Microsoft®

SQL Server 2000 High Availability

Volume 1
Planning

patterns & practices
proven practices for predictable results

ISBN: 0-7356-1835-6

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, Outlook, Visual SourceSafe, 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
Introduction to High Availability **1**
Microsoft SQL Server 2000 High Availability Series . . . 1
Planning Guide . . . 3
Solution Guide . . . 4
Understanding High Availability . . . 4
Designing a High-Availability Data Center . . . 5
Setting High-Availability Goals . . . 5
Identifying and Analyzing Barriers to High Availability . . . 7
Determining and Evaluating High-Availability Solutions . . . 9

Chapter 2
Overcoming Barriers to High Availability **11**
Overcoming Environmental Barriers . . . 12
Overcoming Hardware Barriers . . . 14
Overcoming Communication and Connectivity Barriers . . . 15
Overcoming Software Barriers . . . 15
Overcoming Service Barriers . . . 17
Overcoming Process Barriers . . . 18
Process Management . . . 18
Incident Management . . . 19
Change Management . . . 22
Maintenance Management . . . 23
Configuration Management . . . 24
Overcoming Application Design Barriers . . . 25
Overcoming Staffing Barriers . . . 26

Chapter 3
Recovering a Data Center by Using Database Backups 27
Solving High-Availability Barriers Using Backups . . . 28
Storage Subsystem Failure . . . 28
Application or User Errors . . . 29
Developing a Backup-and-Recovery Solution . . . 30
Measuring Total Recovery Time . . . 31
Improving Total Recovery Time . . . 31
Using a Third-Party Backup-and-Recovery Solution . . . 34
Using Snapshots to Improve the Recovery Time . . . 34
Using Third-Party Solutions to Increase Functionality . . . 39

Chapter 4
Preventing Downtime by Using Redundant Components 41
Using Server-Class Hardware . . . 41
Using a Redundant Storage Subsystem . . . 42
RAID . . . 43
Controller Cards . . . 45
Data and Log File Placement . . . 46
Using Redundant Network Cards . . . 46

Chapter 5
Minimizing Downtime by Using Redundant Servers 47
Using Failover Clustering to Maintain a Redundant Server . . . 48
Failover Clustering Architecture . . . 48
Failover Clustering Advantages . . . 51
Failover Clustering Limitations and Disadvantages . . . 52
Using Log Shipping to Maintain a Redundant Server . . . 52
Log Shipping Architecture . . . 53
Firewall Considerations . . . 57
Log Shipping Role Change . . . 57
Log Shipping Advantages . . . 59
Log Shipping Limitations and Disadvantages . . . 59
Using Transactional Replication to Maintain a Redundant Server . . . 60
Transactional Replication Architecture . . . 60
Firewall Considerations . . . 63
Transactional Replication Role Change . . . 63
Transactional Replication Advantages . . . 64
Transactional Replication Limitations and Disadvantages . . . 65
Redirecting Client Network Traffic to a Promoted Secondary Server . . . 65
Choosing Among the Technology Solutions . . . 68
Summary . . . 69

Appendix

Contents of a Run Book 71
Resource Information 71
Contact Information 72
Hardware Components 72
Software Components 73
Procedural Information 75
Operational Tasks 75
Emergency Tasks 77

Index 79

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 2material, 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 0.1.

Table 0.1. 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

Introduction to High Availability

Component failure, power outages, user errors, application memory leaks, and other circumstances can make a data center unavailable. To increase the availability of a data center, you must develop and follow procedures to prepare for and minimize the impact of downtime. You must also perform regular backups to prepare for catastrophic disasters and database corruption. You can prevent component failures from causing the unavailability of a data center by deploying redundant components. You can minimize the time a data center is unavailable due to server failures and planned maintenance by using redundant servers.

Increasing the availability of a data center saves money by keeping a data center available to business users and customers, and it may save a business in the event of a catastrophic disaster. You also need to consider that the availability of a data center can have an impact far beyond your own company, such as affecting client accounts and medical records.

The budget for increasing the availability of a data center is dictated by the cost of the database not being available. The higher the cost of unavailability, the more sophisticated and expensive the solutions should be to prepare for all types of disasters and to minimize the resulting downtime.

Microsoft SQL Server 2000 High Availability Series

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 high-availability data center, this series is for you.

The Microsoft SQL Server 2000 High Availability Series is designed for business decision makers, Microsoft Certified Solution Providers, Microsoft Consulting Services, IT professionals, and developers responsible for application or infrastructure development and deployment.

This series assumes you have a basic understanding of the following areas of SQL Server 2000 and Microsoft Windows® 2000 operating system:

- Windows 2000 Active Directory® service and DNS
- Microsoft Cluster Service (MSCS)
- Database design for online transaction processing (OLTP) and online analytical processing (OLAP)
- Application design for databases
- SQL Server operational procedures
- SQL Server backup and restore procedures
- SQL Server security model
- Redundant Array of Independent Drives (RAID)
- Storage Area Networks (SANs)
- Network Load Balancing (NLB)

The series consists of a Planning Guide and a Solution Guide. The Planning Guide helps you design a data center to achieve the level of availability needed for the business environment. The Solution Guide helps you implement selected server redundancy solutions to minimize unavailability caused by server failures and planned downtime.

The series provides guidance in achieving a highly available data center, including:

- Understanding barriers to high availability
- Developing procedures to minimize the risk and length of downtime
- Using backups to recover from server failure and database corruption
- Using redundant components to prevent downtime
- Evaluating server redundancy solutions
- Implementing server redundancy solutions

Note: This documentation is not intended to replace the product documentation for Windows 2000 or SQL Server 2000.

The Planning Guide provides a solid springboard for making effective decisions to help you build a highly available data center.

Planning Guide

The Planning Guide analyzes the barriers to availability and provides solutions to each of these barriers. It contains the following chapters:

- **Chapter 1: "Introduction to High Availability"**—This chapter defines high availability, discusses setting availability goals, and lists the barriers you must address to successfully deploy a highly available data center. By the end of this chapter, you will understand high availability and how to set goals for achieving the level of availability required for the business environment.
- **Chapter 2: "Overcoming Barriers to High Availability"**—This chapter discusses each of the barriers to high availability and the solutions to overcome these barriers. By the end of this chapter, you will understand the procedures that should be part of every SQL Server 2000 high-availability data center and the steps that you must take to prepare for, prevent, and minimize disaster.
- **Chapter 3: "Recovering a Data Center by Using Database Backups"**—This chapter discusses using database backups to recover from catastrophic disaster caused by hard-disk failure, by application or user error, or by hardware-induced database corruption. By the end of this chapter, you will understand how the SQL Server backups increase availability, and how third-party backup solutions increase restoration performance and functionality.
- **Chapter 4: "Preventing Downtime by Using Redundant Components"**—This chapter discusses using component redundancy to increase the availability of a single data-center server. This redundancy includes using server-class computers to provide component redundancy, using RAID arrays to provide storage redundancy, and using multiple networks to provide network redundancy. By the end of this chapter, you will understand the steps that you can take to prevent downtime caused by the failure of a single component.
- **Chapter 5: "Minimizing Downtime by Using Redundant Servers"** — This chapter discusses server redundancy solutions that minimize the time required to recover from a server failure. By the end of this chapter, you will understand the redundant server solutions that you can deploy to minimize downtime. You will also be able to decide among these solutions based on the unique requirements of the business environment.

After you read the Planning Guide, you will understand how proper procedures increase availability, backups help recover from catastrophic disasters, redundant components prevent downtime, and redundant servers minimize downtime.

Solution Guide

The Solution Guide documents the steps required to implement each of the server redundancy solutions discussed in Planning Guide Chapter 5. The Solution Guide contains the following chapters:

- **Chapter 1: "Implementing Failover Clustering"**—This chapter provides the steps to implement failover clustering, including the steps to fail over to a standby node and to fail back to the primary node.
- **Chapter 2: "Implementing Log Shipping"**—This chapter provides the steps to implement log shipping, including the steps to fail over to a standby server and to fail back to the primary server.
- **Chapter 3: "Implementing Transactional Replication"**—This chapter provides the steps to implement transactional replication for high availability, including the steps to fail over to a subscriber and fail back to the original server.
- **Chapter 4: "Implementing Network Load Balancing"**—This chapter provides the steps to implement network load balancing, including the steps to fail over to a standby server and to fail back to the primary server.
- **Chapter 5: "Implementing Remote Mirroring and Stretch Clustering"**—This chapter provides the steps to implement a stretch cluster, including the steps to fail over to a remote standby node and to fail back to the primary node.

Understanding High Availability

In the context of this series, *high availability* means increasing the availability of the data center itself. You increase the availability of a data center by:

- Developing and following well-documented procedures
- Ensuring the ability to recover from any type of disaster
- Avoiding downtime and the need for restoration by using component and server redundancy
- Minimizing how long data is unavailable after an accident or emergency
- Providing the ability to sustain operations during a recovery period after an accident or emergency
- Avoiding system downtime caused by planned maintenance

Even if you ensure the actual availability of the data center, users may perceive the data center as unavailable when they cannot access it. Some reasons users may not be able to access the data center are improper application design, inadequate security, network failures, and DNS problems. Although this series identifies these issues as barriers to high availability, resolving these issues is beyond the scope of this series.

Designing a High-Availability Data Center

To design a high-availability data center, start by doing the following:

- Set high-availability goals for scheduled and unscheduled downtime.
- Identify and analyze the actual and perceived barriers to high availability.
- Determine and evaluate the solutions that overcome each barrier to high availability.

After completing these tasks, you are ready to implement the appropriate processes and solutions to increase the availability of the data center.

Setting High-Availability Goals

The first step in increasing the availability of a data center is to set high-availability goals. To do this, the decision-making group must thoroughly understand how the end-user community uses the data center and the importance of the data center to the profitability of the company.

Do the following when setting high-availability goals:

- **Identify stakeholders**—Setting high-availability goals is the responsibility of many parties, and these goals must be appropriate to all stakeholders. The impact of high-availability goals on database administrators as well on business users and customers must be evaluated. For example, although business users and customers might want 99.999-percent availability, database administrators must make clear the cost of achieving this high-availability goal.
- **Establish the value of availability**—The value of availability determines the budget for achieving that availability. To decide how much money to invest in a high-availability data center, you must understand the value of the data center to the business. For example, each unavailable hour might cost a heavily used commerce Web site $100,000 in sales. This business impact can justify a major investment to prevent and minimize downtime and to ensure continuing customer goodwill. You must also understand that the costs of availability are nonlinear. For example, a five-minute outage may have far wider consequences and costs than five separate one-minute breaks in service.
- **Evaluate recovery point versus recovery time**—When setting high-availability goals, you must determine if it's more important to restore the data center to its exact state before failure, or to recover quickly, or both. The answer to this question is a critical factor in determining the server redundancy solution. You must determine if a solution that results in lost transactions is inconvenient, damaging, or catastrophic.

- **Plan when to maintain the data center—**To determine the best high-availability solution, you must understand when users need the data center. For example, if the data center is not heavily used or is not used at all at certain times, you can perform maintenance operations during these low-use times at reduced cost. You must, however, pay attention to whether people in different time zones use the data center in different time windows. If use in different time zones eliminates a time window, you must find an alternative approach for maintenance.

The ideal data center availability is 24 x 7 x 365 availability, or 100-percent availability. The percentage of uptime you should strive for is some variation of 99.*x* percent — with an ultimate goal of five nines, or 99.999 percent. Three nines (99.9 percent) is an achievable level of availability using a single data-center server. Achieving five nines (99.999 percent) is unrealistic for a single data-center server because this level of availability permits only about five total minutes of downtime in a calendar year. Four nines (99.99 percent) is achievable, however, by using fault resilient clusters with automatic failover. Five nines is achievable using advanced fault tolerant computers. This guide addresses the steps required to achieve these levels of availability, including the prerequisites to these technology solutions.

Even minimal scheduled downtime — such as 2 hours a month, or 24 hours a year — reduces availability to 99.73 percent. You can increase availability to 99.93 percent by reducing scheduled downtime to 30 minutes a month, or 6 hours a year. If you use the primary data center-server only for production purposes and perform database backups, health checks, and other tasks on secondary servers that have copies of the same data, the chances of achieving 99.999-percent availability increase.

So how many nines should you realistically pursue for a data center? Table 1.1 shows the levels of availability that have been achieved by some leading companies whose businesses depend upon high availability.

Table 1.1: Achievable levels of availability

Company	Level of availability	Description
NASDAQ	99.97%	Technology stock exchange with 2 million transactions a day (200 a second)
Barnes & Noble	99.98%	Electronic retailer with 5.6 million visitors a month
Quote.com	99.99%	Online financial site delivers 8.6 million page views a day
Buy.com	99.99%	Internet superstore with more than 2,000 concurrent visitors per minute

Because you can schedule planned system outages to have the least possible impact on a business, planned downtime is frequently treated differently than unplanned downtime. Whether planned downtime must be factored into the availability equation depends on business needs. A goal of four or five nines of availability for unplanned outages during scheduled business hours requires less of an investment than 24 x 7 availability, which must include both planned and unplanned system outages.

Identifying and Analyzing Barriers to High Availability

A high-availability barrier is anything that limits a data center's availability. Because it is impossible to protect a business from every barrier that might arise, you must estimate the effect of each barrier in advance and determine the barriers that are cost effective to overcome. To determine an appropriate high-availability solution, you must first identify and analyze each barrier for the following:

- The probable time the system will be down or that the barrier will cause the problem
- The probability that the barrier will occur and cause unavailability
- The estimated cost to overcome the barrier compared to the estimated cost of the unavailability

For example, to analyze the high-availability risk that a user will accidentally delete a portion of their data, you might do the following:

1. Estimate the time the data center will be unavailable because of this barrier to high availability.
   a. If a copy of the data exists on a redundant server (and the error is discovered before it is duplicated to the redundant server), the deleted data can simply be copied back to the production server.
   b. If a copy of the data does not exist on a redundant server or the data deletion has already occurred on the redundant server, you can restore the deleted data to an alternative server. After the accidentally deleted data is restored to the alternative server, it can be copied from that server to the production server by using Transact-SQL.
   c. If recovering from this user error requires a restoration from backup on the primary server, the time the data center is unavailable includes the time to restore from backup plus the time needed to resubmit the transactions that occurred after the deletion (if these transactions are available).
2. Estimate the probability that this barrier will occur. The probability is affected by the application design and the training provided to the data center users.

3. Estimate the cost to overcome this barrier. The cost to prevent unavailability resulting from this barrier depends on the solution you choose. In addition, the cost to overcome this barrier may include additional user training and perhaps a redesign of the application.
   a. If you maintain a redundant copy of the production database on a secondary server, the cost to overcome the barrier is measured by the cost of keeping a redundant copy of the data plus the time it takes to restore that data to the production server. This solution minimizes downtime, but it costs more to implement.
   b. If you rely on database backups, the cost to overcome the barrier is measured by the time it takes to restore the data from backup plus the time it takes to resubmit the transactions that occurred after the deletion. This solution results in more downtime, but it costs less to implement.

Note: When you evaluate the cost to overcome a barrier, remember that a solution that overcomes one barrier may also overcome numerous additional barriers. The cost of all of the barriers resolved by a solution must therefore be weighed against the cost of the solution. For example, keeping a redundant copy of the production database on a secondary server can overcome many barriers.

Barriers to high availability can be actual barriers or perceived barriers. *Actual unavailability* means that the data center is actually down. *Perceived unavailability* means that the data center is functioning, but is not available to the business user or customer because of intervening problems, such as a network, Web site, or DNS failure.

You must carefully evaluate each process and system element to identify and analyze actual and perceived barriers to availability. These barriers include the following:

- **Environmental issues —** Problems with the data-center environment itself can reduce availability. Environmental issues include inadequate cabling, power outages, communication line failures, fires, and other disasters.
- **Hardware issues —** Problems with any piece of hardware used by the data center can reduce availability. Hardware issues include power supply failures, inadequate processors, memory failures, inadequate disk space, disk failures, network card failures, and incompatible hardware.
- **Communication and connectivity issues —** Problems with the network can prevent users from connecting to the data center. Communication and connectivity issues include network cable failures, inadequate bandwidth, router or switch failure, DNS configuration errors, and authentication issues.

- **Software issues —** Software failures and upgrades can reduce the availability of a data center. Software failure issues include downtime caused by memory leaks, database corruption, viruses, and denial of service attacks. Software upgrade issues include downtime caused by application software upgrades and service pack installations.
- **Service issues —** Services that you obtain from outside a business can exacerbate a failure and increase unavailability. Service issues include poorly trained staff, slow response time, and out-of-date contact information.
- **Process issues —** The lack of proper processes can cause unnecessary downtime and increase the length of downtime caused by a hardware or software failure. Process issues include inadequate or nonexistent operational processes, inadequate or nonexistent recovery plans, inadequate or nonexistent recovery drills, and deploying changes without testing.
- **Application design issues —** Poor application design can reduce the perceived availability of a data center. Application issues include excessive blocking and locking, hard-coding of server names and IP addresses, and use of duplicate SQL Server logins.
- **Staffing issues —** Insufficient, untrained, or unqualified staff can cause unnecessary downtime and lengthen the time to restore availability. Staffing issues include insufficient training materials, inadequate training budget, insufficient time for training, and inadequate communication skills.

This series addresses the identification and analysis of each of these barriers to high availability.

Determining and Evaluating High-Availability Solutions

The high-availability solutions discussed in this series include procedural processes, database backups, redundant components, and redundant servers. All these solutions are required to achieve a highly available data center. The remaining chapters in the Planning Guide discuss issues related to these solutions.

After reading the Planning Guide, see the Solution Guide to learn how to implement the server redundancy solutions presented in Planning Guide Chapter 5.

2

Overcoming Barriers to High Availability

To achieve high availability, you must develop solutions to each barrier listed in Chapter 1. Overcoming these barriers requires following well-established best practices for high availability. To minimize and eliminate downtime, you must develop processes for carrying out operational and emergency tasks and escalating issues within your organization. You must also anticipate and plan for all types of failures, using redundant components to prevent failure and redundant servers to minimize downtime caused by unplanned failures.

To help organizations implement the required processes, Microsoft has developed Microsoft Operations Framework (MOF). MOF is a collection of best practices, principles, and models designed to help businesses achieve reliability, availability, supportability, and manageability for mission-critical applications built using Microsoft products and technologies. MOF divides the IT life cycle into identifiable tasks and functions, such as change management, system administration, security administration, and problem management. The best practices and policies discussed in this chapter are based on the recommendations contained in MOF. For more information, see "Better Manage Your IT Systems with the Microsoft Operations Framework" on the Microsoft Web site at *http://www.microsoft.com/MOF*.

Overcoming Environmental Barriers

You must design the data center to ensure high availability. The term *data center* refers to medium and large facilities. You can increase the availability of your data center by including the following:

- **Raised floors —** Raised floors provide space for the massive amounts of cable required for a data center, simplify the process of adding and moving equipment, and allow cool air to be pushed under raised floors and directed at servers and other heat-sensitive equipment. Without a raised floor, staff members can more easily trip over a cable and cause a server to fail. Also, if it is difficult for your staff to add or move equipment, these tasks can lengthen unavailability when equipment must be added or moved.
- **Fire suppression systems —** Good smoke detectors and fire extinguishers are crucial. Use a gas-and-water system to suppress fires. Install smoke detectors and temperature sensors throughout the data center so that you can monitor conditions and control them in zones. Also be sure that your staff can manually start and stop the fire suppression system.
- **Temperature controls —** Computer equipment reliability is better in cool conditions. Try to keep the data center at 68 degrees Fahrenheit (20 degrees Celsius). Although desktop computers and individual servers have fans to cool the CPU, fans do not cool the air enough for data-center servers and other heat-sensitive equipment. Do not rely on the centralized air conditioning in the building. Generally it is turned off in the evenings and on weekends. Use a dedicated, redundant cooling system to provide continuous cooling for the data center equipment.
- **Humidity controls —** High humidity can cause condensation on equipment. Very low humidity can lead to excess static electricity. Large fluctuations in humidity can cause circuit boards to expand and contract, damaging circuitry. Try to keep the data center between 40 to 45 percent relative humidity.
- **A redundant power system —** Prepare for both widespread and local power outages to avoid a lengthy service interruption. When a power outage occurs, a battery backup system can supply enough power for an orderly shutdown. If systems must continue to operate, install redundant backup generators to power-critical equipment, including the cooling system.
- **Redundant power supplies —** Within the data center, blown circuits or damaged wiring can cause a power outage to a rack of equipment or to individual components. Redundant power supplies to each rack of equipment can prevent a blown circuit from causing downtime. If a main circuit loses power, a redundant power supply automatically switches the rack to a second power supply.

- **Redundant data connections —** If voice and high-speed data connections to the data center fail, users cannot access the data-center servers and the data center staff will have difficulty communicating with standby sites. Ensure that voice and high-speed data connections are redundant. If one communications carrier loses service or must take its system down for maintenance, a second carrier lets users continue accessing the information they need and allows staff to communicate with standby sites if needed.

  To prevent damage to multiple lines during construction or maintenance, the redundant lines should enter the data center at different locations. For optimal service and cost-effectiveness, locate the data center facility near an Internet hub.
- **Backup systems —** Automated backup devices that mechanically insert and remove tapes are essential for large data centers. Ensure that your backup system can automatically perform all backups required by your disaster recovery plan without intervention by your staff.
- **Off-site storage —** If database backups are stored on site, a disaster at the data center might destroy both the production system and all backups of the production system. Storing backups off-site protects against this barrier. Consider storing backups at a secondary site as well as at an off-site location near the primary site.
- **Security precautions —** Intruders can access data by physically entering the data center, or they can obtain virtual access through a network connection. To stop intruders from physically entering the data center facility, require staff and visitors to provide credentials upon entry, and log all entries. Security cameras can also help. To stop virtual intruders, enforce security on each server, secure the network using a firewall, and require employees to use strong passwords that change frequently. Ensure that your staff stays current with hacking trends.
- **Space —** The data center facility should provide enough room for equipment to be organized, for growth, and for staff. Relocating equipment often causes downtime. In addition, adequate room for employees is important for employee productivity, which can increase availability.
- **Redundant data center facilities —** To provide protection against site-level disasters, deploy redundant servers in a secondary data center facility as well as in the primary data center. This arrangement ensures that you don't lose the secondary server during a catastrophe at the primary site. If all servers must be in the same facility, place them on separate power grids to provide some protection against localized disasters.

Ensuring that the data center itself is properly designed is a key component to increasing the availability of the data center.

Overcoming Hardware Barriers

A high-availability data center requires server-class hardware. Invest in high-quality components, redundant components at every point, and hot-swappable components whenever possible. You can overcome hardware barriers by using the following in the data center:

- **Up-to-date hardware —** Out-of-date components, firmware, or software drivers can cause software incompatibilities and result in availability problems. Installing up-to-date components with correct firmware and software driver revisions minimizes the risk that this barrier will reduce availability.
- **Certified computers, components, and configurations —** Using uncertified computers, components, and configurations for the data center is not supported by Microsoft Product Support Services (PSS). Use only computers, components, and configurations that are listed on the Microsoft Hardware Compatibility List (HCL). To search the HCL for certified computers, components, and configurations, see the "Hardware Compatibility List" on the Microsoft Web site at *http://www.microsoft.com/hcl/default.asp*.
- **Sufficient capacity —** Insufficient storage system, memory, or processor resources can cause the perception of unavailability by causing the data center to respond sluggishly to client requests, resulting in time-out errors. You resolve insufficient memory resources by adding more memory. You resolve insufficient processor resources by adding more or faster processors. You resolve insufficient storage resources by adding more disks, more controllers, and controllers that support multiple channels. More disks are faster than larger disks; more controllers split the I/O to the disks; and multiple channels support multiple pipes. To further increase the performance of the storage subsystem, use a storage area network (SAN). A SAN consists of multiple disks connected to one or more servers using Fibre Channel for high speed connectivity.
- **Redundancy solutions —** Use Microsoft® SQL Server™ database, file, and transaction log backups to protect data against hardware failures. Use redundant components and redundant servers to eliminate or limit downtime caused by hardware failures.

Ensuring that only top quality hardware is used will increase the availability of the data center. Because hardware failures do occur, however, you must also employ redundancy solutions to limit or eliminate downtime caused by hardware failures. For more information on redundancy solutions, see Chapter 4, "Preventing Downtime by Using Redundant Components," and Chapter 5, "Minimizing Downtime by Using Redundant Servers."

Overcoming Communication and Connectivity Barriers

The inability of clients to connect to and communicate with the data center causes a perceived lack of availability. To avoid this problem, ensure that your network is designed for high availability. Overcome communication and connectivity barriers by implementing the following:

- **Redundant networks and network cards —** Network cards and network paths can fail and prevent users from accessing the data center. Ensuring that the data center is accessible over multiple network paths using multiple network cards increases data center availability.
- **Multiple data center access paths —** The failure of a router or switch can prevent users from accessing the data center and can prevent your staff from reaching the Internet when troubleshooting a hardware or software problem. Use multiple switches and routers to ensure that users have a redundant path to the data center and that your staff can access the Internet.
- **Multiple domain controllers —** Users must be able to be authenticated by either SQL Server or the Microsoft Windows® operating system. To ensure that Windows users can be authenticated by the Windows operating system, use multiple domain controllers in the Windows domain to provide redundancy.
- **Multiple DNS servers —** Users must be able to locate the data center to establish a connection to the data center. On a Windows 2000 network, users find resources by using DNS. The failure of a DNS server can prevent users from locating the data center. Ensure that a secondary DNS server exists on the network that can direct users to the data center if the primary DNS server fails.

Ensuring that redundant network paths and components exist ensures that network infrastructure problems do not prevent users from communicating with the data center. Ensuring that multiple domain controllers exists guarantees that Windows users can establish a trusted connection to the data center.

Overcoming Software Barriers

You must ensure that the Windows operating system and the SQL Server service remain available to users. If either becomes unavailable, users cannot access the data center. Software barriers occur due to a variety of causes, including software failures, upgrades and maintenance, database corruption, user or application errors, viruses, and denial of service attacks. Use the following guidance to overcome each of these barriers.

- **Software failures —** The failure of the SQL Server service because of application errors, memory leaks, and excessive locking can cause data center unavailability. You must ensure that the applications deployed are well-designed and tested, that custom extended stored procedures do not cause memory leaks, and that the application design does not generate excessive locking.

- **Upgrades and planned maintenance —** SQL Server service pack installations and planned maintenance can require that the Windows 2000 operating system or the SQL Server service be restarted. A Windows service pack installation requires restarting the Windows operating system. The installation of a SQL Server service pack requires placing SQL Server into single user mode and then restarting SQL Server. In addition, other types of planned maintenance — such as adding additional memory, adding a hard drive, or upgrading a server application — may require a server to be taken offline. To avoid downtime caused by these software barriers, consider implementing either automatic failover clustering or automatic log shipping. For more information about failover clustering and log shipping, see Chapter 5, "Minimizing Downtime by Using Redundant Servers."
- **Database corruption —** Hardware failure can cause database corruption. To limit the downtime caused by database corruption, ensure that you back up all databases regularly and ensure the security of those backups. The only way to recover from database corruption is to recover from backup.
- **User error —** Accidental or malicious deletion or alteration of data can jeopardize the availability of the data center. To limit or eliminate downtime caused by user error, ensure that the database is backed up regularly. Depending on the extent of the error, you may be able to recover from it by using an alternative server to recover the data and applying it to the primary server. In a severe case, you may have to recover the primary server from backup. Without adequate backups, you must reconstruct the data, which will severely affect the availability of the data center.
- **Viruses —** Viruses can cause a data-center server to cease functioning. To reduce the threats caused by viruses, install the latest security patches and deploy virus protection software on all networked computers. Ensure that the virus protection software monitors all incoming and outgoing files and data for viruses.
- **Denial of service attacks —** Malicious users can attack Windows 2000 services and limit the availability of the SQL Server service by sending an overwhelming number of requests for access. To reduce the risk of denial of service attacks, restrict access to legitimate users only. You can use Windows authentication in SQL Server to restrict access to authenticated Windows users only. You can use Internet Information Services (IIS) to authenticate Internet users before they attempt to connect to SQL Server. You can also use IIS to restrict the IP addresses that can access SQL Server through the IIS server.

Eliminating software barriers before they cause a lack of availability is the best way to achieve high availability for a data center. However, software barriers are not always preventable. Application errors, user errors, and hardware subsystem failures can cause database corruption. Because these barriers cannot always be prevented,

you must consider mitigation solutions. For more information on these solutions, see Chapter 3, "Recovering a Data Center by Using Database Backups"; Chapter 4, "Preventing Downtime by Using Redundant Components"; and Chapter 5 "Minimizing Downtime by Using Redundant Servers."

Overcoming Service Barriers

Achieving a highly available data center requires services from outside hardware and software vendors. Unresponsive vendors and poorly trained vendor staff can reduce the availability of the data center. Your relationships with external vendors should include the following:

- **Service Level Agreements —** Negotiate a Service Level Agreement (SLA) with each of your major vendors to ensure a specific level of availability for the portion of the data center affected by their hardware or software. An SLA guarantees that a system will perform to specifications, support required growth, and be available to a given standard. Be sure that you are working with responsive vendors with well-trained staff members. Be sure that your staff is aware of the terms of each SLA. For example, many hardware vendor SLAs have clauses that require support personnel from the vendor or only specific, certified persons from your staff to open the server casing to replace defective components. Failure to comply can result in a violation of the SLA and potential nullification of any vendor warranties or liabilities. SLAs for data centers usually include (at minimum):
  - Percentage of availability (such as 99.99 percent)
  - Maximum number of concurrent users
  - Number of transactions to be supported per unit of time (such as 5,000 per second)
  - Method of contacting support personnel
  - Number of support calls allowable within a given period
  - Response time expected on support issues

  Note: Microsoft requires that an SLA be part of each Windows 2000 Datacenter Server solution.

- **Microsoft support contract —** In addition to an SLA, you should also negotiate a support contract for your Microsoft software.
- **Testing vendor support —** Make arrangements to periodically test escalation procedures by conducting support-request drills. Be sure you also test pagers and phone trees to ensure you have the most recent contact information.

If you are unable to solve a system issue quickly on your own, the absence of an SLA or support contract can increase the length of time the data center is unavailable. If you cannot access the vendor's support center or do not have necessary service account codes, the length of your service outage can increase.

Overcoming Process Barriers

Proper processes eliminate unnecessary downtime and ensure the most rapid recovery possible from service outages. Proper processes also ensure respect for your staff and ensure that you have the clout necessary to make the decisions required to protect the data center.

To overcome process barriers, you must incorporate the following:

- **Process management —** Develop and follow operational processes for performing routine tasks and emergency processes for responding to each type of disaster. Document the steps to follow for each task and disaster, and keep this document current. This document is called a *run book*. For information that the run book should contain, see Appendix A, "Contents of a Run Book."
- **Incident management —** Diagnosing that a failure has occurred and responding to the failure in a timely and proper manner is essential to achieving high availability. Incident management also encompasses monitoring to anticipate disaster before it occurs and to assist in diagnosing failures when they do occur.
- **Change management —** Establish a quality assurance (QA) environment to test all changes before they go into production. Maintaining a duplicate environment enables you to ensure that each alteration you make to the production environment succeeds and does not cause unnecessary downtime.
- **Maintenance management —** Perform periodic maintenance on the data center to prevent problems from occurring and to detect problems before they affect the availability of the data center.
- **Configuration management —** Develop a standard configuration for all data-center servers, such as standardized drive letters, mapped directories, and share names.

Process Management

Well-developed operational and emergency processes increase the availability of the data center by providing your staff with tested steps for resolving issues as they arise. Develop procedures for routine operational processes. Develop a complete disaster recovery plan consisting of the emergency procedures to be followed for each type of emergency. If your staff knows the precise steps that should be followed to repair a user error or to replace a failed disk, the risk of unnecessary downtime is diminished and necessary downtime is kept to a minimum.

Create a script library containing Transact-SQL scripts that your staff should use for routine operations and for recovery from each type of disaster. If your staff always performs the same procedure the same way, you will avoid problems and increase availability.

Ensure that the run book is easily accessible to all personnel who need it. For example, if all the blinking lights on the data-center servers suddenly go dark, an inexperienced engineer on night watch must be enable to locate and use the run book to follow the correct procedures. These procedures include the initial steps that should be followed, emergency phone numbers, and escalation procedures if service cannot be immediately restored.

As maintenance and emergency procedures change, you must update the procedures in the run book. Similarly, you must update contact information in the run book. Log all problems and their resolutions, whether trivial or major, as they occur. The resolution of past problems may help expedite the resolution of a future problem. If you ever have to use your disaster recovery plan, be sure to document the entire process. To determine the right level of detail, imagine how you would explain the sequence of events and your thought process to your peers.

Keep an up-to-date, offsite printed copy or encapsulated copy of your run book. Avoid maintaining a run book that needs a separately installed graphical user interface. Use a versioning tool to track changes to the run book.

Periodically test the disaster recovery plan to ensure that it works and is sufficiently detailed. The consequences of an untested plan can be very costly. Use drills to ensure that each member of your staff knows what they need to do in an emergency. Use a test environment to avoid causing downtime to your production system for these tests. Database backups and log shipping can be used to duplicate the database environment on a test platform. Log all tests of the disaster recovery plan. Note the day, start time, end time, whether or not it was a success, and in the event of a failure, why it happened. Logging this information is crucial for tracking as well as diagnosing.

Incident Management

Incident management consists of determining the cause of a failure, minimizing the time required to respond to a failure, and monitoring to anticipate or diagnose a failure. The availability of the data center is directly affected by how your staff responds to service incidents.

Diagnosing a Failure

After you have restored a server, try to determine the cause of a failure. Although some types of failure are difficult to assess, the following measures can help you in this effort:

- Document all of the processes that were running on the server when it failed. This will assist you in determining any dependencies that may have contributed to the failure.
- Back up the database and the transaction log for later analysis. Restoring the database on an alternative server and then restoring the transaction logs may help determine the cause of the failure.
- Document any error message that appears on the server console.
- Back up the operating system logs from the failed server, and compare them with historical backups of these logs to isolate anomalies.
- Back up and analyze the SQL Server and SQL Server Agent error logs to see what was going on at the SQL Server level at the time of the failure. Compare these logs with historical backups of these logs.
- If C2 auditing is enabled in SQL Server, back up and analyze the C2 log to help determine who or what caused the problem.

Note: The location of all log files should be standardized, well known, and documented in the run book.

Diagnosing the cause of a failure may help you prevent a similar failure in the future and correct the failure more quickly.

Response-Time Practices

To minimize the response time of your staff to high-availability issues, configure each data-center server to send an alert when it detects a condition that might lead to an error. The following practices assist in reducing response time:

- Send alert messages to a predefined group rather than to an individual. Doing so helps to ensure that alert messages are received. Be sure to specify a fail-safe operator in SQL Server to ensure that someone is notified if the designated individual or group is not on duty when the alert fires.
- Forward alerts and events for each of your data-center servers to a centralized server so that your staff can view all alerts and events from there.
- Standardize the physical appearance of all alert messages so that the appropriate response to an alert is easy to determine.

- Clearly document the meaning, priority, and suggested resolutions (or starting points) for each alert type. Clearly define individuals who must be notified if your staff cannot resolve the issue without downtime.
- If you are receiving alerts through e-mail, set up server-based rules in Microsoft Outlook® to prioritize your alerts. Forward high-priority alerts to your cell phone or pager when you are out of the office.
- Ensure that your staff knows whom to notify to escalate the alert and whom to contact if the first contact does not respond within a given period.
- Use a team Web site to display quick status information for each data-center server, alert messages, and broadcasts. Permit each staff member to post alerts, messages about upcoming projects, and system notes.

Monitoring Practices

To manage incidents, monitor the data center to anticipate disaster and to provide assistance in diagnosing problems and failures. Monitor the data center using Windows 2000 Performance Monitor and SQL Server Profiler. Use the following practices to monitor the data center:

- Establish a performance baseline to determine what is normal for each data-center server. This performance baseline shows resource usage changes over time and helps compare when troubleshooting.
  - Choose a large sample interval (more often than every 3 minutes) for a short duration (15 to 30 minutes) to collect information periodically.
  - Choose a smaller interval (every 15 seconds) to get a snapshot of how the system performs normally at different times of the day.
  - Save these samples in a drive other than the data or log file drives.
  - Synchronize the time on all servers to enable you to match information between servers when a problem occurs.
- Monitor for specific performance conditions on each data-center server, and use alerts to warn of impending problems before they affect availability. You can also configure an alert to initiate a job to perform additional monitoring when a specified condition is detected.
- Perform security audits for sensitive data access and administrative changes, and review the audit logs regularly.
- Monitor login failures to detect attacks on SQL Server. Monitor only what will be meaningful in the long-term; eliminate monitoring that does not return value proportionate to its cost.

Change Management

Unmanaged change to the data center is a major cause of data center failures. You must manage the following types of changes:

- **Planned changes —** Planned changes, such as hardware upgrades and changes in software versions, can be managed using a QA environment to test all changes before they are deployed to the production environment.
- **Emergency changes —** Emergency changes, which are changes necessary to avoid or eliminate downtime, cannot be thoroughly tested. These changes should be fully documented and then duplicated, where appropriate, to the QA environment to ensure that the two environments remain synchronized.

Planned Changes

To manage planned changes, establish a QA environment that is an exact copy of the production environment. For example, if your production environment is using automatic failover clustering, the QA environment must use automatic failover clustering. Use database backups or automatic log shipping to duplicate the production databases to the QA environment.

Use the QA environment to test all changes before they go into production. Create test scripts to adequately represent your production load during normal use and to emulate peak usage time. Use the following guidelines when deploying planned changes:

- Check all code — including schemas, scripts, and stored procedures — into a source control product such as Microsoft Visual SourceSafe® version control software. Doing so enables you to clearly identify each version of a script.
- Create a detailed plan for deploying completed code to the production environment, including time estimates and prerequisites such as service pack levels. Be sure that your change process is modular. You should be able to stop and restart (or roll back) the change at any point. Rely on scripts for making changes, and create a rollback script for every change.
- Schedule the change, and staff accordingly. Before you implement any change in the production environment, notify all affected parties.
- Back up the entire environment before implementing a change to ensure a rollback will be smooth if the change must be backed out.
- List every change and all associated information. Save e-mail related to the project, such as any mail showing discarded design ideas that you can use if the current design does not work as planned.
- Allow changes to be made only under the auspices of your staff.

Emergency Changes

When an issue is too urgent to undergo strenuous testing, you must forgo complete testing in the QA environment before implementing the change in the production environment. If time permits, perform at least limited testing in the QA environment. Never introduce a highly risky change into an unstable environment without testing carefully. Make only one change at a time, and record your observations before and after the change. Update the QA environment to ensure it remains in sync with the production environment.

To avoid the worst-case scenario where change becomes uncontrolled and changes are not tracked, follow these guidelines:

- Even in a crisis, allow time for the most recent fix to take effect before trying an additional fix.
- Allow only highly experienced senior staff to be involved in emergency changes.
- In a crisis, require junior staff members to notify a senior staff member that they need assistance.

Although not all changes can be controlled, they can be managed to eliminate the risk of unnecessary downtime.

Maintenance Management

Every active database needs occasional maintenance to detect and prevent problems before they occur. You can run DBCC (**CheckDB**, **CheckAlloc**, **CheckFilegroup**, and **CheckTable**) and DBCC **IndexDefrag** online without blocking updates. The impact is minimal with the recommended settings. Running tasks, such as DBCC **CheckDB** with the **physical_only** option or DBCC **IndexDefrag**, do not affect performance much.

However, running other tasks can affect availability. You must evaluate each task to determine its affect on availability. For example, dropping and recreating an index is faster than running DBCC **IndexDefrag** to perform the same task. If the index is a vital index, however, dropping and creating an index degrades performance until the index is rebuilt. This may be perceived as a lack of availability on a busy system. In the case of a vital index, DBCC **IndexDefrag** is the better choice for availability.

Evaluate each maintenance task to determine its impact on availability. Perform those tasks that have the highest impact on availability during off-peak hours or on a copy of the production database maintained on a secondary server. Performing maintenance tasks on a secondary server enables you to detect problems without affecting the availability of the production server.

In addition to database maintenance tasks, perform regular health checkups on the data-center environment. Periodically check hard disks to determine whether they need to be defragmented.

Configuration Management

Develop a standard configuration for all servers. Set guidelines specifying which drive letters to use for particular components or applications, such as *C* for operating system, *S* for SQL Server binaries, *L* for transaction logs, and *D* for data. Standardize how services, applications, and users access files and directories over your network, whether through mapped drives or shares. Never allow users to access a drive used by the SQL Server data and log files because user access can reduce the drive's availability by generating competing demands for disk access.

Services

Use a well-documented (or the same) domain user account for the SQL Server and SQL Server Agent services on all servers. Never use a real user account to run these services. Create a dummy account, name it clearly, and note in its profile not to delete or tamper with it.

Ensure that the password for this dummy account does not expire. Changing this password can require a lot of planning if you employ linked servers, remote procedure call (RPC), replication, log shipping, or clustering. If you must change the password, use SQL Server Enterprise Manager.

If SQL Server Agent is not running, the data-center server will be unable to issue alerts. To overcome this barrier, configure SQL Server Agent service to restart automatically and use a centralized monitoring system, such as Microsoft Operations Manager 2000 (MOM), that checks the services on each data-center server periodically. For more information, see "Get in Control of IT Operations — Microsoft Operations Management" on the Microsoft Web site at *http://www.microsoft.com/mom*.

Security

Secure each data-center server to ensure that only the people who need access to the servers can access them. Ensure that all administrator accounts use strong passwords. Allow access to critical servers only from trusted Internet Protocol (IP) addresses.

Use the same security model on all servers. For enhanced security, use Windows Authentication mode, rather than Mixed mode. Mixed mode permits users to connect using SQL Server login accounts, which are inherently less secure. Each staff member should log in to SQL Server using their Windows user account rather than the **sa** account. This enables auditing of all activity on each data-center server. Never use the **sa** account in code; you should be able to change the **sa** account password frequently, without notifying anyone or revising any code.

For more information, see:

- "SQL Server 2000 Security" on the Microsoft Web site at *http://www.microsoft.com/technet/prodtechnol/sql/maintain/security/sql2ksec.asp*.
- "Security Operations Guide for Windows 2000 Server" on the Microsoft Web site at *http://www.microsoft.com/technet/treeview/default.asp?url=/TechNet/security/prodtech/windows/windows2000/staysecure/DEFAULT.asp*.

Overcoming Application Design Barriers

High-availability solutions cannot compensate for bad application design. Poorly written application queries can cause blocking and excessive locking, which reduces the perceived availability of the data-center server. At worst, such queries can cause SQL Server to fail. While these application design issues can affect perceived and actual availability, they are beyond the scope of this document. There are a number of specific application design issues, however, that you must overcome to deploy the redundant server solutions discussed in Chapter 5, "Minimizing Downtime by Using Redundant Servers." These application design issues include the following:

- Do not hard-code server names, instance names, and IP addresses into the application. Creating n-tier applications that connect to the data center using COM+ objects enables you to create a more flexible disaster recovery plan.
- Use the same collation across all SQL Server instances and databases in the data center. Using a common collation enables you to easily restore a database to any server in the data center. If different servers and databases use different collations, you complicate your failover options because restoring a database with one collation to a server with a different collation requires precise coding. Your disaster recovery plan must take this into account.
- If you are using SQL Server logins, you must give application users unique names across all SQL Server instances in the data center to prevent user name conflicts when a failover occurs. Conflicts can occur if two applications share the same user name with different rights and responsibilities. These conflicts occur only when SQL Server logins are used. The uniqueness of Windows logins is enforced by the Active Directory® service.
- Do not code an application for a specific service pack level, although you might want to code for a minimum service pack level. If your disaster recovery plan requires one server to host more than one application, and if the application is not compatible with a certain service pack, the application will not be highly available.

You should ensure that software developers incorporate these best practices in all future development projects.

Overcoming Staffing Barriers

Only entrust qualified and trained employees to operate the data center and carry out the disaster recovery plan during a data center emergency. Unqualified or untrained employees can magnify a disaster and hamper a recovery.

You can facilitate shared ownership and collective responsibility for the data center by rotating team roles, using spare time effectively, presenting new information during meetings and training sessions, and promoting effective communication within the team.

The best practices for overcoming staffing barriers to high availability include the following:

- **Assigning roles —** Assign specific tasks and roles to individuals, and be sure each staff member knows what is expected to minimize confusion. Assign a primary and at least one secondary staff member to each server.
- **Rotating roles —** Rotating staff members among roles allows each staff member to learn new responsibilities and skills. You should rotate staff among projects, among classes of servers, and among production and QA environments. As each staff member becomes familiar with other systems, the number of issues that he or she can handle increases. As a result, team members do not need to call each other as frequently for answers, which can save valuable minutes when a server is unavailable.
- **Educating the staff —** Use spare time to educate team members about unfamiliar projects or systems, or about any new techniques. Provide books, journals, Web sites, and newsgroups for continual learning. Hold short weekly lunchtime presentations on high-availability issues. Invest in professional training and internal cross-training for your staff.
- **Increasing communication skills —** Encourage the development of both spoken and written communication skills. Increasing communication skills might involve formal training or courses in presentation skills, technical writing, and other communication skills. By establishing a high level of communication among staff members, you improve the staff's response time. Staff members must be instantaneously available to each other by cell phone or by e-mail during working hours — and sometimes during nonworking hours.

If your staff is well integrated, able to perform many roles, highly trained, and possesses excellent communication skills, the availability of the data center increases.

3

Recovering a Data Center by Using Database Backups

Despite the best-laid plans, you cannot prevent every disaster that might befall a data center. You must therefore ensure that you can recover the data center from backups after a catastrophic failure, such as a user error or a hardware-induced database corruption. This chapter discusses using database backups to ensure a complete recovery of the data center in the event of a disaster. It focuses on techniques to optimize the Microsoft® SQL Server™ 2000 recovery process and reduce the total recovery time to meet data center availability goals. A basic understanding of SQL Server backup and recovery procedures is assumed. For more information about SQL Server backup and recovery procedures, see "Backing Up and Restoring Databases" on the MSDN Web site at *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/adminsql/ad_bkprst_9zcj.asp*.

This chapter also discusses third-party solutions that extend the SQL Server backup and recovery technology to provide additional performance and functionality to meet the availability requirements of large databases.

Organizations that experience total data loss and do not have an adequate disaster-recovery strategy experience severe disruption to their business. An organization's ability to recover quickly from an outage or a disaster — from a component failure to the complete destruction of a site — directly contributes to its survival. Although other strategies should be implemented to minimize the need for recovery from backup, recovery from backup is the only solution to some types of failures and errors.

Solving High-Availability Barriers Using Backups

Recovery from database backups resolves data-center availability problems caused by storage subsystem failures and application or user errors. Some storage subsystem failures can also be resolved by using component redundancy, as discussed in Chapter 4, "Preventing Downtime by Using Redundant Components." Application or user errors can frequently be resolved only by using backups. In addition, catastrophic disasters, such as fires or floods, may require restoring from backup. Although server redundancy strategies can also be employed to minimize the need for recovery from backup when these types of disasters occur, do not overlook the importance of a good backup and recovery plan. When all other mitigation efforts fail, recovery from backup is the only solution.

Resolving data-center availability barriers caused by storage subsystem failures and application or user error are discussed later in this chapter.

Storage Subsystem Failure

A storage subsystem failure can result in data being unreadable or damaged. Using backups is the simplest and least expensive way to recover from unreadable or damaged data. The most common database problems caused by storage subsystem failures are inability to access data because of a complete hard-disk failure and torn pages caused by controller failure or a partial hard-disk failure.

- **Hard-disk failure —** A hard-disk failure is generally reported to the operating system as read or write errors. In the absence of redundant drives or servers, media failure causes the data center to become partially or completely unavailable. Recovery from this type of error, in the absence of redundancy, requires you to replace the failed drive. If the failed drive contains a data file, you must recover it. If the failed drive contains a transaction log file, you may have lost the most recent transactions. If these transactions are lost, users have to manually resubmit them.
- **Torn Pages —** A torn page is an incomplete write of a page caused by the operating system's inability to write to all of the disk sectors comprising a page. A page with incomplete data is a corrupted page. Torn pages are caused by disk-drive or disk-controller failures. A page is 8 KB and must always be read or written as a unit by the operating system on behalf of SQL Server. If an incomplete write occurs, SQL Server 2000 detects this as a torn page.

  If you experience torn pages, you must replace the device causing the error and repair the damage. You can repair the damage directly, recover data from backup, or fail over to an alternative server with a copy of the database that does not have the error. If you choose to repair the damaged data directly, run DBCC **CHECKDB** using the **physical_only** option to determine the extent of the damage. If the damage is to only one or more indexes, you can rebuild the

indexes rather than restoring them. This reduces the length of the unavailability. If the damage affects data pages, you need to recover from backup. When restoring from backup, identify the data files containing errors and recover only those data files containing errors. It is faster to recover one or more data files in a database than to recover the entire database.

Note: When recovering from torn pages, back up the tail of the transaction log before you begin recovery. This ensures that you can recover all committed transactions.

Although you can use redundant components and redundant servers to reduce or eliminate downtime caused by storage-system failures, sometimes database backups are the only way to recover from this barrier to availability.

Application or User Errors

Application and user errors can result in dropped tables, deleted rows, and incorrect updates. The error type, extent of the damage, and amount of work done before the error was detected determine the difficulty and type of repair. Repairing application or user errors manually and merging corrected data into the production database is time consuming and prone to errors.

When serious errors occur or when damage is extensive, restoring the database to a point in time before the error occurred is frequently the only solution. The SQL Server 2000 recovery tools perform recovery to any specific point in time if the database is using the full recovery model. If the database is using the bulk-logged recovery model, you can recover only to the end of a transaction log backup. If you are using the bulk-logged recovery model, performing frequent transaction log backups enables you to recover to a point that is close to the time the error occurred.

The primary disadvantage of recovering a database to an earlier point in time is that all work done since the error occurred is overwritten and therefore must be redone (or accepted as lost). The second big disadvantage is that the database will be unavailable for the duration of the recovery. When recovering to an earlier point, you must recover the entire database to that point. With a large database, recovering the entire database to an earlier point can take a long time unless you are using one of the third-party backup solutions discussed later in this chapter.

To avoid the problems associated with recovering the entire database to an earlier point, you can recover the entire database or just the affected data files to an alternative server, specifying an earlier point for the recovery. This leaves the production database available and preserves the work performed since the error occurred. In addition, restoring just the file or filegroups affected by the error is much faster than restoring an entire database. On the alternative server, you can then manually extract the deleted or damaged data or tables from the new database and merge or

insert it into the production database. Merging and inserting the damaged data into the production database without causing further damage to the production database is time consuming and risky. Often this type of repair is undertaken only in cases in which it is imperative that no committed work be lost or in which the time it takes to recover to an earlier point is too long.

Performing frequent full and differential database and transaction log backups helps reduce the time required to recover from application or user error. Redundant components and redundant servers do little to protect against this type of barrier to high availability. Adequate application testing and user training is the best defense against application and user errors.

Microsoft works closely with the following vendors, which offer tools that facilitate recovery from application or user errors:

- BMC Software
- Lumigent

These third-party tools include log analyzers that provide detailed information on the history of a database. Such tools can be used to locate the problem and identify a point in time for recovery. In addition, these tools can generate SQL queries that can undo an error. Other third-party tools can extract information directly from backups, reducing the amount of data that must be restored and merged with the current database to repair the problem.

Developing a Backup-and-Recovery Solution

Understanding your availability goals and the types of problems resolved by database restorations helps you develop a backup-and-recovery solution for your data center. After you have determined the acceptable length of downtime from each of the barriers that you intend to resolve by using database backups, use the following steps to develop a backup-and-recovery solution:

1. Determine whether you can meet these availability requirement by using the SQL Server backup-and-recovery tools. Measure the time required to recover from backup by measuring the total time to completely or partially recover a database from backup. If you cannot reduce the total recovery time to meet the availability requirement using the SQL Server 2000 recovery tools, you must either implement one of the third-party solutions discussed later in this chapter or revise the availability requirement.
2. After you have determined the technology solution, develop a database backup regimen consisting of full, differential, and transaction log backups that enables you to recover the database in the allotted recovery time.

3. Develop recovery procedures for each type of barrier. Include these procedures in the run book and in the associated scripts in your script library.

Your backup-and-recovery solution must be able to recover a damaged database in the data center in the allotted recovery time. To determine whether the SQL Server 2000 recovery tools can meet the availability requirement, use an identical test environment to measure the time required to recover the data center from database backups. Test combinations of full, differential, and transaction log backups. The longer the time between each full database backup, the longer the total recovery time if a database fails just before the next full database backup is scheduled to occur. Determine the best use of differential database backups between full database backups to reduce the total recovery time.

Measuring Total Recovery Time

Total recovery time is the time required to completely restore a database from backup, not just the time required to restore a single backup file. There are three methods you can use to obtain the total throughput for the recovery process:

- Use the message displayed by the SQL Server recovery process. The recovery process displays a message indicating the total number of pages transferred and the time in seconds for the transfer. Knowing that a page is 8 KB, you can calculate an average data rate. For small databases, the overhead involved in starting the process can account for a significant amount of the total time. This overhead can be ignored as insignificant for large databases, however.
- Use the length of the data-transfer phase by using messages recorded in the SQL Server error log. Using the times reported by these messages and the number of pages transferred, you can compute the average data rate excluding overhead.
- Use the backup throughput performance counter in System Monitor. This counter gives the instantaneous throughput while the recovery process is running.

Use this total recovery time calculation to optimize your backup schedules to achieve the best possible total recovery time. If this is still not sufficient to meet your availability requirements, use a third-party solution or revise your availability requirement.

Improving Total Recovery Time

You can take a number of steps to improve the recovery time by using the SQL Server 2000 recovery tools. These steps include improving the write performance of the data disk, striping the backup devices, backing up to disk rather than to tape, and adjusting the frequency and type of database backups.

Data Disks

During recovery from backup, SQL Server writes data from the backup device to the disks containing the database data files. To improve the performance of the data disks, follow these guidelines:

- **Use multiple data files —** To increase recovery write performance, create multiple data files for the database and spread these data files evenly across multiple matched disks. SQL Server automatically writes to multiple data files in parallel for increased performance. It is approximately three times faster to recover to three data files spread across three disks than to recovery to a single large data file on one disk.
- **Use a disk array—**To improve recovery write performance even further, place the data files on a disk array configured for either striped mirroring or striping with parity. Striped mirroring provides the best performance for database recovery. Striping with parity provides better performance than no array, but it has significantly slower write performance than striped mirroring. For more information on disk arrays, see Chapter 4, "Preventing Downtime by Using Redundant Components."

Use one of the above methods to increase the performance of the data disks and to reduce the total recovery time.

Backup Devices

During recovery from backup, SQL Server reads data from the backup device and streams the data to SQL Server, which then places it on the data disks. To improve the performance of the backup device, follow these guidelines:

- **Use a disk backup device —** To increase recovery read performance, recover from a disk backup device rather than a tape backup device. Recovering from backup files on a disk array is faster than recovering from a backup file on a single disk. Keeping the most recent database backups on a disk backup device can drastically reduce the total recovery time. For fault tolerance, be sure backups to disk are also archived to tape.
- **Use multiple backup devices —** Another way to improve recovery read performance is to stripe the backup file across multiple backup devices. SQL Server automatically reads from multiple backup devices in parallel for increased performance.
- **Use separate SCSI controllers —** To improve read throughput, do not share SCSI controllers between data disks and backup tape devices. This degrades recovery performance and can stop the recovery process entirely. To maximize read throughput, use a dedicated SCSI controller for each tape device.

- **Enable hardware compression —** To maximize read throughput, enable hardware compression on tape drives.

Use above methods to increase the performance of the backup device and to reduce the total recovery time.

Adjusting the Frequency and Type of Database Backup

To recover a database to the time of failure or to a specific point in time, you must restore the most recent full database backup, restore the most recent differential database backup, restore all transaction log file backups that are more recent than the last full or differential database backup, and manually initiate recovery. The time required to fully recover a database depends on the number and size of these backup files. To decrease the number and total size of these files and improve the total recovery time, follow these guidelines:

- **Perform more frequent full database backups—**Perform a full database backup as often as is practical to reduce the amount of data that must be recovered in addition to the most recent full-database backup. The practical frequency of a full database backup depends on the time required to perform a full backup and the impact of performing it on the data center. The longer the time between each full database backup, the more data has to be recovered from differential and transaction log backup files if the failure occurs just before the next full database backup. The more data that has to be recovered from differential and transaction log backup files, the longer the total recovery time.
- **Perform differential database backups—**Perform one or more differential database backups between each full database backup to reduce the number of transaction log backups that must be recovered to restore the database to the selected point in time. You need to recover only the transaction log backups that are more recent than the differential backup that is restored. The time required to perform a differential database backup is proportional to the number of extents modified since the most recent full database backup rather than the size of the actual database. As a result, performing a differential database backup on a large database can be much faster than performing a full database backup. The less frequently you perform full database backups, the more useful it is to perform frequent differential database backups.
- **Perform frequent transaction log backups—**Perform frequent transaction log backups to make it easier to recover to a specific point in time. If you use the bulk-logged recovery model for the database, you can recover only to the end of a transaction log file. More frequent transaction log backups give you more points in time to choose from. If you use the full recovery model for the database, you can recover to any point in time, but you can recover only to a read-only mode to the end of a transaction log file. Recovering to a read-only mode helps you determine when an error occurred.

- **Perform file or filegroup backups —** To assist in recovering rapidly to an alternative server, to resolve an application or a user error, or to recover to the time of a failure resulting from a disk failure, perform file or filegroup backups in addition to full database backups. File or filegroup backups enable you to more quickly recover a single data file or a filegroup to the production system or to an alternative system. Although you can recover a file or filegroup from within a full database backup, this takes longer than recovering from a file or filegroup backup because the recovery process must find the appropriate backup from within the full backup file. Performing file or filegroup backups provides the most performance benefit when used with medium and large databases. With large databases, also use differential file or filegroup backups between each file or filegroup backup. For more information about the benefits of differential backups, see "Perform Differential Database Backups" earlier in this chapter.

Use the above methods to decrease the number and total size of the backup files required to recover a database and reduce the total recovery time.

Using a Third-Party Backup-and-Recovery Solution

If you need additional performance or functionality beyond that provided by SQL Server 2000, consider implementing a third-party backup-and-recovery solution. Third-party solutions include support for extremely quick backups (snapshots) and platform backups, heterogeneous operating system backups, tape silos, tape RAID, and remote tape backups.

Using Snapshots to Improve the Recovery Time

In addition to the conventional backup technology used by the SQL Server 2000 recovery tools, SQL Server 2000 provides a high-performance backup interface called the Virtual Device Interface for Backup (VDI), for performing high-speed backups (snapshots) using third-party hardware and software. VDI is based on a shared memory design, which allows data to be transferred extremely rapidly with little overhead between SQL Server and a third-party snapshot solution. Third-party vendors offer two types of snapshot technologies, split mirror and copy-on-write, which accept the VDI data stream to back up a database at a very high speed. During restore, SQL Server receives the VDI stream from the snapshot to recover the database at a very high speed.

You can combine snapshot backups with differential, file differential, and transaction log backups to recover a database to the point of failure or to a specific point in time. Snapshots and conventional backups are recorded in **msdb**, so you can use SQL Server Enterprise Manager to identify the backups required for a particular restore sequence.

There are many technology variations, and the technologies are evolving. Consult with your storage and backup vendors to determine the exact features and benefits of their solutions. All vendors offer a variation of either split mirror snapshots or copy-on-write snapshots. The general features and functionality of these two types of snapshots are discussed below.

Split-Mirror Snapshots

Split-mirror snapshots are based on a striped-mirror storage solution. A split-mirror snapshot is an exact copy of a database at the time the mirror is split. This split-mirror snapshot is referred to differently by different vendors. The most common names are clone or business continuance volume (BCV). The backup of the database is a split-mirror snapshot plus a small amount of descriptive data created by SQL Server describing the snapshot.

You can use split-mirror snapshots to recover from application and user errors. You can rapidly recover the production database to a specific point in time or recover the snapshot to an alternative server to resolve application or user errors. You generally do not use split-mirror snapshots to recover from storage subsystem failures because mirroring itself, along with duplexed controllers, generally protects the database from hard-disk and controller failures. You can also use split-mirror snapshots to initialize a standby server for log shipping, a subscriber for transactional replication, or a test server for the QA environment.

Split mirror snapshots work as follows:

- **Splitting the mirror —** You perform a backup by splitting one drive from the mirrored array of drives. The split is a very fast operation. SQL Server suspends writes briefly to prevent torn pages; this is the only impact on the availability of the database. Typical split times are measured in seconds.

  Note: Use a three-way mirror so that the production volumes remain fault-tolerant after the mirror is split.

- **Remirroring —** In preparation for another backup, the clone volume must be rejoined to the mirror of the production volume. During this remirroring operation, the vendor software reconciles the clone volume with the data on the mirror. This reconciliation process occurs in the background, and the database remains fully available for updates. Some vendors copy only changed data from the production volume to the clone volume, while other vendors copy the entire contents of the production volume to the clone volume.
- **Recovering —** During recovery, the clone volume is the master and the SQL Server reconciles the production volume with the clone volume. As part of the recovery, the vendor software supplies SQL Server with the description of the backup being restored. With many vendors, the data is presented to SQL Server

immediately, and the reconciliation occurs in the background. In this case, the restoration occurs in seconds, and the database can be recovered and made available immediately. Users can actually use the data on the clone volume while the reconciliation occurs. To recover to a time more recent than the time of the snapshot, the database can be restored without recovery, and then you can apply differential, file differential, and/or transaction log backups before you initiate recovery.

Split-mirror snapshots eliminate the single most time-consuming phase of the recovery process — the restoration of a full database backup. This is the single most important benefit of split-mirror snapshots. Recovery to a different server can also be accomplished very quickly, providing a fast way to create a copy of the database for maintenance and reporting tasks.

The major disadvantage of this technology is its cost. Because split-mirror snapshots are based on mirroring, the number of separate physical disks required increases the cost of this solution. Other disadvantages include the following:

- **Volume-based backup —** Split mirrors work at the volume level. You must back up or restore an entire volume at one time. This means that you cannot store more than one database on a volume, or you will not be able to back up or restore the databases independently. Restoring one database corrupts the mirror of the other database.
- **Time to remirror limits frequency of backup —** Remirroring can take a significant amount of time for large databases. Although the database is available to users while the reconciliation occurs in the background, you cannot perform another backup until the reconciliation is complete. In addition, the length of time between backups increases the amount of data that must be rolled forward using transaction log backups after a failure. Transaction log backups must be applied to a snapshot to recover the database to a time more recent than the time of the snapshot.
- **Performance during remirror and recovery —** There is some performance degradation during remirroring and recovery. This degradation is typically small.
- **Maintaining a backup during remirror —** Remirroring destroys the snapshot, eliminating your backup. To solve this problem, you can:
  - Add a fourth mirror and create two clones. In this case, to preserve the most recent backup, always remirror the oldest clone when preparing to make another backup.
  - Copy the snapshot to tape or to other media. If you choose this method, consider the additional restore time required to get the copy of the clone back on disk if it is needed in an emergency.

- **No reusability of a backup after restoration**—Recovering a split-mirror backup converts that backup into a database. The backup no longer exists. Consider multiple clones or copying a clone to tape.

Microsoft works closely with the following storage vendors that offer split-mirror backup technology:

- EMC Corporation
- Hewlett-Packard Company
- Hitachi Data Systems

These storage vendors work with a number of third-party backup software products, such as CommVault Systems and Veritas, to provide support for split-mirror snapshots.

Copy-on-Write Snapshots

A copy-on-write snapshot is a copy of all the original disk blocks in a database that changed since the snapshot was created. Copy-on-write snapshots are implemented entirely in software and are available on all classes of storage. The copy-on-write snapshot and the production database share the majority of their disk blocks. They differ only in those disk blocks that have been modified since the snapshot was created. The older the snapshot is, the more likely it is that more blocks will change. An older snapshot also requires a larger space to maintain the snapshot. Ultimately, the snapshot can occupy as much storage as the original if a new snapshot is not created. Some vendors support multiple snapshots of the same volumes at different times.

You can use copy-on-write snapshots to recover from application and user errors. While copy-on-write snapshots are not as fast as split mirror snapshots, recovery using copy-on-write snapshots is significantly faster than recovery using conventional backups if the snapshot is recovered in place. Copy-on-write snapshots provide no protection against storage subsystem failures. To protect the data center against media failure, use a disk array (mirror, striping with parity, or striped mirror) or back up the copy-on-write snapshot to a disk or to a tape backup device.

Copy-on-write snapshots work as follows:

- **Creating the snapshot —** To create the snapshot, the snapshot software sets up an empty catalog of changed blocks, which takes little time. Initially, the snapshot shares all disk blocks with the production database.
- **Maintaining the snapshot —** When a write to the production data occurs, the change catalog is checked to determine whether the disk block that SQL Server is writing has changed since the snapshot was created. This read of the change

catalog on every write slightly degrades write performance. The action taken by SQL Server and the performance impact depends on whether the disk block was previously changed.

- If the disk block SQL Server is writing to has not been changed since the snapshot was created, the original disk block is copied from the production disk and saved to the snapshot. SQL Server then writes the change to the production disk. Copying this original block from the production disk to the snapshot seriously degrades write performance.
- If the disk block that SQL Server is writing to has changed, SQL Server simply writes the change to the production disk because the original block has already been written to the snapshot.

- **Backing up the snapshot —** When copying the snapshot to a backup device, the backup process reads the change catalog to determine where to find each block of data in the snapshot. This is a relatively slow process. Data blocks that have not changed are read from the production disk, and data blocks that have changed are read from the snapshot. When you back up the snapshot, ensure that a description of the snapshot is included in the backup. Whether this occurs automatically depends on the vendor's implementation of this technology.
- **Recovering —** Recovering the production database directly from the snapshot is the fastest restoration method. The original saved blocks are copied back to the production database, restoring it to its state at the time of the snapshot. The time to accomplish the restoration is proportional to the number of blocks that were modified since the snapshot was created. To recover to a time more recent than the time of the snapshot, the database can be restored without recovery, and then you can apply differential, file differential, and/or transaction log backups before you initiate recovery. Restoring the snapshot from a backup of the snapshot to the same or to an alternative server is identical to restoring a database using conventional SQL Server 2000 recovery tools.

The major advantage of copy-on-write snapshots over split-mirror snapshots is cost. Copy-on-write snapshots are less expensive because they require no special hardware and relatively little hard-disk space. The majority of the space in a copy-on-write snapshot is shared with the production database. In addition, with some vendors, you can create and maintain multiple backups simultaneously.

The major disadvantage of copy-on-write snapshots compared to split-mirror snapshots is the time required for recovery. Restoring a copy-on-write snapshot takes longer than a split-mirror snapshot, although it is faster than a conventional backup. Another disadvantage of copy-on-write snapshots is the performance degradation on disk writes. Copy-on write snapshots also provide no protection against media failure.

Using Third-Party Solutions to Increase Functionality

Third-party backup solutions also offer additional backup functionality, including support for the following:

- **Full platform backup —** Although SQL Server backup supports only SQL Server databases, third-party solutions provide the ability to back up the entire operating system when the SQL Server databases are backed up.
- **Tape silos —** Third-party backup solutions support media changers, which automatically change tapes as needed.
- **Backup device sharing —** Third-party backup solutions support remote tape devices and shared central libraries. Some third-party solutions support backup streams from many nodes to be multiplexed onto a single backup device.
- **Rich media management and backup catalog —** Third-party backup solutions offer sophisticated media management and tracking, including media life cycle management.
- **Tape RAID —** Some third-party party backup solutions support tape RAID for additional reliability.

The additional functionality provided by third-party solutions increases availability of the entire platform and the availability of the data center.

For more information, see:

- "Backup and Restore Solution" on the Microsoft Web site at *http://microsoft.com /technet/treeview/default.asp?url=/technet/ittasks/maintain/backuprest /default.asp*.
- "Microsoft Systems Architecture: Internet Data Center" on the Microsoft Web site at *http://www.microsoft.com/technet/treeview/default.asp?url=/technet /itsolutions/idc/default.asp*.

4

Preventing Downtime by Using Redundant Components

Hardware redundancy prevents downtime caused by hardware failures by detecting a failing component before it actually fails and bypassing a failure when it does occur. To achieve server hardware redundancy, deploy server-class hardware, a redundant storage subsystem, and redundant network cards. This chapter discusses using redundant components within a server to prevent downtime. Chapter 2, "Overcoming Barriers to High Availability" discusses the importance of redundant network infrastructure components, such as switches and routers, to ensure connectivity to the data center.

Using Server-Class Hardware

Server-class hardware monitors each critical server component for failure, notifies an administrator when a failure occurs, and includes redundant components that enable the server to work around the failure. Server-class hardware includes some or all of the following features:

- **Redundant power supplies —** Redundant server and disk array power supplies provide a secondary power supply if the primary power supply fails.
- **Redundant fans —** Redundant fans ensure that sufficient cooling exists inside of the server if a cooling fan fails.
- **Redundant storage subsystem —** A redundant storage subsystem provides protection against the failure of a single disk drive or controller. (See "Using a Redundant Storage Subsystem" later in this chapter.)
- **Redundant memory —** Redundant memory provides memory if a memory bank fails.
- **ECC memory —** Error-correcting code (ECC) memory detects and corrects single-bit errors and takes the memory offline if a double-bit error occurs.

- **Redundant network interface cards —** Using redundant network interface cards (NICs) ensures that clients can connect to the data center if a NIC or a network connection fails. (See "Using Redundant Network Cards" later in this chapter.)
- **Power-on monitoring —** When the server is initially turned on, the server detects startup failure conditions, such as abnormal temperature conditions or a failed fan.
- **Prefailure monitoring —** While the server is running, prefailure conditions are monitored. If a component, such as a power supply, hard disk, fan, or memory, is beginning to fail, an administrator is notified before the failure actually occurs. For example, a failure detected by ECC memory is corrected by the ECC memory or routed to the redundant memory, preventing a server failure. An administrator is immediately notified to rectify the memory problem.
- **Power failure monitoring —** When a power failure occurs, system shutdown software ensures a graceful shutdown if necessary in conjunction with an uninterruptible power supply (UPS).

Advanced server-class hardware can also include the following data-protection features:

- Lock-stepped processors, which are two processors that execute the same instruction stream and that crosscheck each other
- End-to-end checksums on data sent to storage devices
- Parity checking on internal buses

Using server-class hardware minimizes the likelihood of hardware failures and uses monitoring and redundancy to limit the effect of a failure on data-center availability. Server-class hardware provides a number of notification options, including on-screen messages, server logs, e-mail, and pager notifications. Some servers also include a battery-powered card that stores failure information that can be accessed even when a server has completely failed.

Using a Redundant Storage Subsystem

To ensure that your Microsoft® SQL Server™ data center is highly available, disks containing the operating system, SQL Server system and user databases and applications must be supported by a redundant storage subsystem. Even if the contents of a drive are read-only, it still takes time to replace a failed hard disk and to restore its contents. The data files and the transaction log files for these databases should be placed on separate disk arrays for maximum recoverability.

External disk arrays provide the best availability. The external disk array should contain redundant power supplies, redundant cooling fans, and redundant controllers. The disk array can be attached to the server by using either SCSI or FibreChannel. Although SCSI is less expensive, FibreChannel provides higher bandwidth and enhanced performance. The disk array can be managed by the

server or managed independently in a storage area network (SAN) using third-party software. A SAN is independent of any particular server or operating system and is attached to the servers using it through a dedicated FibreChannel network.

To provide disk redundancy in the event of a site failure, third-party hardware manufacturers provide remote mirror solutions that duplicate the locally redundant storage system to a remote site and provide transactional consistency between the sites. Remote mirror solutions require a SAN at each site and a dedicated fiber cable between sites that are no more than 100 kilometers apart. Although a remote mirror provides disk redundancy to a remote site, if you want a solution that provides automatic failover of the data center to a remote site, deploy a stretch cluster. For more information about stretch clusters, see Chapter 5, "Minimizing Downtime by Using Redundant Servers."

RAID

Configure the disks in an attached disk array or a SAN as a redundant array of independent disks (RAID). RAID has different implementations, with different levels of fault tolerance and performance. The most popular and common RAID implementations are:

- Striping (Raid 0)
- Mirroring (Raid 1)
- Striping with distributed parity (Raid 5)
- Striped mirroring (Raid 10 or Raid 1+0)
- Mirrored striping (Raid 0+1)

Different vendors use different terminology for implementations of striped mirroring and mirrored striping. Many use the terms interchangeably and incorrectly.

Note: Many manufacturers support additional RAID implementations providing advanced capabilities. These additional implementations are referred to in different ways, including *RAID 10E*, *RAID 30*, *RAID 50*, and *RAID 53*.

Regardless of the RAID implementation you choose, have at least one drive installed as a hot spare so that there is continuous fault tolerance when an active drive fails. Disk array controllers and SAN devices can automatically substitute a failed drive with an installed spare. Replace the failed drive immediately so that you always have a hot spare.

Striping

Striping writes logical blocks of data across two or more disks, creating a single logical volume with no redundant information between the disks. Striping is very fast and inexpensive, but it provides no fault tolerance. Do not use striping with a high-availability data center.

Mirroring

Mirroring writes all data across two or more disks, creating a single logical volume with completely redundant information on each disk. Mirroring provides a high level of availability; the mirrored volume can survive the failure of any disk in the mirror. Because each disk in the mirror provides redundancy, the available capacity of the mirrored volume is equal to the size of a single disk. (In a two-disk mirror, 50 percent of the disk capacity is used for data protection.) With mirroring, read and write performance is asymmetrical. Read performance is very fast because data is available from each disk in the mirror. Read performance can be even faster if the RAID array or SAN controller supports simultaneous reads from each member of the mirror. Writing to a mirror is slower than writing to a single drive because the data must be written to multiple disks. If the disks are on separate I/O buses using separate controllers — a process known as *duplexing* — you can achieve higher availability by eliminating the controller as a point of failure. Mirroring requires a minimum of two drives to implement.

Striping with Distributed Parity

Striping with distributed parity writes logical blocks of data across multiple disks, creating a single volume with parity blocks spread equally among all disk drives. Parity blocks enable recovery of the data on the RAID volume if a single disk fails. If two disks fail, however, the RAID volume fails. Striping with distributed parity is less expensive than mirroring because only the capacity equal to one disk is used for data protection.

Read performance is faster than mirroring because the data is read simultaneously from multiple disks. Disk writes are slower than mirroring because each write takes four I/O operations to create and write the data and parity blocks. These I/O operations occur consecutively; each I/O operation must wait for the previous I/O operation to complete. Because striping with distributed parity is more economical than mirroring, it is the most common RAID configuration. When a disk fails, however, read and write performance immediately decreases. Every read or write request to the failed drive initiates a verification process against all other drives. This reduction in performance continues until the failed disk is replaced and the system completely rebuilds the RAID volume.

Striped Mirroring

Striped mirroring writes logical blocks of data across two or more mirrored sets of disks, creating a single volume with no redundant information between the mirrored sets of disks. Each mirrored set of disks consists of two or more disks. Striped mirroring provides the same high level of availability as mirroring. The available capacity of a striped mirror is equal to that of each disk in the mirror set (in a two-disk mirror set, 50 percent of the disk capacity is used for data protection). Unlike

mirroring, however, striped mirroring provides symmetrical performance. Read performance is very fast because data is available from each disk in the mirror. In striped mirror implementation, RAID array and SAN controllers support simultaneous reads from each member of the mirror, which maximizes read performance. Write performance is also fast because writes occur simultaneously across each mirrored set.

With striped mirroring, if a drive fails, read and write performance is slightly reduced, but only for data stored on the mirror set containing the failed disks. Read and write performance to and from the other mirrored sets is not affected. Striped mirroring can survive the failure of multiple drives, provided at least one drive in each mirrored set survives.

Striped mirroring is the best solution for a database server requiring high performance and high availability. Capacity can be a problem, however, because of the number of disks required and their cost.

Mirrored Striping

Mirrored striping writes logical blocks of data across two or more disks, creating a single logical volume with no redundant information between the disks, and then mirrors the volume. Read and write performance is very fast, but mirrored striping is not as fault tolerant as striped mirroring. When a single disk in a stripe fails, the data remains available in the other stripe, but the stripe is broken. With one broken stripe, the failure of a disk in the remaining stripe causes the data to become unavailable. In addition, when a stripe fails, read performance is slower because only one stripe is functional.

Note: Because many vendors confuse the terms *RAID 10*, *RAID 1+0*, and *RAID 0+1*, use the terms *mirrored striping* and *striped mirroring*. To maximize availability, use striped mirroring.

Controller Cards

Use multiple SCSI or Fibre Channel controller cards in the external array for redundancy and to enhance performance through automatic load balancing. Choose cards with enough channels to split the logical grouping of disks — that is, data and logs — to reduce I/O contention.

Not all write caching is safe for a database server to use. Be sure that your disk controller prevents the uncontrolled reset of the caching controller, has on-board battery backup, and uses mirrored or ECC memory. Do not implement write caching unless the hardware vendor guarantees that the write cache includes all features required to prevent data loss.

If you are using failover clustering, using an internal caching controller can cause data corruption unless the controller mirrors its cache to all nodes of the cluster.

With an internal caching controller in a cluster, the cache in the controller on a failed node may contain completed transactions that are unknown to the surviving node (unless cache mirroring is used).

Data and Log File Placement

Place the data and log devices for each database on separate disks, using as many channels as possible to ensure redundancy in case a disk or mirrored set fails. Splitting the I/O between disks reduces performance issues caused by bottlenecks, which users can perceive as a lack of data-center availability. Use the following guidance when determining the type of storage redundancy for data and log files:

- **Data file drives —** For databases larger than a single disk, use striped mirroring for all data files for maximum availability and maximum performance. For databases that can be contained on a single disk, use mirroring. For very large databases spanning many disks, use striping with distributed parity unless the additional cost of striped mirroring can be justified.
- **Transaction log file drive —** Unless transaction log performance is an issue, use mirroring for all transaction log files. Where transaction log performance is an issue, use striped mirroring to increase performance. Do not use striping with distributed parity because of its poor write performance.

Note: The disks containing the operating system should be separate from the data and transaction log files. They should also be mirrored to ensure the operating system's availability.

Using Redundant Network Cards

To ensure that users can access the data center, use redundant network interface cards (NICs), and use NIC teaming to provide automatic failover between the NICs in the event of a failure. NIC teaming combines two or more physical NICs into a single logical NIC, which ensures that the data center always has an active link to the network. To use NIC teaming, connect each NIC card to a different switch on a different subnet. NIC teaming requires software from the NIC vendor, and each NIC is configured to use a common virtual IP address. When all NICs are working properly, their combined bandwidth is pooled for increased performance. When a teamed NIC begins to fail, the software stops using the failing NIC and routes all network communication over the remaining NIC or NICs. This failover process is transparent to the operating system and other devices on the network.

5

Minimizing Downtime by Using Redundant Servers

The final step in increasing the availability of a data center is to deploy one or more redundant servers to minimize downtime when a server fails. Use a redundant server solution only after you have followed the recommendations in the preceding chapters. Deploying and maintaining a redundant server solution adds complexity and cost to a data center, and if you do not follow the recommendations in the preceding chapters, your efforts might actually decrease data center availability. If you follow the recommendations and you deploy one or more of the redundant server solutions discussed in this chapter, you can achieve 99.99 percent to 99.999 percent availability for the data center.

A redundant server solution can minimize downtime caused by:

- Planned maintenance that requires taking a data-center server offline
- A hardware failure that causes a data-center server to become unavailable while the site remains available
- A disaster that causes an entire site to be unavailable, such as a regional power failure, a natural disaster, or man-made disaster

You can minimize data-center downtime that can result from these types of server unavailability by deploying one or more of the following server redundancy solutions:

- Failover clustering
- Log shipping
- Transactional replication

Note: Each of these server redundancy solutions is discussed in this chapter, followed by a decision tree that helps you choose among them.

Using Failover Clustering to Maintain a Redundant Server

Use failover clustering to increase the availability of a data center unless you cannot justify its financial cost or setup complexity. Failover clustering reduces the downtime caused by a server failure to less than one minute by automatically detecting a server failure and initiating the failover to a secondary server. Failover clustering requires no administrative intervention during the failover, and clients connect seamlessly to the promoted secondary server after the automatic failover.

Failover clustering uses the Microsoft Cluster Service (MSCS), which is a feature of Microsoft® Windows® 2000 Advanced Server and Windows 2000 Datacenter Server. MSCS allows two to four servers (nodes) to be linked in a server cluster utilizing a private network and a shared disk array. MSCS allows cluster resources to fail over from the primary node to one of the secondary nodes if specified cluster resources become unavailable on the primary node. MSCS supports two node clusters on Windows 2000 Advanced Server and four node clusters on Windows 2000 Datacenter Server.

Failover clustering allows you to install up to 16 Microsoft SQL Server™ virtual servers on one MSCS cluster. A SQL Server virtual server is a clustered instance of SQL Server.

Failover Clustering Architecture

Each node in an MSCS cluster is an independent computer with its own hardware resources and operating system. Each node also shares a hard-disk array with other nodes in the cluster. The primary node controls and uses the shared hard-disk array until MSCS initiates a failover to a secondary node. When you install SQL Server 2000 in a cluster, SQL Server setup installs the executable files on each node and installs the data and log files on the shared disk array. When a node fails, MSCS starts SQL Server on the designated secondary node, and the secondary node takes control of the shared hard disk array. SQL Server attaches the data and log files on the shared disk array to the newly started SQL Server instance.

Figure 5.1 illustrates the components of a SQL Server virtual server in a failover cluster.

Figure 5.1
Virtual instance in a failover cluster

A SQL Server virtual server includes the following components:

- **Shared storage resource —** One or more disks that are part of a cluster group and that are available to multiple nodes of a cluster, but to only one node at a time
- **Private storage resource —** One or more disks that are available only to each node in the cluster and that contain the operating system and executable files
- **SQL Server network name —** The virtual name assigned to the virtual server during installation, which is the name that all clients use to connect the clustered instance of SQL Server regardless of the node on which the instance is running
- **SQL Server IP address —** The virtual IP address of the virtual server, which is the IP address that all clients use to connect the clustered instance of SQL Server regardless of the node on which the instance is running
- **SQL Server resources —** The clustered SQL Server resources for a SQL Server instance include the SQL Server, SQL Server Agent, and Microsoft Search services (the clustered resources)

SQL Server 2000 supports single-instance clusters and multiple-instance clusters.

- **Single-instance clusters —** In a single-instance cluster, only one virtual server is installed in the cluster. The data and log files for the virtual server are installed on the shared storage resource for the cluster, and the executable files for the virtual server are installed on the private storage resource for each node. The

virtual server is owned by the primary node, and each secondary node is in a wait state. When the primary node fails or is demoted, a secondary node is enabled. When the node is enabled, SQL Server resources start on this node and take control of the data and log files in the shared storage resource. If each node is configured with identical hardware and software resources, the virtual server performs identically on the secondary node after the failover.

- **Multiple-instance clusters —** In a multiple-instance cluster, two or more virtual servers are installed in the cluster. The data and log files for each virtual server are installed on a shared storage resource dedicated to that virtual server. When the primary node for a virtual server fails or is demoted, the secondary node takes control of the shared storage resource for the virtual server. Because each virtual server has a dedicated shared storage resource, no other virtual servers are affected by this failover.

When determining whether to implement a multiple-instance cluster, assess the expected load of the database applications on each node, and determine whether a single node can handle the combined load during a failover. If a single node cannot handle the combined workload of all virtual servers, use two single-instance clusters instead, each with its own dedicated hardware resources.

MSCS uses a private network between nodes for monitoring. MSCS on the primary node monitors the SQL Server resource group to determine when to initiate a failover. MSCS on the secondary node monitors the shared storage resource to determine when to initiate a failover. MSCS uses the following mechanisms to determine when it should initiate a failover:

- **Shared storage resource mechanism —** MSCS on the primary node renews its reservation of the shared storage resource every three seconds. MSCS on the secondary node attempts to reserve the same shared storage resource every five seconds. MSCS on the secondary node will never reserve the shared storage resource if MSCS on the primary node continues to renew its reservation of the shared storage resource. If MSCS on the primary node fails to renew its reservation of the shared storage resource, however, MSCS on the secondary node seizes control of the shared storage and initiates a failover.
- **SQL Server resources mechanism —** MSCS on the primary node runs two tests to determine whether the SQL Server resources are functioning —a **LooksAlive** check and an **IsAlive** check. If either check determines that the SQL Server resources are no longer functioning, MSCS determines that a resource failure has occurred. Depending on the failover threshold configuration for the cluster, MSCS on the primary node either attempts to restart the failed SQL Server resource or immediately initiates a failover to the designated secondary node. If MSCS initiates a failover, it relinquishes its reservation of the shared storage resource.

If a failover occurs because the primary node fails, MSCS starts the resources in the SQL Server resource group on the secondary node. On startup, SQL Server initiates automatic recovery for each database, rolling forward all completed transactions and rolling back all incomplete transactions. When SQL Server completes automatic recovery, SQL Server is available to its consumers with the same virtual server name and IP address as before the failover. The time required for the failover is short, generally under one minute. The actual time is governed by the number and size of the transactions that must be rolled forward or back at startup and by the speed of the shared storage resource.

The end user, client application, Web server, and middle-tier component (such as Microsoft Transaction Server) experiences a break in the connection to the virtual server when a failover initiates. Cluster-aware applications detect the failover and reconnect automatically to the virtual server, which is running on a new node. If a SQL Server client is not cluster aware, the user must manually reconnect the client to the virtual server.

For more information, see "SQL Server 2000 Failover Clustering" on the Microsoft Web site at *http://www.microsoft.com/SQL/techinfo/administration/2000/failovercluster.asp*.

Failover Clustering Advantages

There are a number of advantages to using failover clustering rather than log shipping or transactional replication. These advantages include the following:

- **Automatic failover —** Switching between nodes occurs automatically whenever a failure is detected. Failover occurs when a SQL Server resource, the Windows operating system, or essential hardware on the primary node fails.
- **Simple manual failover —** You can manually switch the virtual server from the primary node to a secondary node. Simple manual failover enables you to perform normal maintenance on one node while the virtual server continues to run on another node.
- **Transparent to clients —** Failover is relatively transparent to the clients consuming SQL Server resources. After a failover is complete, clients simply reconnect to the new primary node (the promoted secondary node) with the same virtual name and IP address to resume work. Although clients must resubmit any incomplete transactions, they do not need to connect to a different server name or IP address because the failover cluster uses a virtual server name and IP address.
- **Transactionally current —** When MSCS initiates a failover, all committed transactions are preserved and are available to clients when the failover process is complete.

- **Rapid failover —** Completing the failover is fast, depending upon how many transactions need to be rolled forward or rolled back and the speed of the shared storage resource.

These advantages make failover clustering the first choice for increasing data center availability.

Failover Clustering Limitations and Disadvantages

There are a number of limitations and disadvantages to using failover clustering rather than log shipping or transactional replication. These limitations and disadvantages include the following:

- **Proximity requirement —** The round-trip latency of network communication between nodes must not exceed 500 milliseconds, or MSCS will initiate failovers when no actual failure has occurred. This means that all cluster nodes must be located close to one another, unless a third-party clustering solution is used to place nodes in geographically dispersed locations (a stretch cluster). If you deploy a stretch cluster, you must use a third-party solution that appears on the Hardware Compatibility List. To search the HCL for certified computers, components, and configurations, see the "Hardware Compatibility List" on the Microsoft Web site at *http://www.microsoft.com/hcl/default.asp*. Microsoft Product Support Services (PSS) does not support any clustering solutions that have not been certified. For more information, see article Q280743, "Windows Clustering and Geographically Separate Sites," in the Microsoft Knowledge Base at *http://support.microsoft.com/default.aspx?scid=kb;[LN];Q280743*. Third-party clustering solutions are more expensive and more complex than MSCS.
- **Hardware requirement —** MSCS requires specifically certified hardware.
- **Software requirement —** MSCS is available only with Windows 2000 Advanced Server and Windows 2000 Datacenter Server, and failover clustering is available only with SQL Server 2000 Enterprise Edition.

If these disadvantages outweigh the advantages of failover clustering for a data center, you should consider log shipping to determine whether it provides a better high-availability solution for the data center.

Using Log Shipping to Maintain a Redundant Server

Log shipping is designed as an inexpensive high-availability solution that can be used to protect a database against the failure of the primary server. Use log shipping rather than failover clustering if you cannot justify the financial cost and complexity of failover clustering. Also use log shipping with failover clustering to protect against site-level disasters as well as local server failure. Log shipping enables you to maintain a copy of the production database on one or more secondary servers and to easily promote one of the secondary servers to become the new primary server.

Log shipping uses SQL Server backup and restore capabilities to provide database redundancy between the production database on a primary server and a standby database on one or more secondary (standby) servers. Log shipping requires an administrator to detect a failure and initiate a role change. A role change promotes a standby server to a primary server by recovering the production database on the standby server. The role change process is not automatic and takes at least several minutes to complete. It can take significantly longer with large databases. After a role change, clients must connect to a different server with a different server name and IP address. Unlike with failover clustering, virtual server names and IP addresses are not incorporated into log shipping.

Log Shipping Architecture

SQL Server 2000 Enterprise Edition has built-in support for log shipping, whereas SQL Server 2000 Standard Edition requires a custom implementation of log shipping. The built-in log shipping architecture consists of a primary server, one or more standby servers, and a monitoring server. These servers perform the following roles:

- **Primary server —** The *primary server* is a server running SQL Server 2000 and containing the production database. You create and use a full database backup of this database to initialize a standby database on each standby server. SQL Server Agent jobs make periodic transaction log backups of the production database to capture changes made to the production database.
- **Standby servers —** *Standby servers* are servers running SQL Server 2000 and containing an unrecovered copy of the production database from the primary server. SQL Server Agent jobs periodically copy transaction log backups from the primary server and restore these transaction log backups, without recovery, to each standby database. The restoration process updates the standby database on each standby server with new transactions from the primary server. A standby database can be recovered, and the standby server can replace the primary server if the primary server fails or must be taken offline for maintenance.
- **Monitoring server —** A *monitoring server* is a server running SQL Server 2000 that monitors the status of the log shipping jobs on the primary server and on each standby server. To enable you to detect a failure to the primary server or to any standby server, use a server other than the primary server or a standby server as the monitoring server.

To create a custom log shipping solution, create each of these roles manually.

Note: You can use a standby server to contain standby databases from multiple primary servers and a monitoring server to monitor multiple primary server-standby server pairs.

Figure 5.2 illustrates how log shipping works.

Figure 5.2
Log shipping

When you configure log shipping, you restore a copy of the production database without recovery on each standby server. Thereafter, log shipping uses the following SQL Server Agent jobs and alerts to 1) maintain an up-to-date copy of a production database on one or more standby servers, and 2) report the success or failure of these jobs to the monitoring server and to administrators:

- **Transaction log backup job —** The *transaction log backup job* runs on the primary server and backs up the transaction log of the production database on the primary server according to a schedule. This job also records the results of each execution in a status table on the monitoring server. To ensure that this backup remains available if a disk fails, use a fault tolerant storage location for the backup file. Share these backup files so that the transaction log copy job on each standby server can copy them, but strictly limit access through this share to protect sensitive information. Schedule the backup job to run every 1 to 15 minutes. The more frequent the backup, the smaller the size of each backup file and the more closely you can synchronize the standby database on each standby server.
- **Transaction log backup copy job —** The *transaction log backup copy job* runs on a standby server and copies each transaction log backup from the network share on the primary server to the standby server according to a schedule. This job also records the results of each execution in a status table on the monitoring server. Schedule this job to run with the same frequency as the transaction log backup job. Copying each backup to a standby server as soon as it is created is important because the purpose of log shipping is to have an up-to-date copy of the production database on a standby server in case the primary server fails.
- **Transaction log restore job —** The *transaction log restore job* runs on a standby server and restores transaction log backups to the standby database according to a schedule. Restore the transaction logs without recovery to allow additional transaction logs to be restored. To minimize the time required to complete a role change, schedule the restore job to run with the same frequency as the copy job. This keeps the standby database synchronized to within several minutes of the production database. If you also want to use the standby database for reporting, however, you cannot restore the transaction logs as frequently without interfering with its use for reporting. SQL Server requires exclusive access to the standby database to restore transaction logs or the transaction log restore job will fail. To ensure that the transaction log restore job does not fail, configure the restore job to terminate all users every time it runs.

  You must choose between maximum synchronicity and additional functionality. For example, if you run the transaction log restore job once a day, users can access the standby database for reporting, but it will take longer to recover the standby database and bring the standby server online to replace a failed primary server. Delaying the application of transaction log backups to the standby database increases the likelihood that you will discover user errors, application errors, or database corruption before the backups containing those errors are restored to the standby database.
- **Backup alert —** The *backup alert* runs on the monitoring server and monitors the success and failure of transaction log backup jobs on the primary server. Specify the threshold for alerting an administrator that backup jobs are failing. To reduce the risk of receiving false alerts, set a threshold that avoids false alerts by allowing for large transaction log backup files.

- **Out of sync alert —** The *out of sync alert* runs on the monitoring server and monitors the synchronicity between the production database and the standby database. Specify the threshold for alerting an administrator that the standby database has exceeded a specified level of synchronicity. To reduce the risk of receiving false alerts, set a threshold that tolerates the time required to copy large transaction logs across the network and to restore them on the standby server.

If you are using SQL Server Enterprise Edition, use the Database Maintenance Plan Wizard to create these jobs and alerts automatically. If you are using SQL Server Standard Edition, create these jobs manually.

Because log shipping copies only database objects in the production database, database objects that are stored in the **master** and **msdb** databases or in the file system must be manually copied and recreated on a standby server either before or when a role change is initiated. To accomplish this, copy the following objects on the primary server:

- **Server logins —** Create a Data Transformation Services (DTS) job by using the transfer logins task that transfers logins from the primary server to a standby server. Create a separate DTS job for each standby server. Back up the **sysxlogins** table by using the Bulk Copy Program (BCP) utility to capture the server-level logins. Store this backup file on each standby server. Update this backup each time new logins are added to the primary server.
- **Other database objects —** Generate Transact-SQL scripts to recreate all jobs, alerts, and user-defined error messages on the primary server. Execute these scripts on each standby server. Update these scripts as these objects change, and use these scripts to update each standby server.
- **DTS package —** Store a copy of each DTS package on each standby server. Update the copy of each DTS package as they change.

The log shipping jobs keep the standby database synchronized or ready to be synchronized. Manually copying and scripting objects stored outside of the production database ensures that the standby database can take the place of the production database during a failure or planned maintenance.

If you are using the standby server for standby databases from multiple SQL Server instances, the following considerations apply:

- SQL Server logins from different SQL Server instances may have identical names with different permissions, which can create a security conflict.
- Database objects from different SQL Server instances must have different names, or they overwrite each other.
- DTS packages from different SQL Server instances or servers must have different names, or they overwrite each other.

If you take these considerations into account, a SQL Server instance on a standby server will be able to contain the standby database from multiple primary servers.

Firewall Considerations

If the primary server and a standby server are separated by a firewall, you need to open the following ports on the firewall:

- **Port 1433 —** This port is required for SQL Server operations between default instances. This port must be open in both directions. If you are using a named instance of SQL Server, you must open the port used by the named instance.
- **Port 135 —** This port is required so that SQL Server Agent can copy transaction log backup files and other synchronization files to the shared folder on the standby server
- **Ports 137, 138, and 139 or 445 —** These ports are required to use Uniform Naming Convention (UNC) shares.

These are the only ports that need to be opened for log shipping. Other Windows services and application may require additional ports to be opened.

Log Shipping Role Change

Unlike failover clustering that supports automatic failover, changing log shipping server roles is a manual process. You must determine when to initiate a role change and manually execute a number of tasks to complete it. Changing log shipping roles requires you to perform the following tasks on the selected standby server:

- **Back up transaction log —** If the primary server is still functioning, terminate all user connections and perform a final backup of the transaction log to capture all completed transactions that have not yet been backed up. Back up the transaction log by using the **No Recovery** option. Doing so enables you to restore transaction log files to log-shipped database without restoring it from a full database backup. Disable the transaction log backup job, and restrict client access to the production database to ensure that clients cannot change the database. SQL Server Enterprise Edition provides the **sp_change_primary_role** system-stored procedure to accomplish this task.

  If the primary server fails, completed transactions that are not backed up and copied to a standby server are lost. End users have to resubmit these transactions.
- **Copy transaction log backup —** If the primary server is still functioning, copy any transaction log backup files that are not already copied to the standby server that you want to promote. SQL Server Enterprise Edition provides the **sp_change_secondary_role** system-stored procedure to accomplish this task.

  If the primary server fails, you will not be able to restore to the standby database any transaction log backup files that were not copied to a standby server before the primary server failed.

- **Restore transaction log backups, with recovery —** On the standby server, restore, with recovery, all transaction log backup files that are not already restored to the standby database. After recovery, the standby database on the standby server will be able to accept client connections. SQL Server Enterprise Edition provides the **sp_change_secondary_role** system-stored procedure to accomplish this task.
- **Resolve SQL logins —** Resolve SQL Server logins to user accounts in the standby database. Use the **sp_resolve_logins** system-stored procedure to load the saved BCP file containing logins from the primary server. This stored procedure resolves the SID values between the production database and the standby database. This task must be performed after the standby database is recovered.

When these tasks are complete, the new primary server is ready to receive client requests. Because the production database is residing on a different server with a different server name and IP address, however, you must manually point clients to this new server. You accomplish this task in different ways, depending on the design of the client applications and the use of middle-tier objects. You may be able to change the server name or IP address in a COM+ object or in DNS, or you may have to make a change to the Data Source Name (DSN) at each client computer. You may also be able to create a Network Load Balancing (NLB) cluster and virtualize the server name and IP address if the primary server and the standby server are on the same subnet. For more information about using NLB with log shipping, see "Redirecting Client Network Traffic to a Promoted Secondary Server" later in this chapter.

Note: If an application has a hard-coded server name, you cannot use log shipping without recoding the application or using an alias of the same name on the client that you redirect to the primary server when the primary server changes.

After a log shipping role change, you can either continue to use the new primary server as the primary server or bring the original primary server back online after it has been repaired or the maintenance task is finished. If you continue to use the new primary server as the primary server, enable log shipping to the original primary server (configure it as a standby server) to provide for database redundancy if the new primary server fails. If you want to bring the original primary server back online, back up and copy the transaction log from the new primary server to the old primary server, and then recover the original primary database. When these tasks are complete, you can point clients back to this original server and re-enable log shipping to the standby server to reinstate redundancy.

For more information about SQL Server log shipping, see article Q314515, "Frequently Asked Questions — SQL Server 2000 — Log Shipping," in the Microsoft Knowledge Base at *http://support.microsoft.com/default.aspx?scid=KB;EN-US;q314515&*.

Log Shipping Advantages

There are a number of advantages to using log shipping rather than failover clustering or transactional replication. These advantages include the following:

- **Hardware requirement —** Log shipping does not require specifically certified hardware.
- **Software requirement —** Built-in log shipping is a standard component of SQL Server 2000 Enterprise Edition and custom log shipping can be manually configured with any SQL Server 2000 edition.
- **Less network traffic than transactional replication —** Log shipping generates less network traffic than transactional replication does.
- **Site protection —** Log shipping to a remote standby server provides protection against catastrophic site failure.

These advantages make log shipping the logical choice for a high-availability solution if you cannot justify the financial cost and setup complexity of failover clustering, or if you want to protect against server and site disasters in a single solution. You can also use log shipping with failover clustering to provide an inexpensive high-availability solution for site-level disasters with an automatic failover solution for local server failures and routine maintenance operations.

Log Shipping Limitations and Disadvantages

There are a number of limitations and disadvantages to using log shipping rather than failover clustering or replication. These limitations and disadvantages include the following:

- **Manual detection and role change —** Log shipping requires manual intervention by a database administrator (DBA) to detect a failure and to initiate a role change. Initiating a role change requires using a number of stored procedures to recover the standby database. In addition, you must manually ensure that required objects or files not stored in the production database are kept synchronized between the primary server and the standby server.
- **Dual purpose versus minimal latency —** You cannot use a standby database for secondary purposes, such as reporting or routine maintenance tasks, without increasing the transactional latency of the standby server and increasing the time required to bring a standby server online when the primary server fails.
- **Potential transaction loss —** If the primary server fails, completed transactions that have not been backed up and copied to a standby server may be unrecoverable. You can reduce this risk by using fault tolerant storage for the transaction logs, increasing the frequency of transaction log backups, using fault tolerant storage for the transaction log backups, and immediately copying transaction log backups to fault tolerant storage on a standby server.

- **Lack of transparency to clients —** After a role change, clients must connect to a different server with a different server name and IP address. You must manually modify each client to connect to this new server or incorporate an intermediate mechanism to accomplish this task for these clients. Intermediate mechanisms include NLB, DNS, and COM+ components. Each of these intermediate mechanisms requires a manual modification by a DBA to point clients to the new server name and IP address. This lack of transparency increases the time required to complete a role change.

If these disadvantages outweigh the advantages of log shipping for a data center, you should consider transactional replication to determine whether it provides a better high-availability solution for the data center. If neither log shipping nor transactional replication provides the high-availability advantages required for the data center, reconsider failover clustering.

Using Transactional Replication to Maintain a Redundant Server

Use transactional replication rather than failover clustering or log shipping if you want an inexpensive solution that allows a secondary server to be used for reporting as well as for redundancy. Transactional replication can also be used with failover clustering to provide an inexpensive solution to site-level disasters and provide one or more servers that can also be used for reporting. Because transactional replication is not designed for high availability, promoting a secondary server is not simple. In addition, reverting to the original primary server after a secondary server is promoted requires a complete database restoration. Finally, transactional replication requires preparation of the schema for replication. Identity columns and triggers must be set to **Not For Replication**; literals must be defined with apostrophes rather than quotation marks; primary keys must be defined for every table; and database objects must not include unresolved references.

Transactional replication uses SQL Server Agent jobs to copy each change made to a production database to a copy of the database on one or more secondary servers. Transactional replication requires an administrator to detect a failure, disable replication, and designate one of the subscribers as the new primary server. This process is not automatic and takes at least several minutes to complete. It can take significantly longer with a large database. After a role change, clients must connect to a different server with a different server name and IP address. Unlike with failover clustering, virtual server names and IP addresses are not incorporated into transactional replication.

Transactional Replication Architecture

The transactional replication architecture consists of a primary server (the publisher), a distribution server (the distributor), and one or more secondary servers (the subscribers). These server roles perform the following functions:

- **Publisher —** The *publisher* is the server containing the production database (the publication database).
- **Distributor —** The *distributor* is the server containing the *snapshot folder* and the distribution database. The snapshot folder contains a snapshot of the publication database. The distribution database stores changes made to the publication database since the snapshot was taken. In a high-availability solution, the distributor role should be on the same server as one of the subscribers to eliminate additional points of failure. To increase availability in the event of a site failure, place the distributor/subscriber in a separate site from the publisher.
- **Subscribers —** A *subscriber* is a server containing a copy of the publication database that is periodically updated from the distribution database.

Figure 5.3 illustrates how transactional replication works. This diagram illustrates the placement of the distributor role on a subscriber. A single subscriber is shown for the sake of exposition. In a high-availability solution, you may use multiple subscribers, which are geographically distributed.

Figure 5.3
Transactional replication

SQL Server uses the following SQL Server Agent jobs (agents) to initialize and maintain an up-to-date copy of the publication database on one or more subscribers:

- **Snapshot agent —** The *snapshot agent* creates the initial snapshot of the publication database and stores this snapshot in a shared folder on the distributor. The snapshot agent places a mark in the transaction log demarking the time of the snapshot. The log reader agent uses this mark to determine the transactions to copy to update the snapshot on subscribers.
- **Log reader agent —** The *log reader agent* monitors the transaction log for the publication database and copies new transactions to the distribution database. For minimum latency, run the log reader agent continuously.
- **Distribution agent —** The distribution agent initializes a copy of the publication database on each subscriber by using the initial snapshot. Thereafter, the distribution agent periodically copies transactions stored in the distribution database and applies them to the publication database copy on each subscriber to keep the database current.

Because transactional replication copies only database objects in the publication database, database objects that are stored in the **master** and **msdb** databases or in the file system must be manually recreated on a subscriber either before or when a role change is initiated. To accomplish this, perform the following tasks on the publisher:

- **Server logins —** Create a DTS job by using the transfer logins task that transfers logins from the publisher to a subscriber. Create a separate DTS job for each subscriber. Back up the **sysxlogins** table by using BCP to capture the server-level logins. Store this backup file on each subscriber. Update this backup each time new logins are added to the publisher.
- **Other database objects —** Generate Transact-SQL scripts to recreate all jobs, alerts, and user-defined error messages on the primary server. Execute these scripts on each subscriber. Update these scripts as these objects change, and use these scripts to update each subscriber.
- **DTS jobs —** Store a copy of each DTS job on each subscriber. Update the copy of each DTS job as these jobs change.

The transactional replication jobs keep the standby database synchronized. Manually copying and scripting objects stored outside of the publication database ensures that the database copy on a subscriber can take the place of the publication database if a failure occurs.

Firewall Considerations

If the publisher and the subscriber/distributor, or a subscriber and the subscriber/distributor, are separated by a firewall, open the following ports on the firewall:

- **Port 1433 —** This port is required for SQL Server operations between default instances. This port must be open in both directions. If you are using a named instance of SQL Server, you must open the port used by the named instance.
- **Ports 137, 138, and 139 or 445 —** These ports are required if you are delivering the initial snapshot by using UNC shares.

These are the only ports that need to be opened for transactional replication. Other Windows services and application may require additional ports to be opened.

Transactional Replication Role Change

Using a subscriber as the new primary server is a manual process. You must manually stop the replication process and resolve SQL Server logins and user accounts. When these steps are complete, clients can connect to the new primary server. Changing roles includes the following tasks:

- **Disabling replication —** If the distributor is still functioning, disable the transactional replication jobs and place the publication database in single-user mode.
- **Resolving SQL logins —** Resolve SQL Server logins to user accounts in the database copy on the subscriber. Use the **sp_resolve_logins** system-stored procedure to load the saved BCP file containing logins from the publisher. This stored procedure resolves the SID values between the publication database and the database copy on the subscriber.

When these tasks are complete, the new primary server is ready to receive client requests. Because the production database resides on a different server with a different server name and IP address, however, you must manually point clients to this new server. You accomplish this task in different ways, depending on the design of the client applications and the use of middle-tier objects. You may be able to change the server name or IP address in a COM+ object or DNS, or you may have to make a change to the DSN at each client computer. You may also be able to use an NLB cluster and virtualize the server name and IP address if the publisher and the subscriber are on the same subnet. For more information about using NLB with transactional replication, see "Redirecting Client Network Traffic to a Promoted Secondary Server" later in this chapter.

Note: If an application has a hard-coded server name, you cannot use transactional replication without recoding the application or using an alias of the same name on the client that you redirect to the primary server when the primary server changes.

After a transactional replication role change, you can either continue to use the new primary server as the primary server or bring the original primary server back online after it has been repaired. If you continue to use the new primary server as the primary server, enable transactional replication to the original primary server (configure it as a subscriber). If you want to bring the original primary server back online, you must place the production database in single-user mode, back up the production database on the new primary server, and restore the database backup to the original primary server. After restoring the original primary server, you can point clients back to this original server and re-enable transactional replication to the original subscriber.

For more information, 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*.

Transactional Replication Advantages

There are a number of advantages to using transactional replication rather than failover clustering or log shipping. These advantages include the following:

- **Dual use —** You can use the database copy on a subscriber for reporting or routine database maintenance tasks without reducing the transactional latency of the database copy.
- **Minimal transactional latency —** You can reduce the transactional latency between the publication database and database copy on the subscriber to several seconds on a high-speed network. The minimal transactional latency for a standby database with log shipping is several minutes.
- **Hardware requirement —** Transactional replication does not require specifically certified hardware.
- **Software requirement —** Transactional replication is a standard component of all editions of SQL Server 2000.
- **Site protection —** Transactional replication to a remote subscriber provides protection against catastrophic site failure.

These advantages make transactional replication the logical choice for a high-availability solution if you require dual use or minimal latency. If neither of these advantages is required for the data center, you should reconsider log shipping or failover clustering.

Transactional Replication Limitations and Disadvantages

There are a number of limitations and disadvantages to using transactional replication rather than log shipping or failover clustering. These limitations and disadvantages include the following:

- **Manual detection and role change —** Transactional replication requires manual intervention by a DBA to detect a failure and initiate a role change. In addition, you must manually ensure that required objects or files not stored in the publication database are kept synchronized between the publisher and each subscriber.
- **More network traffic than log shipping —** Transactional replication generates more network traffic than log shipping.
- **Potential transaction loss —** If the publisher or distributor fails, completed transactions that have not been copied to subscribers may be unrecoverable. You can reduce this risk by using fault tolerant storage for the transaction logs.
- **Lack of transparency to clients —** After a server role change, clients must connect to a different server with a different server name and IP address. You must manually modify each client to connect to this new server or to incorporate an intermediate mechanism to accomplish this task for these clients. Intermediate mechanisms include NLB, DNS, and COM+ components. Each of these intermediate mechanisms requires a manual modification by a DBA to point clients to the new server name and IP address. The lack of transparency increases the time required to complete a role change.
- **Complex setup and maintenance —** Setting up and maintaining transactional replication is complex.
- **Complex failback —** Changing back to the original primary server after a role change is complex.

These disadvantages outweigh the advantages of transaction replication unless you absolutely require dual use or extremely minimal transactional latency.

Redirecting Client Network Traffic to a Promoted Secondary Server

After the failure of a primary server, you must redirect client network traffic to the promoted secondary server. Although this happens automatically with failover clustering, you must perform this task manually with log shipping and

transactional replication. Although you can perform this task at each client when the secondary server is promoted, you can use a number of mechanisms to expedite this process, including the following:

- **Network Load Balancing —** You can create an NLB cluster to direct client network traffic to a promoted secondary server. NLB is a Windows service that you run on the primary server and on one or more secondary servers to create an NLB cluster. An NLB cluster provides a single server name and address to all clients that want to connect to the data center. While the primary server is functioning properly, you configure NLB to direct all client requests to the primary server. If the primary server fails, you promote a secondary server and then reconfigure the NLB cluster to direct all client requests to the promoted secondary server. With NLB, clients are not redirected to a secondary server until that server is ready to receive client requests.

  Note: Because NLB cannot redirect client requests across subnets, all servers in an NLB cluster must reside on the same subnet. To use NLB to redirect client network traffic to a remote site, you can configure a virtual private network connection between the sites.

- **Alternate server list in client application —** You can design client applications with a list of alternate servers. With this approach, if a client application fails in an attempt to connect to the primary server, you specify a number of retries. After the retries fail, the client application attempts to connect to a secondary server from its list of alternate servers. If a connection to the first secondary server fails, the client application attempts to connect to the next secondary server on the list.

  With an alternate server list, clients may attempt to connect to a secondary server that is not ready to receive client requests and attempt to connect when the primary server has not actually failed. Although custom applications can easily be modified to include a list of alternate servers, third-party applications and utilities cannot be easily modified. The users of these applications need to be aware of the alternate list and make the alternate connection manually.

- **DNS redirection —** You can use DNS to redirect client network traffic to a promoted secondary server. If the primary server fails, you promote a secondary server and then modify the DNS entry for the primary server name to point to the standby server's IP address. Then you rename the promoted secondary server to the name of the primary server. This approach does not work for clients using an IP address rather than a server name to connect to the data center.

  Note: You must change the DNS entry for the primary server in both the forward lookup zone and the reverse lookup.

- **ODBC DSN redirection —** You can use ODBC for client connections to the data center. With ODBC, the client application uses a Data DSN to connect to the data center. The DSN specifies the name of the server to which the client application connects. If the primary server fails, you promote a secondary server and then modify the DSN to point to the promoted secondary server. You can automate the modification of these DSNs by using batch files to copy the modified DSN to all clients.
- **COM+ component redirection —** You can use COM+ components for client connections to the data center. With a COM+ component, the client application connects to the COM+ component on an intermediate server, and the COM+ component specifies the name of the server to which the client application connects. If the primary server fails, you promote a secondary server and then modify the COM+ component to point to the promoted secondary server.

In addition to these methods for redirecting client network traffic in general, DTS packages have their own options for redirection. You can use an alias for the server name or use a dynamic properties task to set your connection object:

- **Alias —** Use the Client Network Utility or **local**.
  - If you are using a named instance of SQL Server or need to refer to another server, create an alias on both servers by using the Client Network Utility. Use the same alias, but point it to the local *<server name>\<instance name>* of the server on which the package is stored.
  - If you are referring to a default instance of SQL Server on the machine on which the package will run, select **local** from the server dropdown list.
- **Dynamic properties task —** You can use the dynamic properties task in the DTS package to set your connection object. Using the dynamic properties task allows the DTS package to refer to the server and instance on which it is running. You do not need to modify each task or connection when the DTS package runs on a different computer.

Choose among the client redirection mechanisms discussed above to redirect client network traffic to a promoted secondary server and to allow DTS tasks to execute properly on the promoted secondary server.

Choosing Among the Technology Solutions

The cost of the high-availability solution, the complexity of its setup and administration, the ease of failover and failback, and the transparency of failover and failback to clients determine the appropriate high-availability solutions for a data center. The choice of operating system and edition of SQL Server affects which high-availability options are available. Failover clustering and built-in log shipping require SQL Server 2000 Enterprise Edition. NLB requires Windows 2000 Advanced Server or Windows 2000 Datacenter Server.

When choosing among these high-availability solutions, consider failover clustering first. Failover clustering provides simple, quick, automatic, and transparent failover, guaranteeing that no transactions are lost when a server fails. The hardware and software required to set up failover clustering is relatively expensive, however. In many environments, the direct and indirect costs resulting from data center unavailability justify the cost of this solution. Failover clustering does not protect against site destruction without the use of a third-party stretch cluster solution, which is expensive and complex. For some environments, such as e-commerce solutions, however, stretch clustering can be justified.

If you do not need automatic failover and cannot justify the cost of failover clustering, log shipping is the most appropriate solution. Log shipping does not require expensive hardware, and setting it up is relatively simple. Log shipping is designed for high availability, and changing server roles in log shipping is easier than changing server roles in transactional replication. Log shipping is also an appropriate, inexpensive solution to protect a data center against site destruction when remote failover clustering cannot be justified.

If you want to use a secondary server for reporting or routine maintenance tasks and offload these tasks from the primary server for performance, you can do one of the following:

- Increase the log shipping latency to enable a log shipping standby server to be used for reporting or maintenance tasks. Increasing the latency, however,decreases the availability of the data center by increasing the time required to recover the standby database if the primary server fails.
- Maintain two log shipping standby servers, using one for reporting and maintenance tasks and the other for high availability.
- Use transactional replication as the high-availability solution.

To minimize the impact of a failover or a role change to a client, use failover clustering. If you cannot justify the cost of this solution, incorporate intermediate mechanisms, such as NLB, DNS, or COM+ components.

Summary

To overcome each barrier to high availability, you must perform an analysis and determine the best solution. No one solution will overcome all of the barriers to high availability. You must use a combination of well-documented policies and procedures, well-designed client applications, regular database backups, redundant hardware, and redundant servers. You must weigh the cost of overcoming a barrier against the cost of not overcoming that barrier when you determine the appropriate solution. When evaluating a solution, be sure that the solution resolves the barrier it was intended to solve. For example, implementing failover clustering does not solve downtime caused by database corruption or user error.

After you have determined the appropriate server redundancy solution for a data center, use the accompanying Solution Guide to implement the solution.

Appendix

Contents of a Run Book

A run book should contain all of the information you and your staff need to perform day-to-day operations and to respond to emergency situations. This information should include the following:

- Resource information about the data center and its hardware and software
- Process information, including step-by-step procedures for operational and emergency processes

The run book should contain all necessary information to enable a staff member to perform any process, from performing a backup to failing over to a remote site.

Resource Information

The run book should contain the following types of detailed resource information to help your staff perform routine operational tasks and respond quickly and efficiently to data center emergencies:

- **Contact information —** Detailed information about each database administrator (DBA), the building facilities staff, utility companies, and all hardware and software vendors
- **Hardware components —** Detailed information about hardware components of the data center
- **Software components —** Detailed information about software components of the data center

Keeping this critical resource information current and readily available to your staff reduces downtime when disaster strikes.

Contact Information

Record detailed information regarding each individual or company that you or your staff may need to contact in an emergency. This detailed contact information should include the following:

- Contact information for each DBA at the primary site, including his or her role in the operational and disaster recovery process
- Contact information for the building facilities staff, the power company, the phone company, and other applicable utilities companies
- Contact information for your remote site, if you have one, and for all DBAs at that site
- Hardware, software, and service vendor support phone numbers, e-mail addresses, account numbers, and login and password information for related Web sites
- Contact information for other server applications on the server, including developers, analysts, testers, and managers affected by a change to the application, related systems, or processes

In addition, record any additional contact information that might be useful in troubleshooting and repairing the data center, such as useful e-mail discussion lists and Web sites.

Hardware Components

Record detailed information regarding each hardware component in the data center, including the following:

- **Server hardware**
  - Model and serial number
  - Brand and speed of the processor
  - Amount and configuration of memory
  - Version of the BIOS
  - Dates and version numbers of firmware
  - NIC cards, including their vendors and model numbers
  - SCSI host adapter or fiber channel cards, including their vendors and model numbers
- **Local storage hardware**
  - Type, size, and number of drives, including cache if any
  - Logical disk configuration

  - RAID levels
  - Disk controller information (including write cache settings)
  - Dates and versions of firmware for drives and controllers
  - Special options used, such as allocation units
- **Disk arrays and storage area networks**
  - Vendor and model
  - Type, size, and number of drives, including cache if any, and controller to which the disk is connected
  - Logical disk configuration
  - RAID level
  - Number of controllers and number of channels
  - Disk controller information (including write cache settings)
  - Dates and versions of firmware for drives and controllers
  - Special options used, such as allocation units

In addition, record all additional information about the data center hardware that might be useful in troubleshooting and repairing the data center. For example, record a map of the physical wiring of specific drives to specific array controllers.

Software Components

Record detailed information about each software component in the data center:

- **All software**
  - Serial numbers and/or license keys
  - The network share location for all software installed on the server, including all service packs, hardware drivers, and hot fixes
  - The onsite and offsite location of all software CDs, including license keys and serial numbers
  - The location of the written documentation for all software
- **Windows 2000**
  - Operating system version, with service pack level and hot fixes
  - Server name, IP address, and role in the domain
  - Customized settings, including terminal server and registry settings
  - Information on related systems, including contacts, configuration information, and documentation of data interfaces
  - Local administrator account name and password

- **MSCS**
  - Cluster configuration, including all cluster IP addresses, cluster name, cluster nodes, and cluster resource groups
  - User accounts authorized to administer the cluster
- **Microsoft® SQL Server™**
  - Installation information, including service pack levels, hot fixes, instance names, server collation, ports, pipes, configuration options, virtual IP name and address, database file locations, file groups, service logins and passwords, e-mail account, and enabled network protocols
  - Information about file shares used by the SQL Server and SQL Server Agent service accounts and the associated permissions on those shares
  - Database collations if different from the server collation
  - Server roles, database schemas, user accounts, permissions, database roles, custom error messages, and the location of scripts to recreate these objects
  - List of all automated SQL Server Agent jobs (specifically including all backup jobs), what they do, who is notified, their corresponding code for each job step, the time or times they run, and the location of scripts to recreate the jobs
  - List of all alerts, what they do, the associated error number or performance condition, who is notified, and the location of scripts to recreate the alerts
  - Linked server, remote server, replication, and log-shipping configuration information
  - Distributed database and distributed partition information, including information such as Data Dependent Routing Tables and distributed transaction marks
  - List and location of all DTS Packages, including associated login and password information
  - List, location, and purpose of all custom code that runs on the server, and the location of a backup copy of this code
  - Names and locations of client tools installed to connect to remote database connections (for example, to heterogeneous data sources), and necessary configuration and connection information
  - List of additional features in use and relevant configuration information, such as Extensible Markup Language (XML) support for Internet Information Services (IIS), Active Directory® service support, and Data Source Names (DSNs)
- **Analysis Services**
  - Data source and transfer information, including all associated jobs
  - Location and storage format of the Analysis Services repository

- Analysis Services repository backup job information and storage location
- Location of data files
- Security architecture, including logins, database roles, and cube roles

In addition, record all additional information about the software that might be useful in troubleshooting and repairing the data center. For example, record the staff members who are most familiar with custom applications.

Procedural Information

Develop and document procedures for each operational and emergency task that you and your staff perform. Whenever possible, develop Transact-SQL scripts for each of these tasks and automate the execution of these scripts by using SQL Server jobs or DTS packages. The procedural information should include the detailed steps and scripts for performing the following tasks utilizing both SQL Server Enterprise Manager and Transact-SQL scripts:

Operational Tasks

The DBA staff performs many routine operational tasks. To avoid problems, your staff should perform these tasks by using the same procedures each time. Record step-by-step procedures for performing each of the following types of routine operational tasks:

- **Security tasks**
  - Changing the domain user account and password used by SQL Server and SQL Server Agent
  - Creating new logins and database user accounts
  - Changing SQL Server user passwords
  - Performing standard and C2 security audits
  - Scripting login information
  - Scripting application roles and recording passwords
  - Scripting linked or remote servers
  - Restoring logins and database users to another SQL Server instance
- **System administration tasks**
  - Starting and stopping the operating system
  - Starting and stopping SQL Server services
  - Changing SQL Server configuration settings
  - Setting database options
  - Applying SQL Server service packs

  - Changing the server name
  - Manually backing up a database
  - Manually backing up a transaction log
- **Monitoring tasks**
  - Monitoring CPU usage
  - Monitoring disk activity
  - Monitoring memory usage
  - Viewing current locks
  - Viewing current activity
  - Viewing the last command batch for a specified connection
  - Viewing the data and log space information for a database
  - Viewing the oldest active transaction in the database
  - Viewing the procedure cache usage
  - Viewing general statistics about SQL Server activity and usage
  - Identifying and analyzing bottlenecks
- **Data collection tasks**
  - Archiving system and application logs in the event viewer
  - Archiving SQL Server error logs and SQL Server Agent logs
  - Archiving SQL Server setup logs
  - Archiving the cluster log file
  - Archiving **sqldiag.exe** output
  - Capturing output from **sysperfinfo** and **sysprocesses**
  - Capturing output from MPS Report tool if available
- **Troubleshooting tasks**
  - Testing TCP/IP sockets client connections
  - Testing named pipes connections
  - Troubleshooting deadlocks
  - Troubleshooting failover clustering
  - Troubleshooting replication
  - Troubleshooting log shipping
  - Troubleshooting MS DTC transactions
  - Troubleshooting orphan users

In addition to the foregoing, add step-by-step instructions for other tasks that you and your staff perform regularly.

Emergency Tasks

Record the appropriate response to each type of emergency that may affect the data center. Although the precise tasks vary depending upon the high availability solutions implemented, have a planned and tested response to each of the following types of emergencies:

- Natural disasters
- Power outages
- Server failures
- Hardware component failures
- User database corruption
- System database corruption
- Application failures
- Network failures
- Web server or other necessary server failures

Depending upon the high availability solutions implemented for the data center, the detailed steps will include MSCS failover and failback steps, log-shipping role change steps, transactional replication role change steps, and database restoration steps. These procedures should document the process of determining when to initiate a failover or a role change and how affected users are notified. These procedures must include steps to verify the system's state before bringing a restored system or database online. They should also include escalation steps in case the first attempt to restore availability fails.

Index

A

access, restricting, 16
alert messages, 21
alternate server list in client application, 66
application design, overcoming bad, 25
application errors, 29
application errors, recovering
  with copy-on-write snapshots, 37
  with split-mirror snapshots, 35
applications, client, alternate server list in, 66
auditing security, 21
availability, evaluating tasks for effect on, 23

B

backup alert, log shipping, 55
backup devices, 32
backup systems, 13
backup-and-recovery solutions
  developing, 30
  third-party, 34
backups
  differential, 33
  filegroup, 34
  frequency of, adjusting, 33
  full, frequency of, 33
  full/differential, 30
  recovering data centers with, 27
  snapshot, 34
  snapshot, combining with other snapshots, 34
  snapshot, copy-on-write, 37
  snapshot, copy-on-write vs. split-mirror, 38
  snapshot, split-mirror, 35
  solving high availability barriers with, 28
  split-mirror snapshots, 35
  transaction log, 33
  type of, adjusting, 33
barriers to high availability, 4, 7–8
  application design, 25
  communication/connectivity, 15
  environmental, 12
  evaluating costs to overcome, 8
  hardware, 14
  overcoming, 11
  process, 18
  service, 17
  software, 15
  solving, with backups, 28
  staffing, 26
business continuance volumes (BCVs), 35

C

change management, 18, 22
  emergency changes, 23
clones, 35
COM+ component redirection, 67
configuration management, 18, 24
contact information, 71–72
controller cards, 45
controlling temperature, 12
copy-on-write snapshots, 37
  vs. split-mirror snapshots, 38
corrupted databases, 16
costs, evaluating, 8

D

data centers
  access paths, multiple, 15
  change management, 22
  designing, 5
  downtime, minimizing, 47
  facility space, 13
  monitoring, 21
  planning maintenance, 6
  recovering with backups, 27
  redundant facilities, 13
  security, 24
  servers, configuring to send error alerts, 20
  Service Level Agreements, 17
  support contracts, 17
  uptime, goal for, 6
  viruses, 16
data connections, redundant, 13
data disks, improving performance of, 32
data files
  multiple, 32
  placement of, 46
databases
  corruption, 16
  recovering. *See also* backups
  recovering, improving time for, 31
  restoring, 29
  restoring, recovery time for, 31
denial of service attacks, 16
designing high-availability data centers, 5
diagnosing failures, 20
differential backups, 33
disaster recovery
  script library for, 19
  testing, 19
disk arrays, 32
  external, advantages of, 42
  external, controller cards for, 45
  managing, 42
disk backup devices, 32
disk redundancy, remote mirror solutions for, 43
distribution agent, 62
DNS redirection, 66

DNS servers, 15
documentation, 18
  in run books, 75
domain controllers, 15
DOS attacks, 16
downtime
  minimizing, with redundant servers, 47
  preventing, with hardware, 41
  scheduled, 6
drive letters, standardizing, 24
DTS packages, 67
duplexing, 44

E

educating staff, 26
emergency changes, managing, 23
emergency tasks, in run books, 77
environmental barriers, 12
errors
  application. *See* application errors
  user. *See* user errors
evaluating costs, 8
external disk arrays
  advantages of, 42
  controller cards for, 45

F

failover clustering, 45
  advantages of, 51
  architecture, 48
  as preferred solution, 68
  automatic, 51
  disadvantages of, 52
  vs. log shipping, 52
  maintaining redundant servers with, 48
  proximity requirement, 52
  transparency, 51
failures, diagnosing, 20
FibreChannel vs. SCSI, 42
filegroup backups, 34
fire suppression systems, 12
firewalls
  log shipping and, 57
  transactional replication and, 63
floors, raised, 12

H

hard-disk failures, 28
hardware
  compatibility, checking, 14
  compression, enabling, 33
  information in run books, 72
  quality, importance of, 14
  redundancy, 41
  server-class, 41
  up-to-date, importance of, 14
HCL, 14
high availability, 1, 4
  achievable levels of, 6
  barriers to. *See* barriers to high availability
  benefits of, 1
  choosing solutions, 68
  designing for, 5
  recovery point vs. recovery time, 5
  setting goals for, 5
  solutions, determining/evaluating, 9
  value, establishing, 5
humidity controls, 12

I

identifying stakeholders, 5
If SQL Server Agent is not running, the data-cente, 24
incident management, 18, 19

L

log analyzers, 30
log files, placement of, 46
log reader agent, 62
log shipping, 52
  advantages of, 59
  architecture, 53
  backup alert, 55
  configuring, 54
  database synchronization, 56
  disadvantages of, 59
  vs. failover clustering, 52
  firewalls and, 57
  monitoring servers, 53
  out of sync alert, 56
  potential transaction loss, 59
  primary server, 53
  primary server, bringing back online, 58
  role changing, 57
  site protection, 59
  SQL logins, resolving, 58
  standby servers, 53
  transaction log, backing up, 57
  transaction log backup copy job, 55
  transaction log backup, copying, 57
  transaction log backup job, 55
  transaction log backups, restoring, 58
  transaction log restore job, 55
  transparency, lack of, 60
login failures, monitoring, 21

M

maintenance
  management, 18, 23
  planned, 16
  tasks, evaluating effect on availability, 23
managing
  configuration, 24
  disk arrays, 42
  emergency changes, 23
  incidents, 18
  maintenance, 23
  planned changes, 22
  processes, 18
measuring recovery time, 31
Microsoft Cluster Service. *See* MSCS clusters
Microsoft Hardware Compatibility List (HCL), 14
Microsoft Operations Framework (MOF), 11
mirrored striping, 45
mirroring, 44
MOF, 11
monitoring
  data centers, 21
  login failures, 21

MSCS clusters, 48
nodes in, 48
shared storage resource mechanism, 50
multiple-instance clusters, 49

N

name conflicts, preventing, 25
network cards
redundant, 15, 46
teaming, 46
network load balancing, 66
networks, redundant, 15
NLB, 66

O

ODBC DSN redirection, 67
off-site storage, 13
operating systems, choosing, 68
operational tasks, in run books, 75
out of sync alert, log shipping, 56

P

performance baselines, 21
planned changes, managing, 22
power failure monitoring, 42
power supplies, redundant, 12
power-on monitoring, 42
prefailure monitoring, 42
procedures
developing, 18
documentation, 18
documenting in run book, 75
process barriers to high availability, 18
process management, 18

Q

QA environment, establishing, 22

R

RAID, 43
implementations, popular/common, 43
implementations, read/write performances of, 43
mirrored striping, 45
mirroring, 44
striped mirroring, 44
striping, 43
striping with distributed parity, 44
raised floors, 12
recovering
data centers, with backups, 27
databases, 29. *See also* backups
databases, improving time for, 31
recovery point vs. recovery time, 5
recovery time
improving, 31
total, measuring, 31
redirecting traffic to secondary server, 65
redirection, 67
redundant
data center facilities, 13
data connections, 13
hardware, 41
network cards, 15, 46
networks, 15
power supplies, 12
power system, 12
servers, maintaining with failover clustering, 48
servers, maintaining with log shipping, 52
servers, maintaining with transactional replication, 60
servers, minimizing downtime with, 47
solutions, 14
storage subsystems, 42
redundant array of independent disks. *See* RAID
remirroring, 35
remote mirror solutions, 43
response time, reducing, 20
restoring databases, 29
recovery time for, 31
restricting access, 16
rotating staff roles, 26
run book
accessibility of, 19
contact information in, 72
emergency task information in, 77
hardware component information in, 72
maintaining off-site copy, 19
operational tasks in, 75
procedural information in, 75
resource information in, 71
software information in, 73
updating, 19
run books, 71

S

scheduled downtime, 6
script libraries, 19
SCSI controllers, separate, 32
SCSI vs. FibreChannel, 42
searching Microsoft Hardware Compatibility List, 14
security, 13, 24
audits, 21
server-class hardware, 41
servers, 60
configuring to send error alerts, 20
failure, diagnosing, 20
hard-coding names, why not to, 25
performance baselines, 21, 25
redirecting traffic to secondary, 65
redundant, maintaining with failover clustering, 48
redundant, maintaining with log shipping, 52
redundant, minimizing downtime with, 47
security, 24
virtual. *See* virtual servers
service barriers to high availability, 17
Service Level Agreements (SLAs), 17
service packs, why not to code for specific level of, 25
services, 24
single-instance clusters, 49
SLAs. *See* Service Level Agreements (SLAs)

snapshot agent, 62
snapshot backups, 34
  combining with other backups, 34
  copy-on-write, 37
  copy-on-write vs. split-mirror, 38
  split-mirror, 35
  split-mirror, advantages and disadvantages, 36
software
  barriers to high availability, 15
  failures, 15
  information in run books, 73
  planned maintenance, 16
  upgrades, 16
split-mirror snapshots, 35
  advantages and disadvantages, 36
  vs. copy-on-write snapshots, 38
SQL Server
  automatic recovery, 51
  edition, choosing, 68
  failures of, 15
  maintenance, planned, 16
  multiple-instance clusters, 49
  multiple-instance clusters, assessing need for, 50
  resources mechanism, 50
  restricting access, 16
  services, 24
  single-instance clusters, 49
  user error, 16
  virtual servers, 49
staffing, 26
stakeholders, identifying, 5
standardizing drive letters, 24
storage, ensuring sufficient, 14
storage subsystems
  failures of, 28
  redundant, 42
striped mirroring, 44
striping, 43
striping with distributed parity, 44
support contracts, 17

T

tasks, evaluating for effect on availability, 23
temperature controls, 12
testing disaster recovery plans, 19
torn pages, 28
transaction log backups, 33
transactional replication, 60
  advantages, 64
  architecture, 60
  database synchronization, 62
  disabling replication, 63
  disadvantages, 65
  firewalls and, 63
  role changes, 63–64
  server roles, 61
  SQL Server logins, resolving, 63
  transparency, lack of, 65

U

updating run book, 19
upgrades, planning, 16
user errors, 16, 29
  recovering, with copy-on-write snapshots, 37
  recovering, with split-mirror snapshots, 35

V

Virtual Device Interface for Backup (VDI), 34
virtual servers, 49, 50
viruses, 16

W

write caching, 45