Using Sybase Failover in A High Availability System

Adaptive Server

Version 12.0

Document ID: 31651-01-1200-03

Last revised: November 1999

Sybase, Inc., 6475 Christie Avenue, Emeryville, CA 94608

Contents

About This Book

Audience

This manual is intended for Sybase System Administrators and Database Owners.

How to use this book

This book describes how to install, configure, and use Sybase Failover in a high availability system.

- Chapter 1, “What is High Availability?” introduces the concepts of a high availability system and Sybase’s Failover.
- Chapter 2, “Failover and Failback” provides an overview of the concepts of failing over and failing back between Adaptive Servers in a high availability system.
- Chapter 3, “Asymmetric and Symmetric Setup” discusses the differences between asymmetric and symmetric configurations.
- Chapter 4, “Modes of Failover” describes the different modes in which Adaptive Server operates when configured for failover.
- Chapter 5, “Proxy Databases, User Databases, and Proxy System Tables” discusses the concepts of proxy databases, the effect of failover on user databases, and the concepts of proxy system tables.
- Chapter 6, “Running do_advisory” describes how to configure two Adaptive Servers for failover.
- Chapter 7, “Configuring Adaptive Server for Failover on HP” describes the steps for configuring Failover on HP.
- Chapter 8, “Configuring Adaptive Server for Failover on IBM AIX” describes the steps for configuring Failover on IBM AIX.
- Chapter 9, “Configuring Adaptive Server for Failover on Digital UNIX TruCluster” describes the steps for configuring Failover on Digital Unix TruCluster.
- Chapter 10, “Configuring Adaptive Server for Failover on Sun” describes the steps for configuring Failover on Sun Cluster.
- Chapter 11, “Configuring Adaptive Server for Failover on Windows NT” describes the steps for configuring Failover on Windows NT.

- Chapter 12, “Troubleshooting Second Point of Failures” describes methods of troubleshooting second point of failures.
- Chapter 13, “Changes to Commands, System Procedures, System Databases, and New dbcc Commands, and Functions” describes how commands, system procedures, and system databases change when Adaptive Server is configured for failover.
- Chapter 14, “Open Client Functionality in a Failover Configuration” describes changes required for Open Client to work with Sybase’s ‘Failover.
- The Glossary at the end of the book defines the terms used specifically with Sybase’s Failover.

Other sources of information

Use the Sybase Technical Library CD and the Technical Library Product Manuals web site to learn more about your product:

- Technical Library CD contains product manuals and technical documents and is included with your software. The DynaText browser (included on the Technical Library CD) allows you to access technical information about your product in an easy-to-use format.

  Refer to the *Technical Library Installation Guide* in your documentation package for instructions on installing and starting Technical Library.
- Technical Library Product Manuals web site is an HTML version of the Technical Library CD that you can access using a standard web browser. In addition to product manuals, you’ll find links to the Technical Documents web site (formerly known as Tech Info Library), the Solved Cases page, and Sybase/Powersoft newsgroups.

  To access the Technical Library Product Manuals web site, go to Product Manuals at http://sybooks.sybase.com.

Sybase certifications on the web

Technical documentation at the Sybase web site is updated frequently.

v **For the latest information on product certifications and/or the EBF Rollups:**

1. Point your web browser to Technical Documents at http://techinfo.sybase.com.
2. In the Browse section, click on What’s Hot.
3. Select links to Certification Reports and EBF Rollups, as well as links to Technical Newsletters, online manuals, and so on.

- **If you are a registered SupportPlus user:**

  1. Point your web browser to Technical Documents at http://techinfo.sybase.com.
  2. In the Browse section, click on What's Hot.
  3. Click on EBF Rollups.

     You can research EBFs using Technical Documents, and you can download EBFs using Electronic Software Distribution (ESD).
  4. Follow the instructions associated with the SupportPlus^{SM} Online Services entries.

- **If you are not a registered SupportPlus user, and you want to become one:**

  You can register by following the instructions on the Web.

  To use SupportPlus, you need:

  1. A Web browser that supports the Secure Sockets Layer (SSL), such as Netscape Navigator 1.2 or later
  2. An active support license
  3. A named technical support contact
  4. Your user ID and password

- **Whether or not you are a registered SupportPlus user:**

  You may use Sybase's Technical Documents. Certification Reports are among the features documented at this site.

  1. Point your web browser to Technical Documents at http://techinfo.sybase.com
  2. In the Browse section, click on What's Hot.
  3. Click on the topic that interests you.

If you need help

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support. If you cannot resolve a problem using the manuals or online help, please have the designated person contact Sybase Technical Support or the Sybase subsidiary in your area.

CHAPTER 1

What is High Availability?

This chapter includes the following sections:

What is High Availability?

A high availability cluster includes two machines that are configured so that, if one machine (or application) is brought down, the second machine assumes the workload of both machines. Each of these machines is called one **node** of the high availability cluster. A high availability cluster is typically used in an environment that must always be available, for example, a banking system to which clients must connect continuously, 365 days a year.

Sybase's Failover product enables Adaptive Server to work in a high availability cluster in an active-active configuration. That is, both the nodes in the cluster include Adaptive Servers managing independent workloads, and are capable of taking over each other's workload in the event of a failure. The Adaptive Server that takes over the workload is called a **secondary companion**, and the Adaptive Server that fails is called the **primary companion**. Together they are **companion servers**. This movement from one node to another is called **failover**. After the primary companion is ready to resume its workload, it is moved back to its original node. This movement is called **failback**. Clients connected to the failed Adaptive Server automatically reestablish their network connections via the second machine.

You must tune your operating system to successfully manage both Adaptive Servers during failover. See your operating system documentation for information about reconfiguring your system for high availability.

Note An Adaptive Server configured for Failover can be shutdown using the **shutdown** command only after you have suspended it from companion configuration both at the server level and at the platform level. See the configuration chapter of this manual for your platform for more information

---

Requirements for Failover

The two Adaptive Servers in a high availability system must have similar configurations. They must both be:

- Running any version 12.0 or higher Adaptive Server
- Running the latest version of Open Client™
- At the same release level
- The must have a compatible configuration.

- Running Component Integration Services (CIS)
- Running a high availability subsystem (or example Sun Cluster, Windows NT running the Microsoft Server Cluster, and so on).
- Configured for either parallel or non-parallel processing.

Resource Requirements

Adaptive Servers configured as companions in a high availability system have different resource requirements than Adaptive Servers that function individually. These different resource requirements exists because the secondary companion must process all the work during failover. This is true even if the companions are setup asymmetrically. Consequently, an Adaptive Server in a high availability system has higher resource requirements than it would have if it was a single server. For more information, see "Single System Presentation" on page 7.

The following are some of the resource requirements that you should consider when you configure Adaptive Server as a cluster companion (your site will have its own set of resource requirements that must be addressed).

- Logins, roles, and databases– You must set the number of logins, roles, and databases for the cluster equal to the total number for one Adaptive Server.
- **number of user connections** – Each companion must be configured for the total number of user connections required for the system, not each companion.
- **number of open databases** – Each companion must be configured for the total number of open databases required for the system, not each companion.
- *srids* – Each companion must be configured for the total number of *srids* required for the system, not each companion.
- **number of devices** – Each must be configured for the total number of devices used by the cluster, not the number of devices used individually. That is, if one companion uses 14 devices and the second uses 23, each Adaptive Server must be configured 37 for its **number of devices**.
- The **sp_configure** option **number of open databases** on an Adaptive Server configured for Failover is reduced by two to ensure a successful failover. That is, if you currently have the **number of open databases** set to 10, you will only be able to open eight databases.

- The **sp_configure** option **number of user connections** on an Adaptive Server configured for Failover is reduced by two to ensure a successful failover. That is, if you currently have the **number of user connections** set to 50, you will only be able to have 48 user connections.

Applications Running with Sybase's Failover

Client applications that connect to companion servers must re-link their libraries with the libraries included with failover software. See "CTLIB Application Changes" on page 196 for more information about using Open Client with failover.

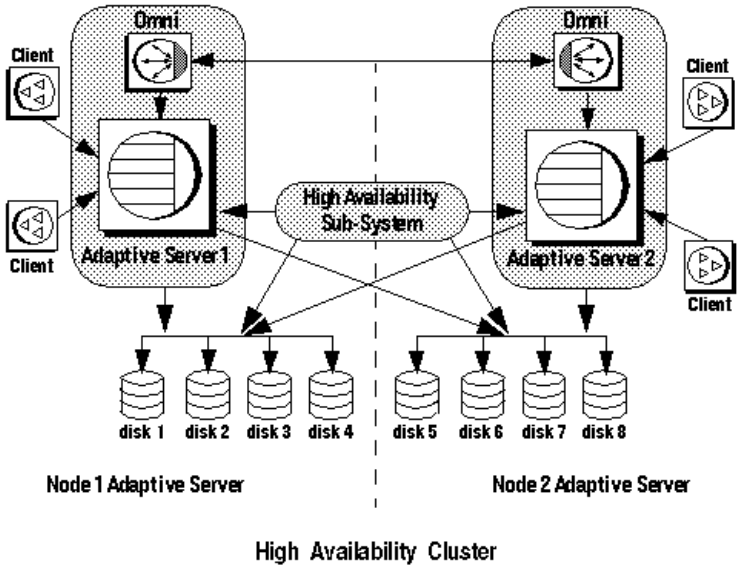
How Does Sybase's Failover Work with High Availability?

A high availability system includes the hardware and software necessary to limit the amount of downtime a system suffers. Sybase's Failover is one piece of software in this high availability system. It provides the ability for the companion to withstand a single point of failure in the cluster. That is, after the primary companion fails over to the secondary companion, subsequent fail overs are not addressed.

A system that uses Sybase Failover includes only two machines. Each machine is one **node** of the high availability **cluster**. Each Adaptive Server is either a **primary companion** or **secondary companion**. Each companion performs the work during operations; the secondary companion takes over the workload when the primary companion fails or is brought down. The primary companion can be brought down for any number of reasons: scheduled maintenance, system failure, power outage, and so on. The event of the second server assuming another server's workload is called **failover**. The event of moving the workload back to the original server once it is up and running again is called **failback**. Figure 1-1 describes a typical configuration consisting of two Adaptive Servers.

Included with the operating system is a high availability subsystem (for example, Sun Cluster for Sun, Microsoft Cluster Server for Windows NT, and so on) that detects and broadcasts to the cluster that part of the system is crashing or is being shut down for maintenance. When Adaptive Server goes down, the high availability subsystem informs the second machine to take over its workload. Any clients connected to the Adaptive Server going down are reconnected to the second Adaptive Server. Figure 1-1illustrates a high availability cluster comprised of two machines using Sybase's Failover:

Figure 1-1: High availability system using Sybase's Failover



The machines in Figure 1-1 are configured so that each machine can read the other machine's disks, although not at the same time (all the disks that are failed over should be shared disks).

For example, if Adaptive Server 1 is the primary companion, and it crashes, Adaptive Server 2, as the secondary companion, reads its disks (disks 1 – 4) and manages any databases on them until Adaptive Server 1 can be rebooted. Any clients connected to Adaptive Server 1 are automatically connected to Adaptive Server 2.

Single System Presentation

One of the hallmarks of a cluster system is that users are unaware that they are logged into a system comprised of two Adaptive Servers. To the users, it appears as if they are logging into a single system with access to all the databases on the cluster. Applications, also, only see a single system. They log into either of the companions and have access to all the databases on the cluster.

However, the System Administrator must treat the system as being comprised of two distinct Adaptive Servers. Both Adaptive Server's must be installed and configured individually, and their configuration may not be exactly the same. Both individual Adaptive Servers as well as the cluster may require system maintenance.

Special Considerations for Sybase's Failover

The following are functionality of Adaptive Server that requires special consideration while configuring a Sybase's Failover.

Using Failover with Disk Mirroring

Sybase's failover and the high availability system enable users to access data while the server to which they were originally connected is down. However, neither of these systems prevent disk failures. To make sure you do not lose any data because of a disk failure, you should use Sybase's failover in conjunction with a data protection mechanism, such as disk mirroring or RAID.

Sybase disk mirroring is not supported in an Adaptive Server companion cluster, and is disabled when you issue **sp_companion** to configure the Adaptive Servers as companions. Use a third-party vendor mirroring system to protect your disk devices.

installhasvss Script

The stored procedures required for failover are not included with the *installmaster* script. Run the *installhasvss* script to install the stored procedures and perform many of the tasks required to configure Adaptive Server for failover. *installhasvss* is located in the *$SYBASE/ASE-12_?0/scripts* directory.

On Windows NT, this script is *insthasv*, and is located in *%SYBASE%\ASE-12_0\scripts*

Note Do not run the *installmaster* script after running *installhasvss*. Do not use an *installhasvss* script that is a different release than Adaptive Server.

For more information, see the configuration chapter for your platform.

SYB_HACMP Server Entry

The *installhasvss* script creates an entry in *sysservers* for a server named SYB_HACMP. Before the Adaptive Server is configured as a companion, the SYB_HACMP server entry points to the local server. The SYB_HACMP *sysservers* entry allows the primary companion to communicate with the secondary companion using their respective entries in the interfaces file. The SYB_HACMP server entry should not be used to create any queries or stored procedures with the companion server.

Never drop the SYB_HACMP server entry. If this entry is inadvertently dropped, you must first re-run *installmaster* and then *installhasvss*.

Define User-Defined Data types in Adaptive Servers Before Configuring Them for Failover

Updates to tables that include either java or user-defined data types are not synchronized after Adaptive Servers in high availability system are configured as primary and secondary companion servers. For example, if a table in the *pubs2* database on the primary companion stores java objects as column data, updates to this column are not propagated to the proxy table. Instead, you must manually update any changes made to columns that store user-defined data types.

And, for another example, if your *pubs2* database on the primary companion includes a table that uses user-defined data types, the *pubs2* proxy table on the secondary companion does not include any updates that you made to *pubs2* on the primary companion.

Adaptive Server and Two-Phase Commit Transactions

Adaptive Servers configured as companion servers using Sybase Failover do not support SYB2PC transactions, which use the Sybase two-phase commit protocol.

CHAPTER 2

Failover and Failback

This chapter describes the characteristics of the failover and failback modes. It includes the following sections:

What is Failover?

Failover is the process of moving databases, metadata, and user connections from a failed or shut down primary companion to a secondary companion so that users can still access data. There are three sequential steps for failover:

1 System failover – The primary node fails over to the secondary node.

2 Companion failover – The primary companion fails over to the secondary node.

3 Connection failover – Connection with the failover property (for example, **isql -Q**) fails over to the secondary companion.

Steps two and three are described in detail below. See your high availability subsystem documentation for a description of step one.

During failover, the secondary Adaptive Server detects the primary Adaptive Server's failure through the operating system's high availability system and initiates the failover mechanism, which performs the following:

1 Performs a **disk reinit** to remap the master device path name to its local drive. **disk reinit** does not disturb the contents of the master device.

2 Mounts the master database, recovers, and brings it online.

3 Maps each of the devices listed in the primary companion's *sysdevices* to the secondary companion's *sysdevices* and performs a **disk reinit** on the disks.

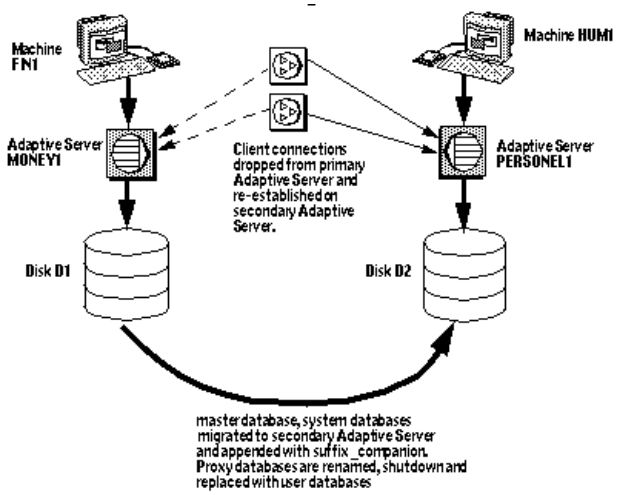
4 Mounts all the primary companion databases on the secondary companion. The secondary companion brings all databases online, after performing recovery from the logs. *tempdb* and *model* are not mounted. Proxy databases are mounted with the name *comp_dbid_dbname*.

Each database the secondary companion mounts has the suffix *_companion* appended to its name (for example, the master database becomes *master_companion*, *sybsystemprocs* becomes *sybsystemprocs_companion*, and so on). The secondary Adaptive Server adds this suffix to ensure the unique identity of the databases currently on its system. The user databases do not have the *_companion* suffix appended to their name; they are guaranteed to be unique.

5 User connections with the failover property (for example, **isql -Q**) and clients using the CS_FAILOVER property are retained and reestablished on the secondary companion. Uncommitted transactions must be resubmitted.

Figure 2-1 describes the failover process:

Figure 2-1: The failover process



Once the secondary companion receives the failover message from the high availability system, no new transactions are started on the clients connected to the primary companion. Any transactions that are not complete at the time of failover are rolled back. After failover is complete, clients or users must resubmit rolled-back transactions.

Client Connections During Failover

Clients with the failover property reconnect automatically during failover. To accommodate this, you must add a line labeled “hafailover” to the interfaces file to provide the connection information necessary for the client to connect to the secondary companion. You can add this line using either a file editor or the **dsedit** utility.

The following interfaces file entry is for an asymmetric configuration between the primary companion PERSONEL1 and its secondary companion MONEY1. It includes an additional *hafailover* entry that enables clients connected to PERSONEL1 to reconnect to MONEY1 during failover:

```
PERSONEL1

   master tli tcp /dev/tcp \x00029f7g82d63ce70000000000000000
   query tli tcp /dev/tcp \x00029f7g82d63ce70000000000000000
   hafailover MONEY1
```

On Windows NT, the connection information is included in the *sql.ini* file, which also includes an entry for *hafailover*. The following is a *sql.ini* entry for a symmetric configuration between the MONEY1 and PERSONEL1 companions:

```
[MONEY1]
 query=TCP,FN1,9835
 master=TCP,FN1,9835
 hafailover=PERSONEL1
[PERSONEL1]
 query=TCP,HUM1,7586
 master=TCP,HUM1,7586
 hafailover=MONEY1
```

For more information about adding this information to the interfaces file, see the configuration chapter for your platform.

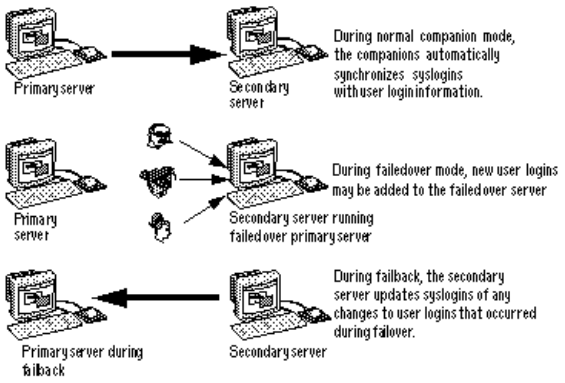
Client applications must re-send any queries that were interrupted by failover. See Chapter 14, “Open Client Functionality in a Failover Configuration” for more information about client applications.

User Logins in Failover

During normal companion mode, companions automatically synchronize any changes to user logins, access and security information, and so on. Any logins added during failover are automatically added to the primary companion when it gets updated during failback. Any uncommitted transactions must be resubmitted and any options set at the session level must be re-established once the companion has successfully failed over.

The synchronization process is described in Figure 2-2:

Figure 2-2: Synchronizing syslogins between primary and secondary servers



All user roles and privileges are maintained after failover.

What is Failback?

When the primary companion or machine is prepared to resume operation, the user with the ha_role performs a failback to return the servers to normal companion mode. Because failback temporarily shuts down the databases of the failed-over companion, you should choose a time for the failback when the application load is light. If you choose a time when the Adaptive Server is very busy, failback succeeds, but it is very slow, and the performance of the secondary companion is sacrificed. Choosing the appropriate time for failback can dramatically reduce the amount of time the clients have to wait to reconnect.

Performing Failback

Failback is accomplished in four steps:

1 Prepare for Failback.

Note IBM HACMP for AIX automatically fails back when the primary node is ready to resume normal companion mode. See Chapter 8 Configuring Adaptive Server for Failover on IBM AIX for more information.

Issuing **sp_companion** with the **prepare_failback** option forces the secondary companion to release the database devices and the databases. Issue **prepare_failback** from the secondary companion. The syntax is:

```
sp_companion server_name 'prepare_failback'
```

Where *server_name* is the name of the secondary companion. The secondary companion issues messages similar to the following during failback:

```
Step:Access across the servers verified
Step:Primary databases are shutdown in secondary
Step:Primary databases dropped from current secondary
Step:Primary devices released from current secondary
Step:Prepare failback for primary server  complete
(return status = 0)
```

The last step of **prepare_failback** is to move the devices back to the primary node according to individual platform subsystem.

2 Reboot Adaptive Server on the Primary Machine.

The high availability subsystem reboots the primary companion automatically.

3 Run **do_advisory**.

Run **sp_companion** with the **do_advisory** option to make sure there are no attribute settings that will prevent the failback operation from succeeding. For more information about **do_advisory**, see Chapter 6, “Running do_advisory”

4 Resume Normal Companion Mode

After failback is complete, issue **sp_companion** from the primary companion (the companion that originally failed) to return to normal companion mode. For example:

```
sp_companion PERSONEL1, resume
Step: Checkin to See if the remote server is up
 Step:Access across the servers verified
 Step: User information syncup succeeded
 (return status = 0)
```

See the configuration chapter for your platform for more information about **sp_companion resume**.

Note You cannot connect clients with the failover property (for example **isql -Q**) until you issue **sp_companion resume**. If you try to reconnect them after issuing **sp_companion prepare_failback**, the client hangs until you issue **sp_companion resume**

Cluster Locks in a High Availability Node

The companions in a high availability cluster must have their user information synchronized. Operations that affect the configuration of the companions are called cluster operations, and are usually initiated by **sp_companion**. Because the companions must be synchronized, clients performing cluster operations that affect the configuration of the node are only allowed to run in serial, not parallel. That is, only one client can perform a cluster operation at a time.

Before a client performs a cluster operation, it obtains a *cluster-wide lock*, which prevents any other client from performing a cluster operation at the same time. The cluster lock is not released until both companions are synchronized. If a client cannot obtain a cluster lock, its cluster operation fails. Even though the operations are performed in serial, there is no queue for the clients; you must resubmit the failed cluster operations.

Generally, you will never notice a cluster lock. They do not affect any other transactions that occur in the database, only cluster operations. However, if the client connection that holds the cluster lock fails during its cluster operation (for example, if you terminate a cluster operation using **Control – c** before it is finished). The client that failed leaves behind a lock that blocks the next client attempting to obtain a cluster lock.

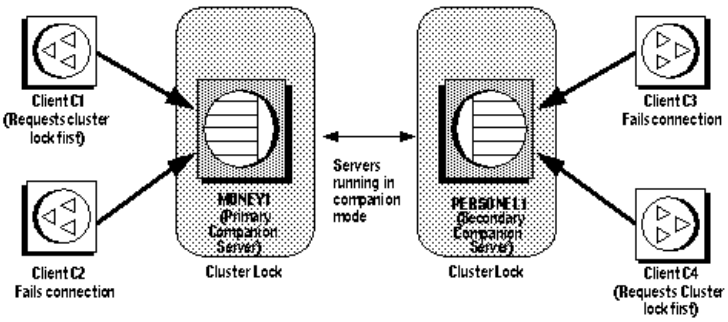
Issue **dbcc ha_admin** to release leftover cluster locks:

```
dbcc ha_admin server_name clusterlock
```

For more information about **dbcc ha_admin**, see “dbcc Options for High Availability Systems” on page 192.

Figure 2-3 describes two companion servers to which four clients are connecting. All of them attempting to perform cluster operations:

Figure 2-3: Clients connecting for cluster operations



1 Client connections C1 and C2 simultaneously attempt to obtain a cluster-wide lock to perform a cluster operation.

2 Client C1 connects to MONEY1 first and receives the cluster-wide lock.

3 Client C2 cannot get a cluster-wide lock, so it cannot perform a cluster operation.

4 Clients C3 and C4 attempt to obtain a cluster-wide lock from PERSONEL1 while C1 is performing its cluster operation.

5 Clients C3 and C4 cannot obtain a cluster-wide lock because it is held by C1.

6 After client C1 is done with its cluster operation, it releases the cluster-wide lock.

7 Client connections C2, C3, and C4 inform the SA that they were not able obtain a cluster-wide lock. The SA can resubmit these client connections for their cluster operations after client C1 has released its cluster-wide lock.

CHAPTER 3

Asymmetric and Symmetric Setup

This chapter describes asymmetric and symmetric setups for Adaptive Server in a high availability system. It includes the following sections:

Asymmetric and Symmetric Configuration

You can configure companion servers either asymmetrically or symmetrically. You must configure companions asymmetrically before you can configure them symmetrically.

Asymmetric Companion Configuration

An asymmetric configuration consists of two Adaptive Servers running on separate machines. The primary Adaptive Server performs the work during day-to-day operations, while the secondary Adaptive Server is prepared to take over the work during a system failure or a scheduled maintenance. The secondary companion is an independent Adaptive Server, and can have its own applications running. To configure for failover, the secondary companion must be a newly installed Adaptive Server version 12.0 and cannot yet have any user logins or user databases. After configuration is complete, you can add user logins and databases to the secondary companion.

When you install and configure Adaptive Server for failover, Adaptive Server is in single-server mode. Use **sp_companion** to change it from single-server mode to a companion server in an asymmetric setup.

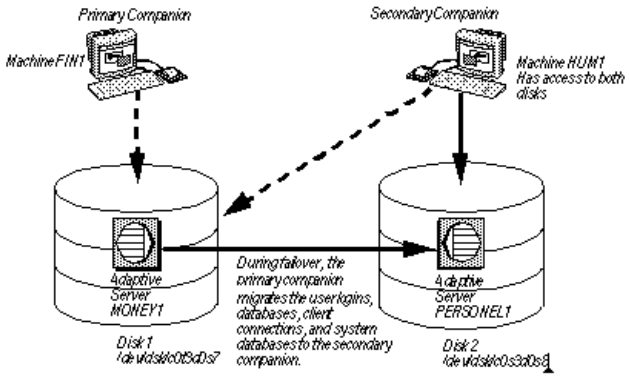
The primary companion issues messages similar to the following when you configure Adaptive Server for failover:

```
sp_companion "MONEY1", configure

 Step: Server 'PERSONEL1' is alive and cluster aware
 Changing physical name of server 'SYB_HACMP' from 'PERSONEL1' to 'MONEY1'
 Step: Access verified from Server:'PERSONEL1' to Server:'MONEY1'
 Step: Server 'MONEY1' is alive and cluster aware
 Changing physical name of server 'SYB_HACMP' from 'MONEY1' to 'PERSONEL1'
 Step: Access verified from Server:'MONEY1' to Server:'PERSONEL1'
 Step: Companion servers configuration check succeeded
 Step: Server handshake succeeded
 Step: Master device accessible from companion
 Step: Added the servers 'PERSONEL1' and 'MONEY1' for cluster config
 Step: Server configuration initialization succeeded
 Step: Synchronizing server logins from companion server
 Step: Synchronizing remoteserver from companion server
 Step: Synchronizing roles from companion server
 Step: Synchronizing server-wide privs from companion server
 Step: User information syncup succeeded
 Step: Server configured in normal companion mode
```

Figure 3-1 describes an asymmetric configuration:

Figure 3-1: Asymmetric configuration in a high availability system



In this setup, MONEY1 is the primary companion and fails over to PERSONEL1, the secondary companion. Both disks are visible to machine HUM1, which connects to machine FIN1 with a dual-ported SCSI. Because this is an asymmetric setup, PERSONEL1 cannot fail over to MONEY1. Disk 1 must be a shared disk, and disk 2 can be a local disk.

See “Configure for Asymmetric Configuration” on page 77 for information about configuring Adaptive Server for an assymetric setup.

Performance of Adaptive Server in an Asymmetric Configuration

During normal companion mode, the performance of the system procedures that update user information (**sp_addlogin**, **sp_addrole**, and so on) and commands like **create database** is slightly degraded because the primary companion must perform the command locally and then synchronize this information with the secondary companion. For example, if you add user “joe” to the primary companion, both the primary companion and the secondary companion must update *syslogins* to include this new user.

Performance after failover depends on the configuration of the secondary companion. If the secondary server is configured similarly to the primary companion's server, the performance should be similar before and after failover. However, if the secondary server is not as robust (for example, has less memory or fewer CPUs) as the primary server, then the performance after failover will be degraded. The performance of the secondary companion can also be degraded after failover because it is now running both the primary companion and any applications it was running before the failover.

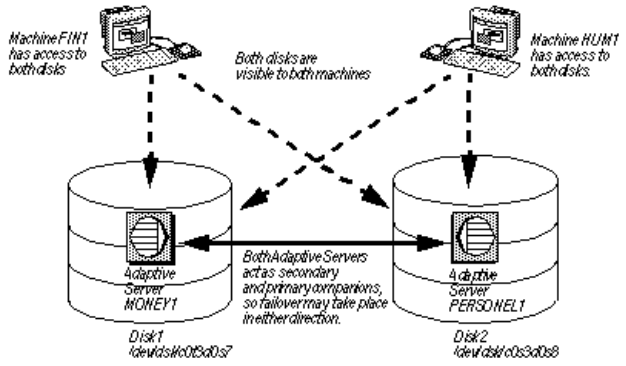
Symmetric Companion Configuration

Like asymmetric configuration, symmetric configuration consists of two fully functional Adaptive Servers running on separate machines, with their own system devices, system databases, user databases, and user logins. However, when failover occurs, either of the Adaptive Servers acts as a primary or secondary companion for the other Adaptive Server.

Before you configure two Adaptive Servers as symmetric companions, you must first configure them for asymmetric companions.

Figure 3-2 describes a symmetric configuration for failover between a financial department machine (FIN1 running Adaptive Server MONEY1) and a human resources machine (HUM1 running Adaptive Server PERSONEL1):

Figure 3-2: Symmetric configuration in a high availability system



During scheduled maintenance or system failure, either MONEY1 fails over to PERSONEL1 or PERSONEL1 fails over to MONEY1. For this configuration, both Disk 1 and Disk 2 are shared disks.

Performance of Adaptive Server in a Symmetric Configuration

During normal companion mode, do not run both Adaptive Servers in a symmetric configuration at the full capacity of their system resources (for example, they could run at 60% of the possible configuration for user connections, data cache, remote server connections, and so on). This allows the secondary companion to manage both the failed over Adaptive Server and its own Adaptive Server during failover mode with a reasonable level of performance. If the Adaptive Servers maximize their system resources, failover still succeeds, but performance may be poor.

Determining the Name of the Companion Server with *@@hacmpservername*

Use the *@@hacmpservername* global variable to determine the name of the companion server. The syntax is:

```
select @@hacmpservername
```

For example, if you issue this command from primary companion MONEY1 you receive output similar to this:

```
select @@hacmpservername

------------------------------
PERSONE1

(1 row affected)
```

Auditing in A High Availability System

This section describes the special considerations for auditing in a system configured for Sybase Failover.

Configure a companion for auditing the same as you would configure a server that does not use failover. If a primary companion is configured for auditing, the secondary companion checks to determine whether it also needs to be configured for auditing as well. For more information see "Auditing in A High Availability System" on page 27

All updates to user and security information (for example, **sp_addlogin**, **sp_addrole**, and so on) are done on both the systems in transactional fashion. This keeps the user and security data identical on both the companions.

Table 3-1 describes changes to the auditing configuration parameters.

Table 3-1: Auditing configuration parameters

Configuration Parameter	Functionality in Sybase Failover
auditing	Both companions must be configured the same for this parameter. Checked as quorum attribute, or when explicitly listed with **do_advisory**. Turning this parameter on and off is not synchronized dynamically for the companions. You must manually update the remote companion if you change this parameter locally.
allow procedure grouping	Both companions must be configured the same for this parameter. Checked as quorum attribute, or when explicitly listed with **do_advisory**.
max roles enabled per user	Both companions must be configured the same for this parameter. Checked as quorum attribute, or when explicitly listed with **do_advisory**.
unified login required	Both companions must be configured the same for this parameter. Checked as quorum attribute, or when explicitly listed with **do_advisory**.
secure default login	Both companions must be configured the same for this parameter. Checked as quorum attribute, or when explicitly listed with **do_advisory**.

Configuration Parameter	Functionality in Sybase Failover
systemwide password expiration	Both companions must be configured the same for this parameter. Checked as quorum attribute, or when explicitly listed with **do_advisory**.
use security services	Both companions must be configured the same for this parameter. Checked as quorum attribute, or when explicitly listed with **do_advisory**.
check password for digit	Both companions must be configured the same for this parameter. Checked as quorum attribute, or when explicitly listed with **do_advisory**.
minimum password length	Both companions must be configured the same for this parameter. Checked as quorum attribute, or when explicitly listed with **do_advisory**.
maximum failed logins	Both companions must be configured the same for this parameter. Checked as quorum attribute, or when explicitly listed with **do_advisory**.

Setting Auditing Options

You can configure auditing options (global, database-wide, and per-login) for each companion server on a per-node basis. That is, each companion has its own auditing setting. Global options are not synchronized between the companions.

During failover, database-wide options are audited as they are currently configured.

After failover:

- Auditing continues to enforce the global options, and database-wide options run the same as before.
- Users are still allowed to set their database-wide options.
- The audit options of the local domain are used for both local and remote logins (that is, either the failed over primary companion or local secondary companion logins)

Audit Trails and Sybase Failover

Audit trails are logged in the audit tables of *sybsecurity* database. During failover, *sybsecurity* for the failed server is mounted as *sybsecurity_companion* on the secondary companion. However, audit trails are always placed in the audit table of the current server. That is, after failover any new audit trails are placed in the audit table of the secondary companion. Also, auditing configuration changes and auditing record changes that are made on one companion are not implemented on the other companion. For example, if you change one of the auditing configuration parameters on the primary companion, this change would not be made on the secondary companion. And, if a user performs makes a change to a database on the primary companion that requires an audit record, this audit record is not made on the secondary companion as well.

On failback, no audit trails are transferred from the failed-over domain to the failed domain.

sybsecurity and Sybase Failover

Sybsecurity database is created by *installsecurity* as part of audit installation. If it exists in either companion during the initial configuration of Sybase Failover, it must exists in both companions.

CHAPTER 4

Modes of Failover

This chapter describes the different modes that Sybase Failover moves through during its operation. It includes the following sections:

What are Modes?

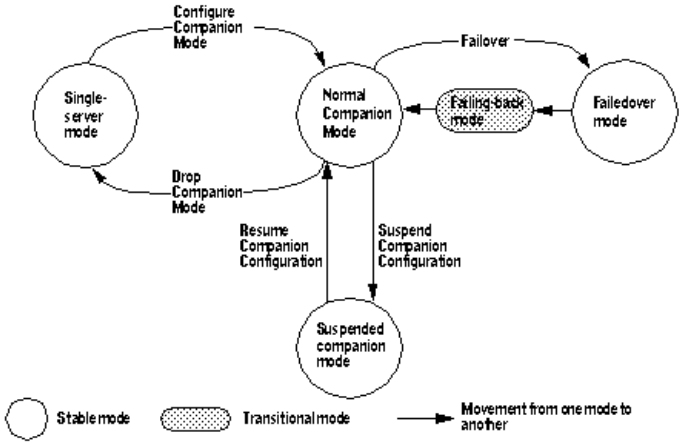
High availability consists of a series of modes in which Adaptive Server runs during its operation. There are two types of modes, **stable** and **transitional**. A stable mode is a system state in which Adaptive Server can exist for an extended period of time, such as the day-to-day operation of Adaptive Server.

Stable modes include:

- Single-server mode
- Normal companion mode
- Failover mode
- Suspended companion mode

The failback transitional mode occurs when Adaptive Server shifts from failed over mode to normal companion mode. The failback transitional mode is typically of a very short duration. The different modes and the movement that the primary companion makes while changing modes are shown in Figure 4-1:

Figure 4-1: Modes of operation for high availability



Before you can configure two Adaptive Servers as companions, both must be in single-server mode, which is the default mode of a newly installed Adaptive Server after running *installhasvss*. After you configure the Adaptive Servers as companions, they are in one of three stable modes:

- Normal companion mode

- Failed-over mode
- Suspended companion mode

Determining the Companion's Mode

You can issue **sp_companion** without any options to display the mode the companion is currently in. For example:

```
sp_companion
Server 'MONEY1' is alive and cluster configured.
Server 'MONEY1' is configured for HA services.
Server 'MONEY1' is currently in 'Symmetric normal' mode.
```

Companion MONEY1 is configured for symmetric failover and is currently running in normal companion mode.

Determining the Mode with *@@cmpstate*

You can also determine the mode using the *@@cmpstate* global variable. At the **isql** prompt, enter:

```
select @@cmpstate
```

Table 4-1 describes values that *@@cmpstate* returns:

Table 4-1: @@cmpstate return values

@@cmpstate	Companion Mode
0	Single server
1	Reserved
2	Secondary normal
3	Secondary suspended
4	Secondary failover
5	Secondary failback
6	Reserved
7	Primary normal
8	Primary Suspended
9	Primary failback
10	Reserved
11	Symmetric normal
12	Symmetric failover

@@cmpstate	Companion Mode
13	Symmetric failover
14	Symmetric suspended
15	Reserved

The Different Modes of a Companion Server

This sections describe each modes in detail.

Single-Server Mode

In this mode, Adaptive Server acts as a standalone server. A newly installed Adaptive Server is in single-server mode by default.

Normal Companion Mode

When both companions are running and are configured for failover, they operate in normal companion mode. This is the mode in which the day-to-day operations of Adaptive Server occur. For asymmetrical systems, this means that the primary companion can failover to the secondary companion. For a symmetric system, this means that either companion can fail over to the remaining companion.

Suspended Mode

Use suspended mode to temporarily suspend the companions from normal companion mode. In suspended mode, both servers act as single servers. Suspended mode is useful for performing system maintenance because you can start and stop the Adaptive Server and associated resources without risking failover.

Even though the companions cannot fail over, the nodes upon which they are working can still fail over; you must perform some platform-specific steps to suspend node failover. Also, before you shut down a companion in suspended mode, you must perform some platform-specific tasks. See the chapter for your platform for more information.

Many utilities and commands are severely restricted in suspended mode. See Chapter 13, “Changes to Commands, System Procedures, System Databases, and New dbcc Commands, and Functions” for more information.

Note Always suspend companion mode from the secondary companion

---

To suspend a companion from running in normal companion mode for any length of time (typically for maintenance), enter:

```
sp_companion 'primary_server_name', 'suspend'
```

For example, to suspend primary companion MONEY1 from normal companion mode with its secondary companion PERSONEL1, issue the following from MONEY1:

```
sp_companion "PERSONEL1", suspend
```

The companion produces messages similar to the following:

```
Step: Server 'MONEY1' is alive and cluster aware
 Step: Access verified from Server:'MONEY1' to Server:'PERSONEL1'
 Step: Server 'PERSONEL1' is alive and cluster aware
 Step: Access verified from Server:'PERSONEL1' to Server:'MONEY1'
 Step: Companion servers configuration check succeeded
 Step: Access across the servers verified
```

Failback Mode

Adaptive Server must enter the transitional failback mode to move from failover mode on the secondary companion to normal companion mode on the primary companion.

Failback mode is a planned even. That is, it is only done when the SA determines that the system is ready to resume normal operations. Use **sp_companion "prepare_failback"** to initiate failback and migrate the failed-over Adaptive Server to its original node. “Performing Failback” on page 16 describes the steps necessary to perform failback.

Resuming Normal Companion Mode from Suspended Mode

To resume normal companion mode:

```
sp_companion "primary_server_name", resume
```

For example, to resume normal companion mode between primary companion MONEY1 and its secondary companion PERSONEL1, issue the following from MONEY1:

```
sp_companion "PERSONEL1", resume
```

The companion from which you issued the command produces messages similar to the following:

```
Step: Server 'MONEY1' is alive and cluster aware
 Step: Access verified from Server:'MONEY1' to Server:'PERSONEL1'
 Step: Server 'PERSONEL1' is alive and cluster aware
 Step: Access verified from Server:'PERSONEL1' to Server:'MONEY1'
 Step: Companion servers configuration check succeeded
 Step: Checkin to See if the remote server is up
 Step: Access across the servers verified
 Step: Synchronizing server logins from companion server
 Step: Synchronizing remoteserver from companion server
 Step: Synchronizing roles from companion server
 Step: Synchronizing server-wide privs from companion server
 Step: User information syncup succeeded
```

Drop Failover Mode

To permanently disable companion mode, enter:

sp_companion "*server_name*", 'drop'

The companion from which you issued the command produces messages similar to the following:

```
Step:Local server 'MONEY1' is alive  and cluster aware
 Step:HA Versions and DLL check succeeded
 Step:Access across the servers verified
 Step: Removed the servers 'MONEY1' and 'PERSONEL1' for cluster config
 (return status = 0)
```

After completing this command the two Adaptive Servers are no longer companion servers and are running in single-server mode.

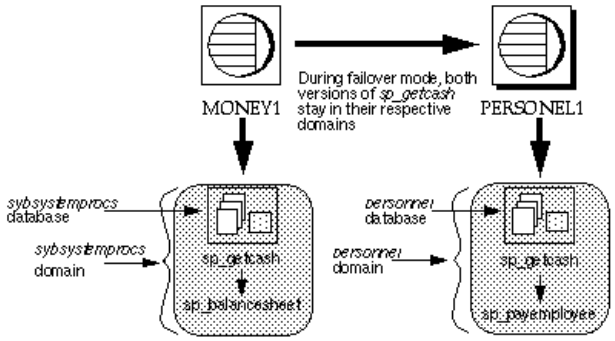
Note **Drop** is an irreversible operation. Once you have reverted the companion servers to single-server mode, you must dump, drop, and reload all user databases to reconfigure them as companions.

If the companion you drop is in a symmetric setup, the cluster automatically assumes an asymmetric setup between the companions.

Domains

Both the primary and the secondary companions can have stored procedures, users, and devices with the same names. Adaptive Servers configured for failover use *domains* to determine which database these objects belong to. For example, using the financial/human resources configuration outlined in Figure 3-1 and Figure 3-2, suppose both Adaptive Servers MONEY1 and PERSONEL1 have a stored procedure named **sp_getcash**, as described in Figure 4-2:

Figure 4-2: Domains during failover



In MONEY1, **sp_getcash** (which issues a secondary stored procedure named **sp_balancesheet**) is defined in the domain of *sybsystemprocs*. In PERSONEL1, **sp_getcash** (which issues a secondary stored procedure named **sp_payemployee**) is defined in the domain of the database *personnel*. During failover, even though *sybsystemprocs* for MONEY1 migrates to PERSONEL1 as *sybsystemprocs_companion*, its domain does not change, nor do the objects that are defined for this domain. Users that issue **sp_getcash** in *sybsystemprocs* for MONEY1 during failover mode still issue the correct secondary-stored procedure, **sp_balancesheet**.

The concept of domains is transparent to the users; they issue the same stored procedure or use their same login and login password.

System procedures that are stored in the master database are not controlled by domains. System procedures should never have a dependency on an object that are stored in the master database.

CHAPTER 5

Proxy Databases, User Databases, and Proxy System Tables

This chapter describes how proxy databases and tables are used in a failover system. This chapter includes the following sections:

For complete information about proxy databases and tables, see the *Component Integration Services User's Guide*.

Proxy Databases

Proxy databases are not created by default when you configure the Adaptive Servers as companions. They are created in the remote server only if you configure for Failover using the **with_proxydb** option of **sp_companion**. The discussion in this chapter assumes you used **sp_companion** with the **with_proxydb** option. The companions still failover whether or not you include the **with_proxydb** option when you configure the Adaptive Servers as companion server; the proxy database are created dynamically as they are needed. For more information about **sp_companion with_proxydb**, see the *Adaptive Server Reference Manual*.

Databases in companion servers are either primary or proxy databases. Primary database are where the data is physically located. Each proxy database corresponds to a primary database; it has the same name as the primary database, and proxy entries for all the objects in the primary database, but it contains no data.

After you configure the companions for failover and the proxy databases are created, the user databases are visible to both companions. This means that you can perform transactions on a primary database from either companion. For example, if a primary companion named PERSONEL1 includes a database named *salary*, its secondary companion, MONEY1, includes a *salary* proxy database. You can perform inserts, updates, and deletes on *salary* from either MONEY1 or from PERSONEL1. Also, *sysdatabases* on either companion lists the *salary* database. For example, the following query produces the same result on both PERSONEL1 and MONEY1:

```
1> select name from sysdatabases
 name
 ------------------------------
 master
 model
 salary
 sybsystemdb
 sybsystemprocs
 tempdb
```

How are Proxy Databases Created?

Adaptive Server uses Component Integration Services (CIS) to create the proxy databases. Both the primary Adaptive Server and the secondary Adaptive Server must have CIS running before they are configured for Sybase Failover. To determine if you have CIS running, enter:

```
sp_configure "enable cis"

 Parameter Name      Default    Memory Used  Config Value  Run Value
 ------------------  ---------  -----------  ------------  ---------
 enable cis                  1            0             1          1
```

A Run Value of 1 indicates that CIS is running.

For information about configuring Adaptive Server for CIS, see the *Component Integration Services User's Guide*.

CIS performs the following when it creates the proxy databases:

1 If you do not specify a size or a database device, estimates the size of the database required to contain all the proxy tables.

2 Creates all proxy tables. These act as placeholders for the tables and views found in the primary companion's database.

3 Imports the metadata (column names, size, indexes, and so on) from the primary companion.

4 Grants all permissions on the proxy tables to *public*.

5 Adds the user *guest* to the proxy database.

6 Sets the database status to indicate that the database is a proxy database. The status is indicated in the *status3* column of *sysdatabases*. **sp_helpdb** includes information about whether a database is either a proxy or primary database.

When Are Proxy Databases Created?

Proxy database are only created if you configured for Sybase Failover using the **sp_companion...with_proxydb** option. After the companions are configured with this option, proxy databases are created during the following situations:

- Proxy databases for all the primary companions user databases are created when you create a companion configuration.
- Proxy databases are created for any new user databases you create in the primary companion using the **create database** command.

- During failover, the secondary companion first mounts the primary databases and then drops the proxy databases. During failback, the secondary companion reverses the process, amounting the primary databases and then re-creating the proxy databases.

Size of the Proxy Databases

When Adaptive Server creates a proxy database, it checks the number of tables and views in the primary database and calculates the amount of space required to accommodate the same number of proxy tables in the proxy database. Each proxy table requires eight pages (one extent). Each index on a proxy table also requires eight pages. Adaptive Server also adds either an additional 10 percent or 500 page – whichever is larger–to the database to allow for table growth.

As a result, the size of the proxy databases depends on the number of tables and views in the primary database. Proxy databases do not have a default size; the minimum size is at least the size of the *model* database.

Behavior of Commands and System Procedures in Proxy Databases

The behavior of some commands and system procedures changes if you issue them in proxy databases.

Changes to Commands in Proxy Databases

For most commands, it does not matter whether you issue them from within the primary database or the proxy database; only the primary database is updated. However, the following list describes the commands that you cannot issue from within the proxy database:

- **create** or **drop procedure**
- **create** or **drop view**
- **create** or **drop trigger**
- **create** or **drop rule**
- **create** or **drop default**

You must run **dump** and **load** database commands from the primary companion. If you issue these commands from the proxy database, they will only update the proxy database; they will not update the primary companion.

Changes to System Procedures in Proxy Databases

System procedures always perform their tasks *locally*. That is if you issue any system procedure in a proxy database, any changes it makes do not appear in the primary database, and vice-versa.

System procedures begin with either the **sp**_ or **xp**_ prefix.

Issuing User-defined Stored Procedures in Proxy Databases

Whether you issue a user-defined stored procedure from the primary database or the proxy database, the user-defined stored procedure performs its tasks within the primary database. That is, if you issue **user_created_proc** on the *pubs2* primary database or from the *pubs2* proxy database it always executes on the *pubs2* proxy database, you see the same output. However, if you issue the system procedure from within a proxy database, it is treated differently depending on the following:

- CIS first looks for the system procedure in the local server. If it finds the system procedure locally, it is executed as a local stored procedure.
- If the system procedure cannot be found locally, it is forwarded to the primary companion as a remote procedure call (RPC).
- If it is a user defined stored procedure, it is turned into an RPC and forwarded to the primary companion.

Generally, users do not see any difference whether they issue the system procedure from within the proxy or primary database.

System procedures issued in a companion configuration are processed according to the same rules as for a single server. For a description of how system procedures are processed, see the *Adaptive Server Reference Manual.*

sp_dboption Does Not Update Proxy Databases

If you use **sp_dboption** to change the database options on the primary database, these changes are not automatically forwarded to the proxy databases on the secondary companion. You must set the **sp_dboption** on the proxy database as well.

For example, if you use **sp_dboption** to change the *pubs2* database so that **select into bulkcopy/pllsort** is on the primary companion, the *pubs2* proxy database on the secondary companion is not set.

Manually Updating the Proxy Databases

alter database allows you to manually re-synchronize your proxy databases with their primary databases using the **for proxy_update** option. You must issue this command from the *master* database

```
alter database <dbname>
 [existing options]
 [for proxy_update]
```

for proxy_update is useful for synchronizing changes to the primary databases that are not automatically migrated to the proxy databases. For example, if you rename the primary database using **sp_rename**, the proxy database is not automatically renamed. However, if you issue the **alter database... for proxy_update** after renaming the database, the proxy database is rebuilt using the new database name.

If you enter **for proxy_update** with no other options (for example, **alter database pubs2 for proxy_update**), the size of the database is not extended; instead, the proxy tables are dropped from the proxy database and then re-created using the metadata from the primary companion's database.

If you use **alter database** to extend the size of the database, the proxy table update is performed after the size extensions are made.

for proxy_update is supported for all external data sources, not just the primary companion in a cluster environment. Also, a database does not have to be created with the **for proxy_update** clause for it to be manually updated. If you specify a default storage location, either through the **create database** command or **sp_defaultloc**, the primary companion's metadata can be synchronized with the metadata at the remote storage location.

For more information about **alter database**, see the *Adaptive Server Reference Manual*.

Proxy System Tables in *master*

Proxy system tables enable a secondary companion to access the primary companion's system tables. One extent is allocated for the proxy system tables in *sysobjects*. You cannot drop these proxy system tables. Proxy system tables use the following naming syntax:

`rmt_ha_`*system_table_name*

Table 5-1 lists the proxy system tables in the secondary companions *sysobjects*:

Table 5-1: Proxy table names in secondary companion's sysobjects

Proxy System Table name	System Table Name
rmt_ha_sysalternates	*sysalternates*
rmt_ha_sysattributes	*sysattributes*
rmt_ha_sysconfigures	*sysconfigures*
rmt_ha_sysdatabases	*sysdatabases*
rmt_ha_syslanguages	*syslanguages*
rmt_ha_sysloginroles	*sysloginroles*
rmt_ha_syslogins	*syslogins*
rmt_ha_sysmessages	*sysmessages*
rmt_ha_sysobjects	*sysobjects*
rmt_ha_sysprotects	*sysprotects*
rmt_ha_sysremotelogins	*sysremotelogins*
rmt_ha_sysresourcelimits	*sysresourcelimits*
rmt_ha_sysroles	*sysroles*
rmt_ha_sysservers	*sysservers*
rmt_ha_syssessions	*syssessions*
rmt_ha_syssrvroles	*syssrvroles*
rmt_ha_systhresholds	*systhresholds*
rmt_ha_systypes	*systypes*
rmt_ha_sysusers	*sysusers*

CHAPTER 6

Running *do_advisory*

This chapter describes how to run **sp_companion** with the **do_advisory** option and includes these sections:

What is the *do_advisory* Option?

When you perform a cluster operation (for example, moving from failover mode to normal companion mode), either companion may have attribute settings that prevent the cluster operation from succeeding. For example, the secondary companion may be configured with a stack size that is too small to accommodate both companions during failover mode, or the companions may be configured for different languages.

To prevent these problems, the **sp_companion** command includes a **do_advisory** option which checks hundreds of attribute settings for each of the companions and issues warnings about any settings that will prevent a successful cluster operation. The attributes do not necessarily have to have the same values on both companions; for many attributes, the values must only be compatible between the two companions. **sp_companion do_advisory** does not change any of the attributes, it only advises you about any potential problems.

sp_companion...do_advisory is not triggered automatically (for example, during a **sp_companion...resume**). You should run **sp_companion...do_advisory** periodically to make sure there are no compatibility issues between your companions that will prevent a successful failover.

do_advisory allows you to specify the granularity of the attributes you want to investigate. You can either look at all the attributes, or you can specify subsets of attributes. When you specify that you want to look at all the attributes, **sp_companion** issues a list of all the attributes that will prevent a successful cluster operation.

The subset consists of *group*, *base*, or *quorum* attributes. A group attribute comprises a broad set of server settings (for example, all the login attributes or all the space attributes); a base attribute comprises specific settings within the group attributes (for example, user logins or CIS settings). When you specify that you want to investigate a subset of attributes, **do_advisory** only reports the attributes of this subset that will prevent a successful cluster operation.

Quorum attributes are configuration parameters that **sp_companion** checks every time it is run, regardless of whether or not you specify group or base attributes. If **sp_companion** finds that a quorum attribute is set such that it will prevent a successful cluster operation, the command fails. For more information, see “Quorum Attributes” on page 54.

- *Application group* – Checks to make sure the configuration settings for the applications running on the local companion are compatible with the remote companion. The application group includes the following:

Charsets – Verifies that the character sets for which the secondary companion is configured includes all the character sets for which the primary companion is configured.
Java Archives – Checks to make sure the Java archive on the primary companion has the same name and class definition on the secondary companion. If a class definition belongs to java archive on the primary companion, it must belong to the same java archive on the secondary companion.

Note These are not automatically synchronized; if you configure one companion for Java, you must manually configure the other as well.

Languages – Verifies that the languages for which the secondary companion is configured includes all the languages for which the primary companion is configured.
Remote servers – Checks that remote server entries used by the application on the primary companion are the same on the secondary, if they exist. This ensures that server names and the associated server IDs used by the companions are unique and consistent within the cluster.
All default server entries (including SYB_BACKUP, local server name, companion server name, SYB_HACMP, local XP server, and companion XP server) are automatically synchronized.
Sort order – Verifies that the sort orders for which the secondary companion is configured includes all the sort orders for which the primary companion is configured.
Time ranges – Verifies that time range definitions defined and used by the primary companion must be the same used by the secondary companion, if they exist.
User types – Checks to make sure that all user-defined data type definitions in *master* used by an application on the primary companion are defined the same on the secondary companion, if they exist.

- *Config group* – Checks for compatibility between configuration parameters defined in the configuration file (located in *$SYBASE/server_name.cfg*). Configuring the Adaptive Server as companions does not automatically synchronize the configuration options. The config group includes the following base attributes:

  CIS – Verifies that CIS is correctly configured for the cluster operation.
  DTM – Verifies that the Distributed Transaction Manager parameters are compatible between the companions.

Disk i/o – Makes sure the disk configuration (disk i/o structures, **allow sql server async i/o**, and so on) is compatible between the companions.
ESP – Makes sure the extended stored procedures are compatible between the companions.
Errorlog – Makes sure that the error log information (**event logging**, **event log computer name**, and so on) is compatible between the companions.
General config – Verifies that all the general configuration parameters (those set in the configuration file) are correctly set for the cluster operation.
Java – Makes sure that Java is either enabled or disabled for both companions.
Languages – Makes sure that both companions have the same language, character set, and sort order.
Network – Makes sure that the network related parameters (**allow remote access**, **default network packet size**, and so on) are compatible between the companions.
Parallel – Verifies that the parallel configuration parameters (**max parallel degree**, **memory per worker process**, and so on) are compatible between the companions.
Q Diag – Verifies that the Q Diagnostic attributes (**autostart collector**, **sql text pipe active**, and so on) are compatible between the companions.
Security – Verifies that the security configuration (auditing, allow procedure grouping, and so on) for the companions is compatible.

- *Database group* – Checks that database attributes are compatible between the companions. The database group includes:

  Unique Dbid – Verifies that database IDs on the primary companion are not used on the secondary companion.

  Note If a user database ID conflicts with a system database ID on the secondary companion (for example *sybsystemprocs*), you must drop and recreate the system database on the secondary companion.

- *Devices group* – Checks that device attributes are compatible between the companions. The devices group includes:

  Devnames – Verifies that logical device names on the primary companion are not used on the secondary companion.

- *Logins group* – Verifies that login and permissions are consistent between the primary and secondary companions.

Logins – All user information (logins, permissions, and so on) defined on the primary companion must be defined, available, and compatible on the secondary companion, if it exists. Logins on the primary companion are checked that they have unique names and *suids* on the secondary companions, if they exist. The logins group also checks that remote logins, external logins, aliases, (in *master*), and user names (in *master*) are compatible across the companions. **do_advisory** automatically corrects any issues that it finds with a value of 1 (for example, a login that exists on the primary companion that does not conflict with any logins on the secondary companion, but does not exists in secondary).

Default login incompatibilities of *probe*, *qcollector*, *qrepositiry*, and so on are fixed automatically.

- *Roles group* – Verifies that roles are consistent between the primary and secondary companions.

  Roles – Verifies that all user-defined roles, login roles and server wide permissions are compatible.

- *Space group* – Verifies that the secondary companion has sufficient space available for the primary companion databases during failover.

  Master Space – Estimates the space required to synchronize the metadata during the initial configuration of the companion server or during **sp_companion...resume**.
  Proxydb Space – Estimates the space required for creating the proxy databases (when you configure the companion servers with **with_proxydb**).

How Do I Run the *do_advisory* Option?

The syntax for **sp_companion do_advisory** is:

```
sp_companion server_name, do_advisory [, all | help | group_attribute_name |
base_attribute_name ]
```

where:

- *server_name* is the name of the remote Adaptive Server.
- **all** indicates that you want information about both the group and the base attributes.
- **help** prints the **sp_companion do_advisory** syntax and a list of the group and base attributes

- *group_attribute_name* is the name of the group attribute upon which you want **sp_companion** to report.
- *base_attribute_name* is the name of the base attribute upon which you want **sp_companion do_advisory** to report.

sp_companion do_advisory output includes:

- Attribute name – The name of the attribute that **sp_companion do_advisory** is investigating.
- Attribute type – The type of attribute. For example, the type might be CIS, disk i/o, General Config (these are the configuration parameters set in the *server_name.cfg* file).
- Local value – The value of the attribute on the companion from which you entered **sp_companion do_advisory** command.
- Remote value – The value of the attribute on the remote companion.
- Advisory – After accessing the attributes on the two companions, **sp_companion do_advisory** prints its findings in the Advisory column. The values in this column are:

  0 – The attributes will not affect the cluster operation.

  1 – The attributes are not configured for the best configuration, but they will not prevent a cluster operation.

  2 – The attributes need to be altered before proceeding with the cluster operation.

  For example, the following checks the attributes between Adaptive Servers MONEY1 and PERSONEL1:

```
sp_companion "MISS", do_advisory, 'all'
go

Attribute Name   Attrib Type  Local Value  Remote Value  Advisory
--------------   -----------  -----------  ------------  ------
cis connect time  CIS         1            0             2
cis rpc handling  CIS         1            0             2
max cis remote se CIS         10           25            2

(1 row affected)
(return status = 0)
```

In this example, the attributes **cis rpc handling**, **max cis remote connections**, and **max cis remote servers** all have a value of 2 under the Advisory column, which indicates that these attributes will prevent a successful companion configuration between MONEY1 and PERSONEL1. Note that the Local Values for these three attributes are different from the Remote Values. The companions must be reconfigured to have the same or compatible values.

Quorum Attributes

Whether or not you include the **do_advisory** option, if you issue **sp_companion** with either the **configure** or **resume** option, **sp_companion** checks a select group of attributes to make sure the companions have compatible values. These are called quorum attributes. If one of the companions has a value for a quorum attribute that is not compatible with the other companion, **sp_companion** fails.

Note If **sp_companion** issues a message stating that a quorum attribute prevented it from successfully finishing, run **sp_companion. . . do_advisory** for a list of the problem attributes.

The following is a list of the quorum attributes:

- **enable cis**
- **cis packet size**
- **max cis remote connections**
- **max cis remote servers**
- **number of devices**
- **esp execution stack size**
- **start mail session**
- **xp_cmdshell context**
- **default character set id**
- **default language id**
- **default sortorder id**
- **disable character set conversions**
- **enable repagent thread**
- **allow backward scans**
- **allow netsted triggers**
- **allow resource limits**
- **parition groups**
- **size of auto identity columns**

- **SQL perform integration**
- **cfg read committed with lock**
- **enable Java**
- **enable DTM**
- **number of DTX participants**
- **strict dtm enforcement**
- **allow remote access**
- **default network packetsize**
- **max network packetsize**
- **max parallel degree**
- **number or remote logins**
- **number of remote sites**
- **max parallel degree**
- **Charsets**
- **Java Archives**
- **Languages**
- **Remote Servers**
- **Sort order**
- **Time Ranges**
- **User Types**
- **Unique Dbid**
- **Devnames**
- **Logins**
- **Roles**

CHAPTER 7

Configuring Adaptive Server for Failover on HP

This chapter lists the steps necessary to configure Adaptive Server for Failover on HP. It includes the following sections:

Configure Hardware and Operating System for High Availability

Sybase high availability requires the following hardware and system components:

- Two homogenous, networked systems with similar configurations in terms of resources like CPU, memory, etc.
- These systems should be equipped with the high availability subsystem package and the associated hardware.
- Must have devices that are accessible to both nodes.
- The system must have a logical volume manager (LVM) to maintain unique device pathnames across the cluster nodes.
- Vendor provided mirroring, not Sybase mirroring, should be used for media failure protection.

  See your hardware and operating system documentation for information about installing platform specific high availability software.

Prepare Adaptive Server to Work with The High Availability Subsystem

Perform the tasks in this section to prepare Adaptive Server for a high availability configuration

Install Adaptive Servers

Install both the primary and the secondary servers. They must be installed in the same location on each node. The primary companion can be either a newly installed Adaptive Server, or it can be upgraded from a previous version of Adaptive Server with existing databases, users, and so on. The secondary companion must be a newly installed Adaptive Server and cannot have any user logins or user databases. This is to make sure that all user logins and database names are unique within the cluster.After configuration for failover is complete, you can add user logins and databases to the secondary companion.

If you are installing on the local disk, make sure any databases are created on the multihost disk.

See your the installation documentation for your platform for information about installing and configuring Adaptive Server.

Add Entries for Both Adaptive Servers to the Interfaces File

The interfaces file for both primary and secondary companion must include entries for both companions. For example, the interfaces file for the servers used in the examples above would have entries for both MONEY1 and PERSONEL1. The server entry in the interfaces file must use the same network name that is specified in sysservers. For information about adding entries to the interfaces file, see the installation documentation for your platform.

Add Entries to *interfaces* File for Client Connections During Failover

To enable clients to reconnect to the failed over companion, you must add an additional line to the interfaces file. By default, clients connect to the port listed in the query line of the server entry. If that port is not available (because that server has failed over), the client connects to the server listed in the *hafailover* line of the server entry. Here is a sample interfaces file for a primary companion named MONEY1 and a secondary companion named PERSONEL1:

```
MONEY1
    master tcp ether FIN1 4100
    query tcp ether FIN1 4100
    hafailover PERSONEL1
```

Use **dsedit** to add entries to the interfaces file. If the interfaces entries already exist, you must modify them to work for Failover.

See the the Utility Programs manual for your platform for information about **dsedit**.

Set the Value of *$SYBASE* the Same on a Local File System

If you installed *$SYBASE* on a local file system, *$SYBASE* must point to the same directory name on both companions. You can accomplish this by either:

- Making sure that the *$SYBASE* release directory on each companion is created in the same directory.
- If the companions have the *$SYBASE* release directory in different locations, create a directory with the same path on both companions that acts as a symbolic link to the actual *$SYBASE* release directory.

  For example, even though primary companion MONEY1 has a release directory of */usr/u/sybase1* and PERSONEL1 has uses */usr/u/sybase2* as its release directory, their *$SYBASE* must point to the same path.

  Both MONEY1 and PERSONEL1 have */SYBASE*, which they establish as a symbolic link to their respective *$SYBASE* release directories. On MONEY1, */SYBASE* is a link to */usr/u/sybase1*, and on PERSONEL1, */SYBASE* is a link to */use/u/sybase2*.

If you installed $SYBASE on a local file system, you must also have copies of both companion's RUNSERVER files in *$SYBASE/ASE-12_0/install* on both nodes.

The sybha Executable

The sybha executable provides the ability for the Adaptive Server High Availability Basis Services library to interact with each platform's high availability cluster subsystem. The Adaptive Server High Availability Basis Services library calls sybha. sybha is located in the $SYBASE/ASE-12_0/bin directory. Before sybha can run, you must change its ownership and permissions. You must also edit a file named *sybhauser* in the *$SYBASE/ASE-12_0/install* directory. This file contains a list of the users who have System Administrator privileges on the cluster. Sybase strongly recommends that severely limit the number of users who have System Administrator privileges on the cluster.

As root, perform the following:

1 Change directory to *$SYBASE/ASE-12_0/bin* directory:

```
cd $SYBASE/ASE-12_0/bin
```

2 Change the ownership of the **sybha** to root:

```
chown root sybha
```

3 Modify the file permissions for **sybha** to 4755

```
chmod 4755 sybha
```

4 Change directory to *$SYBASE/ASE-12_0/install* directory.

```
cd $SYBASE/ASE-12_0/install
```

5 Add the users to the *sybhauser* file who need to administer the high availability subsystem. These logins must be in the format of UNIX login IDs, not Adaptive Server logins. For example:

```
sybase
coffeecup
spooner
venting
howe
```

6 Change the permissions of sybhauser to root:

```
chown root sybhauser
```

7 Modify the file permissions for *sybhauser* so it can only be modified by root:

```
chmod 644 sybhauser
```

Create New Default Device Other Than Master

By default, the master device is the default device in a newly installed Adaptive Server. This means that, if you create any databases (including the proxy databases used by failover), they are automatically created on the master device. However, adding user databases to master makes it more difficult to restore the master device from a system failure. To make sure that the master device contains as few extraneous user databases as possible, create a new device using **disk init**. Use **sp_diskdefault** to specify the new device as the default before you configure Adaptive Server as a companion for failover. For example, to add a new default device named *money_default_1* to the MONEY1 Adaptive Server, enter:

```
sp_diskdefault money1_default1, defaulton
```

The master device continues to also be a default device until you specifically issue the following to suspend it as the default device:

```
sp_diskdefault master, defaultoff
```

See the *Adaptive Server Reference Manual* for more information about **disk init** and **sp_diskdefault**.

Add the Local Server to *sysservers*

Using **sp_addserver**, add the local server as the local server in *sysservers* using the network name specified in the interfaces file. For example, if the companion MONEY1 uses the network name of MONEY1 in the interfaces file:

```
sp_addserver MONEY1, local, MONEY1
```

You must reboot Adaptive Server for this change to take effect.

Add Secondary Companion to *sysservers*

Add the secondary companion as a remote server in *sysservers*:

```
sp_addserver server_name
```

By default, Adaptive Server adds the server with an *srvid* of 1000. You do not need to reboot Adaptive Server for the change to take effect.

Run *installhasvss* to Install HA Stored Procedures

Note You must perform the tasks described in Add Entries for Both Adaptive Servers to the Interfaces File, above, before running *installhasvss*. If you run *installhasvss* before performing these tasks you will have to re-run *installmaster* to re-install all the system stored procedures.

The *installhasvss* script performs the following tasks to configure Adaptive Server for failover:

- Installs the stored procedures required for failover (for example, **sp_companion**).
- Installs the SYB_HACMP server in *sysservers*.

You must have System Administrator privileges to run the *installhasvss* script.

installhasvss in located in the *$SYBASE/ASE-12_0/scripts* directory. To execute the *installhasvss* script, enter:

```
$SYBASE/OCS-12_0/bin/isql -Usa -Ppassword -Sservername < $SYBASE/ASE-
12_0/scripts/installhasvss
```

installhasvss prints messages as it creates stored procedures and creates the SYB_HACMP server.

Assign *ha_role* to SA

You must have the **ha_role** on both Adaptive Servers to run **sp_companion**. To assign the **ha_role**, issue the following from **isql**:

```
sp_role "grant", ha_role, sa
```

You must log out and then log back in to the Adaptive Server for the change to take effect.

Verify Configuration Parameters

You must enable the following configuration parameters before you configure Adaptive Server for failover:

- **enable CIS** – Enables Component Integration Services (CIS). This configuration parameter is enabled by default.

- **enable xact coordination** – Enables Distributed Transaction Management (DTM). This configuration parameter is enabled by default.
- **enable HA** – Enables Adaptive Server to function as a companion in a high availability system. **enable HA** is off by default. This configuration is static, so you must reboot Adaptive Server for it to take effect. This parameter causes a message to be written to your errorlog stating that you have started the Adaptive Server in a high availability system.

See the *System Administration Guide* for information about enabling configuration parameters.

Configuring HP for Failover

This section describes the steps for preparing your HP MC/ServiceGuard high availability subsystem for Sybase's Failover. This section assumes:

- You are familiar with HP MC/ServiceGuard.
- You have configured a two-node cluster hardware for MC/ServiceGuard
- You have installed HP MC/ServiceGuard version 11.05 on both nodes running under HPUX 11.0.
- The cluster system has been installed and configured.
- You have set up volume groups to contain all the database devices in the cluster on the shared disk devices.
- All the shared volume groups are already part of the cluster configuration.

See your HP documentation Managing MC/ServiceGuard for more information about installing, configuring, and managing MC/ServiceGuard.

Create the Package Configuration

The package configuration process defines the Adaptive Server and its associated resources that are run by the package manager when a package starts on a node in the cluster. The package configuration also includes a prioritized list of cluster nodes on which the package runs as well as defines the different types of failover the package allows. You must defined a package for each companion server.

Note The name of the Adaptive Server specified in the interfaces file must be the same as the name of the HP MC/ServiceGuard package

---

For example, for the companion servers described in this manual, you would create a package named MONEY1 for primary companion MONEY1 and another package named PERSONEL1 for secondary companion PERSONEL1

Note You can use either SAM or MC/ServiceGuard commands to create and customize your package configuration file. This document describes the steps using MC/ServiceGuard commands. See the HP MC/ServiceGuard document for information on how to use SAM to perform these operations

---

As root, perform the following steps for both the primary and secondary companions:

1 Create a subdirectory on the primary node in the */etc/cmcluster* directory to contain the package information for your primary companion. For example, to create a directory for primary companion MONEY1:

```
mkdir /etc/cmcluster/MONEY1
```

2 Change the permissions for this directory so it is only accessible by root:

```
chmod 700 /etc/cmcluster/MONEY1
```

3 Create the same subdirectory on the secondary node. For example, to create this directory on machine FIN1 for primary companion MONEY1:

```
rsh FIN1 chmod 700 /etc/cmcluster/MONEY1
```

4 Change the permissions for this directory so it is only accessible by root:

```
rsh FIN1 "mkdir /etc/cmcluster/MONEY1"
```

5 Generate a package configuration template for the primary companion using the **cmmakepkg** command. This command uses the following syntax:

```
/usr/sbin/cmmakepkg -p
/etc/cmcluster/subdirectory_name/companion_name.ascii
```

Where *subdirectory_name* is the name of the subdirectory you created in step 1, and *companion_name* is the name of the companion for which you are configuring the package. For example, to create a package configuration template for primary companion, MONEY1:

```
/usr/sbin/cmmakepkg -p
/etc/cmcluster/MONEY1/MONEY1.ascii
```

6 Edit the configuration template file you just created so it specifies the package name, a prioritized list of nodes, the location of the control script, and the failover parameters for each package.

The following are the edits made to the MONEY1.ascii configuration file (your edits will be different):

```
PACKAGE_NAME                    MONEY1
FAILOVER_POLICY                 CONFIGURED_NODE
FAILBACK_POLICY                 MANUAL
NODE_NAME                       FIN1
NODE_NAME                       HUM1
RUN_SCRIPT                      /etc/cmcluster/MONEY1/MONEY1.cntl
HALT_SCRIPT                     /etc/cmcluster/MONEY1/MONEY1.cntl
```

```
SERVICE_NAME                    MONEY1
SERVICE_FAIL_FAST_ENABLED       NO
SERVICE_HALT_TIMEOUT            300
```

Copy this file to the subdirectory on the second node you created in step 3. For example, to copy the *MONEY1.ascii* file using **rcp**:

```
rcp /etc/cmcluster/MONEY1/MONEY1.ascii HUM1:/etc/cmcluster/MONEY1/MONEY1.ascii
```

Edit the *ASE_HA.sh* Script

The *ASE_HA.sh* template script configures the high availability subsystem to start, stop, and monitor Adaptive Server for failover. The *ASE_HA.sh* template script is included in the *$SYBASE/ASE-12_0/install* directory. Make a copy of this script in the package subdirectory you created in step 1, above, and modify it to include the environment variables for your cluster environment. Both the primary and secondary companions require a copy of this script. As root, perform the following steps:

1 If you are currently using a script to configure Adaptive Server applications to run in your high availability system, make a backup copy of this file. For example, if you have a script named *SYBASE1.sh*, copy it to *SYBASE1.sh.backup*. Otherwise proceed to step 2

2 On the primary node, change to the package subdirectory under */etc/cmcluster*. For example, if you are configuring the primary companion MONEY1:

```
cd /etc/cmcluster/MONEY1
```

3 Copy the *ASE_HA.sh* template script from the *$SYBASE/ASE-12_0/install* directory to the primary companion's package subdirectory. Use the following syntax for the package template name:

```
<package_name>.sh
```

Where *package_name* is the name of the companion server you are configuring. For example, to make a copy of the *ASE_HA.sh* file for MONEY1:

```
cp ASE_HA.sh /etc/cmcluster/MONEY1/MONEY1.sh
```

4 Edit the *server_name.sh* file for your environment. Edit the lines that include "__FILL_IN__" (and any other lines that require editing for you site). This is a list of these lines:

Where:

- ASE_12_0 – specifies the version of Adaptive Server. Set this to:
  – yes if both servers are using Sybase ASE version 12.0 or greater,
  – no if you are using earlier versions of Adaptive Server.
- ASE_HAFAILOVER – specifies whether you are using Sybase Failover. Set this to:
  – yes if you are using Sybase Failover,
  – no if you are using mode 0 failover
- BASIC_FAILOVER – is set to either “yes” or “no:”
  - yes - Use the failover mechanisms provided by the HP MC/ServiceGuard high availability subsystem if it determines the servers are running in modes that allow failover. When a failover occurs, the script first checks if the companions are in a correct mode to perform a failover. If the companions are not enabled for Sybase’s Failover (that is, they are running in single-server mode), the script attempts to start up the primary companion on the secondary node.
  - no - Do not revert to mode 0 failover. That is, if BASIC_FAILOVER is set to no, failover does not happen at either the node or the companion level.
- PACKAGE_NAME - the name of the package as specified in the MC/ServiceGuard package configuration script.

Note You must specify the value of the PACKAGE_NAME to be the same as the companion name. For example, if the PRIM_SERVER value is MONEY1, the value of the PACKAGE_NAME must be MONEY1 as well.

- MONITOR_INTERVAL – The amount of time – in seconds – this scripts waits between checks to see if the Adaptive Server process is alive.
- SHUTDOWN_TIMEOUT - The maximum amount of time – in seconds – to wait for a companion server abort to complete before killing the SYBASE Adaptive Server process. The SHUTDOWN_TIMEOUT protects a hung companion server that prevents the halt script from completing. The value of SHUTDOWN_TIMEOUT must be less than the time out variable set in the package configuration file

- RECOVERY_TIMEOUT – is the maximum amount of time the high availability subsystem waits, in seconds, before determining the companion failed to start. Make sure you set this number that is sufficiently long enough for a loaded companion to reboot. RECOVERY_TIMEOUT is also used as the maximum amount of time the subsystem waits for failover and failback to complete.
- SYBASE - The location in which the Sybase products are installed. This value is automatically set to PRIM_SYBASE if you are on primary host and to SEC_SYBASE if you are on the secondary host
- SYBASE_ASE – is the installation directory of Sybase Adaptive Server products. The default is ASE-12_0.
- SYBASE_OCS – is the installation directed of Sybase Open Client products. The default is OCS-12_0
- SYBUSER - The name of the user who starts the Adaptive Server session
- HALOGIN – is the login of the user with the **sa_role** and **ha_role**. This has to be the same on both the primary and secondary companion.
- HAPWD – is the password for the HA_LOGIN. This has to be the same on both the primary and secondary companion.

Note The HA_LOGIN and HA_PWD must be the same name and password used when you are configuring Adaptive Server as a companion server. (that is, running **sp_companion**.

- PRIM_SYBASE – is the path to the directory in the primary node in which the Adaptive Server products are installed. If you are using local devices, the location must be the same on both nodes. If you are using a shared device, this location must be different on both nodes.
- PRIM_ASE_HOME - The path to the directory in which the Adaptive Server products are installed on the primary node. The default is *$SYBASE/$SYBASE_ASE*
- PRIM_SERVER – is the name of the primary companion.
- PRIM_HOSTNAME – is the name of the primary node.

- PRIM_CONSOLE_LOG – is the full path to the errorlog for the current primary companion session. This can be any file that has sufficient space and is writable by SYBUSER. The default is *$SYBASE/$SYBASE_ASE/install/server_name.cs_log*
- PRIM_RUNSCRIPT – is the name of the RUNSERVER file that is used to bring up the primary companion. The default is *$SYBASE/$SYBASE_ASE/install/RUN_server_name*
- SEC_SYBASE – is the directory in which the Adaptive Server products are installed on the secondary node. If you are using local devices, the location must be the same on both nodes. If you are using a shared device, this location must be different on both nodes.
- SEC_ASE_HOME - The path to the directory in which the Adaptive Server products are installed on the secondary node. The default is *$SYBASE/$SYBASE_ASE*
- SEC_SERVER – is the name of the secondary companion.
- SEC_HOSTNAME – is the name of the secondary node.
- SEC_CONSOLE_LOG – is the full path to the errorlog for the current secondary companion session. This can be any file that has sufficient space and is writable by SYBUSER. The default is *$SYBASE/$SYBASE_ASE/install/server_name.cs_log*
- ISQL – is the path to the isql binary. The default is *$SYBASE/$SYBASE_ASE/install/server_name.cs_log*

The example below shows the settings in *MONEY1.sh* for the primary companion MONEY1 running on host FIN1, and for the secondary companion PERSONEL1, running on host HUM1. Both of these use a local file system. During failover, MONEY1 restarts on the HUM1 if PERSONEL1 is down or not in companion mode:

Table 7-1: Settings for MONEY1 in the ASE_HA.sh script

Variable	Setting
ASE_12_0	yes
ASE_HAFAILOVER	yes
BASIC_FAILOVER	yes
PACKAGE_NAME	MONEY1
MONITOR_INTERVAL	5
SHUTDOWN_TIMEOUT	60
RECOVERY_TIMEOUT	300
HALOGIN	“SA”

Variable	Setting
HAPASSWD	“Odd2Think
PRIM_SYBASE	*/opt/sybase*
PRIM_SERVER	MONEY1
PRIM_HOSTNAME	FIN1
PRIM_CONSOLE_LOG	*$PRIM_SYBASE/ASE-12_0/install/MONEY1.cs_log*
SEC_SYBASE	*/opt/sybase*
SEC_SERVER	PERSONEL1
PRIM_HOSTNAME	HUM1
SEC_CONSOLE_LOG	*$PRIM_SYBASE/ASE-12_0/install/PERSONEL1.cs_log*

5 Change the permission on the file to 700 so it is only readable, writable, and executable by root. For example, to change permissions for *MONEY1.sh*:

```
chmod 700 MONEY1.sh
```

6 Distribute the script to the secondary node. For example, to distribute the file to the secondary node HUM1:

```
rcp /etc/cmcluster/MONEY1/MONEY1.sh
HUM1:/etc/cmcluster/MONEY1/MONEY1.sh
```

7 Repeat the above steps for the secondary companion.

The secondary companion package script uses values for PRIM_SERVER, PRIM_HOST, PRIM_SYBASE, SEC_SERVER, SEC_HOST, and SEC_SYBASE that are the opposite of the primary companion package script. Below are the values for PERSONEL1.sh:

Table 7-2: Settings for PERSONEL1 in the ASE_HA.sh script

Variable	Setting
ASE_12_0	yes
ASE_HAFAILOVER	yes
BASIC_FAILOVER	yes
PACKAGE_NAME	MONEY1
MONITOR_INTERVAL	5
SHUTDOWN_TIMEOUT	60
RECOVERY_TIMEOUT	300
HALOGIN	“SA”
HAPASSWD	“Odd2Think

Variable	Setting
PRIM_SYBASE	*/opt/sybase*
PRIM_SERVER	PERSONEL1
PRIM_HOSTNAME	HUM1
PRIM_CONSOLE_LOG	*$PRIM_SYBASE/ASE-12_0/install/MONEY1.cs_log*
SEC_SYBASE	*/opt/sybase*
SEC_SERVER	MONEY1
PRIM_HOSTNAME	FIN1
SEC_CONSOLE_LOG	*$PRIM_SYBASE/ASE-12_0/install/PERSONEL1.cs_log*

Create the Package Control Script

The package control script contains the information necessary to:

- Run the companion servers in the package
- Monitor the companion servers
- Respond to failure
- Halt the package

For security reasons, the control script must reside in a directory that includes *cmcluster* in its path.

Each package requires a separate control script. The control script is placed in the package subdirectory under */etc/cmcluster* is given the same name that it has in the package configuration file. It must be executable.

Perform the following as root:

1 Use the **cmmakepkg** utility to generate a package control script template for the primary companion in the same directory you created in step 1 on page 68. The **cmmakepkg** utility uses the following syntax:

```
/usr/sbin/cmmakepkg -s
/etc/cmcluster/package_name/companion_name.cntl
```

Where *package_name* is the name of the directory you created in step 1 on page 68, and *companion_name* is the name of the companion you are configuring.

For example, to create a package control script for primary companion MONEY1:

```
/usr/sbin/cmmakepkg -s
/etc/cmcluster/MONEY1/MONEY1.cntl
```

2 Edit the package control script to reflect your cluster environment

Follow the steps below to edit your package control script:

1 Define the volume groups that are used by this companion server package:

```
VG[0]=""
```

For example, if primary companion MONEY1 uses volume group *ha_vg1*, enter the following:

```
VG[0]="ha_vg1"
```

2 If you are using a shared file system, define the logical volumes and file system in the following line in the FILESYSTEMS section of the script:

```
LV[0]="";FS[0]="", FS_MOUNT_OPT[0]="-Fvxfs -o rw, suid, log, mincache,
dync, blkclear, detainlog, largefiles"
```

For example, if primary companion MONEY1 has data on a *ha_fs1* file system on logical volume *ha_lv1*:

```
LV[0]="ha_lv1";FS[0]="/ha_fs1", FS_MOUNT_OPT[0]=""
```

3 Enter the command to halt the companion service. Enter this command inside the **customer_defined_halt_cmds** function. This command includes the location of the *ASE_HA.sh* file (described in "Edit the ASE_HA.sh Script" on page 67). Before editing, this section looks similar to:

```
function customer_defined_halt_cmds
{
# ADD customer defined run commands.
: # do nothing instruction, because a function must contain some command.

test_return 52
}
```

Edit the function to include the **halt** command. For example, to include the **halt** command for companion MONEY1:

```
function customer_defined_halt_cmds
{
# ADD customer defined run commands.
: # do nothing instruction, because a function must contain some command.

/etc/cmcluster/MONEY1/MONEY1.sh halt
test_return 52
```

```
}
```

4 Move to the START OF CUSTOMER DEFINED FUNCTIONS section of *companion_name.cntl* and enter the command to start the companion service. Enter this command inside the **customer_defined_run_cmds** function. This command includes the location of the *ASE_HA.sh* file (described in "Edit the ASE_HA.sh Script" on page 67). Before editing this section looks similar to:

```
function customer_defined_run_cmds
{
# ADD customer defined run commands.
: # do nothing instruction, because a function must contain some command.

test_return 51
}
```

Edit the function to include the **start** command. For example, to include the **start** command for companion MONEY1:

```
function customer_defined_run_cmds
{
# ADD customer defined run commands.
: # do nothing instruction, because a function must contain some command.

/etc/cmcluster/MONEY1/MONEY1.sh start
test_return 51
}
```

5 Define the script that monitors the companion server process as a service in the SERVICE NAMES AND COMMANDS section of the script:

```
SERVICE_NAME[0]=""
SERVICE_CMD[0]=""
SERVICE_RESTART[0]=""
```

For example, configure monitoring for primary companion MONEY1:

```
SERVICE_NAME[0]="MONEY1"
SERVICE_CMD[0]="/etc/cmcluster/MONEY1/MONEY1.sh monitor"
SERVICE_RESTART[0]="-R"
```

6 Distribute the script to each node in the cluster. For example, to distribute the script to the secondary node HUM1:

```
# rcp /etc/cmcluster/MONEY1/MONEY1.cntl
HUM1:/etc/cmcluster/MONEY1/MONEY1.cntl
```

7 Repeat these steps for the secondary companion.

Verify and Distribute the Configuration

Perform the following steps to verify and distribute the configuration.

1 Use the **cmcheckconf** utility to verify that the package configuration file is correct. **cmcheckconf** uses the following syntax:

```
cmcheckconf -C /etc/cmcluster/cmclconfig.ascii -P
/etc/cmcluster/package_name/primary_companion_name.ascii
-p /etc/cmcluster/secondary_package_name/secondary_companion_name.ascii
```

Where *package_name* is the name of the directory you created in step 1 on page 68, *primary_companion_name* is the name of the companion you are configuring, and *secondary_companion_name* is the name of its secondary companion. For example, to verify the package configuration file for MONEY1:

```
cmcheckconf -C /etc/cmcluster/cmclconfig.ascii -P
/etc/cmcluster/MONEY1/MONEY1.ascii
-p /etc/cmcluster/PERSONEL1/PERSONEL1.ascii
```

2 Perform the following steps to distribute the binary cluster configuration file:

- Issue the **vgchange** command to activate the cluster lock volume group so that the lock disk can be initialized:

  ```
  /usr/sbin/vgchange -a y /dev/vglock
  ```

- Use the **cmapplyconf** utility to generate the binary configuration file and distribute it across the nodes. **cmapplyconf** uses the following syntax:

  ```
  /usr/sbin/cmapplyconf -v -C
  /etc/cmcluster/cmclconf.ascii -P
  /etc/cmcluster/primary_package_name/primary_com
  panion_name.ascii
  -p
  /etc/cmcluster/secondary_package_name/secondary
  _companion_name.ascii
  ```

  Where *primary_package_name* is the name of the directory you created in step 1 on 68, *primary_companion_name* is the name of the companion you are configuring, and similar definitions for *secondary_package_name* and *secondary_companion_name*. For example, to generate a binary configuration file for MONEY1:

```
 # cmapplyconf -v -C /etc/cmcluster/cmclconf.ascii -P
/etc/cmcluster/MONEY1/MONEY1.ascii
```

```
-p /etc/cmcluster/PERSONEL1/PERSONEL1.ascii
```

- Issue the **vgchange** command to deactivate the cluster lock volume group:

```
/etc/sbin/vgchange -a n /dev/vglock
```

Note The cluster lock volume group must be activated only on the node from which you issue the **cmapplyconf** command so that the lock disk can be initialized. When you configure the cluster, the cluster lock volume group must be active only on the configuration node and deactivated on all other nodes. Make sure you deactivate the cluster lock volume group on the configuration node after **cmapplyconf** is executed.

Note You must run **cmcheckconf** and **cmapplyconf** any time you make changes to the cluster and package configuration files.

Start Up Both the Primary and Secondary Companions

At this point, you are ready to start the package which starts and monitors the Adaptive Server. As root, start the primary companion using the following syntax:

```
/usr/sbin/cmrunpkg -n node_name primary_companion_name
```

For example, to start primary companion MONEY1 on node FIN1:

```
/usr/sbin/cmrunpkg -n FIN1 MONEY1
```

Start the secondary companion using the same command and syntax.

Configure Companion Servers for Failover

Perform the tasks in this section to configure the Adaptive Servers as primary and secondary companions in a high availability system.

Run *sp_companion* With *do_advisory* Option

You must configure the secondary companion with sufficient resources to perform the work of both servers during failover. The secondary companion may have attributes that will prevent a successful cluster operation. For example, if both the primary and secondary companions are configured for 250 user logins, during failover, the secondary companion only has the resources for half the number of potential user logins necessary. Instead, both MONEY1 and PERSONEL1 should be configured for 500 user logins.

The **sp_companion do_advisory** option checks the configuration options on both the primary and the secondary companion to make sure a cluster operation (such as configuring an Adaptive Server as a secondary companion) will be successful. **sp_companion do_advisory** advises you of any configuration options that should be changed.

See Chapter 6, “Running do_advisory” for a complete description of the **sp_companion do_advisory** option.

Configure for Asymmetric Configuration

Use **sp_companion** to configure the primary companion for asymmetric configuration:

```
sp_companion "primary_server_name", configure, with_proxydb, login_name,
password
```

Where:

- *primary_server_name* is the name of the primary Adaptive Server as defined in the interfaces file entry and in *sysservers*.
- The *with_proxydb* indicates that proxy databases are created on the secondary companion for all databases other than system databases. Any subsequent databases that are added also create proxy databases.
- *login_name* is the name of the user performing this cluster operation (they must have the ha_role).

- *password* is the password of the person performing this cluster operation

This example configures an Adaptive Server named PERSONEL1 as a secondary companion:

```
sp_companion "PERSONEL1", configure, with_proxydb, null, sa, Odd2Think
Server 'MONEY1' is alive and cluster configured.
Step: Access verified from Server:'MONEY1' to Server:'PERSONEL1'
Server 'PERSONEL1' is alive and cluster configured.
Step: Access verified from Server:'PERSONEL1' to Server:'MONEY1'
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
.....
Step: Companion servers configuration check succeeded
Step: Server handshake succeeded
Step: Master device accessible from companion
Step: Added the servers 'MONEY1' and 'PERSONEL1' for cluster config
Step: Server configuration initialization succeeded
Step: Synchronizing server logins from companion server
Step: Synchronizing remoteserver from companion server
Step: Synchronizing roles from companion server
Step: Synchronizing server-wide privs from companion server
Step: User information syncup succeeded
Step: Server configured in normal companion mode
```

If there are user databases during the **sp_companion** configuration, you see messages similar to these:

```
Step: Created proxy database 'pubs2'
 Step: Proxy status for database has been set. Please Checkpoint the database
'pubs2'
 Step: Server configured in normal companion mode"
 Starting companion watch thread
```

Before you configure the companions for symmetric configuration, you must first configure them for asymmetric configuration.

See "Asymmetric Companion Configuration" on page 22 for more information about asymmetric configuration.

Configure for Symmetric Configuration

After you configure your companions for asymmetric failover, you can configure them for symmetric configuration. In a symmetric configuration, both servers act as primary and secondary companions. See Figure 3-2 on page 24 for a description of symmetric configuration.

Issue **sp_companion** from the secondary companion to configure it for symmetric configuration. Use the same syntax as for asymmetric configuration. See "Configure for Asymmetric Configuration," above, for a description of the syntax for **sp_companion**.

The following example adds an Adaptive Server named MONEY1 as the secondary companion to the Adaptive Server named PERSONEL1 described in "Configure for Asymmetric Configuration" on page 77:

```
sp_companion 'MONEY1', configure, with_proxydb, null, sa, Think2Odd
Server 'MONEY1' is alive and cluster configured.
Step: Access verified from Server:'MONEY1' to Server:'PERSONEL1'
Server 'PERSONEL1' is alive and cluster configured.
Step: Access verified from Server:'PERSONEL1' to Server:'MONEY1'
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
.....
Step: Companion servers configuration check succeeded
Step: Server handshake succeeded
Step: Master device accessible from companion
Step: Added the servers 'MONEY1' and 'PERSONEL1' for cluster config
Step: Server configuration initialization succeeded
Step: Synchronizing server logins from companion server
Step: Synchronizing remoteserver from companion server
Step: Synchronizing roles from companion server
Step: Synchronizing server-wide privs from companion server
Step: User information syncup succeeded
Step: Server configured in normal companion mode
```

Administering Sybase's Failover

This section describes information about:

- Failing back to the primary companion
- Resuming normal companion mode
- Suspending normal companion mode
- Troubleshooting information about common problems with Sybase's Failover.

Failing Back to the Primary Companion and Resuming Normal Companion Mode

Failback moves the primary companion's shared disks from the secondary node back to the primary node and starts the primary companion on the primary node. Failback is a planned event. To failback to the primary companion:

1 Issue **sp_companion** command from the secondary companion to verify that it is in failover mode.

Note The high availability subsystem automatically restarts the primary companion.

---

2 Issue the following from the secondary companion:

```
sp_companion primary_companion_name,
prepare_failback
```

Where *primary_companion_name* is the name of primary companion server.

For example, to fail back the primary companion MONEY1, issue the following from the secondary companion PERSONEL1:

```
sp_companion MONEY1, prepare_failback
```

3 From the primary companion, issue:

```
sp_companion secondary_companion_name, resume
```

Where *secondary_companion_name* is the name of the secondary companion server. For example, to resume normal companion mode between primary companion MONEY1 and secondary companion PERSONEL1:

```
sp_companion PERSONEL1, 'resume'
```

4 Issue **sp_companion** with out any options from either companion to make sure you are in normal companion mode.

Note You cannot connect clients with the failover property (for example **isql -Q**) until you issue **sp_companion resume**. If you do try to reconnect them after issuing **sp_companion prepare_failback**, the client hangs until you issue **sp_companion resume**.

Suspending Companion Mode

Suspended mode temporarily disables the ability of the primary companion to failover to the secondary companion. When you move the companions to suspended mode, synchronization between the companions does not occur, and the primary companion cannot failover to the secondary companion. However, suspended mode is very useful for performing such maintenance tasks as changing configuration parameters. Perform the following steps to switch from normal companion mode to suspended mode:

1 As root, issue **cmhaltserv** to disable the monitoring process so that it does not trigger a failover when you shut down the companion server:

```
cmhaltserv -v <primary_package_name>
```

Where *primary_package_name* is the name of the primary package, which is also the same as the name of the primary companion. For example, to disable the monitoring process for primary companion MONEY1:

```
cmhaltserv -v MONEY1
```

2 Move the companions from normal companion mode to suspended mode. Issue the following from the secondary companion:

```
sp_companion primary_server_name, suspend
```

For example, To suspend primary companion MONEY1, issue the following from secondary companion PERSONEL1:

```
sp_companion MONEY1, suspend
```

You can now shut down the primary companion as necessary and it will not failover to secondary companion.

Resuming Normal Companion Mode from Suspended Mode

To resume normal companion mode between two companions that have been moved to suspend mode:

1 As root, issue **cmhaltpkg** from primary node to shutdown down the primary companion:

```
cmhaltpkg primary_package_name
```

Where *primary_package_name* is the name of the primary package, which is the same as the name of the primary companion server. For example to halt the MONEY1 package:

```
cmhaltpkg MONEY1
```

2 As root, issue **cmmodpkg** and **cmrunpkg** from the primary companion to run the package which restarts the primary companion:

```
cmmodpkg -e primary_package_name
cmrunpkg primary_package_name
```

Where *primary_package_name* is the name of the primary package, which is the same as the name of the primary companion server. For example to run the MONEY1 package to restart the MONEY1 primary companion:

```
cmmodpkg -e MONEY1
cmrunpkg MONEY1
```

Dropping Companion Mode

To drop companion mode, issue:

```
sp_companion companion_name, "drop"
```

Dropping companion mode is an irreversible process; you must reconfigure the Adaptive Servers companion servers before you they will failover in a high availability system and retain all the functionality that Sybase's Failover provides. However, the nodes upon which the Adaptive Servers are running are still a monitored by the high availability subsystem.

If you drop the companion mode while the monitor script is running, the script continues to monitor the server for any down or hung instances. If you plan to shutdown the server and do not want the node to failover, you need to kill the monitor process by issuing:

```
/usr/sbin/cmhaltsrv service_name
```

For example, to halt the service for primary companion MONEY1:

```
/usr/sbin/cmhaltsrv MONEY1
```

Alternatively, you can halt the package, reactivate the volume group, and then restart the companion only.

If you do not kill the monitor process and it detects that the companion has gone down, it triggers a failover to the secondary node. It restarts the primary companion on the secondary node depending on your settings for BASIC_FAILOVER.

Troubleshooting Sybase Failover on HP

This section includes troubleshooting information about common errors.

Error Message 18750

If a companion server issues error message 18750, check the *@@cmpstate* of your servers. If your primary companion is in normal companion mode, but the secondary companion is in secondary failover mode, your cluster is in an inconsistent state, and you need to manually recover from this. This inconsistent state may be caused by an **sp_companion 'prepare_failback'** command failing on the secondary companion. You can determine whether this happened by examining the log on the secondary node. To recover from this, perform the following steps manually:

1. Shut down both the primary and the secondary companions by halting both their packages.
2. Reboot the secondary companion by starting the package for the secondary companion.
3. Repair all databases marked "suspect." To determine which databases are suspect, issue:

   ```
   select name, status from sysdatabases
   ```

   Databases marked suspect have a *status* value of 320.
4. Allow updates to system tables:

   ```
   sp_configure "allow updates", 1
   ```
5. For each suspect, failed-over database, perform the following:

```
1> update sysdatabase set status=status-256 where name='database_name'
2> go
1> dbcc traceon(3604)
2> go
1> dbcc dbrecover(database_name)
2>go
```

6. From the secondary companion, issue:

```
sp_companion primary_companion_name, prepare_failback
```

   For example, from secondary companion PERSONEL1:

   ```
   sp_companion MONEY1, prepare_failback
   ```

Make sure that this command executes successfully.

7 Resume normal companion mode. From the primary companion, issue:

```
sp_companion secondary_companion, resume
```

For example, from the primary companion MONEY1:

```
sp_companion PERSONEL1, resume
```

Recovering from a Failed *prepare_failback*

During a failback, if **prepare_failback** was executed successfully on the secondary companion but the primary companion fails to boot, perform the following to rollback and then reissue the **prepare_failback** command:

1 Check the primary companion's errorlog the HP MC/ServiceGuard package log, or the system log to find the reason the server failed to boot, and correct the problems.

2 If the package for the primary companion is running on the primary node, halt the package.

3 Login to the secondary companion and issue:

```
dbcc ha_admin ("", "rollback_failback")
dbcc ha_admin ("", "rollback_failover")
```

4 Verify secondary companion is in normal companion mode

5 As root, start up the package for the primary companion to run on secondary node.

```
/usr/sbin/cmrunpkg -n <secondary_node> primary_companion_package_name
```

Your secondary companion is now in failover mode. Once you verify that everything is ready for the primary companion to failback to normal companion mode, you can issue **sp_companion...prepare_failback**.

Location of Error Logs

Sybase's Failover and HP MC/ServiceGuard includes the following error logs:

- */var/adm/syslogs/syslog.log* – contains the output of HP MC/ServiceGuard cluster-level activities as well as operating system level activities.

- */etc/cmcluster/<package_name>/<package_name>.cntl.log* – contains the output of the HP MC/ServiceGuard package activities and Sybase's Failover activities from the companion start, stop, and monitor script

  For output from the companion start, stop, and monitor script, search for "SYBASE HA".

  For MC/ServiceGuard package failure, search for the string "ERROR".

- *$PRIMARY_CONSOLE_LOG* - The location of this log is defined in */etc/cmcluster/<package_name>/<package_name>.sh*. This errorlog includes information from the last execution of Adaptive Server from the *ASE_HA.sh* script.

CHAPTER 8

Configuring Adaptive Server for Failover on IBM AIX

Perform the tasks in this chapter to configure Adaptive Server for Failover on IBM AIX.

Configure Hardware and Operating System for High Availability

Sybase high availability requires the following hardware and system components:

- Two homogenous, networked systems with similar configurations in terms of resources like CPU, memory, etc.
- These systems should be equipped with the high availability subsystem package and the associated hardware.
- Must have devices that are accessible to both nodes.
- The system must have a logical volume manager (LVM) to maintain unique device pathnames across the cluster nodes.
- Vendor provided mirroring, not Sybase mirroring, should be used for media failure protection.

  See your hardware and operating system documentation for information about installing platform specific high availability software.

Requirements for Running Sybase's Failover on IBM AIX

Configuring for high availability on IBM HACMP requires:

- 2 hardware-compatible nodes running HACMP for AIX, Version 4.2.2, that are configured in the same cluster.
- Each node has 3 IP addresses, one for service, one for boot, and one for standby. The standby IP address should be on a different subnet from the other two.
- Shared disk devices that are set up for the high availability system between the nodes.
- Shared logical volume groups that are set up to contain all the database devices in the cluster. Make sure that both nodes have the same major number for each of the shared volume groups that you define in the cluster. In this chapter, these resources are referred to as:
  – *shared_vg1* for the primary node
  – *shared_vg2* for the secondary node

  See the *HACMP for AIX Installation or Administration Guide* for information about installing the high availability subsystem.

Sybase also recommends that you identify the following resources in advance.

- A shared volume group name for the primary node (for example, *shared_vg1*).
- A shared volume group name for the secondary node (for example, *shared_vg2*).
- A resource group name for the primary companion (for example, *resgrp1*).
- A resource group name for the secondary companion (for example, *resgrp2*).
- The name of the primary companion name.
- The name of the secondary Adaptive Server companion name.

Special Considerations for Running Adaptive Server on HACMP for AIX

When the primary companion fails over on HACMP 4.2.2, the entire node fails over, not just the primary companion. During this node failover, the IP address of the servicing host (the primary node) is swapped with another standby address. In some networking environments, this may cause all the processes on the initial IP address to freeze and eventually time out. Because of this, when you use Sybase's Failover with HACMP on AIX:

- Do not allow clients to log in directly to the primary node
- Limit the primary node to running only one high availability application at a time

Prepare Adaptive Server to Work with the High Availability Subsystem

Perform the tasks in this section to prepare Adaptive Server for a high availability configuration.

Install Adaptive Servers

Before you install Adaptive Server, start the HACMP services on the same node on which you are installing the Adaptive Server. To make sure that the HACMP node is running on its service IP address and not the boot or standby IP address.

Install both the primary and the secondary servers. You can install the companions on either local or shared file systems. If they are installed on shared file systems, the file system must not be the same. This is to prevent the file systems from overwriting each other during a device failover. For example, you can install the primary companion on */node1_sybase*, but install the secondary companion on */node1_sybase*.

If the servers are installed on local file system, the name of the file systems must be the same. For example, both the primary and the secondary companion could be installed in */sybase*.

The file systems that contain *$SYBASE* must be either local or shared; you cannot mix local and shared file systems for *$SYBASE* in the cluster.

The database devices for the primary companion must be devices in the shared volume group on the primary node (for example, *shared_vg1*), so the volume group for this node must be "varied on."

If you are creating an asymmetric configuration, you can use any device (either shared or local) for the database device. If you are creating a symmetric configuration, you must use a device in the shared volume group on the secondary node (for example, *shared_vg2*) for its database devices, so the volume group for this node must be "varied on."

The primary companion can be either a newly installed Adaptive Server, or it can be upgraded from a previous version of Adaptive Server with existing databases, users, and so on.

The secondary companion must be a newly installed Adaptive Server without any user logins or user databases. This ensures that all user logins and database names are unique within the cluster. After configuration for failover is complete, you can add user logins and databases to the secondary companion.

See your the installation documentation for your platform for information about installing and configuring Adaptive Server.

Add Entries for Both Adaptive Servers to the Interfaces File

The interfaces file for both primary and secondary companion must include entries for both companions. For example, the interfaces file for the servers used in the examples above would have entries for both MONEY1 and PERSONEL1. The server entry in the interfaces file must use the same network name that is specified in *sysservers*. For information about adding entries to the interfaces file, see the installation documentation for your platform.

Add Entries to *interfaces* File for Client Connections During Failover

To enable clients to reconnect to the failed over companion, you must add an additional line to the interfaces file. By default, clients connect to the port listed in the query line of the server entry. If that port is not available (because that server has failed over), the client connects to the server listed in the *hafailover* line of the server entry. Here is a sample interfaces file for a primary companion named MONEY1 and a secondary companion named PERSONEL1:

```
MONEY1
    master tcp ether FIN1 4100
    query tcp ether FIN1 4100
    hafailover PERSONEL1
```

Use **dsedit** to add entries to the interfaces file. If the interfaces entries already exist, you must modify them to work for Failover.

See the the Utility Programs manual for your platform for information about **dsedit**.

Set *$SYBASE* the Same on a Local File system

If you installed *$SYBASE* on a local file system, *$SYBASE* must point to the same directory name on both companions. You can accomplish this by either:

- Making sure that the *$SYBASE* release directory on each companion is created in the same directory.
- If the companions have the *$SYBASE* release directory in different locations, create a directory with the same path on both companions that acts as a symbolic link to the actual *$SYBASE* release directory.

  For example, even though primary companion MONEY1 has a release directory of */usr/u/sybase1* and PERSONEL1 has uses */usr/u/sybase2* as its release directory, their *$SYBASE* must point to the same path.

  Both MONEY1 and PERSONEL1 have */SYBASE*, which they establish as a symbolic link to their respective *$SYBASE* release directories. On MONEY1, */SYBASE* is a link to */usr/u/sybase1*, and on PERSONEL1, */SYBASE* is a link to */use/u/sybase2*.

If you installed $SYBASE on a local file system, you must also have copies of both companion's RUNSERVER files in *$SYBASE/ASE-12_0/install* on both nodes.

The sybha Executable

The sybha executable provides the ability for the Adaptive Server High Availability Basis Services library to interact with each platform's high availability cluster subsystem. The Adaptive Server High Availability Basis Services library calls sybha. sybha is located in the $SYBASE/ASE-12_0/bin directory. Before sybha can run, you must change its ownership and permissions. You must also edit a file named *sybhauser* in the *$SYBASE/ASE-12_0/install* directory. This file contains a list of the users who have System Administrator privileges on the cluster. Sybase strongly recommends that severely limit the number of users who have System Administrator privileges on the cluster.

As root, perform the following:

1 Change directory to *$SYBASE/ASE-12_0/bin* directory:

```
cd $SYBASE/ASE-12_0/bin
```

2 Change the ownership of the **sybha** to root:

```
chown root sybha
```

3 Modify the file permissions for **sybha** to 4755

```
chmod 4755 sybha
```

4 Change directory to *$SYBASE/ASE-12_0/install* directory.

```
cd $SYBASE/ASE-12_0/install
```

5 Add the users to the *sybhauser* file who need to administer the high availability subsystem. These logins must be in the format of UNIX login IDs, not Adaptive Server logins. For example:

```
sybase
coffeecup
spooner
venting
howe
```

6 Change the permissions of sybhauser to root:

```
chown root sybhauser
```

7 Modify the file permissions for *sybhauser* so it can only be modified by root:

```
chmod 644 sybhauser
```

Verify Configuration Parameters

You must enable the following configuration parameters before you configure Adaptive Server for failover:

- **enable CIS** – Enables Component Integration Services (CIS). This configuration parameter is enabled by default.
- **enable xact coordination** – Enables Distributed Transaction Management (DTM). This configuration parameter is enabled by default.
- **enable HA** – Enables Adaptive Server to function as a companion in a high availability system. **enable HA** is off by default. This configuration is static, so you must reboot Adaptive Server for it to take effect. This parameter causes a message to be written to your errorlog stating that you have started the Adaptive Server in a high availability system.

See the *System Administration Guide* for information about enabling configuration parameters.

Add Thresholds to the Master Log

If you have not already done so, you must add a threshold to the master log.

1 Define and execute **sp_thresholdaction** on the master database's log to set a threshold on the number of pages left before a dump transaction occurs. Sybase does not supply **sp_thresholdaction**. See the Adaptive Server Reference Manual for information about creating this system procedure.

2 Place thresholds on the master and *sybsystemprocs* log segments so they do not fill up:

```
sp_addthreshold "master", "logsegment", 250, sp_thresholdaction
sp_addthreshold "sybsystemprocs", "logsegment", 250, sp_thresholdaction
```

3 You must reboot the primary companion for this static parameter to take effect.

Create New Default Device Other Than Master

By default, the master device is the default device in a newly installed Adaptive Server. This means that, if you create any databases (including the proxy databases used by failover), they are automatically created on the master device. However, adding user databases to master makes it more difficult to restore the master device from a system failure. To make sure that the master device contains as few extraneous user databases as possible, create a new device using **disk init**. Use **sp_diskdefault** to specify the new device as the default before you configure Adaptive Server as a companion for failover.

For example, to add a new default device named *money_default_1* to the MONEY1 Adaptive Server, enter:

```
sp_diskdefault money1_default1, defaulton
```

The master device continues to also be a default device until you specifically issue the following to suspend it as the default device:

```
sp_diskdefault master, defaultoff
```

See the *Adaptive Server Reference Manual* for more information about **disk init** and **sp_diskdefault**.

Add The Local Server to *sysservers*

Using **sp_addserver**, add the local server as the local server in *sysservers* using the network name specified in the interfaces file. For example, if the companion MONEY1 uses the network name of MONEY1 in the interfaces file:

```
sp_addserver MONEY1, local, MONEY1
```

You must reboot Adaptive Server for this change to take effect.

Add Secondary Companion to *sysservers*

Add the secondary companion as a remote server in *sysservers*:

```
sp_addserver server_name
```

By default, Adaptive Server adds the server with an *srvid* of 1000. You do not need to reboot Adaptive Server for the change to take effect.

Run *installhasvss* to Install HA Stored Procedures

Note You must perform the tasks described in Add Entries for Both Adaptive Servers to the Interfaces File, above, before running *installhasvss*. If you run *installhasvss* before performing these tasks you will have to re-run *installmaster* to re-install all the system stored procedures.

The *installhasvss* script performs the following tasks to configure Adaptive Server for failover:

- Installs the stored procedures required for failover (for example, **sp_companion**).
- Installs the SYB_HACMP server in *sysservers*.

You must have System Administrator privileges to run the *installhasvss* script.

installhasvss in located in the *$SYBASE/ASE-12_0/scripts* directory. To execute the *installhasvss* script, enter:

```
$SYBASE/OCS-12_0/bin/isql -Usa -Ppassword -Sservername <
../scripts/installhasvss
```

installhasvss prints messages as it creates stored procedures and creates the SYB_HACMP server.

Assign *ha_role* to SA

You must have the **ha_role** on both Adaptive Servers to run **sp_companion**. To assign the **ha_role**, issue the following from **isql**:

```
sp_role "grant", ha_role, sa
```

You must log out and then log back in to the Adaptive Server for the change to take effect.

Configure the IBM AIX Subsystem for Sybase's Failover

Perform the steps in this section to configure IBM AIX for Failover.

Modify the *ASE_HA.sh* Script

The *ASE_HA.sh* script is used to start, stop, and monitor an Adaptive Server in a high availability environment. Adaptive Server includes this script in the *$SYBASE/ASE-12_0/install* directory. You must make a copy of this script and modify it for your environment for both Adaptive Servers running in the cluster. The modifications you make to the script will slightly differ depending on whether the script is for the primary or secondary companion. Each node has a copy of this script at the same location (for example, both nodes have a copy of the script in */usr/u/sybase*), and both copies only have read, write, and execute permissions for "root." An easy way to do this is to first modify both scripts on the same node, copy both the scripts to the other node, and then set the appropriate permissions for the scripts on both nodes.

To modify the script for your environment:

1 Change to the *$SYBASE/ASE-12_0/install* directory.

2 As root, copy *ASE_HA.sh* to the HACMP event handler script directory, usually in */usr/sbin/cluster/events*, and name it:

```
RUNHA_<server_name>.sh
```

where *server_name* is the Adaptive Server to be monitored.

For example, to copy a *ASE_HA.sh* script for a server named MONEY1 to the */usr/sbin/cluster/events* directory, enter:

```
cp ASE_HA.sh /usr/sbin/cluster/events/RUNHA_MONEY1.sh
```

3 You must edit the *RUNHA_server_name.sh* script for your environment. The original *ASE_HA.sh* script contains the variables listed below. Edit the lines that include "__FILL_IN__" (and any other lines that require editing) with the values for your site:

- MONITOR_INTERVAL – is the interval of time, in seconds, *RUNHA_server_name.sh* waits between checks to see if the dataserver process is alive.

- RECOVERY_TIMEOUT – is the maximum amount of time the high availability subsystem waits, in seconds, before determining the companion failed to start. Make sure you set this number that is sufficiently long enough for a loaded companion to reboot. RECOVERY_TIMEOUT is also used as the maximum amount of time the subsystem waits for failover and failback to complete.
- SHUTDOWN_TIMEOUT – is the maximum time the high availability subsystem waits for the companion to shutdown before killing it.

Note This value should always be less the amount of time it takes for the HACMP **wait time** parameter to go into a **config_too_long** state. This is 360 seconds by default. If there is a possibility that your server will take longer than this to boot up, you can reconfigure this value by executing:

```
chssys -s clstrmgr -a "-u milliseconds_to_wait"
```

- RESPONSE_TIMEOUT – is the maximum amount of time the subsystem allows for a simple query to return a result set, and is used to diagnose whether or not the companion server is hung. For example, if **isql** fails to establish a connection in 60 seconds, it automatically times out and exits. However, if **isql** successfully connects, but does not return a result set, RESPONSE_TIMEOUT may determine that the companion server is hung. By default, RESPONSE_TIMEOUT is set to 999999.
- ASE_FAILOVER – can be set to either yes or no:
  - yes - Monitors the companion server for hung or dead processes and stops HACMP services on this node so the devices failover to the secondary node. If set to "yes", one must run **sp_companion configure** on the server as well to keep the high availability consistent.
  - no - Do not bring down the HACMP subsystem on this node even if the primary companion fails over. This setting is useful if you need to bring down a companion for maintenance or reconfiguration.

Note If you are configuring an asymmetric setup, set ASE_FAILOVER to "no."

- BASIC_FAILOVER – is set to either "yes" or "no:"

  - yes - Use the failover mechanisms provided by the HACMP subsystem if it determines the servers are running in modes that allow failover. When a failover occurs, the HACMP subsystem monitor first checks if the companions are in a correct mode to perform a failover. If the companions are not enabled for Sybase's Failover (that is, they do have **enable ha** set to 1), or they are running in single-server mode, or if the secondary companion is down, the HACMP subsystem monitor checks if BASIC_FAILOVER is set. If it is, the monitor attempts to start up the primary companion on the secondary node.
  - no - Do not revert to mode 0 failover even if Sybase's Failover criteria is not met. That is, if BASIC_FAILOVER is set to no, failover does not happen at either the node or the companion level.
- retry – is the number of times the HACMP subsystem attempts rebooting on the local node before failing over. Set this to a high number for an asymmetric setup so the secondary companion is more likely to reboot itself if it ever goes down. The default is 0, which means that the companion will not reboot on the same node if it goes down.
- SYBASE_ASE – is the installation directory of Sybase Adaptive Server products. The default is ASE-12_0.
- SYBASE_OCS – is the installation directed of Sybase Open Client products. The default is OCS-12_0
- PRIM_SERVER – is the name of the primary companion.
- SEC_SERVER – is the name of the secondary companion.
- PRIM_HOST – is the name of the primary host or service interface name.
- SEC_HOST – is the name of the secondary host or service interface name.
- PRIM_SYBASE – is the directory to which the $SYBASE environment variable should be set on the primary host. If you are using local devices, the location must be the same on both nodes. If you are using a shared device, this location must be different on both nodes.

- SEC_SYBASE – is the directory to which the $SYBASE environment variable should be set on the secondary host. If you are using local devices, the location must be the same on both nodes. If you are using a shared device, this location must be different on both nodes.
- PRIM_SYBASE_HOME – is the path to the directory in the secondary host in which the Adaptive Server products are installed. Usually this is *$SYBASE/$SYBASE_ASE*
- PRIM_ISQL – is the path to the isql binary on the primary host.
- SEC_ISQL – is the path to the isql binary on the secondary host.
- HA_LOGIN – is the login of the user with the **sa_role** and **ha_role**. This has to be the same on both the primary and secondary companion.
- HA_PWD – is the password for the HA_LOGIN. This has to be the same on both the primary and secondary companion.
- PRIM_RUNSCRIPT – is the name of the RUNSERVER file that is used to bring up the primary companion.
- PRIM_CONSOLE_LOG – is the full path to the errorlog for the current primary companion session. This can be any file that has sufficient space and is writable by root. The default is $SYBASE/install.
- SEC_CONSOLE_LOG – is the full path to the errorlog for the current secondary companion session. This can be any file that has sufficient space and is writable by root. The default is $SYBASE/install.

4 Edit the script for the primary companion. The example below shows the settings in the *RUNHA_MONEY1.sh* script for primary companion MONEY1 running on host FIN1, and for secondary companion PERSONEL1 running on host HUM1. In this example, when the primary companion shuts down, the monitor script tries one time to reboot the primary companion on node FIN1. If this fails, the script shuts down the HACMP services on FIN1 and moves the database devices for MONEY1 to PERSONEL1 on HUM1. If PERSONEL1 is down or in an inconsistent state, the script starts MONEY1 on HUM1

Variable	Primary Companion
ASE_FAILOVER	yes
BASIC_FAILOVER	yes
RETRY	1

Variable	Primary Companion
PRIM_SERVER	MONEY1
PRIM_HOST	FIN1
HA_LOGIN	“sa”
HA_PWD	“Odd1Think
PRIM_CONSOLE_LOG	*$SYBASE/$SYBASE_ASE/install/MONEY1.CS_log*
SEC_SERVER	PERSONEL1
SEC_HOST	HUM1
SEC_CONSOL_LOG	*$SYBASE/$SYBASE_ASE/install/PERSONEL.CS_log*

5 Edit the script for the secondary companion. These values will differ depending on whether you are using an asymmetric or a symmetric setup.

If this is an asymmetric setup, the values for PRIM_SERVER should be the same as SEC_SERVER (the name of the secondary companion). PRIM_HOST should be the same as SEC_HOST, and PRIM_SYBASE should be the same as SEC_SYBASE.

If this is a symmetric setup, the values for the PRIM_SERVER, PRIM_HOST, PRIM_SYBASE, SEC_SERVER, SEC_HOST, and SEC_SYBASE in the secondary companion script are the opposite of what is set in the primary companion script.

Table 8-1 describes the values for the variables for both an asymmetric setup and a symmetric setup on primary companion MONEY1 and secondary companion PERSONEL1:

Table 8-1: Values for the seconday companion

Variables	Asymmetric Secondary companion	Symmetric Secondary Companion
RETRY	10	1
ASE_FAILOVER	no	yes
BASIC_FAILOVER	no	yes
PRIM_SERVER	PERSONEL1	PERSONEL1
PRIM_HOST	HUM1	HUM1
SEC_SERVER	PERSONEL1	MONEY1
SEC_HOST	HUM1	FIN1

Configure the Resource Groups in HACMP

Note You can perform the steps described in this section either from the command line or through the configuration utility SMIT. This document describes only SMIT. See the HACMP for AIX documentation for information about performing these steps the command line.

Shut down the cluster services on both nodes in 'graceful' mode, and then log in to the boot IP addresses of the primary node as 'root' and perform the following tasks.

1 Start SMIT.

2 From the Cluster Resources screen, select Add a Resource Group if you are creating a new resources group, or select Change/Show a Resource Group if you are changing an existing resources group.

3 Enter "cascading" in the Node Relationship field, as described below:

```
Define the Resource Group for the Primary Companion:
Resource Group Name                      [<resgrp1>]
Node Relationship                        [cascading]
Participating Node Names                [<primary_node> <secondary_node>]

Define the Resource Group for the Secondary Companion:
(For Asymmetric Failover Configuration)
Resource Group Name                      [<resgrp2>]
Node Relationship                        [cascading]
Participating Node Names                 [<secondary_node>]

(For Symmetric Failover Configuration)
Resource Group Name                      [<resgrp2>]
Node Relationship                        [cascading]
Participating Node Names                [<secondary_node> <primary_node>]
```

4 Configure each of the resource groups defined in step 2 on page 104. For the Application Server field, enter the name of the primary companion. Enter the information in all the required fields, such as IP Label, Volume Groups, and File systems. Repeat this step for each of the companions.

5 Define the primary and secondary companions as application servers in HACMP Cluster Resources. Select either Add Application Server or Change Application Server, and enter these values:

- For the Start/Stop Scripts, enter the name of the scripts you created in step 2 on page 99.

- For primary and symmetric secondary companions, enter “monitor” and “failover” as the arguments for the start and stop scripts, respectively.
- For an asymmetric secondary companion, use “monitor” and “stop” as the arguments for the start and stop script, respectively.

For example:

```
Define the Primary Companion Server
Server Name   [<primary_ase>]
Start Script  [/usr/sbin/cluster/events/RUN_<primary_ase>_ha monitor]
Stop Script   [/usr/sbin/cluster/events/RUN_<primary_ase>_ha failover]

Define the Secondary Companion Server:
(For Asymmetric Failover Configuration)
Server Name   [<secondary_ase>]
Start Script  [/usr/sbin/cluster/events/RUN_<secondary_ase>_ha monitor]
Stop Script   [/usr/sbin/cluster/events/RUN_<secondary_ase>_ha stop]

(For Symmetric Failover Configuration)
Server Name   [<secondary_ase>]
Start Script  [/usr/sbin/cluster/events/RUN_<secondary_ase>_ha monitor]
Stop Script  [/usr/sbin/cluster/events/RUN_<secondary_ase>_ha failover]
```

6 Synchronize the cluster resources. Using SMIT on the node on which you have performed steps 1 through 6, go to the Cluster Resources screen and select Synchronize Cluster Resources. This propagates the changes you made to all the other nodes within the same cluster. In some cases, you may need to stop the HACMP services and reboot both the nodes before performing the synchronization. Make sure the synchronization does not produce any errors.

Configure Companion Servers for Failover

Perform the tasks in this section to configure the Adaptive Servers as primary and secondary companions in a high availability system.

Run *sp_companion* With *do_advisory* Option

You must configure the secondary companion with sufficient resources to perform the work of both servers during failover. The secondary companion may have attributes that will prevent a successful cluster operation. For example, if both the primary and secondary companions are configured for 250 user logins, during failover, the secondary companion only has the resources for half the number of potential user logins necessary. Instead, both MONEY1 and PERSONEL1 should be configured for 500 user logins.

The **sp_companion do_advisory** option checks the configuration options on both the primary and the secondary companion to make sure a cluster operation (such as configuring an Adaptive Server as a secondary companion) will be successful. **sp_companion do_advisory** advises you of any configuration options that should be changed.

See Chapter 6, “Running do_advisory” for a complete description of the **sp_companion do_advisor** option.

Configure for Asymmetric Configuration

Use **sp_companion** to configure the primary companion for asymmetric configuration:

```
sp_companion "primary_server_name", configure, proxy_device_name, login_name,
password, cluster_login, cluster_login_password.
```

Where:

- *primary_server_name* is the name of the primary Adaptive Server as defined in the interfaces file entry and in *sysservers*.
- The *proxy_device_name* is the device where the proxy user databases are created (if no proxy device is specified, the default devices for the server will be used for proxy databases). For more information, see “sp_companion” on page 183.
- *login_name* is the name of the user performing this cluster operation (they must have both the sa_role and the ha_role).

- *password* is the password of the person performing this cluster operation

This example configures an Adaptive Server named PERSONEL1 as a secondary companion:

```
 sp_companion "PERSONEL1", configure, null, sa, "Odd2Think"
Server 'MONEY1' is alive and cluster configured.
Step: Access verified from Server:'MONEY1' to Server:'PERSONEL1'
Server 'PERSONEL1' is alive and cluster configured.
Step: Access verified from Server:'PERSONEL1' to Server:'MONEY1'
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
.....
Step: Companion servers configuration check succeeded
Step: Server handshake succeeded
Step: Master device accessible from companion
Step: Added the servers 'MONEY1' and 'PERSONEL1' for cluster config
Step: Server configuration initialization succeeded
Step: Synchronizing server logins from companion server
Step: Synchronizing remoteserver from companion server
Step: Synchronizing roles from companion server
Step: Synchronizing server-wide privs from companion server
Step: User information syncup succeeded
Step: Server configured in normal companion mode
```

If there are user databases during the **sp_companion** configuration, you see messages similar to these:

```
Step: Created proxy database 'pubs2'
 Step: Proxy status for database has been set. Please Checkpoint the database
'pubs2'
 Step: Server configured in normal companion mode"
 Starting companion watch thread
```

Before you configure the companions for symmetric configuration, you must first configure them for asymmetric configuration.

See "Asymmetric Companion Configuration" on page 22 for more information about asymmetric configuration.

Configure for Symmetric Configuration

After you configure your companions for asymmetric failover, you can configure them for symmetric configuration. In a symmetric configuration, both servers act as primary and secondary companions. See Figure 3-2 on page 24 for a description of symmetric configuration.

Issue **sp_companion** from the secondary companion to configure it for symmetric configuration. Use the same syntax as for asymmetric configuration. See "Configure for Asymmetric Configuration," above, for a description of the syntax for **sp_companion**.

The following example adds an Adaptive Server named MONEY1 as the secondary companion to the Adaptive Server named PERSONEL1 described in "Configure for Asymmetric Configuration" on page 104:

```
sp_companion 'MONEY1', configure, null, sa, MyPassword, sa_cluster_login,
MyClusterPassword
Server 'MONEY1' is alive and cluster configured.
Step: Access verified from Server:'MONEY1' to Server:'PERSONEL1'
Server 'PERSONEL1' is alive and cluster configured.
Step: Access verified from Server:'PERSONEL1' to Server:'MONEY1'
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
.....
Step: Companion servers configuration check succeeded
Step: Server handshake succeeded
Step: Master device accessible from companion
Step: Added the servers 'MONEY1' and 'PERSONEL1' for cluster config
Step: Server configuration initialization succeeded
Step: Synchronizing server logins from companion server
Step: Synchronizing remoteserver from companion server
Step: Synchronizing roles from companion server
Step: Synchronizing server-wide privs from companion server
Step: User information syncup succeeded
Step: Server configured in normal companion mode
```

Bring Up Primary Companion as a Monitored Resource

Follow the instructions in this section to start the primary companion as a service that is monitored by the high availability subsystem.

Note Before you begin monitoring the primary companion, make sure you no longer need to shut down the primary server for maintenance or other purposes. Once you begin monitoring the primary companion, you must move it to suspended mode to bring it down. If you are unsure, start the primary server using the startserver script in *$SYBASE/ASE-12_0/install*, finish configuring the companion, then reboot the server using the steps described here.

To start the primary companion as a resource that is monitoring for failover:

1 Stop the HACMP services on the primary node.

2 Check */tmp/hacmp.out* to make sure the node_down event completed, then restart the HACMP services either by using SMIT or by executing this command as "root" at the command line:

```
/usr/sbin/cluster/etc/rc.cluster -boot '-N' '-b'  '-i'
```

This automatically executes the *RUNHA_<server_name>.sh* monitor script, which brings up the primary companion and monitors it during crash or hang situations.

Repeat this process on the secondary node to bring up the secondary companion.

Administering Sybase's Failover

This section describes information about:

- Failing back to the primary companion
- Resuming normal companion mode
- Suspending normal companion mode
- Troubleshooting information about common problems with Sybase Failover.

Failing Back to the Primary Node

Failback automatically occurs on HACMP. When HACMP is brought up on the primary node, the *stop_server* event on the secondary node triggers the monitoring script to execute **sp_companion 'prepare_failback'**.

To fail back to the primary node, make sure that the secondary companion is in secondary failover mode, and bring up HACMP services on the primary node. To make sure that **sp_companion 'prepare_failback'** was executed successfully, search for this string in */tmp/hacmp.out*:

```
SYBASE HA MONITOR: Prepare_failback was successful.
```

Note Before you start the HACMP services on the primary node, make sure that the secondary node is up and the secondary companion is running in secondary failover mode. If the secondary companion or secondary node is not up and running, do not bring up the primary companion. If both nodes are down, or the HACMP services has stopped on both nodes, always restart the secondary node and its HACMP services before restarting the primary node.

Manually Failing Back

Note If the automatic failback failed, examine the logs to make sure that the high availability system performed the following steps. If it did not, you can perform them manually. You must perform them in the sequence described below.

---

1 Stop the HACMP subsystem with the **takeover** mode on the primary node. This shuts down the primary companion and fails over its resources to the secondary companion.

2 Shutdown and then restart your secondary companion.The *RUNHA_servername.sh* restarts the companion automatically after you shut it down if RETRY is set to a value greater than 0.

3 Log in as LOGIN_NAME to the secondary companion through isql and make sure that it is running in secondary failover mode.

4 Issue **sp_companion 'prepare_failback'**. For example, to fail back from the secondary companion PERSONEL1:

```
sp_companion MONEY1, 'prepare_failback'
```

5 Restart HACMP on the primary node.

6 Log in to the primary companion using isql and make sure that it is running in primary failback mode.

7 Issue sp_companion 'resume'. For example, to resume companion mode for primary companion MONEY1:

```
sp_companion PERSONEL1, 'resume'
```

Note You cannot connect clients with the failover property (for example **isql -Q**) until you issue **sp_companion resume**. If you do try to reconnect them after issuing **sp_companion prepare_failback**, the client hangs until you issue **sp_companion resume**.

Suspending Companion Mode

If you must shut down the primary companion for maintenance but do not want to fail over to the secondary companion, you must temporarily suspend companion mode. When the companion mode is suspended, synchronization between the companions does not occur, and the primary companion cannot fail over to the secondary companion. However, suspended mode is very useful for performing such maintenance tasks as changing configuration parameters:

1 To move to suspended mode, issue:

```
sp_companion <primary_server_name>, suspend
```

For example, to suspend primary companion MONEY1:

```
sp_companion MONEY1, suspend
```

2 Kill the monitoring process so it does not trigger a failover when the companion server goes down. As "root," enter:

```
ps -ef|grep "RUNHA_<server_name>.sh monitor"
kill -9 <pid>
```

For example, to kill the monitoring process for MONEY1 which has a *pid* of 2509:

```
ps -ef|grep "RUNHA_MONEY1.sh monitor"
kill -9 2509
```

3 Shut down the primary companion.

After killing the monitoring process, you can bring the companion server down as many times as necessary and it will not failover.

Restarting Shutdown Companion During Suspended Mode

Use the start up script in *$SYBASE/ASE-12_0/install* to restart the primary companion without it being monitored:

```
startserver -f ./RUN_<server_name>
```

For example, to start the MONEY1 companion:

```
startserver -f ./RUN_MONEY1
```

If you use this script to start a companion server, it will not fail over when the server goes down, even if it is configured to do so. Use this method only if you are doing maintenance, and you are certain that you do not want the server databases to be accessible when the server is down.

Resuming Normal Companion Mode

The steps for resuming normal companion mode are slightly different depending on whether you are moving from suspended mode or from failover mode.

Resuming Normal Companion Mode from Suspended Mode

To resume normal companion mode between two companions that have been moved to suspended mode:

1 Shut down the primary companion if it is not already.

2 Stop the HACMP services on the primary node in "graceful" mode.

3 Restart the HACMP services on the primary node.

Resuming Normal Companion Mode

To resume normal companion mode between two companions that are in failover mode, simply restart the HACMP services on the primary node, and perform the following:

1 Check that both companions are in failback mode by issuing sp_companion with no parameters.

2 Resume normal companion mode by issuing:

```
sp_companion secondary_server_name, resume
```

For example to issue normal companion mode for primary companion PERSONEL1:

```
sp_companion PERSONEL1, resume
Server 'MONEY1' is alive and cluster configured.
Step: Access verified from Server:'MONEY1' to Server:'PERSONEL1'
Server 'PERSONEL1' is alive and cluster configured.
Step: Access verified from Server:'PERSONEL1' to Server:'MONEY1'
Step: Checkin to See if the remote server is up
Step: Synchronizing server logins from companion server
Step: Synchronizing remoteserver from companion server
Step: Synchronizing roles from companion server
Step: Synchronizing server-wide privs from companion server
Step: User information syncup succeeded
sys_id ses_id      ses_id2     ses_status Purged from s1.
------ ----------- ----------- ---------- ------------------------------

(0 rows affected)
sys_id ses_id      ses_id2     ses_status Copied to s1.
------ ----------- ----------- ---------- ------------------------------

(0 rows affected)
sys_id ses_id      ses_id2     ses_status Purged from s2.
------ ----------- ----------- ---------- ------------------------------
(0 rows affected)
Step: Syssession information syncup succeeded
```

Dropping Companion Mode

To drop companion mode, issue:

```
sp_companion companion_name, "drop"
```

Dropping companion mode is an irreversible process; you must reconfigure the Adaptive Servers companion servers before you they will failover in a high availability system and retain all the functionality that Sybase's Failover provides. However, the nodes upon which the Adaptive Servers are running are still a monitored by the high availability subsystem.

If you drop the companion mode while the *RUNHA_<servername>.sh* script is running, the script continues to monitor the server for any down or hung instances. If you plan to shutdown the server and do not want the node to failover, you need to kill the monitor process by issuing:

```
kill -9 ‘ps -ef | grep "RUNHA_<servername>.sh monitor" | grep -v grep | awk
’{print $2}’‘
```

If you do not kill the monitor process, it triggers a failover of the resources when it detects that the companion has gone down, and tries to restart the companion from either the primary or secondary node, depending on your settings for RETRY and BASIC_FAILOVER.

Troubleshooting Failover on HACMP for AIX

This section includes troubleshooting information about common errors.

Error Message 18750

If a companion server issues error message 18750, check the *@@cmpstate* of your servers. If your primary companion is in normal companion mode, but the secondary companion is in secondary failover mode, your cluster is in an inconsistent state, and you need to manually recover from this. This inconsistent state may be caused by an **sp_companion 'prepare_failback'** command failing on the secondary companion. You can determine whether this happened by examining the HACMP log on the secondary node, (located in */tmp/hacmp.out*). To recover from this, perform the following steps manually:

1 Shut down both the primary and the secondary companions.

2 Reboot the secondary companion.

3 Repair all databases marked “suspect.” To determine which databases are suspect, issue:

```
select name, status from sysdatabases
```

Databases marked suspect have a *status* value of 320.

4 Allow updates to system tables:

```
sp_configure “allow updates”, 1
```

5 For each suspect failed over database, perform the following:

```
1> update sysdatabase set status=status-256 where name='database_name'
2> go
1> dbcc traceon(3604)
2> go
1> dbcc dbrecover(database_name)
2> go
```

6 From the secondary companion, issue:

```
sp_companion primary_companion_name, prepare_failback
```

For example, from primary companion MONEY1:

```
sp_companion MONEY1, prepare_failback
```

Make sure that this command executes successfully.

7 Restart the HACMP services on the primary node.

Recovering From a Failed *prepare_failback*

During a failback, if **prepare_failback** was executed successfully on the secondary companion but the primary companion fails to boot, perform the following to rollback and then reissue the **prepare_failback** command:

1 Check the primary companion’s errorlog and the HACMP error log to find the reason the server failed to boot, and correct the problems.

2 Stop the HACMP services on the primary node with **takeover**

3 Login to the secondary companion as LOGIN_NAME, and issue:

```
dbcc ha_admin ("", "rollback_failback")
dbcc ha_admin ("", "rollback_failover")
```

Your companion servers should both be back in the failover mode

4 Restart HACMP on the primary node.

Location of Failover Logs

Sybase's Failover includes the following logs. These logs are helpful for investigating and diagnosing errors encountered during HACMP failover:

- */tmp/hacmp.out* – contains output of the HACMP activities, as well as the output from the *RUNHA_server_name.sh* monitoring script. For general HACMP failure, search for the string "ERROR". For output of the *RUNHA_server_name.sh* script, search for "SYBASE HA MONITOR".

  After determining the reason for the failure, correct it, then go to the Cluster Recovery Aids screen of SMIT and do a Recover From Script Failure, before continuing.

  If a node does not include a sufficient amount of space in a particular file system, HACMP hangs in the middle of a failover or failback process, which results in a **config_too_long** lock. If this occurs, you must clean up the full directories, then start SMIT and move to the Cluster Recovery Aids screen and perform a Recover From Script Failure before continuing.

- $PRIM_CONSOLE_LOG – the location of this log is defined in the *RUNHA_server_name.sh* monitoring script. This error log includes the Adaptive Server information from the last execution of the *RUNHA_server_name.sh* script.

CHAPTER 9

Configuring Adaptive Server for Failover on Digital UNIX TruCluster

Perform the tasks in this chapter to configure Adaptive Server for Failover on DEC TruCluster.

Configure Hardware and Operating System for High Availability

Sybase high availability requires the following hardware and system components:

- Two homogenous, networked systems with similar configurations in terms of resources like CPU, memory, etc.
- These systems should be equipped with the high availability subsystem package and the associated hardware.
- Must have devices that are accessible to both nodes.
- Vendor provided mirroring, not Sybase mirroring, should be used for media failure protection.

  See your hardware and operating system documentation for information about installing platform specific high availability software.

Requirements for Running Sybase's Failover on Digital Unix

Configuring for high availability on Digital Unix TruCluster requires:

- 2 hardware-compatible nodes running Compaq Tru64 version 4.0D with Compaq TruCluster version 1.5.
- Each node has 2 IP addresses, one for the node and one for the TruCluster services.
- A TruCluster service name for the primary companion (for example, *primary*).
- A TruCluster service name for the secondary companion (for example, *secondary*).
- The name of the primary companion name must be the same its TruCluster service name.
- The name of the secondary Adaptive Server companion must be the same as its TruCluster service name.

Prepare Adaptive Server to Work with The High Availability Subsystem

Perform the tasks in this section to prepare Adaptive Server for a high availability configuration.

Install Adaptive Servers

Install both the primary and the secondary servers. You can install the companions on either local or shared file systems. If they are installed on shared file systems, the file system must not be the same. This is to prevent the file systems from overwriting each other during a device failover. For example, you can install the primary companion on */node1_sybase*, but install the secondary companion on */node1_sybase*.

If the servers are installed on local file system, the name of the file systems must be the same. For example, both the primary and the secondary companion could be installed in */sybase*.

The file systems that contain *$SYBASE* must be either local or shared; you cannot mix local and shared file systems for *$SYBASE* in the cluster.

The database devices for the primary companion must be on the shared SCSI bus disks.

If you are creating an asymmetric configuration, you can use any device (either shared or local) for the database device. If you are creating a symmetric configuration, you must use a device on the shared SCSI bus disks.

The primary companion can be either a newly installed Adaptive Server, or it can be upgraded from a previous version of Adaptive Server with existing databases, users, and so on.

The secondary companion must be a newly installed Adaptive Server without any user logins or user databases. This ensures that all user logins and database names are unique within the cluster. After configuration for failover is complete, you can add user logins and databases to the secondary companion.

See your the installation documentation for your platform for information about installing and configuring Adaptive Server.

Add Entries for Both Adaptive Servers to the Interfaces File

The interfaces file for both primary and secondary companion must include entries for both companions. For example, the interfaces file for the servers used in the examples above would have entries for both MONEY1 and PERSONEL1. The server entry in the interfaces file must use the same network name that is specified in *sysservers*. For information about adding entries to the interfaces file, see the installation documentation for your platform.

Add Entries to *interfaces* File for Client Connections During Failover

To enable clients to reconnect to the failed over companion, you must add an additional line to the interfaces file. By default, clients connect to the port listed in the query line of the server entry. If that port is not available (because that server has failed over), the client connects to the server listed in the *hafailover* line of the server entry. Here is a sample interfaces file for a primary companion named MONEY1 and a secondary companion named PERSONEL1:

```
MONEY1
    master tcp ether FIN1 4100
    query tcp ether FIN1 4100
    hafailover PERSONEL1
```

Use **dsedit** to add entries to the interfaces file. If the interfaces entries already exist, you must modify them to work for Failover.

See the the Utility Programs manual for your platform for information about **dsedit**.

Set *$SYBASE* the Same on a Local File system

If you installed *$SYBASE* on a local file system, *$SYBASE* must point to the same directory name on both companions. You can accomplish this by either:

- Making sure that the *$SYBASE* release directory on each companion is created in the same directory.
- If the companions have the *$SYBASE* release directory in different locations, create a directory with the same path on both companions that acts as a symbolic link to the actual *$SYBASE* release directory.

  For example, even though primary companion MONEY1 has a release directory of */usr/u/sybase1* and PERSONEL1 has uses */usr/u/sybase2* as its release directory, their *$SYBASE* must point to the same path.

Both MONEY1 and PERSONEL1 have */SYBASE*, which they establish as a symbolic link to their respective *$SYBASE* release directories. On MONEY1, */SYBASE* is a link to */usr/u/sybase1*, and on PERSONEL1, */SYBASE* is a link to */use/u/sybase2*.

If you installed $SYBASE on a local file system, you must also have copies of both companion's RUNSERVER files in *$SYBASE/ASE-12_0/install* on both nodes.

The sybha Executable

The sybha executable provides the ability for the Adaptive Server High Availability Basis Services library to interact with each platform's high availability cluster subsystem. The Adaptive Server High Availability Basis Services library calls sybha. sybha is located in the $SYBASE/ASE-12_0/bin directory. Before sybha can run, you must change its ownership and permissions. You must also edit a file named *sybhauser* in the *$SYBASE/ASE-12_0/install* directory. This file contains a list of the users who have System Administrator privileges on the cluster. Sybase strongly recommends that severely limit the number of users who have System Administrator privileges on the cluster.

As root, perform the following:

1 Change directory to *$SYBASE/ASE-12_0/bin* directory:

```
cd $SYBASE/ASE-12_0/bin
```

2 Change the ownership of the **sybha** to root:

```
chown root sybha
```

3 Modify the file permissions for **sybha** to 4755

```
chmod 4755 sybha
```

4 Change directory to *$SYBASE/ASE-12_0/install* directory.

```
cd $SYBASE/ASE-12_0/install
```

5 Add the users to the *sybhauser* file who need to administer the high availability subsystem. These logins must be in the format of UNIX login IDs, not Adaptive Server logins. For example:

```
sybase
coffeecup
spooner
venting
```

```
howe
```

6 Change the permissions of sybhauser to root:

```
chown root sybhauser
```

7 Modify the file permissions for *sybhauser* so it can only be modified by root:

```
chmod 644 sybhauser
```

Verify Configuration Parameters

You must enable the following configuration parameters before you configure Adaptive Server for failover:

- **enable CIS** – Enables Component Integration Services (CIS). This configuration parameter is enabled by default.
- **enable xact coordination** – Enables Distributed Transaction Management (DTM). This configuration parameter is enabled by default.
- **enable HA** – Enables Adaptive Server to function as a companion in a high availability system. **enable HA** is off by default. This configuration is static, so you must reboot Adaptive Server for it to take effect. This parameter causes a message to be written to your errorlog stating that you have started the Adaptive Server in a high availability system.

See the *System Administration Guide* for information about enabling configuration parameters.

Add Thresholds to the Master Log

If you have not already done so, you must add a threshold to the master log.

1 Define and execute **sp_thresholdaction** on the master database's log to set a threshold on the number of pages left before a dump transaction occurs. Sybase does not supply **sp_thresholdaction**. See the Adaptive Server Reference Manual for information about creating this system procedure.

2 Place thresholds on the master and sybsystemprocs log segments so they do not fill up:

```
sp_addthreshold "master", "logsegment", 250, sp_thresholdaction
sp_addthreshold "sybsystemprocs", "logsegment", 250, sp_thresholdaction
```

3 You must reboot the primary companion for this static parameter to take effect.

Create New Default Device Other Than Master

By default, the master device is the default device in a newly installed Adaptive Server. This means that, if you create any databases (including the proxy databases used by failover), they are automatically created on the master device. However, adding user databases to master makes it more difficult to restore the master device from a system failure. To make sure that the master device contains as few extraneous user databases as possible, create a new device using **disk init**. Use **sp_diskdefault** to specify the new device as the default before you configure Adaptive Server as a companion for failover.

For example, to add a new default device named *money_default_1* to the MONEY1 Adaptive Server, enter:

```
sp_diskdefault money1_default1, defaulton
```

The master device continues to also be a default device until you specifically issue the following to suspend it as the default device:

```
sp_diskdefault master, defaultoff
```

See the *Adaptive Server Reference Manual* for more information about **disk init** and **sp_diskdefault**.

Add the Local Server to *sysservers*

Using **sp_addserver**, add the local server as the local server in *sysservers* using the network name specified in the interfaces file. For example, if the companion MONEY1 uses the network name of MONEY1 in the interfaces file:

```
sp_addserver MONEY1, local, MONEY1
```

You must reboot Adaptive Server for this change to take effect.

Add Secondary Companion to *sysservers*

Add the secondary companion as a remote server in *sysservers*:

```
sp_addserver server_name
```

By default, Adaptive Server adds the server with an *srvid* of 1000. You do not need to reboot Adaptive Server for the change to take effect.

Run *installhasvss* to Install HA Stored Procedures

Note You must perform the tasks described in Add Entries for Both Adaptive Servers to the Interfaces File, above, before running *installhasvss*. If you run *installhasvss* before performing these tasks you will have to re-run *installmaster* to re-install all the system stored procedures.

The *installhasvss* script performs the following tasks to configure Adaptive Server for failover:

- Installs the stored procedures required for failover (for example, **sp_companion**).
- Installs the SYB_HACMP server entry in *sysservers*.

You must have System Administrator privileges to run the *installhasvss* script.

installhasvss in located in the *$SYBASE/ASE-12_0/scripts* directory. To execute the *installhasvss* script, enter:

```
$SYBASE/OCS-12_0/bin/isql -Usa -Ppassword -Sservername
< ../scripts/installhasvss
```

installhasvss prints messages as it creates stored procedures and creates the SYB_HACMP server.

Assign *ha_role* to SA

You must have the **ha_role** on both Adaptive Servers to run **sp_companion**. To assign the **ha_role**, issue the following from **isql**:

```
sp_role "grant", ha_role, sa
```

You must log out and then log back in to the Adaptive Server for the change to take effect.

Configure the Digital Unix Subsystem for Sybase's Failover

Perform the steps in this section to configure Digital Unix's TruCluster for Failover.

Modify the *ASE_HA.sh* Script

The *ASE_HA.sh* script is used to start, stop, and monitor an Adaptive Server in a high availability environment. Adaptive Server includes this script in the *$SYBASE/ASE-12_0/install* directory. You must make a copy of this script and modify it for your environment for both Adaptive Servers running in the cluster. The modifications you make to the script will slightly differ depending on whether the script is for the primary or secondary companion. Each node has a copy of this script at the same location (for example, both nodes have a copy of the script in */usr/u/sybase*), and both copies only have read, write, and execute permissions for "root." An easy way to do this is to first modify both scripts on the same node, copy both the scripts to the other node, and then set the appropriate permissions for the scripts on both nodes.

To modify the script for your environment:

1 Change to the *$SYBASE/ASE-12_0/install* directory.

2 As root, copy *ASE_HA.sh* to a file named *RUNHA_<server_name>.sh*

where *server_name* is the Adaptive Server to be monitored.

For example, to copy a *ASE_HA.sh* script for a server named MONEY1:

```
cp ASE_HA.sh RUNHA_MONEY1.sh
```

3 You must edit the *RUNHA_server_name.sh* script for your environment. The original *ASE_HA.sh* script contains the variables listed below. Edit the lines that include "__FILL_IN__" (and any other lines that require editing) with the values for your site:

- MONITOR_INTERVAL – is the interval of time, in seconds, *RUNHA_server_name.sh* waits between checks to see if the dataserver process is alive.
- RECOVERY_TIMEOUT – is the maximum amount of time the high availability subsystem waits, in seconds, before determining the companion failed to start. Make sure you set this number that is sufficiently long enough for a loaded companion to reboot. RECOVERY_TIMEOUT is also used as the maximum amount of time the subsystem waits for failover and failback to complete.
- SHUTDOWN_TIMEOUT – is the maximum time the high availability subsystem waits for the companion to shutdown before killing it.

  Note This value should always be less the amount of time configured for the action script **time_out** in the **asemgr**

- RESPONSE_TIMEOUT – is the maximum amount of time the subsystem allows for a simple query to return a result set, and is used to diagnose whether or not the companion server is hung. For example, if **isql** fails to establish a connection in 60 seconds, it automatically times out and exits. However, if **isql** successfully connects, but does not return a result set, RESPONSE_TIMEOUT may determine that the companion server is hung. By default, RESPONSE_TIMEOUT is set to 999999.
- ASE_FAILOVER – can be set to either yes or no:
  - yes - Monitors the companion server for hung or dead processes and relocates the TruCluster service to the secondary node. If set

to "yes", one must run **sp_companion...configure** on the server as well to keep the high availability consistent.

- no - Do not relocate the TruCluster subsystem on this node even if the primary companion fails over. This setting is useful if you need to bring down a companion for maintenance or reconfiguration.

Note If you are configuring an asymmetric setup, set ASE_FAILOVER to “no.”

NOTE:** This should only be set to "yes" if BOTH the servers are running ASE version 12.0 or later. Servers of previous versions should have this set to "no".

- BASIC_FAILOVER – is set to either “yes” or “no:”
  - yes - Use the failover mechanisms provided by the TruCluster subsystem if it determines the servers are running in modes that allow failover. When a failover occurs, the TruCluster subsystem monitor first checks if the companions are in a correct mode to perform a failover. If the companions are not enabled for Sybase’s Failover (that is, they do have **enable ha** set to 1), or they are running in single-server mode, or if the secondary companion is down, the TruCluster subsystem monitor checks if BASIC_FAILOVER is set. If it is, the monitor attempts to start up the primary companion on the secondary node.
  - no - Do not revert to mode0 failover (mode 0 restarts the primary companion on the secondary node, and does not involve Sybase’s Failover; for more information) even if Sybase’s Failover criteria is not met. That is, if BASIC_FAILOVER is set to no, failover does not happen at either the node or the companion level.
- retry – is the number of times the TruCluster subsystem attempts rebooting on the local node before failing over. Set this to a high number for an asymmetric setup so the secondary companion is more likely to reboot itself if it ever goes down. The default is 0, which means that the companion will not reboot on the same node if it goes down.
- SYBASE_ASE – is the installation directory of Sybase Adaptive Server products. The default is ASE-12_0.
- SYBASE_OCS – is the installation directory of Sybase Open Client products. The default is OCS-12_0

- PRIM_SERVER – is the name of the primary companion.
- SEC_SERVER – is the name of the secondary companion.
- PRIM_HOST – is the name of the primary host or service interface name.
- SEC_HOST – is the name of the secondary host or service interface name.
- PRIM_SYBASE – is the directory to which the $SYBASE environment variable should be set on the primary host. If you are using local devices, the location must be the same on both nodes. If you are using a shared device, this location must be different on both nodes.
- SEC_SYBASE – is the directory to which the $SYBASE environment variable should be set on the secondary host. If you are using local devices, the location must be the same on both nodes. If you are using a shared device, this location must be different on both nodes.
- PRIM_SYBASE_HOME – is the path to the directory in the primary host in which the Adaptive Server products are installed. Usually this is *$SYBASE/$SYBASE_ASE*.
- PRIM_SYBASE_HOME – is the path to the directory in the primary host in which the Adaptive Server products are installed. Usually this is *$SYBASE/$SYBASE_ASE*
- PRIM_SYBASE_HOME – is the path to the directory in the secondary host in which the Adaptive Server products are installed. Usually this is *$SYBASE/$SYBASE_ASE*
- PRIM_ISQL – is the path to the isql binary on the primary host.
- SEC_ISQL – is the path to the isql binary on the secondary host.
- HA_LOGIN – is the login of the user with the **sa_role** and **ha_role**. This has to be the same on both the primary and secondary companion.
- HA_PWD – is the password for the HA_LOGIN. This has to be the same on both the primary and secondary companion.
- PRIM_RUNSCRIPT – is the name of the RUNSERVER file that is used to bring up the primary companion.
- PRIM_CONSOLE_LOG – is the full path to the errorlog for the

current primary companion session. This can be any file that has sufficient space and is writable by root. The default is $SYBASE/install.

- SEC_CONSOLE_LOG – is the full path to the errorlog for the current secondary companion session. This can be any file that has sufficient space and is writable by root. The default is $SYBASE/install.

4 Edit the script for the primary companion. The example below shows the settings in the *RUNHA_MONEY1.sh* script for primary companion MONEY1 running on host FIN1, and for secondary companion PERSONEL1 running on host HUM1. In this example, when the primary companion shuts down, the monitor script tries one time to reboot the primary companion on node FIN1. If this fails, the script shuts down the TruCluster services on FIN1 and moves the database devices for MONEY1 to PERSONEL1 on HUM1. If PERSONEL1 is down or in an inconsistent state, the script starts MONEY1 on HUM1

Variable	Primary Companion
ASE_FAILOVER	yes
BASIC_FAILOVER	yes
RETRY	1
PRIM_SERVER	MONEY1
PRIM_HOST	FIN1
HA_LOGIN	“sa”
HA_PWD	“Odd1Think
PRIM_CONSOLE_LOG	*$SYBASE/$SYBASE_ASE/install/MONEY1.CS_log*
SEC_SERVER	PERSONEL1
SEC_HOST	HUM1
SEC_CONSOL_LOG	*$SYBASE/$SYBASE_ASE/install/PERSONEL.CS_log*

5 Edit the script for the secondary companion. These values will differ depending on whether you are using an asymmetric or a symmetric setup.

If this is an asymmetric setup, the values for PRIM_SERVER should be the same as SEC_SERVER (the name of the secondary companion). PRIM_HOST should be the same as SEC_HOST, and PRIM_SYBASE should be the same as SEC_SYBASE.

If this is a symmetric setup, the values for the PRIM_SERVER, PRIM_HOST, PRIM_SYBASE, SEC_SERVER, SEC_HOST, and SEC_SYBASE in the secondary companion script are the opposite of what is set in the primary companion script.

Table 9-1 describes the values for the variables for both an asymmetric setup and a symmetric setup on primary companion MONEY1 and secondary companion PERSONEL1:

Table 9-1: Values for the seconday companion

Variables	Asymmetric Secondary companion	Symmetric Secondary Companion
RETRY	10	1
ASE_FAILOVER	no	yes
BASIC_FAILOVER	no	yes
PRIM_SERVER	PERSONEL1	PERSONEL1
PRIM_HOST	HUM1	HUM1
SEC_SERVER	PERSONEL1	MONEY1
SEC_HOST	HUM1	FIN1

Configure Companion Servers for Failover

Perform the tasks in this section to configure the Adaptive Servers as primary and secondary companions in a high availability system.

Run *sp_companion* with *do_advisory* Option

You must configure the secondary companion with sufficient resources to perform the work of both servers during failover. The secondary companion may have attributes that will prevent a successful cluster operation. For example, if both the primary and secondary companions are configured for 250 user logins, during failover, the secondary companion only has the resources for half the number of potential user logins necessary. Instead, both MONEY1 and PERSONEL1 should be configured for 500 user logins.

The **sp_companion do_advisory** option checks the configuration options on both the primary and the secondary companion to make sure a cluster operation (such as configuring an Adaptive Server as a secondary companion) will be successful. **sp_companion do_advisory** advises you of any configuration options that should be changed.

See Chapter 6, “Running do_advisory” for a complete description of the **sp_companion do_advisor** option.

Configure for Asymmetric Configuration

Use **sp_companion** to configure the primary companion for asymmetric configuration:

```
sp_companion "primary_server_name", configure, with_proxydb, login_name,
password
```

Where:

- *primary_server_name* is the name of the primary Adaptive Server as defined in the interfaces file entry and in *sysservers*.
- The *with_proxydb* indicates that proxy databases are created on the secondary companion for all databases other than system databases. Any subsequent databases that are added also create proxy databases.
- *login_name* is the name of the user performing this cluster operation (they must have both the **sa_role** and the **ha_role**).

- *password* is the password of the person performing this cluster operation

This example configures an Adaptive Server named PERSONEL1 as a secondary companion:

```
 sp_companion "PERSONEL1", configure, with_proxydb, null, sa, "Odd2Think"
Server ’MONEY1’ is alive and cluster configured.
Step: Access verified from Server:’MONEY1’ to Server:’PERSONEL1’
Server ’PERSONEL1’ is alive and cluster configured.
Step: Access verified from Server:’PERSONEL1’ to Server:’MONEY1’
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
.....
Step: Companion servers configuration check succeeded
Step: Server handshake succeeded
Step: Master device accessible from companion
Step: Added the servers ’MONEY1’ and ’PERSONEL1’ for cluster config
Step: Server configuration initialization succeeded
Step: Synchronizing server logins from companion server
Step: Synchronizing remoteserver from companion server
Step: Synchronizing roles from companion server
Step: Synchronizing server-wide privs from companion server
Step: User information syncup succeeded
Step: Server configured in normal companion mode
```

If there are user databases during the **sp_companion** configuration, you see messages similar to these:

```
Step: Created proxy database ‘pubs2’
 Step: Proxy status for database has been set. Please Checkpoint the database
'pubs2'
 Step: Server configured in normal companion mode”
 Starting companion watch thread
```

Before you configure the companions for symmetric configuration, you must first configure them for asymmetric configuration.

See “Asymmetric Companion Configuration” on page 22 for more information about asymmetric configuration.

Configure for Symmetric Configuration

After you configure your companions for asymmetric failover, you can configure them for symmetric configuration. In a symmetric configuration, both servers act as primary and secondary companions. See Figure 3-2 on page 24 for a description of symmetric configuration.

Issue **sp_companion** from the secondary companion to configure it for symmetric configuration. Use the same syntax as for asymmetric configuration. See "Configure for Asymmetric Configuration," above, for a description of the syntax for **sp_companion**.

The following example adds an Adaptive Server named MONEY1 as the secondary companion to the Adaptive Server named PERSONEL1 described in "Configure for Asymmetric Configuration" on page 129:

```
sp_companion 'MONEY1', configure, with_proxydb, null, sa, MyPassword
Server 'MONEY1' is alive and cluster configured.
Step: Access verified from Server:'MONEY1' to Server:'PERSONEL1'
Server 'PERSONEL1' is alive and cluster configured.
Step: Access verified from Server:'PERSONEL1' to Server:'MONEY1'
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
.....
Step: Companion servers configuration check succeeded
Step: Server handshake succeeded
Step: Master device accessible from companion
Step: Added the servers 'MONEY1' and 'PERSONEL1' for cluster config
Step: Server configuration initialization succeeded
Step: Synchronizing server logins from companion server
Step: Synchronizing remoteserver from companion server
Step: Synchronizing roles from companion server
Step: Synchronizing server-wide privs from companion server
Step: User information syncup succeeded
Step: Server configured in normal companion mode
```

Bring Up Primary Companion as a Monitored Resource

Follow the instructions in this section to start the primary companion as a service that is monitored by the high availability subsystem.

Note Before you begin monitoring the primary companion, make sure you no longer need to shut down the primary server for maintenance or other purposes. Once you begin monitoring the primary companion, you must move it to suspended mode to bring it down. If you are unsure, start the primary server using the startserver script in *$SYBASE/ASE-12_0/install*, finish configuring the companion, then reboot the server using the steps described here.

To start the primary companion as a resource that is monitoring for failover:

1 Restart the TruCluster services by executing this command as “root” at the command line:

```
asemgr -r server_name
```

This automatically executes the *RUNHA_<server_name>.sh* monitor script, which brings up the primary companion and monitors it during crash or hang situations.

Repeat this process on the secondary node to bring up the secondary companion.

Administering Sybase's Failover

This section describes information about:

- Failing back to the primary companion
- Resuming normal companion mode
- Suspending normal companion mode
- Troubleshooting information about common problems with Sybase Failover.

Failing Back to the Primary Node

The monitor scripts executes**sp_companion...prepare_failback** when you are failing back the TruCluster service to the primary node.

To fail back to the primary node, make sure that the secondary companion is in secondary failover mode, and relocates the TruCluster services on the primary node using:

```
asemgr -m server_name host_name
```

Where *server_name* is the name of the primary companion, and *host_name* is the name of the host on which it is running.

To make sure that **sp_companion 'prepare_failback'** was executed successfully, search for this string in *$SYBASE/$SYBASE_ASE/install/server_name.ha_log*:

```
SYBASE HA MONITOR: Prepare_failback was successful.
```

Note Before you relocate the TruCluster service on the primary node, make sure that the secondary node is up and the secondary companion is running in secondary failover mode. If the secondary companion or secondary node is not up and running, do not relocate the TruCluster service. If both nodes are down, restart the secondary node before restarting the primary node.

Manually Failing Back

Note If the automatic failback failed, examine the logs to make sure that the high availability system performed the following steps. If it did not, you can perform them manually. You must perform them in the sequence described below.

1 Relocate the primary node's TruCluster service to the secondary node. This shuts down the primary companion and fails over its resources to the secondary companion.

2 Shutdown and then restart your secondary companion.The *RUNHA_servername.sh* restarts the companion automatically after you shut it down if RETRY is set to a value greater than 0.

3 Log in as LOGIN_NAME to the secondary companion through isql and make sure that it is running in secondary failover mode.

4 Issue **sp_companion 'prepare_failback'**. For example, to fail back from the secondary companion PERSONEL1:

```
sp_companion MONEY1, 'prepare_failback'
```

5 Relocate the TruCluster service back to the primary node.

6 Log in to the primary companion using isql and make sure that it is running in primary failback mode.

7 Issue sp_companion 'resume'. For example, to resume companion mode for primary companion MONEY1:

```
sp_companion PERSONEL1, 'resume'
```

Note You cannot connect clients with the failover property (for example **isql -Q**) until you issue **sp_companion resume**. If you do try to reconnect them after issuing **sp_companion prepare_failback**, the client hangs until you issue **sp_companion resume**.

Suspending Companion Mode

If you must shut down the primary companion for maintenance but do not want to fail over to the secondary companion, you must temporarily suspend companion mode. When the companion mode is suspended, synchronization between the companions does not occur, and the primary companion cannot fail over to the secondary companion. However, suspended mode is very useful for performing such maintenance tasks as changing configuration parameters:

1 To move to suspended mode, issue:

```
sp_companion <primary_server_name>, suspend
```

For example, to suspend primary companion MONEY1:

```
sp_companion MONEY1, suspend
```

2 Kill the monitoring process so it does not trigger a failover when the companion server goes down. As "root," enter:

```
ps -ef|grep "RUNHA_<server_name>.sh monitor"
kill -9 <pid>
```

For example, to kill the monitoring process for MONEY1 which has a *pid* of 2509:

```
ps -ef|grep "RUNHA_MONEY1.sh monitor"
kill -9 2509
```

3 Shut down the primary companion.

After killing the monitoring process, you can bring the companion server down as many times as necessary and it will not failover.

Restarting Shutdown Companion During Suspended Mode

Use the start up script in *$SYBASE/ASE-12_0/install* to restart the primary companion without it being monitored:

```
startserver -f ./RUN_<server_name>
```

For example, to start the MONEY1 companion:

```
startserver -f ./RUN_MONEY1
```

If you use this script to start a companion server, it will not fail over when the server goes down, even if it is configured to do so. Use this method only if you are doing maintenance, and you are certain that you do not want the server databases to be accessible when the server is down.

Resuming Normal Companion Mode

The steps for resuming normal companion mode are slightly different depending on whether you are moving from suspended mode or from failover mode.

Resuming Normal Companion Mode from Suspended Mode

To resume normal companion mode between two companions that have been moved to suspended mode:

1 Shut down the primary companion if it is not already.

2 Issue the following to restart the TruCluster service on the primary node:

```
asemgr -r server_name
```

Resuming Normal Companion Mode

To resume normal companion mode between two companions that are in failover mode, simply restart the TruCluster services on the primary node, and perform the following:

1 Check that both companions are in failback mode by issuing sp_companion with no parameters.

2 Resume normal companion mode by issuing:

```
sp_companion secondary_server_name, resume
```

For example to issue normal companion mode for primary companion PERSONEL1:

```
sp_companion PERSONEL1, resume
Server 'MONEY1' is alive and cluster configured.
Step: Access verified from Server:'MONEY1' to Server:'PERSONEL1'
```

```
Server 'PERSONEL1' is alive and cluster configured.
Step: Access verified from Server:'PERSONEL1' to Server:'MONEY1'
Step: Checkin to See if the remote server is up
Step: Synchronizing server logins from companion server
Step: Synchronizing remoteserver from companion server
Step: Synchronizing roles from companion server
Step: Synchronizing server-wide privs from companion server
Step: User information syncup succeeded
sys_id ses_id      ses_id2     ses_status Purged from s1.
------ ----------- ----------- ---------- ------------------------------

(0 rows affected)
sys_id ses_id      ses_id2     ses_status Copied to s1.
------ ----------- ----------- ---------- ------------------------------

(0 rows affected)
sys_id ses_id      ses_id2     ses_status Purged from s2.
------ ----------- ----------- ---------- ------------------------------
(0 rows affected)
Step: Syssession information syncup succeeded
```

Dropping Companion Mode

To drop companion mode, issue:

```
sp_companion companion_name, "drop"
```

Dropping companion mode is an irreversible process; you must reconfigure the Adaptive Servers companion servers before you they will failover in a high availability system and retain all the functionality that Sybase's Failover provides. However, the nodes upon which the Adaptive Servers are running are still a monitored by the high availability subsystem.

If you drop the companion mode while the *RUNHA_<servername>.sh* script is running, the script continues to monitor the server for any down or hung instances. If you plan to shutdown the server and do not want the node to failover, you need to kill the monitor process by issuing:

```
kill -9 ‘ps -ef | grep "RUNHA_<servername>.sh monitor" | grep -v grep | awk
’{print $2}’‘
```

If you do not kill the monitor process, it triggers a failover of the resources when it detects that the companion has gone down, and tries to restart the companion from either the primary or secondary node, depending on your settings for RETRY and BASIC_FAILOVER.

Troubleshooting Failover on TruCluster for Digital Unix

This section includes troubleshooting information about common errors.

Error Message 18750

If a companion server issues error message 18750, check the *@@cmpstate* of your servers. If your primary companion is in normal companion mode, but the secondary companion is in secondary failover mode, your cluster is in an inconsistent state, and you need to manually recover from this. This inconsistent state may be caused by an **sp_companion 'prepare_failback'** command failing on the secondary companion. You can determine whether this happened by examining the TruCluster log on the secondary node, (located in *$SYBASE/SYBASE_ASE/install/server_name.ha_log*). To recover from this, perform the following steps manually:

1 Shut down both the primary and the secondary companions.

2 Reboot the secondary companion.

3 Repair all databases marked "suspect." To determine which databases are suspect, issue:

```
select name, status from sysdatabases
```

Databases marked suspect have a *status* value of 320.

4 Allow updates to system tables:

```
sp_configure "allow updates", 1
```

5 For each suspect failed over database, perform the following:

```
1> update sysdatabase set status=status-256 where name='database_name'
2> go
1> dbcc traceon(3604)
2> go
1> dbcc dbrecover(database_name)
2> go
```

6 From the secondary companion, issue:

```
sp_companion primary_companion_name, prepare_failback
```

For example, from primary companion MONEY1:

```
sp_companion MONEY1, prepare_failback
```

Make sure that this command executes successfully.

7 Restart the TruCluster service on the primary node.

Recovering from a Failed *prepare_failback*

During a failback, if **prepare_failback** was executed successfully on the secondary companion but the primary companion fails to boot, perform the following to rollback and then reissue the **prepare_failback** command:

1 Check the primary companion's errorlog and the TruCluster error log to find the reason the server failed to boot, and correct the problems.

2 Stop the TruCluster service on the primary node with **asemgr -x *server_name***

3 Login to the secondary companion as LOGIN_NAME, and issue:

```
dbcc ha_admin ("", "rollback_failback")
dbcc ha_admin ("", "rollback_failover")
```

Your companion servers should both be back in the failover mode

4 Restart the TruCluster service on the primary node.

Location of Failover Logs

Sybase's Failover includes the following logs. These logs are helpful for investigating and diagnosing errors encountered during TruCluster failover:

- *$SYBASE/SYBASE_ASE/install/server_name.ha_log* – contains output of the TruCluster activities, as well as the output from the *RUNHA_server_name.sh* monitoring script. For general TruCluster failure, search for the string "ERROR". For output of the *RUNHA_server_name.sh* script, search for "SYBASE HA MONITOR".

  After determining the reason for the failure, correct it, and restart the TruCluster service.

- $PRIM_CONSOLE_LOG – the location of this log is defined in the *RUNHA_server_name.sh* monitoring script. This error log includes the Adaptive Server information from the last execution of the *RUNHA_server_name.sh* script.

CHAPTER 10

Configuring Adaptive Server for Failover on Sun

This chapter lists the steps necessary to configure Adaptive Server for Failover on Sun. It includes the following sections:

Configure Hardware and Operating System for High Availability

Sybase high availability requires the following hardware and system components:

- Two homogenous, networked systems with similar configurations in terms of resources like CPU, memory, etc.
- These systems should be equipped with the high availability subsystem package and the associated hardware.
- Must have devices that are accessible to both nodes.
- The system must have a logical volume manager (LVM) to maintain unique device pathnames across the cluster nodes.
- Configure both public and private networks on both the nodes.
- Create volumes and volume groups on the multihost disks.
- Create one logical host on both the primary and secondary hosts.
- Register one or more volume groups on each logical host.
- Use third-party vendor mirroring rather than Sybase mirroring for media failure protection.
- For more information about commands for running Sun Cluster, see the Sun Cluster documentation.

See your hardware and operating system documentation for information about installing platform specific high availability software.

Prepare Adaptive Server to Work with the High Availability Subsystem

Perform the tasks in this section to prepare Adaptive Server for a high availability configuration.

Install Adaptive Servers

Install both the primary and the secondary servers. They must be installed in the same location on each node. The primary companion can be either a newly installed Adaptive Server, or it can be upgraded from a previous version of Adaptive Server with existing databases, users, and so on. The secondary companion must be a newly installed Adaptive Server and cannot have any user logins or user databases. This is to make sure that all user logins and database names are unique within the cluster.After configuration for failover is complete, you can add user logins and databases to the secondary companion.

If you are installing on the local disk, make sure any databases are created on the multihost disk.

See your the installation documentation for your platform for information about installing and configuring Adaptive Server.

Add Entries for Both Adaptive Servers to the Interfaces File

The interfaces file for both primary and secondary companion must include entries for both companions. For example, the interfaces file for the servers used in the setups described in this manual would have entries for both MONEY1 and PERSONEL1. The server entry in the interfaces file must use the same network name that is specified in sysservers. For information about adding entries to the interfaces file, see the installation documentation for your platform.

Add Entries to *interfaces* File for Client Connections During Failover

To enable clients to reconnect to the failed over companion, you must add an additional line to the interfaces file. By default, clients connect to the port listed in the query line of the server entry. If that port is not available (because that server has failed over), the client connects to the server listed in the *hafailover* line of the server entry. Here is a sample interfaces file for a primary companion named MONEY1 and a secondary companion named PERSONEL1:

```
MONEY1
   master tli tcp /dev/tcp \x000224b782f650950000000000000000
   query tli tcp /dev/tcp \x000224b782f650950000000000000000
   hafailover PERSONEL1
```

Use **dsedit** to add entries to the interfaces file. If the interfaces entries already exist, you must modify them to work for Failover.

See the the Utility Programs manual for your platform for information about **dsedit**.

Make the Value of *$SYBASE* the Same for Both Companions

If *$SYBASE* is installed on the local disk, then *$SYBASE* on both companions must point to the same directory path name. This is not necessary if *$SYBASE* is installed on the shared disk. You can accomplish this by either:

- Making sure that the *$SYBASE* release directory on each companion is created in the same directory.
- If the companions have the *$SYBASE* release directory in different locations, create a directory with the same path on both companions that acts as a symbolic link to the actual *$SYBASE* release directory.

  For example, even though primary companion MONEY1 has a release directory of */usr/u/sybase1* and PERSONEL1 has uses */usr/u/sybase2* as its release directory, their *$SYBASE* must point to the same path.

Both MONEY1 and PERSONEL1 have */SYBASE*, which they establish as a symbolic link to their respective *$SYBASE* release directories. On MONEY1, */SYBASE* is a link to */usr/u/sybase1*, and on PERSONEL1, */SYBASE* is a li[nk to */use/u/sybase2*.

Note The remote monitors for either companion do not function if you do not make the value of $SYBASE the same for both companions.

The sybha Executable

The sybha executable provides the ability for the Adaptive Server High Availability Basis Services library to interact with each platform's high availability cluster subsystem. The Adaptive Server High Availability Basis Services library calls sybha. sybha is located in the $SYBASE/ASE-12_0/bin directory. Before sybha can run, you must change its ownership and permissions. You must also edit a file named *sybhauser* in the *$SYBASE/ASE-12_0/install* directory. This file contains a list of the users who have System Administrator privileges on the cluster. Sybase strongly recommends that severely limit the number of users who have System Administrator privileges on the cluster.

As root, perform the following:

1 Change directory to *$SYBASE/ASE-12_0/bin* directory:

```
cd $SYBASE/ASE-12_0/bin
```

2 Change the ownership of the **sybha** to root:

```
chown root sybha
```

3 Modify the file permissions for **sybha** to 4755

```
chmod 4755 sybha
```

4 Change directory to *$SYBASE/ASE-12_0/install* directory.

```
cd $SYBASE/ASE-12_0/install
```

5 Add the users to the *sybhauser* file who need to administer the high availability subsystem. These logins must be in the format of UNIX login IDs, not Adaptive Server logins. For example:

```
sybase
coffeecup
spooner
```

```
venting
howe
```

6 Change the permissions of sybhauser to root:

```
chown root sybhauser
```

7 Modify the file permissions for *sybhauser* so it can only be modified by root:

```
chmod 644 sybhauser
```

Create New Default Device Other Than Master

By default, *master* is the default device in a newly installed Adaptive Server. This means that, if you create any databases (including the proxy databases used by failover), they are automatically created on the master device. However, adding user databases to master makes it more difficult to restore the master device from a system failure. To make sure that the master device contains as few extraneous user databases as possible, create a new device using **disk init**. Use **sp_diskdefault** to specify the new device as the default before you configure Adaptive Server as a companion for failover.

For example, to add a new default device named *money_default_1* to the MONEY1 Adaptive Server, enter:

```
sp_diskdefault money1_default1, defaulton
```

The master device continues to also be a default device until you specifically issue the following to suspend it as the default device:

```
sp_diskdefault master, defaultoff
```

See the *Adaptive Server Reference Manual* for more information about **disk init** and **sp_diskdefault**.

Add the Local Server to *sysservers*

Using **sp_addserver**, add the local server as the local server in *sysservers* using the network name specified in the interfaces file. For example, if the companion MONEY1 uses the network name of MONEY1 in the interfaces file:

```
sp_addserver MONEY1, local, MONEY1
```

You must reboot Adaptive Server for this change to take effect.

Add Secondary Companion to *sysservers*

Add the secondary companion as a remote server in *sysservers*:

```
sp_addserver server_name
```

By default, Adaptive Server adds the server with an *srvid* of 1000. You do not need to reboot Adaptive Server for the change to take effect.

Run *installhasvss* to Install HA Stored Procedures

Note You must perform the tasks described in Add Entries for Both Adaptive Servers to the Interfaces File, above, before running *installhasvss*. If you run *installhasvss* before performing these tasks you will have to re-run *installmaster* to re-install all the system stored procedures.

The *installhasvss* script performs the following tasks to configure Adaptive Server for failover:

- Installs the stored procedures required for failover (for example, **sp_companion**).
- Installs the SYB_HACMP server in *sysservers.*

You must have System Administrator privileges to run the *installhasvss* script.

installhasvss in located in the *$SYBASE/ASE-12_0/scripts* directory. To execute the *installhasvss* script, enter:

```
$SYBASE/OCS-12_0/bin/isql -Usa -Ppassword -Sservername
< ../scripts/installhasvss
```

installhasvss prints messages as it creates stored procedures and creates the SYB_HACMP server.

Assign *ha_role* to SA

You must have the **ha_role** on both Adaptive Servers to run **sp_companion**. To assign the **ha_role**, issue the following from **isql**:

```
sp_role "grant", ha_role, sa
```

You must log out and then log back in to the Adaptive Server for the change to take effect.

Verify Configuration Parameters

You must enable the following configuration parameters before you configure Adaptive Server for failover:

- **enable CIS** – Enables Component Integration Services (CIS). This configuration parameter is enabled by default.
- **enable xact coordination** – Enables Distributed Transaction Management (DTM). This configuration parameter is enabled by default.
- **enable HA** – Enables Adaptive Server to function as a companion in a high availability system. **enable HA** is off by default. This configuration is static, so you must reboot Adaptive Server for it to take effect. This parameter causes a message to be written to your errorlog stating that you have started the Adaptive Server in a high availability system.

See the *System Administration Guide* for information about enabling configuration parameters.

Add Thresholds to the Master Log

If you have not already done so, you must add a threshold to the master log.

1. Define and execute **sp_thresholdaction** on the master database's log to set a threshold on the number of pages left before a dump transaction occurs. Sybase does not supply **sp_thresholdaction**. See the Adaptive Server Reference Manual for information about creating this system procedure.
2. Place thresholds on the master log segment so it does not fill up:

```
sp_addthreshold "master", "logsegment", 250, sp_thresholdaction
```

3. You must reboot the primary companion for this static parameter to take effect.

Configuring the Sun Cluster Subsystem for Sybase's Failover

See the Sun Cluster high availability subsystem manuals for information about installing the high availability subsystem.

This section assumes that the high availability subsystem is already installed.

Note The *$SYBASE/ASE-12_0/install* directories for both companions must include RUNSERVER files for both companions after installing Adaptive Server on the local disks.

---

inst_ha_script sets up the environment for Sybase Failover to run with the Sun Cluster high availability subsystem. *inst_ha_scripts* is located in *$SYBASE/%SYBASE_ASE/install*. Before you run this script, you must edit it so that:

- The *$SYBASE* environment variable points to the correct directory.
- SC_DIR, SYB_DIR, BIN_DIR, and SCSYB_DIR variables are set correctly for your site.

After you have modified *inst_ha_scripts* for your site, as root, run it to:

1 Copy the following scripts to */opt/SUNWcluster/ha/sybase:*

   - *hasybase_fmon*
   - *hasybase_fmon_start*
   - *sybase_ccd_toggles*
   - *sybase_db_restart*
   - *sybase_db_shutdown*
   - *sybase_fm_check*
   - *sybase_fm_init*
   - *sybase_fm_start*
   - *sybase_fm_stop*
   - *sybase_get_lh*
   - *sybase_get_version*
   - *sybase_shutdown*

- *sybase_status*
- *sybase_status_svcs*
- *sybase_svc_abort*
- *sybase_svc_abort_net*
- *sybase_svc_start*
- *sybase_svc_start_net*
- *sybase_svc_stop*
- *sybase_svc_stop_net*

2 Copy the following scripts to */opt/SUNWcluster/bin*

- *hasybase*
- *dbms_utilities*

3 Change the permissions for the files listed in steps 1 and 2 so the owner and group is *bin*, and have their permissions set to 755. For example, to change the permissions for *sybase_svc_stop*, move to */opt/SUNWcluster/bin* and issue:

```
chmod 755 sybase_svc_stop
chown bin sybase_svc_stop
chgrp bin sybase_svc_stop
```

4 Copy the following scripts to */etc/opt/SUNWscsyb*

- *hasybase_support*
- *hasybase_config_V1*

5 Change the permissions for all these files so the owner is *root* and group is *sys*, and have their permissions set to 444. For example, to change the permissions for *hasybase_support*, move to */opt/SUNWcluster/bin* and issue:

```
chmod 444 hasybase_support
chown root hasybase_support
chgrp sys hasybase_support
```

Note This ends the tasks *inst_ha_scripts* performs. You must manually perform the rest of the steps in this section.

1 Create a file named *sybtab* in the */var/opt/sybase* directories for both nodes. This file must be identical on both nodes. Edit *sybtab* to contain:

- The name and release directory location of the primary and secondary companion
- The name and release directories of Backup Server for the primary and secondary companion
- The name of *$SYBASE_ASE* and *$SYBASE_OCS* directories

Use the following syntax for each entry:

```
server_name:$SYBASE path
```

Where *server_name* is the name of the Adaptive Server or Backup Server.

For example, the *sybtab* file for MONEY1 and PERSONEL1 would look similar to:

```
MONEY1:/SYBASE120
MONEY1_back:/SYBASE120
PERSONEL1:/SYBASE120
PERSONEL1_back:/SYBASE120
SYBASE_ASE:ASE-12_0
SYBASE_OCS:OCS-12_0
```

2 Run the following command to make sure the logical hosts are running on both nodes:

```
haget -f mastered
```

haget returns the name of the logical host it is mastering. For example, if this command is run on FIN1, it returns:

```
loghost-MONEY1
```

3 If you have installed the *$SYBASE* directory on a multihost disk, create the setup files for the fault monitor. Copy the following directories (with their subdirectories) and files from *$SYBASE* to */var/opt/sybase*:

- *ctlib.loc*
- *interfaces*
- *charsets/iso_1/*
- *locales/locales.dat*
- *locales/us_english/*

The ctlib.loc file appears in */var/opt/sybase* and in */var/opt/sybase/locales/us_english/iso_1*.

4 Register the Sybase service using the **hareg** command. Run **hareg** on only one node of the cluster. As “root,” enter:

```
hareg -s -r sybase -h loghost-primary_companion,loghost-
secondary_companion
```

Where loghost-*primary_companion* and loghost-*secondary_companion* are the two logical hosts defined on the primary and secondary nodes. For example, to register the Sybase service for primary companion MONEY1 and secondary companion PERSONEL1:

```
hareg -s -r sybase -h loghost-MONEY1,loghost-PERSONEL1
```

For more information about creating logical hosts and the **hareg** command, see your Sun documentation.

5 Check the status of the Sybase service. As "root," issue:

```
hareg
```

hareg should return:

```
sybase  off
```

If the output shows that Sybase service is off, then, still as "root," activate the Sybase service:

```
hareg -y sybase
```

hareg returns:

```
sybase  on
```

6 Register the primary and secondary companions with the logical hosts by issuing the **hasybase** command on either node of the cluster:

```
hasybase insert server_name loghost_name 60 10 120 300 srvlogin/srvpasswd
/$SYBASE/$SYBASE_ASE/install/RUNSERVER_file_name
```

Where:

- *server_name* – is the name of the companion server
- *loghost-loghost_name* – is the name of the logical host on which the companion server is registered.
- 60,10,120,300 – indicate the probe cycle time, connectivity probe cycle count, probe timeout, and restart delay respectively.
- *srvlogin/srvpasswd* – is the login name and password the cluster subsystem uses to monitor and shut down the ASE server.
- *RUNSERVER_file_name* – is the run server file for the companion server.

For example to register primary companion MONEY1 on loghost *loghost-MONEY1*:

```
hasybase insert MONEY1 loghost-MONEY1 60 10 120 300 sa/
/SYBASE120/ASE-12_0/install/RUN_MONEY1
```

To register secondary companion PERSONEL1 on logical *loghost-PERSONEL1*:

```
hasybase insert PERSONEL1 loghost-PERSONEL1 60 10 120 300 sa/
/SYBASE120/ASE-12_0/install/RUN_PERSONEL1
```

See your Sun documentation for more information about the **hasybase** command.

7 Issue the **hasybase** command to start the primary and secondary companions:

```
hasybase start companion_name
```

This will also invoke the monitors for both companion servers.

Where *companion_name* is the name of the companion you want to start monitoring. For example, to begin monitoring MONEY1:

```
hasybase start MONEY1
```

Note **hasybase** starts the companions automatically if they are not already running when you issue the command.

Configure Companion Servers for Failover

Perform the tasks in this section to configure the Adaptive Servers as primary and secondary companions in a high availability system.

Run *sp_companion* with *do_advisory* Option

You must configure the secondary companion with sufficient resources to perform the work of both servers during failover. The secondary companion may have attributes that will prevent a successful cluster operation. For example, if both the primary and secondary companions are configured for 250 user logins, during failover, the secondary companion only has the resources for half the number of potential user logins necessary. Instead, both MONEY1 and PERSONEL1 should be configured for 500 user logins.

The **sp_companion do_advisory** option checks the configuration options on both the primary and the secondary companion to make sure a cluster operation (such as configuring an Adaptive Server as a secondary companion) will be successful. **sp_companion do_advisory** advises you of any configuration options that should be changed.

See Chapter 6, “Running do_advisory” for a complete description of the **sp_companion do_advisory** option.

Configure for Asymmetric Configuration

Use **sp_companion** to configure the primary companion for asymmetric configuration:

```
sp_companion "primary_server_name", configure, with_proxydb, login_name,
password
```

Where:

- *primary_server_name* is the name of the primary Adaptive Server as defined in the interfaces file entry and in *sysservers*.
- The *with_proxydb* indicates that proxy databases are created on the secondary companion for all databases other than system databases. Any subsequent databases that are added also create proxy databases.
- *login_name* is the name of the user performing this cluster operation (they must have the ha_role).

- *password* is the password of the person performing this cluster operation

This example configures an Adaptive Server named PERSONEL1 as a secondary companion:

```
sp_companion "PERSONEL1", configure, with_proxydb, null, sa, Odd2Think
Server 'MONEY1' is alive and cluster configured.
Step: Access verified from Server:'MONEY1' to Server:'PERSONEL1'
Server 'PERSONEL1' is alive and cluster configured.
Step: Access verified from Server:'PERSONEL1' to Server:'MONEY1'
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
.....
Step: Companion servers configuration check succeeded
Step: Server handshake succeeded
Step: Master device accessible from companion
Step: Added the servers 'MONEY1' and 'PERSONEL1' for cluster config
Step: Server configuration initialization succeeded
Step: Synchronizing server logins from companion server
Step: Synchronizing remoteserver from companion server
Step: Synchronizing roles from companion server
Step: Synchronizing server-wide privs from companion server
Step: User information syncup succeeded
Step: Server configured in normal companion mode
```

If user databases already exist while you are using **sp_companion**, you see messages similar to these:

```
Step: Created proxy database 'pubs2'
 Step: Proxy status for database has been set. Please Checkpoint the database
'pubs2'
 Step: Server configured in normal companion mode"
 Starting companion watch thread
```

See "Asymmetric Companion Configuration" on page 22 for more information about asymmetric configuration.

Configure for Symmetric Configuration

After you configure your companions for asymmetric failover, you can configure them for symmetric configuration. In a symmetric configuration, both servers act as primary and secondary companions. See Figure 3-2 on page 24 for a description of symmetric configuration.

Issue **sp_companion** from the secondary companion to configure it for symmetric configuration. Use the same syntax as for asymmetric configuration. See ““Configure for Asymmetric Configuration” on page 154,” above, for a description of the syntax for **sp_companion**.

The following example adds an Adaptive Server named MONEY1 as the secondary companion to the Adaptive Server named PERSONEL1 described in “Configure for Asymmetric Configuration” on page 154:

```
sp_companion 'MONEY1', configure, with_proxydb, null, sa, Think2Odd
Server 'MONEY1' is alive and cluster configured.
Step: Access verified from Server:'MONEY1' to Server:'PERSONEL1'
Server 'PERSONEL1' is alive and cluster configured.
Step: Access verified from Server:'PERSONEL1' to Server:'MONEY1'
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
.....
Step: Companion servers configuration check succeeded
Step: Server handshake succeeded
Step: Master device accessible from companion
Step: Added the servers 'MONEY1' and 'PERSONEL1' for cluster config
Step: Server configuration initialization succeeded
Step: Synchronizing server logins from companion server
Step: Synchronizing remoteserver from companion server
Step: Synchronizing roles from companion server
Step: Synchronizing server-wide privs from companion server
Step: User information syncup succeeded
Step: Server configured in normal companion mode
```

Administrating Sybase's Failover

This section includes information about using Sybase Failover.

Failing Back to the Primary Companion

Note When you register the logical hosts, disable the automatic failback option. Failback should be a planned event.

Failback moves the primary companion's shared disks from the secondary node back to the primary node and starts the primary companion on the primary node.

1 After your primary host is ready to take over the primary companion, issue this command from the secondary companion:

```
sp_companion primary_companion_name, prepare_failback
```

Where *primary_companion_name* is the name of primary companion server.

This command moves the primary companion's logical host to the primary host.

For example, to fail back the primary companion MONEY1, issue this command from the secondary companion PERSONEL1:

```
sp_companion MONEY1, prepare_failback
```

2 Make sure the primary companion's logical host is moved successfully to the primary host by issuing this command:

```
haget -f mastered
```

The output shows the primary host monitoring the logical host of the primary companion.

3 Start the primary companion:

```
hasybase start primary_companion_name
```

For example to start the primary companion MONEY1:

```
hasybase start MONEY1
```

4 To resume normal companion mode, issue the following from the primary companion.

```
sp_companion secondary_companion_name, resume
```

Where *secondary_companion_name* is the name of the secondary companion server. For example, to resume normal companion mode for primary companion MONEY1:

```
sp_companion PERSONEL1, resume
```

Note You cannot connect clients with the failover property (for example **isql -Q**) until you issue **sp_companion resume**. If you do try to reconnect them after issuing **sp_companion prepare_failback**, the client hangs until you issue **sp_companion resume**.

Suspending Normal Companion Mode

Suspended mode temporarily disables the ability of the primary companion to fail over to the secondary companion. To switch from normal companion mode to suspended mode:

1 Stop the high availability subsystem from monitoring the primary and secondary companion as resources. As "root," issue:

```
hasybase stop primary_companion_name
hasybase stop secondary_companion_name
```

For example, to stop monitoring primary and secondary companions MONEY1 and PERSONEL1:

```
hasybase stop MONEY1
hasybase stop PERSONEL1
```

2 Suspend normal companion mode. From the secondary companion, issue:

```
sp_companion companion_name, suspend
```

For example, to suspend primary companion MONEY1 for maintenance, connect to secondary companion PERSONEL1 and issue:

```
sp_companion MONEY1, suspend
```

To put the entire logical host in maintenance mode, refer to Sun Cluster System Administration Guide for details.

Resuming Normal Companion Mode

To move from suspended mode to normal companion mode:

1 Make sure both companions are running.

2 Begin monitoring the primary and secondary companion as resources. Issue the following as root:

```
hasybase start primary_companion_name
hasybase start secondary_companion_name
```

For example, to begin monitoring primary and secondary companions MONEY1 and PERSONEL1:

```
hasybase start MONEY1
hasybase start PERSONEL1
```

3 Resume normal companion mode. From the secondary companion, issue:

```
sp_companion primary_companion_name, resume
```

For example, to resume normal companion mode for primary companion MONEY1:

```
sp_companion MONEY1, resume
```

Note You cannot connect clients with the failover property (for example **isql -Q**) until you issue **sp_companion resume**. If you do try to reconnect them after issuing **sp_companion prepare_failback**, the client hangs until you issue **sp_companion resume**.

Dropping Companion Mode

To drop companion mode, issue:

```
sp_companion companion_name, "drop"
```

Dropping companion mode is an irreversible process; you must reconfigure the Adaptive Servers companion servers before you they will failover in a high availability system and retain all the functionality that Sybase's Failover provides. However, the companion server are still a monitored by the high availability subsystem. To stop the high availability subsystem from monitoring the companions, issue:

```
hasybase stop companion_server_name
```

Troubleshooting Failover for Sun Cluster

This section includes troubleshooting information about common errors.

- When you shut down a companion, it is restarted on the same node instead of failing over the first time. It fails over on the second shutdown. This is an issue with the Sun Monitor.

  As a workaround, set *restart_delay* to a large value (say, 50000) when you issue *hasybase* insert (step 6 on 158) so the companion always fails over within the time specified by the *restart_delay* value if the companion is shut down. To use this workaround, you must start the companion using the **hasybase start** command; you cannot start the companion using the Sybase RUNSERVER file.

- Sybase has not analyzed the *hasybase_config_V1* file for Adaptive Server version 12.0.

- If any of your nodes have a large number of remote NFS mounts, you may see NFS errors, and your response time from this node may be slow when the logical host is deported from this node. Specifically, when you issue **sp_companion...prepare_failback** from the secondary node, and the primary companions logical host is being deported to the primary host, you will see a slow response from the secondary node. This is temporary, and should revert to the normal response time in a few minutes. To avoid this, make sure your secondary host is working with a normal response time before you issue **sp_companion...resume** from the primary host.

- If your cluster includes only two nodes and does not include any quorum disks, and a node in your cluster fails, split-brain partitions occur and failover does not proceed without user intervention. Every 10 seconds, the system displays:

```
*** ISSUE ABORTPARTITION OR CONTINUEPARTITION ***
```

  along with the commands you must issue to either abort or continue. To continue, issue:

```
scadmin continuepartition <localnode> <clustername>
```

  To avoid this situation, make sure you have quorum disks defined on both nodes.

- Error message 18759. If a companion server issues error message 18750, check the *@@cmpstate* of your servers. If your primary companion is in normal companion mode, but the secondary companion is in secondary failover mode, your cluster is in an inconsistent state, and you need to manually recover from this. This inconsistent state may be caused by an **sp_companion 'prepare_failback'** command failing on the secondary companion. To recover from this, perform the following steps manually:

  a Issue the following to stop monitoring both companion servers:

  ```
  hasybase stop companion_name
  ```

  b Shut down both the primary and the secondary companions.

  c As root, issue the following to move the primary logical host back to the secondary node:

  ```
  haswitch secondary_host_name primary_log_host
  ```

  d restart the secondary companion.

  e Repair all databases marked "suspect." To determine which databases are suspect, issue:

  ```
  select name, status from sysdatabases
  ```

  Databases marked suspect have a *status* value of 320.

  f Allow updates to system tables:

  ```
  sp_configure "allow updates", 1
  ```

  g For each suspect failed over database, perform the following:

```
1> update sysdatabase set status=status-256 where
name='database_name'
2> go
1> dbcc traceon(3604)
2> go
1> dbcc dbrecover(database_name)
2> go
```

  h From the secondary companion, issue:

```
sp_companion primary_companion_name, prepare_failback
```

  For example, from primary companion MONEY1:

  ```
  sp_companion MONEY1, prepare_failback
  ```

  Make sure that this command executes successfully.

  i Issue the following to resume monitoring the primary companion:

```
hasybase start primary_companion_name
```

Recovering from a Failed *prepare_failback*

During a failback, if **prepare_failback** was executed successfully on the secondary companion but the primary companion fails to boot, perform the following to rollback and then reissue the **prepare_failback** command:

1 Check the primary companion's errorlog and the HADBMS error log to find the reason the server failed to boot, and correct the problems.

2 Issue the following to stop monitoring the primary companion:

```
hasybase stop primary_companion_name
```

3 As root, issue the following to move the primary logical host back to the secondary node:

```
haswitch seconary_host_name primary_log_host
```

4 Login to the secondary companion and issue:

```
dbcc ha_admin ("", "rollback_failback")
dbcc ha_admin ("", "rollback_failover")
```

Your companion servers should both be back in the failover mode. For more information about **dbcc ha_admin**, see "dbcc Options for High Availability Systems" on page 192.

5 Reissue **sp_companion...prepare_failback** on the secondary companion.

Location of the Logs

Use this information for debugging your high availability subsystem:

- Adaptive Server error log (the location is defined in the RUNSERVER file)
- Messages from HASYBASE layer (located in */var/opt/SUNWscsyb/hadbms.log*
- Console log located in */var/adm/messages*
- *CCD* logs are located in */var/opt/SUNWCluster/ccd/ccd.log*

CHAPTER 11

Configuring Adaptive Server for Failover on Windows NT

This chapter lists the steps necessary to configure Adaptive Server for Failover on Windows NT. It includes the following sections:

Configure Hardware and Operating System for High Availability

Sybase high availability requires the following hardware and system components:

- Windows NT Enterprise Edition (with Service Pack 3, 4, and 5) and Microsoft Cluster Server installed on both nodes, residing on local disk storage with the same path on both nodes (for example, *C: \WINNT* and *C:\WINNT\Cluster* on both nodes).
- A Microsoft certified cluster. See your Microsoft documentation for a description of what constitutes a certified cluster.
- Adaptive Server software installed on both cluster nodes, with the Sybase release directory (*%SYBASE%*) residing on local disk storage on the nodes (rather than shared disk storage).
- Sybase data devices on shared disk drives.
- Both Adaptive Servers have an independent shared disk (or set of shared disks) for their data device storage. This area of shared disk stores all the companion database device files. The other companion cannot use this area of shared disk for any of its data devices.

Prepare Adaptive Server for High Availability Configuration

Perform the tasks in this section to prepare Adaptive Server for a high availability configuration.

Install Adaptive Servers

Install both the primary and secondary Adaptive Servers according to the instructions in the Windows NT installation guide. Do not use the machine name as the Adaptive Server name.

The primary companion can be either a newly installed Adaptive Server, or it can be upgraded from a previous version of Adaptive Server with existing databases, users, and so on.

The secondary companion must be a newly installed Adaptive Server without any user logins or user databases. This ensures that all user logins and database names are unique within the cluster. After configuration for failover is complete, you can add user logins and databases to the secondary companion.

Place all data and log devices (including the master and *sybsystemprocs* devices) on dedicated shared disks, and the corresponding cluster resources must be in a dedicated MSCS group.

If you are installing on the local disk, make sure any databases are created on the shared disk.

See *Installing Adaptive Server and OmniConnect for Windows NT* for more information.

Changing the Domain Administration Account

After you install the Adaptive Servers, they run under an operating system account known as "LocalSystem." For a regular installation of Adaptive Server, this works fine, however, for a clustered operation, the Adaptive Server must be able to communicate over the network to the other cluster node using Windows NT operating system services. Because the Local System account is not allowed to access any NT operating system services related to the network, it cannot communicate with the other node. You must reconfigure both Adaptive Servers to run under a domain administration account.

To configure Adaptive Server to run as a domain administrator:

1 Start the Services application from the Windows NT Control Panel.

2 Select the service corresponding to the Adaptive Server. Its service name uses this syntax:

   Sybase SQLServer _ *server_name*

   For example, Sybase SQLServer_MONEY1

3 Click Startup to display the service's startup properties dialog box.

4 Select the This Account radio button from the Log On As group.

5 Enter a valid domain administration account name (for example, *MYDOMAIN\AdminUser1*). Enter and then confirm this account's password.

6 Click OK to save these changes.

7 Restart the Adaptive Server to use these changes.

Add Entries for Both Adaptive Servers to *sql.ini*

The *sql.ini* file must include entries for both companions. For example, the *sql.ini* file for the cluster described in this manual would have entries for both MONEY1 and PERSONEL1. The server entry in the *sql.ini* file must use the same network name that is specified in *sysservers*. For information about adding entries to *sql.ini*, see*Installing Adaptive Server and OmniConnect for Windows NT*.

Add Entries to sql.ini for Client Connections During Failover

By default, clients connect to the port listed in the query line of the server entry. If that port is not available (because that server has failed over), the client connects to the server listed in the hafailover line of the server entry in sql.ini. Here is a sample *sql.ini* file for a primary companion named MONEY1 and a secondary companion named PERSONEL1:

```
[MONEY1]
query=TCP,FN1,9835
master=TCP,FN1,9835
hafailover=PERSONEL1
[PERSONEL1]
query=TCP,HUM1,7586
master=TCP,HUM1,7586
hafailover=MONEY1
```

Use **dsedit** to add entries to the *sql.ini* file. If *sql.ini* entries already exist, you must modify them to work for Failover.

See *Installing Adaptive Server and OmniConnect for Windows NT* for information about **dsedit**.

Create New Default Device Other Than Master

The master device is the default device in a newly installed Adaptive Server. This means that, if you create any databases (including the proxy databases used by Failover), they are automatically created on the master device. However, adding user databases to master makes it more difficult to restore the master device from a system failure. To make sure that the master device contains as few extraneous user databases as possible, create a new device using **disk init** (make sure this device is on a dedicated shared disk). Use **sp_diskdefault** to specify the new device as the default before you configure Adaptive Server as a companion for failover. For example, to add a new default device named *money_default_1* to the MONEY1 Adaptive Server, enter:

```
sp_diskdefault money1_default1, defaulton
```

The master device continues to also be a default device until you suspend it as the default device:

```
sp_diskdefault master, defaultoff
```

See the *Adaptive Server Reference Manual* for more information about **disk init** and **sp_diskdefault**.

Add Primary Companion as a Local Server

Using **sp_addserver**, list the local server as the local server in *sysservers* using the network name specified in the *sql.ini* file. For example, if the companion MONEY1 uses the network name of MONEY1 in the *sql.ini* file:

```
sp_addserver MONEY1, local, MONEY1
```

You must reboot Adaptive Server for this change to take effect.

Add Secondary Companion to sysservers

Add the secondary companion as a remote server in *sysservers*:

```
sp_addserver server_name
```

By default, Adaptive Server adds the server with a *srvid* of 1000. You do not need to reboot Adaptive Server for the change to take effect.

Run *insthasv* to Install HA Stored Procedures

Run the *insthasv* script on both Adaptive Servers. The *insthasv* script:

- Installs the stored procedures required for Failover (for example, **sp_companion**).
- Installs the SYB_HACMP server in *sysservers*.

  You must have System Administrator privileges to run the *insthasv* script.

  insthasv is located in the *%SYBASE%\ASE-12_0\scripts* directory. To execute *insthasv*, enter:

```
%SYBASE%\OCS-12_0\bin\isql -Usa -Ppassword -Sservername < %SYBASE\ASE-
12_0\scripts\insthasv
```

  insthasv prints messages as it creates stored procedures and creates the SYB_HACMP server.

Assign *ha_role* to SA

You must have the **ha_role** on both Adaptive Servers to run **sp_companion**. To assign the **ha_role**, issue the following from **isql**:

```
sp_role "grant", ha_role, user_name
```

You must log out and then log back in to the Adaptive Server for the change to take effect.

Verify Configuration Parameters

You must enable the following configuration parameters before you configure Adaptive Server for failover:

- **enable CIS** – enables Component Integration Services (CIS). This configuration parameter is enabled by default.
- **enable xact coordination** – enables Distributed Transaction Management (DTM). This configuration parameter is enabled by default.

- **enable HA** – enables Adaptive Server to function as a companion in a high availability system. **enable HA** is off by default. This configuration is static, so you must reboot Adaptive Server for it to take effect. This parameter causes a message to be written to your error log stating that you have started the Adaptive Server in a high availability system.

  See the *System Administration Guide* for information about enabling configuration parameters.

Run *sp_companion* with *do_advisory* Option

You must configure the secondary companion with sufficient resources to perform the work of both servers during failover. For example, if both the primary and secondary companions are configured for 250 user logins, during failover, the secondary companion has only half the number of potential user logins necessary. MONEY1 and PERSONEL1 should both be configured for 500 user logins.

The **sp_companion do_advisory** option checks the configuration options on both the primary and the secondary companion to make sure a cluster operation (such as configuring an Adaptive Server as a secondary companion) will be successful. **sp_companion do_advisory** advises you of any configuration options you should change.

Configuring Windows NT for Failover

You can configure Failover on Windows NT either from the command line or using the Cluster Administrator. Configuring from the command line is described below; configuring with the Cluster Administrator is described in "Configure Windows NT for Failover Using the Cluster Administrator" on page 172.

If you are configuring for a symmetric setup, you must first configure the cluster for an asymmetric setup.

Configure for Asymmetric Configuration from the Command Line

To configure the primary companion for asymmetric configuration, enter:

```
sp_companion "primary_server_name", configure, with_proxydb, login_name,
password, cluster_login, cluster_login_password
```

Where:

- *primary_server_name* is the name of the primary Adaptive Server as defined in the *sql.ini* file entry and in *sysservers*.
- The *with_proxydb* indicates that proxy databases are created on the secondary companion for all databases other than system databases. Any subsequent databases that are added also create proxy databases.
- *login_name* is the name of the user performing this cluster operation (they must have both the **ha_role** and **sa_role**).
- *password* is the password of the person performing this cluster operation.
- *cluster_login* The login that the high availability subsystem uses to log in to the companion to control it. This login must exist in the primary before running **sp_companion...configure** and must have **sa_role** and **ha_role**.
- *cluster_login_password* is the user's password for logging in to the cluster.

This example configures an Adaptive Server named PERSONEL1 as a secondary companion:

```
1> sp_companion "PERSONEL1", configure, with_proxydb, sa, MyPassword,
sa_cluster_login, MyClusterPassword
2> go
Server 'MONEY1' is alive and cluster configured.
Step: Access verified from Server:'MONEY1' to Server:'PERSONEL1'
Server 'PERSONEL1' is alive and cluster configured.
```

```
Step: Access verified from Server:'PERSONEL1' to Server:'MONEY1'
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
.....
Step: Companion servers configuration check succeeded
Step: Server handshake succeeded
Step: Master device accessible from companion
Step: Added the servers 'MONEY1' and 'PERSONEL1' for cluster config
Step: Server configuration initialization succeeded
Step: Synchronizing server logins from companion server
Step: Synchronizing remoteserver from companion server
Step: Synchronizing roles from companion server
Step: Synchronizing server-wide privs from companion server
Step: User information syncup succeeded
Step: Server configured in normal companion mode
```

If user databases already exist when run **sp_companion**, you see these messages:

```
Step: Created proxy database 'pubs2'
Step: Proxy status for database has been set. Please Checkpoint the database
'pubs2'
Step: Server configured in normal companion mode"
Starting companion watch thread
```

Before you configure the companions for symmetric configuration, you must first configure them for asymmetric configuration.

See "Asymmetric Companion Configuration" on page 22 for more information about asymmetric configuration.

Configure for Symmetric Setup from the Command Line

You must configure your companions for an asymmetric setup before you can configure them for a symmetric setup. In a symmetric configuration, both servers act as primary and secondary companions. See "Symmetric Companion Configuration" on page 24 for a description of symmetric configuration.

Issue **sp_companion** from the secondary companion to configure it for symmetric configuration. Use the same syntax as for asymmetric configuration.

The following example adds an Adaptive Server named MONEY1 as the secondary companion to the Adaptive Server named PERSONEL1 described in “Configure for Asymmetric Configuration from the Command Line” on page 170:

```
1> sp_companion 'MONEY1', configure, with_proxydb, sa, MyPassword,
sa_cluster_login, MyClusterPassword
2> go
Server 'MONEY1' is alive and cluster configured.
Step: Access verified from Server:'MONEY1' to Server:'PERSONEL1'
Server 'PERSONEL1' is alive and cluster configured.
Step: Access verified from Server:'PERSONEL1' to Server:'MONEY1'
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
.....
Step: Companion servers configuration check succeeded
Step: Server handshake succeeded
Step: Master device accessible from companion
Step: Added the servers 'MONEY1' and 'PERSONEL1' for cluster config
Step: Server configuration initialization succeeded
Step: Synchronizing server logins from companion server
Step: Synchronizing remoteserver from companion server
Step: Synchronizing roles from companion server
Step: Synchronizing server-wide privs from companion server
Step: User information syncup succeeded
Step: Server configured in normal companion mode
```

Configure Windows NT for Failover Using the Cluster Administrator

The Cluster Administrator utility is a graphical user interface that walks you through the configuration process. These section assumes that the Microsoft Cluster Server is installed on your system.

1. Create a cluster group. See you Microsoft Cluster Server documentation for information.
2. Move the dedicated shared disks for the companion you are configuring into the cluster group you created in step 1. See you Microsoft Cluster Server documentation for information.
3. Select | Administrative Tools | Cluster Administrator.

4 Select File | Resource | New Resource.

5 On the New Resource screen enter:

- Name – the name of the package you are configuring.
- Description – a brief description of the package. This field is not required.
- Resource type – select Sybase Companion Server
- Group – group in which you want this cluster included. This field is not required

  Click OK to create this group.

6 Select Change Group then select the name of the group to move the physical disk resources (data and log devices) of the primary companion to this new group. Click OK. You see the Possible Owners screen.

7 The Possible Owners screen specifies the nodes on which this resource can be brought online. Both nodes must be listed as possible owners in the right-hand window of this screen. If the list is not correct, use Add or Remove to correct it. Select Next.

8 The Dependencies screen lists the services that must be brought online first before starting this resource. Make sure the shared disk device is listed as a dependency. Select Next.

9 On the ASE Server Information screen enter:

- The name of the Adaptive Server you are configuring as the primary companion.
- The System Administrator login for this companion (this must be “sa”
- The System Administrator password for this login.
- A password check to make sure the password you entered is correct.

  Select Next.

10 Enter the name of the Adaptive Server that is to be the secondary companion in the Companion Server Information field.

   To configure the companions in a symmetric setup, select symmetric.

   Select Next.

11 On the Cluster Parameters screen, select Use System Generated Cluster Login. This provides a system-generated setup log that is used when the cluster logs into the Adaptive Server. Select Next. (Alternatively, you can create the login on the primary companion, assign it both the **sa_role** and **ha_role** before you perform this step.)

12 (Optional) On the Setup Options screen, enter the path to the error log that records the steps made during this configuration (this log is very helpful if you need to call Technical Support). Select Finish.

13 The next screen lists the configuration that you have selected for this cluster configuration.Select Back and re-enter the appropriate data to change any information. When the configuration is correct, select Next to configure this cluster resource.

You see a series of messages as the two Adaptive Servers are configured. If any error messages appear, address the issues and select Next. You do not have to start over again.

When the configuration is complete, the companions are in normal companion mode in either an asymmetric or symmetric setup, depending on what you specified in the Companion Server Information screen.

Configure Microsoft Cluster Server

This section describes the steps for setting the pending timeout and failback properties for the primary companion's cluster resource. If you are configuring a symmetric setup, you must set the properties for both companions.

- When the Microsoft Cluster Server (MSCS) takes the cluster resource for the primary companion on or off line, it allows for a certain amount of time to perform its processing before assuming that the operation will not complete. By default, this amount of time is 180 seconds (3 minutes). This value is known as the "pending timeout," and can be set for each resource in the MSCS cluster.

  For the Sybase Companion Server resource, the pending timeout period must be long enough to boot the Adaptive Server, run recovery on its databases, and possibly execute **sp_companion resume**. For companions that have large databases, it is likely that this processing will take more than 180 seconds, and you should set the pending timeout property to a higher number.

- If you are repairing or restarting the primary node after a failover, MSCS automatically fails back to the primary node as soon as the primary node comes back up unless the MSCS group containing the Sybase Companion Server resource is set to not automatically fail back.

To configure both of these properties:

1 Select Start | Administrative Tools | Cluster Administrator.

2 From the Cluster Administrator window, select Configure an Existing Resource.

3 Select Advanced from the Properties window.

4 Change the Pending Timeout property to a value that is comfortably larger than the longest time the server takes to recover, plus about 2 minutes.

5 Select Failback and make sure the Prevent Failback radio button is selected.

6 Click OK.

Check the MSCS Configuration

Use the Cluster Administrator, to verify the configuration of MSCS is correct:

- There should be a new cluster resource of type “Sybase Companion Server” for each companion that can fail over. In an asymmetric setup, there is one of these resources, for a symmetric setup, there are two of these resources.

  The names of these resources are same as the names of the primary and secondary companions they are managing. For example,.if you created an asymmetric setup where PERSONEL1 is the secondary companion for MONEY1, there should be a new cluster resource called MONEY1.

- The new cluster resources described above should all be in their own group, which is named *companion_name_GRP*, and where *companion_name* is the name of the companion server resources they contain.

- The cluster group described above should contain one for each physical cluster disk upon which the companions data devices reside.

Securing the MSCS Cluster

The Sybase integration software that interfaces MSCS to Adaptive Server requires a login (with **ha_role** and **sa_role**) and password for the Adaptive Server you are configuring as a companion server. This allows the integration software to log into Adaptive Server when it needs to control it for cluster operations.

The login and its password are stored as part of the Windows NT registry Cluster Database (under *HKLM\Cluster*). This information is encrypted to prevent users from obtaining privileged login information by browsing the registry using tools like *REGEDIT.EXE* and *REGEDT32.EXE*. However, as with any reversible encryption, there is a possibility that a user could break the encryption. To address this possibility, Sybase recommends that you protect the appropriate area of the registry using a Discretionary Access Control List (DACL) that allows only administrators access to the information.

Perform the following to encrypt the cluster login and password

1 Run *REGEDT32.EXE*.

2 From the window titled HKEY_LOCAL_MACHINE on Local Machine, Double click on the Cluster folder. A subtree opens containing registry keys.

3 Select the Resources registry key.

4 Select Permissions from the Security menu. A dialog called Registry Key Permissions is displayed.

5 Select Remove from the Registry Key Permissions dialog box to remove all entries displayed except CREATOR OWNER and *machine_name\Administrators*, where *machine_name* is the local machine name. This prevents anyone except administrative users from reading this part of the registry

6 Click OK to commit the changes

Repeat this process on both cluster nodes.

Troubleshooting Sybase Failover on Windows NT

This section includes troubleshooting information about common errors.

Error Message 18750

If a companion server issues error message 18750, check the *@@cmpstate* of your servers. If your primary companion is in normal companion mode, but the secondary companion is in secondary failover mode, your cluster is in an inconsistent state, and you need to manually recover from this. This inconsistent state may be caused by an **sp_companion 'prepare_failback'** command failing on the secondary companion. You can determine whether this happened by examining the log on the secondary node. To recover from this, perform the following steps manually:

1 Reboot the secondary companion.

2 Repair all databases marked "suspect." To determine which databases are suspect, issue:

```
select name, status from sysdatabases
```

Databases marked suspect have a *status* value of 320.

3 Allow updates to system tables:

```
sp_configure "allow updates", 1
```

4 For each suspect, failed-over database, perform the following:

```
1> update sysdatabase set status=status-256 where name='database_name'
2> go
1> dbcc traceon(3604)
2> go
1> dbcc dbrecover(database_name)
2>go
```

5 From the secondary companion, issue:

```
sp_companion primary_companion_name, prepare_failback
```

For example, from secondary companion PERSONEL1:

```
sp_companion MONEY1, prepare_failback
```

Make sure that this command executes successfully.

6 Make sure the primary companion is up and running, a then resume normal companion mode. From the primary companion, issue:

```
sp_companion secondary_companion, resume
```

For example, from the primary companion MONEY1:

```
sp_companion PERSONEL1, resume
```

7 Make sure the Sybase Companion Server resource for the companion relationship is located on the primary node (use **Move Group** to move it if not) and is **Offline**. Then, bring the resource online using the Cluster Administrator.

Recovering from a Failed *prepare_failback*

During a failback, if **prepare_failback** was executed successfully on the secondary companion but the primary companion fails to boot, perform the following to rollback and then reissue the **prepare_failback** command:

1 Check the primary companion's system event log to find the reason the server failed to boot, and correct the problems.

2 Check that the MSCS group that contains the resource for the primary server is located on the secondary node. If not, use **Move Group** to move it there

3 Login to the secondary companion and issue:

```
dbcc ha_admin ("", "rollback_failback")
dbcc ha_admin ("", "rollback_failover")
```

4 Verify secondary companion is in normal companion mode

5 Check that the MSCS resource for the primary server is online. If not, manually bring the resource online using the Cluster Administrator.

6 As root, start up the package for the primary companion to run on secondary node.

```
/usr/sbin/cmrunpkg -n <secondary_node> primary_companion_package_name
```

Your secondary companion is now in failover mode. Once you verify that everything is ready for the primary companion to failback to normal companion mode, you can either issue **sp_companion...prepare_failback** or **Move Group**.

CHAPTER 12

Troubleshooting Second Point of Failures

This chapter discusses common problems that result from secondary point of failures with the high availability subsystem.

Troubleshooting with *dbcc ha_admin*

Sybase's Failover includes **dbcc ha_admin**, which addresses second point of failures. A second point of failure for a high availability system occurs when the primary companion is already in failover mode, and another point in the system fails.

See "dbcc Options for High Availability Systems" on page 192 for information about **dbcc ha_admin** syntax and a complete list of options.

Re-Installing *installmaster* and *installhasvss*

Perform the steps in the following sections to re-install either *installmaster* or *installhasvss*.

Re-Installing *installmaster*

After you install *installmaster* on a companion server, you should only re-run this script if the stored procedures it creates are corrupted, or if you need to install a newer version of *installmaster*. **dbcc ha_admin (' ', state_machine)** temporarily moves the companion to single-server mode so the *installmaster* can safely reinstall or update the stored procedures. Do not run *installmaster* without running **dbcc ha_admin**.

Note Because **dbcc ha_admin** moves the companion to single-server mode, you should only run this command when there is no concurrent activity.

Preform the following to re-install *installmaster*:

1 Run **dbcc ha_admin** to move the local companion server to single-server mode:

```
dbcc ha_admin (' ', 'state_machine', 'halt')
```

Where `' '` is used for a placeholder.

2 Re-run *installmaster*.

3 Run **dbcc ha_admin** to return the companion server to its original mode:

```
dbcc ha_admin (' ', 'state_machine', 'restart')
```

4 You *must* re-install *installhasvss* after you re-install *installmaster*. See "Re-Installing installhasvss" on page 181, for more information.

Re-Installing installhasvss

After you install *installhasvss* on a companion server, you should only re-run this script if the stored procedures it creates are corrupted, or if you need to install a newer version of *installhasvss*. **dbcc ha_admin (' ', state_machine)** temporarily moves the companion to single-server mode so the *installhasvss* can safely reinstall or update the stored procedures. If you attempt to run *installhasvss* without running **dbcc ha_admin**, the companion issues the following error message:

```
Server is not in single-server mode.
Please run dbcc ha_admin (' ', 'state_machine', 'halt') and try again
```

Note Because **dbcc ha_admin** moves the companion to single-server mode, you should only run this command when there is no concurrent activity.

Preform the following to re-install *installhasvss*:

1. Make a note of the *srvnetname* for the SYB_HACMP entry in *sysservers*. When it is configured for Sybase Failover, SYB_HACMP points to the companion server's *svrnetname* (for example, the *srvnetname* for the SYB_HACMP entry on companion server MONEY1 is PERSONEL1). If the local node crashes while you are re-running *installhasvss*, this name is removed from *sysservers*, and you will have to replace it manually.
2. Run **dbcc ha_admin** to move the companion to single-server mode:

   ```
   dbcc ha_admin (' ', 'state_machine', 'halt')
   ```

   Where '' is used for a placeholder.
3. Re-run *installhasvss*. After *installhasvss* finishes, the companion server reverts to its original mode.

If the node crashes after you perform step 2, above, the *srvnetname* of the remote server is removed from *sysservers*. If this occurs, add the name of the remote server to *sysservers* by issuing:

```
sp_addserver SYB_HACMP, null, 'remote_server_svrnetname'
```

Run **dbcc ha_admin** to return the companion server to its original mode:

```
dbcc ha_admin (' ', 'state_machine', 'restart')
```

Using *dbcc ha_admin* to Address Second Point of Failures for Failover and *prepare_failback*

dbcc ha_admin includes the **rollback_failover** and **rollback_failback** options. These **dbcc** options should be only as a last resort, and only by System Administrators who are knowledgeable about the high availability subsystem.

These options allow you to rollback the steps performed by:

- A failover that did not complete because of either a problem with the high availability subsystem (for example, all the disks were not available during the failover, so the companion marks all the databases as suspect) or the secondary companion crashed during the failover.
- A **sp_companion...prepare_failback** the did not complete because of either a problem with the high availability subsystem or the primary companion failed to reboot during the failback steps.

You must preform platform specific steps before you issue either **dbcc ha_admin rollback_failover** or **rollback_failback**. See the configuration chapter for your platform for information.

Error Messages 18805, 18769, 18836

The following are common error messages you may receive:

- Error message 18805 –

```
Warning: Server '%1!' is configured for ASE HA services. The networkname
in its SYB_HACMP entry does not point to the local server. If this is due
to an earlier failed cluster command, refer to the System Administration
Guide
```

  If the local node is running in single-server mode and the srvnetname entry for SYB_HACMP is set correctly, its network name is the same as the local servers network name. This error occurs when SYB_HACMP's network name is set another server's network name. If this occurs because of an earlier failed cluster operation, use **sp_addserver** to set the *srvnetname* of SYB_HACMP to the local servers network name. Note that during normal companion mode, the *svrnetname* for SYB_HACMP *always* points to the remote companion's network name, and should *never* be changed.

- Error message 18769 –

```
The HA cluster is currently in use for other cluster operations. Retry
```

```
the command later. If the problem persists, it may be due to an earlier
failed cluster command; check the System Administration Guide (Error
%1!).
```

All cluster operations receive a cluster-wide lock and then immediately release the lock when they are done. This error occurs when you perform a cluster operation but the previous cluster operation did not release the cluster-wide lock. For information about releasing a cluster-wide lock, see “Cluster Locks in a High Availability Node” on page 18.

- Error message 18836 –

```
Configuration operation '%1!' can not proceed due to Quorum AdvisoryCheck
failure. Please run 'do_advisory' command to find the incompatible
attribute and fix it
```

sp_companion checks a series of quorum attributes to confirm the compatibility between the companion servers. One of your companion servers has attribute settings that are not compatible. Run **do_advisory** for a list of the problem attributes. See Chapter 6, “Running do_advisory” for information.

CHAPTER 13

Changes to Commands, System Procedures, System Databases, and New dbcc Commands, and Functions

This chapter discusses the changes to commands, system procedures, and system databases when Adaptive Server is configured for failover.

Table 13-1: Changes to commands in asymmetric and symmetric mode

Command	Asymmetric setup	Symmetric setup
create role **add role** **drop role** **alter role**	During normal companion mode, any changes made to the primary companion with these commands are synchronized with the secondary companion server. During failover mode, the secondary companion is updated with **create role**, **create role** and **alter role** changes. The primary companion is updated with this information during failback mode. You cannot run **drop role** during failover mode. You cannot run these commands during suspended mode.	These commands have the same behavior in symmetric mode as they have in asymmetric configuration.

Command	Asymmetric setup	Symmetric setup
create database	During normal companion mode, **create database** creates a proxy database on the secondary companion. During failover mode, **create database** is not allowed to run because the primary companion's *model* database is not in failover mode. During failback mode, **create database** is allowed only under special circumstances. You cannot run **create database** during suspended mode.	**create database** has the same behavior in symmetric setup as it has in asymmetric setup.
disk init	During normal companion mode, **disk init** has the same behavior as in symmetric configuration. During failover mode, the secondary server can add devices to its local set by ensuring the unique device name space. During suspended mode, **disk init** cannot run.	During normal companion mode, **disk init** ensures that the secondary companion does not already have a disk with same physical and logical name, and that the secondary companion server can access the device. **disk init** is not allowed to run during failover mode because it cannot verify access to the disk on the primary companion. However, **disk init** is allowed to perform some special duties like log expansion. During suspended mode, **disk init** cannot run.

Command	Asymmetric setup	Symmetric setup
disk mirror **disk remirror** **disk unmirror**	Sybase mirroring is not supported for high availability	Sybase mirroring is not supported for high availability
drop database	During normal companion mode, **drop database** informs the companion server to free the database name space and may request to drop the proxy database. During failover mode, there are no restrictions on the **drop database** command. During suspended mode, you cannot run **drop database**.	This command has the same behavior in symmetric setup as is has in asymmetric setup.
grant **revoke**	During normal companion mode, changes to permissions from these commands are synchronized across the companion servers. During failover mode, there are no restrictions for **grant**. You cannot run **revoke** during failover mode. During suspended mode, you cannot issue either **grant** or **revoke**.	This command has the same behavior in symmetric setup as is has in asymmetric setup.
shutdown **shutdown with nowait**		

Changes to System Procedures in Adaptive Server Configured for Failover

Using proxy databases guarantees unique database names with the cluster, but it does not guarantee unique database IDs. The same database may have a different database ID before and after failover. Because the database IDs may change, system procedures are automatically recompiled after failover to make sure they do not use an incorrect or out-of-date database or object ID from *sysprocedures.*

During failover mode, Adaptive Server performs a domain check to make sure that, if there are system procedures with duplicate names in the two Adaptive Servers, the system procedure in the correct domain is run. This domain check is only performed in failover mode.

System Procedures Hold Table Lock When Modifying System Tables

System procedures cannot acquire table locks on system tables explicitly. However, in a system using Sybase's Failover, system procedures on both companions could attempt to modify the system tables at the same time. To prevent deadlocks, if you issue a system procedure to modify a system table, the system procedure acquires a table lock on the proxy table of the system table it is modifying. That is, if you issue a system procedure to alter the *syslogins* system table on primary companion MONEY1, the system procedure acquires a table lock on the *syslogins* proxy table on the secondary companion, PERSONEL1. The system procedure then modifies the *syslogins* proxy table on PERSONEL1, and the *syslogins* proxy table updates the *syslogins* system table on MONEY1. After the changes are committed, the table locks on the proxy *syslogins* system table are released. Any other system procedures that need to make changes to the same system table are in a queue for that table. After the lock is released, they acquire the table lock.

You can set the amount of time, in seconds, system procedures wait in the queue for the locked proxy system table with the **sp_configure "dtm lock timeout period"** command. For more information, See the **dtm lock timeout period** section of the Distributed Transaction Management (DTM) section of this guide.

Changes to System Procedures in A Failover Configuration

This section describes the system procedures whose behavior changes when Adaptive Server is configured for Failover. After Adaptive Server is configured as a companion server:

- System procedures have no change to their default functionality when they are run in single-server mode.
- You cannot run any of the system procedures listed in Table 13-2 or Table 13-3 during failback mode.
- The first column of Table 13-2 and Table 13-3 “Normal Companion Mode,” describes the behavioral changes for system procedures issued from an asymmetric primary, asymmetric secondary, or symmetric companion.
- The last column of Table 13-2 Table 13-3 “Failover Mode,” describes the behavioral changes for system procedures issued during either asymmetric secondary failover or symmetric failover.

Table 13-2 lists the system procedures that change *server-wide* attributes (for example, the default language or the resource limit):

- During normal companion mode, all the system procedures listed in Table 13-2 must be run from *master*.
- These system procedures cannot be run during asymmetric secondary suspended mode or symmetric suspended mode.
- An X indicates that the system procedure does not run in the listed mode.

Table 13-2: Changes in system procedures that alter server-wide attributes

System procedure	Normal Companion Mode	Asymmetric Primary Suspended Mode	Failover Mode
sp_add_resource_limit			
sp_add_time_range			
sp_addexternlogin			
sp_addlanguage			
sp_addlogin			
sp_addremotelogin			
sp_addserver			
sp_defaultdb			
sp_defaultlanguage			

System procedure	Normal Companion Mode	Asymmetric Primary Suspended Mode	Failover Mode
sp_drop_resource _limit	You must manually run this system procedure on the remote server as well to synchronize the companions	X	X
sp_drop_time_range		X	X
sp_dropexternlogin		X	X
sp_droplanguage		X	X
sp_droplogin		X	X
sp_dropremotelogin		X	X
sp_dropserver		X	X
sp_locklogin			
sp_modify_resource _limit	You must manually run this system procedure on the remote server as well to synchronize the companions		
sp_modify_time_range			
sp_password			
sp_remotelogin			
sp_remoteoption			
sp_serveroption			
sp_setlangalias			

Table 13-3 lists the system procedures that change the attributes of the database in which they are run (such as adding a user, alias, or group to the current database). You cannot run these system procedures from *master* during either secondary suspended or symmetric suspended mode. An X indicates that you cannot run the system procedure in the listed mode.

Table 13-3: System procedures that alter database-wide attributes when they are run in master

System procedure	Normal Companion Mode	Asymmetric Primary Suspended Mode	Failover Mode	Notes
sp_addalias				
sp_addgroup				

System procedure	Normal Companion Mode	Asymmetric Primary Suspended Mode	Failover Mode	Notes
sp_addtype				
sp_adduser				
sp_changedbowner			X	See below for additional restrictions for this system procedure.
sp_changegroup	You must manually run this system procedure on the remote server as well to synchronize the companions			
sp_dropalias		X	X	
sp_dropgroup		X	X	
sp_droptype		X	X	
sp_dropuser		X	X	
sp_renamedb			X	See below for additional restrictions for this system procedure.

sp_changedbowner and **sp_renamedb** cannot run during failover mode, and have the following additional behavior changes:

- **sp_changedbowner** – After you run this procedure on local companion, you must manually run it on the remote server as well to synchronize the companions if the following are true:
  - You are not running this command in *master*.
  - The companion is in suspended or normal companion mode
  - The companion was configured using the **with_proxydb** option.
- **sp_renamedb** – You must first run this system procedure in the primary database and then run it in the proxy database on the remote server, if the following are true:
  - You do not run this command in *master*
  - The companion is in suspended or normal companion mode
  - The companion is configured using the **with_proxydb** option

dbcc Options for High Availability Systems

Sybase Failover includes **dbcc ha_admin**, which addresses second point of failures. Second point of failures a situations in which a cluster operation fails because of problems with the high availability subsystem. For example, if you issue **sp_companion 'prepare_failback'** and the secondary companion crashes. **dbcc ha_admin** provides a method of backing out of the cluster operation. After **dbcc ha_admin** is complete, you can re-issue the cluster operation.

Note **dbcc ha_admin** should only by a System Administrator who is familiar with the high availability subsystem. Issuing this command at the wrong time may only further trouble an already problematic situation.

Table 13-4 includes information about the **dbcc ha_admin** options.

Table 13-4: dbcc ha_admin options

Option Name	Function	Syntax and Comments
rollback_failback	Rolls back the effect of **sp_companion... prepare_failback** and returns the companion to the failover mode. This command works irrespective of the results of **prepare_failback** command.	`dbcc ha_admin (" ", rollback_failback)` Where " " is a required empty placeholder • Can only be used in failback mode. • Any failback threads waiting for the **resume** command are killed with this command. • You may need to perform platform-specific steps to prepare you companions for **rollback_failback** option. See the configuration chapter for your platform for more information. • This command is only issued from the secondary companion.
rollback_failover	Rolls back the effects of failover from the primary companion, and returns it to normal companion mode. **rollback_failover** does not affect the secondary companion.	`dbcc ha_admin (" ", rollback_failover)` Where " " is a required empty placeholder • This command can only be used in failover mode. • You may need to perform platform-specific steps to prepare you companions for **rollback_failover** option. See the configuration chapter for your platform for more information. • **rollback_failover** has no effect on the companion server that failed. The companion server that takes over the failed companion's work load resumes normal companion mode. • This command is only issued from the secondary companion. • This command works even when failover marked the databases "suspect"

Option Name	Function	Syntax and Comments
drop_failoverdb	Only used in failover mode. **drop_failoeverdb** drops the failed-over databases that could not be dropped with the **drop database** command. This command also cleans up the *master_companion* of all the metadata relating to the dropped database	`dbcc ha_admin (" ", drop_failoverdb database_name)` Where " " is a required empty placeholder, and *database_name* is the name of the database you are dropping. • Use as a last resort, when you must drop a database to complete the load of another database.
clusterlock	Releases cluster-wide locks that were left behind by a client performing a cluster operation	`dbcc ha_admin (" ", clusterlock)` For more information about cluster-wide locks and releasing them, see "Cluster Locks in a High Availability Node" on page 18.
state_machine	Moves the companion server to single-server mode.	`dbcc ha_admin (' ', 'state_machine', 'halt')` Where " " is a required empty placeholder. For information about using this option, see "Re-Installing installmaster and installhasvss" on page 180.
session	Invokes clients that are sleeping because of a failed **sp_companion...resume**. Clients that are invoked disconnect from the secondary companion and connect to the primary companion.	`dbcc ha_admin (SYB_HACMP, session, "drop")`

dbcc dbrepair Option for Sybase Failover

Sybase Failover adds the **dropproxydb** option to **dbcc dbrepair**.

Table 13-5: dbcc dbrepair dropproxydb option

Option Name	Function	Syntax and Comments
dropppoxydb	Drops proxy databases.	`dbcc dbrepair(`*`database_name`*`, dropproxydb)` Where *database_name* is the name of the database whose proxy database you are dropping.

CHAPTER 14

Open Client Functionality in a Failover Configuration

This chapter discusses the changes required for Open Client to work with Sybase's Failover.

CTLIB Application Changes

Note An application installed in a cluster must be able to run on both the primary and secondary companions. That is, if you install an application that requires a parallel configuration, the secondary companion must also be configured for parallel processing so it can run the application during failover.

You must modify all of your applications that are written with CTLIB API calls before they can work with Sybase's failover software. The following steps describe the modifications:

1 Set the CS_HAFAILOVER property using the **ct_config** and **ct_con_props** CTLIB API calls. You can set this property at either the context or the connection level. This property is set using the following syntax:

```
ct_config(context, action, CS_HAFAILOVER, buf, buflen, outlen)
ct_con_props(connection, action, CS_HAFAILOVER, buf, buflen, outlen)
```

2 Modify the interfaces file so clients fail over to the secondary companion.

The *interfaces* file includes a line labeled *hafailover* that enables clients for reconnect to the secondary companion when the primary companion crashes or you issue a **shutdown with nowait**, triggering failover.

See "Add Entries for Both Adaptive Servers to the Interfaces File" on page 59 for information about adding this line to the interfaces file.

3 Write application failover messages according to the following parameters:

- As soon as the companion begins to go down, clients receive an informational message that failover is about to occur. Treat this as an informational message in the client error handlers.
- Once the failover property is set (from step 1) and the interfaces file has a valid entry for the *hafailover* server, the client connection is an failover connection, and clients reconnect to the secondary companion appropriately.

  However, if the failover property is set but the interfaces file does not have a entry for the *hafailover* server (or vice-versa), then it is a not a failover connection. Instead, it is a normal non-high availability connection with the failover property turned off. The user must check the failover property to know whether or not the connection was a failover connection.

1 Add return codes.

When a successful failover occurs, the client issues a return value named CS_RET_HAFAILOVER, which is specific to the following CTLIB API calls:

```
ret = ct_results(cmd, result_type)
ret = ct_send(cmd)
```

CS_RET_HAFAILOVER is returned from the API call during a synchronous connection. In an asynchronous connection, these API ‘s issue CS_PENDING and the **callback** function returns CS_RET_HAFAILOVER. Depending on the return code, the customer can do the required processing, such as sending the next command to be executed.

2 Rebuild your applications, linking them with the libraries included with the failover software.

Note You cannot connect clients with the failover property (for example **isql -Q**) until you issue **sp_companion resume**. If you do try to reconnect them after issuing **sp_companion prepare_failback**, the client hangs until you issue **sp_companion resume**.

Glossary

This glossary describes terms used in this book. For a description of Adaptive Server and SQL terms, refer to the *Adaptive Server Glossary*.

Asymmetrical — A high availability system consisting of one primary companion and one secondary companion. In an asymmetric system, only the primary companion can failover. In this system, the secondary Adaptive Server is also known as a "hot stand-by."

Companion Server — Each Adaptive Server in a high availability system is a companion. One of the Adaptive Servers is a **companion** (see below for definition) and the other is the **secondary** companion (see below for definition).

Cluster — A collection of nodes in a high availability system. A cluster for the Adaptive Server high availability system consists of two nodes.

Failback — The planned event during which Adaptive Server is migrated back to, and restarted on, the original machine. This involves moving the failed-over databases, devices, and client connections from the secondary companion to the restarted primary companion.

Failover — During failover, Adaptive Server migrates to another machine which takes over the responsibility of managing the failed over Adaptive Server. Failover can occur because of either a scheduled maintenance or a failure of Adaptive Server or the machine running Adaptive Server.

Failover Mode — The mode of the primary companion after it has failed over and is running on the secondary companion.

High Availability — A system that is designed to reduce the amount of downtime a system suffers.

Node — A machine in a high availability system.

Normal Companion Mode — The mode during which two Adaptive Servers in a high availability system are functioning as independent servers and are configured to failover during a scheduled maintenance or system failure.

Primary companion — The Adaptive Server whose databases and connections are migrated to the secondary Adaptive Server during failover.

Proxy Databases	Place holder databases created on the secondary companion for every user database on the primary companion. Proxy databases reserve the database names so that during failover, all database names are unique on the system. For more information about proxy databases, see Chapter 5, “Proxy Databases, User Databases, and Proxy System Tables”
Secondary companion	The Adaptive Server configured to accept the failed over primary Adaptive Server during failover
Single-Server Mode	The mode of Adaptive Server when it is being configured for high availability. During this mode, Adaptive Server cannot failover.
Suspended Companion Mode	The mode of Adaptive Server after companion mode has been suspended. During this mode, Adaptive Server cannot failover; it is working independently of the other Adaptive Server.
Symmetrical	A high availability system in which two independent Adaptive Servers act as failover servers for each other. That is, each Adaptive Server acts as both a primary and a secondary companion.

Index

A

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

W