Microsoft SQL Server**, and then click **Enterprise Manager**.
2. Expand **Microsoft SQL Servers**, and then expand **SQL Server Group**.
3. Right-click the SQL Server instance, and then click **Properties**.
4. On the **Database Settings** tab in the **Default Data Directory** text box, type or browse to the drive and folder you want for the data files.
5. In the **Default Log Directory** text box, type or browse to the drive and folder you want for the log files, and then click **OK**.
6. Close SQL Server Enterprise Manager.

Setting the TCP/IP Port SQL Server Uses

The default instance of SQL Server usually listens on port 1433. If you are configuring a multiple instance cluster, you must use a different port for each additional instance. Although SQL Server can set this port dynamically at startup, you might want to set each named instance to listen on a specific, unused port. Check with a network administrator for a recommendation. Specify a static port if you have to open ports on a firewall and do not want the SQL Server ports to change.

► **To specify the TCP/IP port for an instance**

1. Click **Start**, point to **Programs**, point to **Microsoft SQL Server**, and then click **Server Network Utility**.
2. In **SQL Server Network Utility**, click the instance of SQL Server for which you want to set the port.

3. In the **Enabled Protocols** list, click **TCP/IP**, and then click **Properties**.
4. In the **Default Port** text box, type the appropriate port, and then click **OK**.
5. Click **OK** to close the **SQL Server Network Utility** dialog box.
6. Repeat these steps for each node participating in the SQL Server virtual server.

Setting Advanced Failover Options

Advanced failover options determine the failover behavior of SQL Server in the MSCS cluster.

► **To specify failover behavior**

1. Click **Start**, point to **Programs**, and then click **Cluster Administrator**.
2. In Cluster Administrator, click the cluster group containing SQL Server.
3. Right-click **SQL Server**, and then click **Properties**.
4. On the **Advanced** tab, use the following information to configure the advanced cluster failover properties. The defaults should not be changed without careful consideration and testing:
   - **Do Not Restart/Restart**—Click **Restart** to fail over to a secondary node if a failure is detected. This is the default.
   - **Affect the Group**—Select this check box to have the failover of SQL Server cause a failover of all resources in the cluster group.
   - **Threshold**—Type the number of attempts that MSCS should make to restart SQL Server before failing over to a secondary node.
   - **Period**—Type the seconds between retry attempts.

     This value is ignored if the threshold is set to 0.
   - **Looks Alive Poll Interval**—Click **Use Value from Resource Type** to use the SQL Server default of 5,000 milliseconds (5 seconds).
   - **Is Alive Poll Interval**—Click **Use Value from Resource Type** to use the SQL Server default of 60,000 milliseconds (60 seconds).
   - **Pending Timeout**—Specify the time SQL Server has in either the Offline Pending or the Online Pending state before MSCS places the resource in Offline or Failed status. The default is 180 seconds.
5. When you are finished, click **OK** to close SQL Server Properties in the Cluster Administrator.

Configuring Service Properties in a Cluster

In a cluster environment, you must use the correct management tool to configure SQL Server services, as follows:

- Use Cluster Administrator to start and stop the services.

- Use the **Properties** dialog box for the SQL Server instance in Enterprise Manager to set the service accounts and passwords for the services.

Failing to use the appropriate tools to configure SQL Server properties can cause SQL Server startup problems. In addition, in a cluster, do not set SQL Server Agent to restart automatically. If SQL Server Agent is set to restart automatically, it can conflict with the cluster services during a failover.

Failing Over to a Secondary Node in a Stretch Cluster

Failover is automatic when using MSCS and SQL Server. You can control how long MSCS takes to detect a failure, and you can take steps in your applications to reduce the time to bring up the standby service. Before making any changes to MSCS or to a client application, test the failover cluster to become familiar with how it fails over under different conditions.

Testing Failover

Use Cluster Administrator to initiate a planned failover. This simulates taking a node offline for planned maintenance.

► **To initiate a planned failover**

1. Click **Start**, point to **Programs**, and then click **Cluster Administrator**.
2. Click the cluster group containing the SQL Server virtual server.

   Notice the cluster that owns each resource in this cluster group.
3. Right-click the cluster group containing the SQL Server virtual server, and then click **Move Group**.

Watch each resource in the group switch from online to offline to online pending, and then to online in the new group. The entire process takes 1 to 2 minutes for a typical environment. Also notice that the resource owner in this cluster group changes to the secondary cluster.

Use one of the following options to simulate an unplanned failover.

► **To simulate an unplanned failover**

1. Do one of the following:
   - Unplug the corporate network cable from the node that owns SQL Server.
   - Reboot the owning node.
   - Unplug the fibre connected to the active SAN.
   - Turn the power off on the owning node.

2. Use Cluster Administrator on the secondary node to watch each resource in the group switch from online to offline to online pending, and then to online.

   The entire process takes less than 1 minute. Notice that the resource owner in this cluster group changes to the secondary cluster.

Minimizing Failover Time

If the time it takes SQL Server to fail over from one node to another needs to be decreased, use the following information to determine how to reduce time to availability.

Understanding Time to Availability

The time it takes from the occurrence of a fault to SQL Server being available on the standby node consists of the following:

- The time to detect and confirm the fault, which you can control
- The time to take the resources offline, which is controlled by MSCS and which you cannot control
- The time to bring the resources online, which MSCS controls and which you can control by ensuring that sufficient SQL Server memory and processor resources are allocated
- The time for SQL Server to initialize, which you can control by limiting long transactions)

SQL Server is usually available within 1 minute or less.

Controlling the Time Required to Detect and Confirm the Fault

The **LooksAlive** and **IsAlive** poll intervals for SQL Server determine the time that it takes MSCS to detect a failure. Smaller intervals create more overhead and increase the possibility of false failure detections — and therefore result in unnecessary failovers. The default for SQL Server the **IsAlive** interval is 60,000 milliseconds (60 seconds), and **LooksAlive** is 5,000 milliseconds (5 seconds). Change these values on the test platform first, and test them thoroughly under a realistic load. If you set the interval lower, be prepared to measure the impact on the application.

Controlling the Time Required to Bring the Resources Online

If the failover node has insufficient available memory at the time of failover, it will take longer to bring SQL Server online after a failover. Although this condition should not occur in a single-instance cluster, this can be a problem in a multiple-instance cluster. In normal operation, each node runs one instance of SQL Server, and each instance uses memory as needed. If a node does not have sufficient memory to run multiple instances simultaneously during a failover, use a lower maximum memory setting to reserve memory for failover.

If at least 30 MB of unallocated physical memory is available, the instance starts in dynamic memory allocation mode. For monitoring purposes, a low-memory condition creates an entry in the SQL Server event log warning you that the operating system did not fulfill the SQL Server memory request. No entry is made in the Application Log. This does not mean that you should attempt to operate SQL Server with only 30 MB of available physical memory.

Controlling the Time Required to Initialize

At startup, SQL Server must verify the condition of each database, rolling forward completed transactions and rolling back any uncommitted transactions that it finds. If client applications use very long transactions, the incomplete transactions can take a long time to roll back. Use a series of short transactions whenever possible. Determine each application's worst-case transaction size, and test the recovery time when this transaction is rolled back in a failover.

Performing a Manual SAN Failover

MSCS does not control the SAN devices, and the failover of the SAN devices is not automatic. Follow the procedures each storage vendor provides to reverse the role of the SAN devices when the active SAN fails. These procedures include additional steps to restore MSCS when the backup SAN is brought online.

When a failure is detected, the quorum drive is generally affected and MSCS stops.

► **To restart MSCS**

1. Bring the backup SAN online.
2. Stop all nodes except one.
3. Stop and restart the remaining node
4. Open **Computer Management**, right-click **Disk Manager**, and then click **Rescan**.
5. Open Cluster Administrator, and verify that all disks are online.

► **To restart disks if MSCS cannot start all disks**

1. Delete each disk resource from its cluster group.
2. Right-click the cluster group, click **New**, and then click **Resource**.
3. Assign the disk the same name, and click **Physical Disk** as the resource type.
4. Assign the correct physical disk to the resource name, and accept the defaults for the remaining settings.

Note: Repeat this procedure for each disk.

Reconnecting Clients

When the SQL Server service activates on the failover node, the application must reconnect to continue operating. How clients reconnect depends on whether an application maintains state or not, and on the ability of the application to reissue the steps in a transaction. Clients reconnect in the following ways:

- In a Web application, application state is usually not maintained. When a user submits a request to the Web server, the application usually makes a new connection. In this environment, no additional retry logic is required. The Web application needs to gracefully handle the error response when the application cannot connect to SQL Server, however.
- In a non-Web application, a connection is often held for a whole session. You must implement logic in the application that detects a broken connection and that attempts to reconnect at appropriate intervals (for example, 30 seconds). The application should be prepared to restart the transaction that was in progress when the connection broke. In some cases, you may have to rely on the application user to retry the submission.

Failing Back to the Original Node

In a single-instance cluster, little reason exists to fail back to the original node. If you have symmetrical servers that are dedicated to the same application, performance is identical on each. Failing back only causes more downtime.

In a multiple-instance configuration, after a failover, two instances of SQL Server run on the same node, so performance decreases for both. Choose a failback time to minimize the impact on clients.

You can set MSCS to automatically fail back to the original node; however, doing so incurs another service outage when the service is moved to the other node. In addition, the failed node may not be ready for the service to fail back, causing more downtime; therefore, enabling automatic failback is not always effective in maintaining high availability.

Optionally, you can set up automatic failback to occur during standard times set for when an outage is acceptable.

► **To configure automatic failback in a specific time slot**

1. Click **Start**, point to **Programs**, and then click **Cluster Administrator**.
2. Right-click the group containing SQL Server.
3. Choose **Properties**.
4. On the **Failback** tab, set the time of day during which you want to allow failback.

Manual failback enables you to control when to stop the service and move it back to the original node when doing so is least disruptive to your clients.

► **To initiate manual failback**

1. Click **Start**, point to **Programs**, and then click **Cluster Administrator**.
2. Right-click the group containing SQL Server.
3. Choose **Move Group**.

Restoring a Node After a Catastrophic Failure

If one of the cluster nodes suffers a catastrophic failure, you might have to rebuild the server. Use the steps in the following procedures to recover from a catastrophic failure.

► **To rebuild the server**

1. On the surviving node, remove the failed node from the virtual server. If you have more than one virtual server on a node, doing so updates all instances.
2. On the surviving node, evict the failed cluster node.
3. Rebuild the failed node.
4. Install SQL Server.

► **To remove a node from a virtual server**

1. Insert the SQL Server 2000 Enterprise Edition CD into the CD-ROM drive.
2. On the **SQL Server 2000 Enterprise Edition** page, click **SQL Server Components**, and then click **Install Database Server**.
3. On the **Welcome** page for the Microsoft SQL Server Installation Wizard, click **Next**.
4. On the **Computer Name** page, type the virtual server name, click **Virtual Server**, and then click **Next**.
5. Leave **Advanced Options** selected, and then click **Next**.
6. Leave **Maintain a Virtual Server for Failover Clustering** selected, and then click **Next**.

   Note: If the server is unavailable, you will receive the following error message: "One or more nodes of Virtual Server are unavailable. Setup cannot modify the virtual server's IP address resource." If this error message appears, click **OK**.

7. On the **Virtual Server Information** page, click **Next**.
8. On the **Cluster Management** page, click the failed node, and then click **Remove**. Click **Next**.
9. On the **Remote Information** page, enter the login name and password of an account that is an administrator for all nodes in the cluster. Click **Next**.
10. In the message box, click **Yes**.
11. Click **Finish**.

► To evict a node from a cluster

1. Click **Start**, point to **Programs**, and then click **Cluster Administrator**.
2. In **Cluster Administrator**, right-click the failed node, and then click **Evict Node**. Click **OK**.
3. Close Cluster Administrator.

► To rebuild a node

1. Reinstall Windows, join the server to the domain, and apply the most recent Windows service pack on the new hardware.
2. Configure the domain user account to administer the cluster. For more information, see "Configuring Accounts and Names and Obtaining IP Addresses" earlier in this chapter.
3. Configure the corporate and private NICs.
4. Install and configure MSCS as an additional node. For more information, see "Installing and Configuring MSCS on Additional Nodes in the Cluster," earlier in this chapter.

► To add the rebuilt node to the virtual server

1. Insert the SQL Server 2000 Enterprise Edition CD into the CD-ROM drive of any node in the cluster.

   You can add a rebuilt node to the virtual server from any node in cluster.
2. On the **SQL Server 2000 Enterprise Edition** page, click **SQL Server Components**, and then click **Install Database Server**.
3. On the **Welcome** page for the Microsoft SQL Server Installation Wizard, click **Next**.
4. On the **Computer Name** page, type the virtual server name, click **Virtual Server**, and then click **Next**.
5. On the **Installation Selection** page, click **Advanced Options**, and then click **Next**.
6. Leave **Maintain a Virtual Server for Failover Clustering** selected, and then click **Next**.
7. On the **Virtual Server Information** page, click **Next**.
8. On the **Cluster Management** page, click the new node, click **Add**, and then click **Next**.
9. In the **Username** text box, specify a domain user account that is an administrator on each node participating in the virtual server configuration. Type the password in the **Password** text box, type the domain name in the **Domain** text box, and then click **Next**.
10. On the **Services Accounts** page, type the domain user account, password, and domain name in the appropriate text boxes, and then click **Next**.
11. Click **Finish** when setup completes. Repeat this process for each instance.

Note: After installing SQL Server 2000 and joining it to the virtual SQL Server, apply the appropriate SQL Server service pack to the rebuilt node. All nodes run at the same service pack level.

References

For more information about topics covered in this chapter, see the following:

- Article Q254321, "INF: Clustered SQL Server Do's, Don'ts, and Basic Warnings," in the Microsoft Knowledge Base at *http://support.microsoft.com /default.aspx?scid=kb;en-us;Q254321*
- Article Q259267, "Microsoft Cluster Service Installation Resources," in the Microsoft Knowledge Base at *http://support.microsoft.com /default.aspx?scid=kb;en-us;Q259267*
- Article Q239885, "INF: How to Change Service Accounts on a SQL Virtual Server," in the Microsoft Knowledge Base at *http://support.microsoft.com /default.aspx?scid=kb;en-us;Q239885*
- Article Q273673, "INF: SQL Virtual Server Client Connections Must Be Controlled by Clients," in the Microsoft Knowledge Base at *http://support.microsoft.com/default.aspx?scid=kb;en-us;Q273673*
- "Microsoft Solution for Systems Architecture: Internet Data Center" on the Microsoft TechNet Web site at *http://www.microsoft.com/technet /treeview/default.asp?url=/technet/itsolutions/idc/pag/pag.asp*

For more information on third-party vendor products used in testing, see the following:

- Compaq servers and SAN devices
- SANCastle Fibre Channel Gateway at *http://sancastle.com/products/*