Oracle8*i*

Standby Database Concepts and Administration

Release 2 (8.1.6)

December 1999

Part No. A76995-01

Oracle8*i* Standby Database Concepts and Administration, Release 2 (8.1.6)

Part No. A76995-01

Primary Authors: Lance Ashdown, Anna Logan

Contributing Authors: Rick Anderson, Mark Johnson, Steve Lee, Barbara Benton

Contributors: Joydip Kundu, Mark Smith, Lawrence To, Jiangbin Luo, Bill Bridge, Janet Blowney

Graphic Designer: Valarie Moore

Contents

List of Figures

List of Tables

Send Us Your Comments

Oracle8*i* Standby Database Concepts and Administration, Release 2 (8.1.6)

Part No. A76995-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the chapter, section, and page number (if available). You can send comments to us in the following ways:

- E-mail - nedc_doc@us.oracle.com
- Fax - 603-897-3819. Attn: Oracle 8*i* Standby Database Documentation
- Postal service:
  Oracle Corporation
  Oracle 8*i* Standby Database Documentation
  One Oracle Drive
  Nashua, NH 03062-2698
  U.SA.

If you would like a reply, please give your name, address, and telephone number below.

____

____

____

If you have problems with the software, please contact your local Oracle Support Services.

Preface

The standby database is one of Oracle Corporation's most powerful and exciting additions to its Oracle8*i* backup and recovery solutions. This guide includes the conceptual and task-oriented information you will need to create, manage, and maintain a standby database.

See Also: For general information about backup and recovery, see the *Oracle8i Backup and Recovery Guide*. For concepts, procedures, and reference material related to the Recovery Manager utility, see the *Oracle8i Recovery Manager User's Guide and Reference*.

What Is New in Oracle8*i*?

This section describes new features relating to standby databases in Oracle release 8.1.5 and release 8.1.6.

Release 8.1.6

A new feature in release 8.1.6 enables you to generate trace files that describe archival activity by setting the LOG_ARCHIVE_TRACE initialization parameter (see Determining Which Archived Logs Have Been Received by the Standby Site on page 4-5).

Release 8.1.5

New features in release 8.1.5 include the following:

- Oracle8*i* provides the means to allow the background archive process (ARCH) or its foreground equivalent to archive online redo log files to multiple destinations (see Specifying Archive Destinations on page 2-16). Up to five destinations can be specified with the initialization parameter: LOG_ARCHIVE_DEST_*n* (where *n* is an integer from 1 to 5). A destination can be a local disk-based file or it can be a user-specified standby database that is either local or remote to the primary database.

- You can run a standby database in *managed recovery mode* so that as archived redo logs are generated on the production database, they are automatically applied to the standby database (see Managed Recovery Mode on page 1-9).
- The *read-only mode* option can be used to make a standby database available for queries and reporting, even while archived redo logs are being transferred from the primary database site (see Testing the Standby Database Without Performing Failover on page 1-10).

Structure

This book contains the following chapters:

Chapter	Contents
Chapter 1, "Standby Database Concepts"	Offers a general overview of the Oracle8*i* standby database.
Chapter 2, "Preparing a Standby Database"	Describes how to create a standby database.
Chapter 3, "Managing a Standby Database"	Provides guidelines for managing a standby database in manual recovery mode, managed recovery mode, and read-only mode.
Chapter 4, "Performing Maintenance on a Standby Database"	Provides step-by-step instructions for typical maintenance operations on a standby database.
Chapter 5, "Standby Database Scenarios"	Describes common standby database scenarios.

Changes to This Book

This manual is new in release 8.1.6. In release 8.1.5, all backup and recovery documentation was located in the *Oracle8i Backup and Recovery Guide*. In release 8.1.6, the backup and recovery documentation is divided into the following books:

- *Oracle8i Backup and Recovery Guide*
- *Oracle8i Recovery Manager User's Guide and Reference*
- *Oracle8i Standby Database Concepts and Administration*

Audience

This guide is for database administrators (DBAs) who administer the backup, restore, and recovery operations of an Oracle database system.

Knowledge Assumed of the Reader

Readers of this guide are assumed to be familiar with relational database concepts and basic backup and recovery administration. They are also assumed to be familiar with the operating system environment under which they are running Oracle.

Conventions

This section explains the conventions used in this manual relating to:

- Text
- Code Examples

Text

This section explains the conventions used within the text.

UPPERCASE Characters

Uppercase text is used to call attention to tablespace names, initialization parameters, and SQL keywords.

For example, "If you create a private rollback segment, the name must be included in the ROLLBACK_SEGMENTS parameter of the initialization parameter file. You can view this information by issuing a SHOW PARAMETER statement in SQL*Plus."

Italicized Characters

Italicized words within text are book titles, new vocabulary, emphasized words, or variables in SQL syntax.

For example, "An *archived redo log* is an online redo log that has been copied offline. You *must* run your database in ARCHIVELOG mode to enable this feature. If you are using Recovery Manager, you can specify an archived redo log in a **backup** command by using the **archivelog like** *'/oracle/archive/arc_*'* subclause."

Bold Characters

Bold words within text are operating system-specific commands.

For example, "Use the Recovery Manager **backup** command to back up your database. Alternatively, use the UNIX **cp** command to copy files."

`Monospaced` Characters

Filenames and directories appear in a monospaced font.

For example, "The information in the primary site `tnsnames.ora` file must correspond to the information in the standby site `listener.ora` file."

Code Examples

SQL and SQL*Plus statements appear separated from the text of paragraphs in a monospaced font. For example:

```
INSERT INTO emp (empno, ename) VALUES (1000, ’SMITH’);
ALTER TABLESPACE users ADD DATAFILE ’users2.ora’ SIZE 50K;
run {
     allocate channel ch1 type disk;
     backup database;
}
```

You can execute SQL and SQL*Plus statements in different environments on different platforms. As much as possible, this guide attempts to provide generic documentation; that is, documentation that is not specific to any operating system or interface. Nevertheless, it is sometimes necessary for illustrative purposes to show how the syntax works at the operating system level. In these cases, this book uses examples from a UNIX command-line interface and employs the % symbol to indicate the operating system prompt.

How to Use This Guide

Every reader of this guide is presumed to have read:

- The beginning of the *Oracle8i Concepts* manual, which provides an overview of the concepts and terminology related to Oracle and a foundation for the more detailed information in this guide. The rest of the *Oracle8i Concepts* manual explains the Oracle architecture and features in detail.
- The chapters in the *Oracle8i Administrator’s Guide* that deal with managing the control file, online redo logs, and archived redo logs.

You will often need to refer to the following reference guides:

- *Oracle8i SQL Reference*
- *Oracle8i Reference*
- *Oracle8i Backup and Recovery Guide*
- *Oracle8i Recovery Manager User's Guide and Reference*

1 Standby Database Concepts

This chapter explains the nature and function of a standby database. It includes the following topics:

What Is a Standby Database?

A standby database is a database replica created from a backup of a primary database. By applying archived redo logs from the primary database to the standby database, you can keep the two databases synchronized.

A standby database has the following main purposes:

- Disaster protection
- Protection against data corruption
- Supplemental reporting

If the primary database is destroyed or its data becomes corrupted, you can perform a failover to the standby database, in which case the standby database becomes the new primary database. You can also open a standby database with the read-only option, thereby allowing it to function as an independent reporting database.

This section contains the following topics:

- Configuration Options
- Advantages and Disadvantages
- Compatibility and Operational Requirements
- Concepts and Terminology

Configuration Options

You can set up a standby database in several different ways, depending on the method for:

- Transferring archived redo logs to the standby site
- Applying archived redo logs to the standby database

For example, Oracle's *managed standby environment* allows the primary database to automatically archive redo logs to the standby database site so long as the standby instance is started. If you implement a *non-managed standby environment*, you must transfer the logs manually.

If the standby database is in *managed recovery mode*, the standby database automatically applies logs received from the primary database. You can also apply logs manually to the standby database by placing it in *manual recovery mode*. At any time you can open the standby database in *read-only mode* for reporting purposes.

The following table explains the possible configurations depending on the environment that you choose:

Environment	Method of Transfer	Standby Database Modes	Network Requirements
Managed	Automatic (or manual if necessary)	Managed recovery, manual recovery, or read-only	Net8
Non-Managed	Manual only	Manual recovery or read-only	None

Most database administrators (DBAs) choose a managed recovery environment. You may prefer a non-managed environment if:

- You do not want to maintain a Net8 connection between the primary and standby sites, which is required for managed recovery.
- You want to create a time lag between the archiving of a log at the primary database and the application of the log to the standby database. A time lag protects against the transfer of corrupted or erroneous data from the primary database to the standby database.

See Also: To learn about the different standby database modes, see Standby Database Modes on page 1-7. To learn about configuration options in the standby database environment, see Configuration of the Standby Database Environment on page 1-19. To learn how to set up a standby database with a time lag, see Scenario 10: Standby Database with a Time Lag on page 5-51.

Advantages and Disadvantages

A standby database can be a powerful tool for both disaster prevention and supplementary reporting. For example, you can:

- Maintain a standby database in a location that is geographically remote from the primary database, or maintain several standby databases in geographically diverse locations.
- Maintain the primary and standby databases on different disk drives of the same machine, so that if the primary database's drive fails, you can activate the standby database and resume normal operations.
- Implement a managed standby configuration, whereby a standby database automatically applies archived redo logs that are automatically shipped to the

standby site by a primary database. In this way, changes to a primary database are regularly propagated to a standby database.

- Make a standby database the new primary database with minimal loss of time and data if the primary database is completely destroyed.
- Provide possible protection against erroneous batch jobs, user errors (for example, truncating the wrong table), or application corruptions on the primary database by *not* applying archived logs containing corrupt data to the standby database. You can then activate the uncorrupted standby database, making it the primary database.

While a standby database can be a tremendous benefit in your backup and recovery strategy, it involves costs as well. For example, a standby database requires:

- An additional computer if you want to maximize disaster prevention by maintaining a standby database on a separate host
- Implementation and maintenance of a Net8 connection if you use the managed standby environment
- Additional system resources and extra storage space no matter which implementation you choose
- Administration of the standby database to mirror some structural operations (for example, adding a tablespace or datafile) performed on the primary database

Compatibility and Operational Requirements

Note the following requirements for maintaining a standby database:

- The primary database must run in ARCHIVELOG mode.
- A standby database in manual recovery mode operates only on Oracle release 7.3 or higher.
- A standby database in managed recovery mode operates only on Oracle release 8.1 or higher.
- A standby database in read-only mode operates only on Oracle release 8.1.5 or higher.
- The redo logs that you apply to the standby database must be either archived or noncurrent online redo logs. Note that you can salvage the transactions in the current redo log by archiving it manually.

- You must use the same version and release of the operating system on the primary and standby hosts. The standby host can, however, use a different directory structure.
- You should use the same version, release, and patch of the Oracle RDBMS for the primary and standby databases so that failover operations are not compromised.
- The primary database and standby database cannot share the same control file.
- If you place your primary and standby databases on the same host, some operating systems will not allow you to mount two instances with the same database name on the same machine simultaneously. Workarounds for this situation exist for every platform.
- You cannot activate a standby database and then return it to managed recovery mode; an activated standby database becomes a normal primary database.

Concepts and Terminology

Familiarize yourself with the following terms, which are used throughout the subsequent chapters:

activation
See failover.

failover
The operation of turning a standby database into a normally functioning primary database. This operation is also called standby database *activation*. Note that after a failover, you cannot switch the standby database back so that it becomes a standby database again.

gap sequence
A sequence of archived redo logs that must be manually applied to a standby database before it can be placed in managed recovery mode.

managed recovery mode
A standby database mode initiated by entering the following SQL*Plus statement:

```
RECOVER MANAGED STANDBY DATABASE;
```

When a standby database runs in managed recovery mode, it automatically applies redo logs received from the primary database.

managed standby environment
A configuration in which a primary database automatically archives redo logs to a standby site. If the standby database is in managed recovery mode, it automatically applies the logs received from the primary database to the standby database. Note that in a managed standby environment, a standby *site* continues to receive archived logs even if the standby *database* is not in managed recovery mode.

manual recovery mode
A standby database mode initiated by issuing the following SQL*Plus statement:

```
RECOVER STANDBY DATABASE;
```

This mode allows you to recover a standby database manually.

non-managed standby environment
Any environment in which the primary database does not automatically archive redo logs to the standby site. In this environment, you must manually transfer archived logs to the standby site and manually apply them.

primary database
A database used to create a standby database. Every standby database is associated with one and only one primary database. A single primary database can, however, support multiple standby databases.

primary site
The location of the primary database. Note that the primary and standby sites can be on separate hosts or on the same host.

read-only mode
A standby database mode initiated by issuing the following SQL statement:

```
ALTER DATABASE OPEN READ ONLY;
```

This mode allows you to query the standby database, but not to make changes to it.

standby database
A database replica created using a backup of your primary database. A standby database has its own initialization parameter file, control file, and datafiles.

standby database environment
The physical configuration of the primary and standby databases. The environment depends on many factors, including:

- The number of standby databases associated with a primary database

- The number of machines used by the databases
- The directory structures of the machines used by the databases
- The network configuration

standby site
The location of the standby database. The standby site can be on the same host as the primary database or on a separate host.

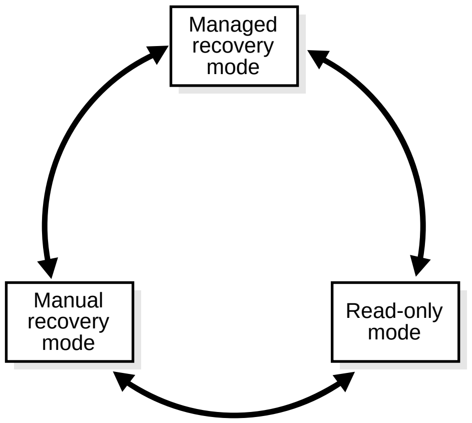
Standby Database Modes

You can perform any of the following mutually exclusive operations on a standby database:

- Maintain it in manual recovery mode
- Maintain it in managed recovery mode
- Open it in read-only mode for queries

Although you cannot run the standby database in more than one mode at the same time, you can switch back and forth between the modes at will. For example, you can run in managed recovery mode, then open read-only, then switch to manual recovery, then return to managed recovery, as shown in Figure 1–1.

Figure 1–1 Switching Between Modes

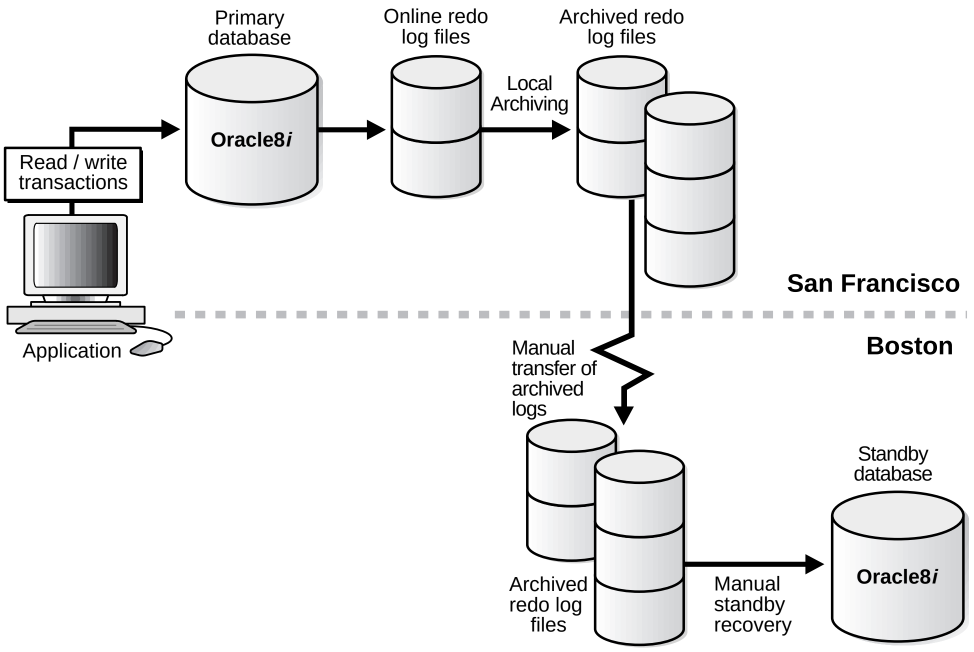


Manual Recovery Mode

You have the option of placing the database in *manual recovery mode*, in which case you must continually and manually transfer and apply archived redo logs to the standby database to keep it synchronized with the primary database.

To perform manual recovery, connect to the standby database using SQL*Plus and issue the RECOVER STANDBY DATABASE statement. Figure 1–2 shows an example of a database in manual recovery mode.

Figure 1–2 Standby Database in Manual Recovery Mode



Manual recovery mode is useful in environments in which you do not want to connect the primary and standby databases through Net8. Also, if for some reason the primary database is unable to automatically transfer archived redo logs to a standby database in a managed recovery environment, you may need to perform manual recovery to update the standby database.

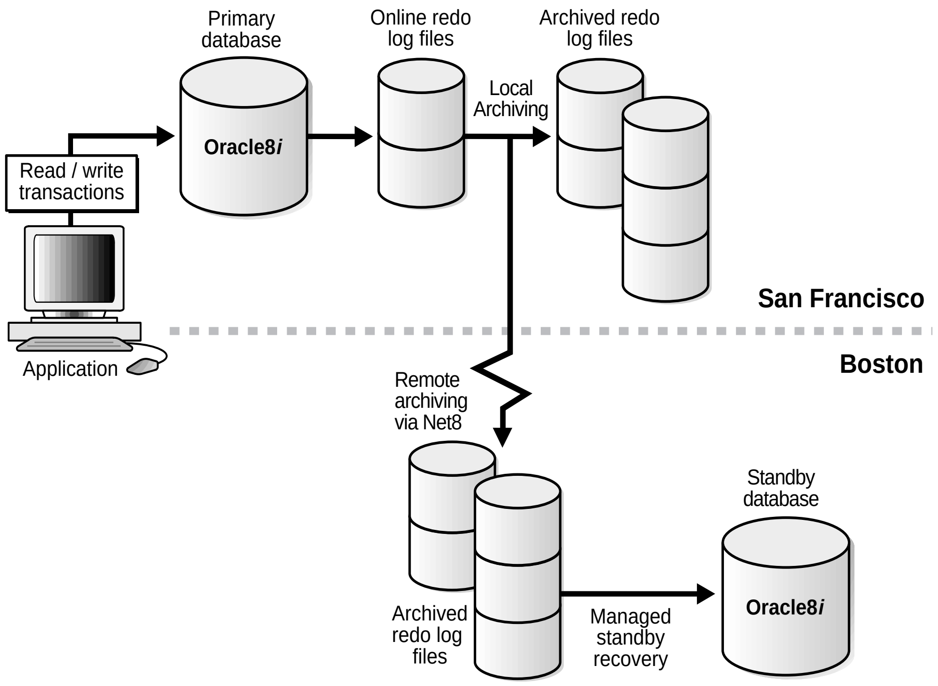
> **See Also:** To learn how to recover a standby database manually, see Placing the Standby Database in Manual Recovery Mode on page 3-3.

Managed Recovery Mode

You can place the standby database in *managed recovery mode*, in which case the standby database automatically applies archived redo logs as it receives them from the primary database. To initiate managed recovery, connect to the standby database using SQL*Plus and issue the RECOVER MANAGED STANDBY DATABASE statement.

The principal advantage of running a database in managed recovery mode is that you do not have to transfer or apply archived redo logs manually: Oracle automates the procedure. For example, Figure 1–3 illustrates a case in which a primary database in San Francisco transmits archived redo logs to a standby site in Boston, where the standby database automatically applies them.

Figure 1–3 Automatic Updating of a Standby Database

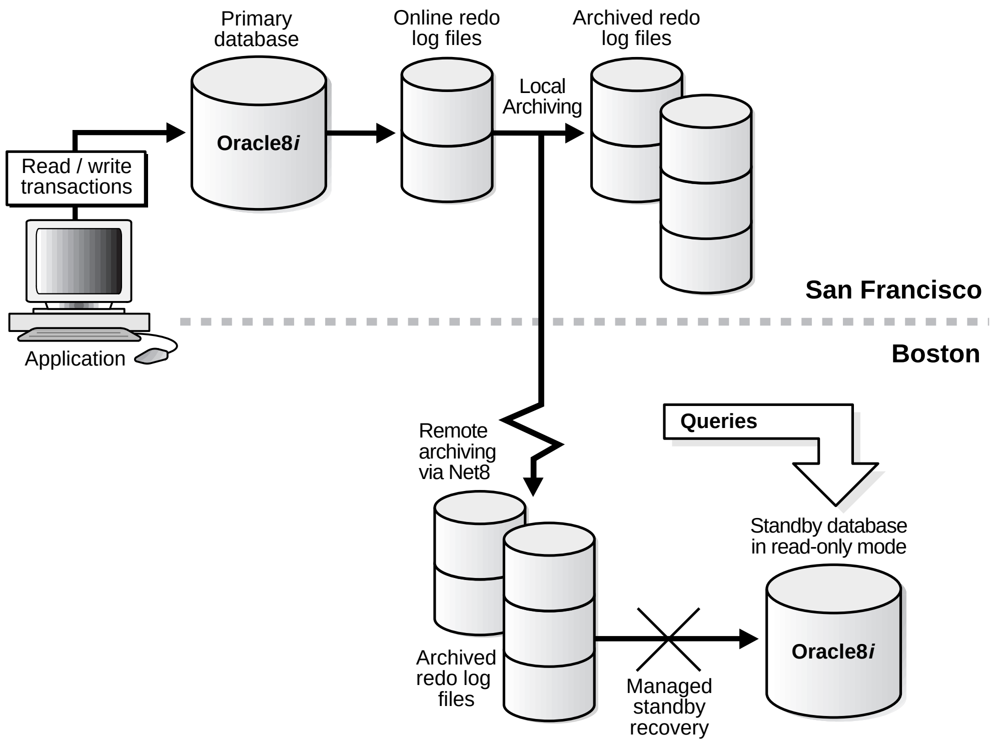


See Also: To learn how to run the standby database in managed recovery mode, see Placing the Standby Database in Managed Recovery Mode on page 3-13.

Read-Only Mode

You can also open your standby database in *read-only mode* after terminating manual or managed recovery. You can then query the database and even store data in temporary tablespaces (so long as they already exist in the standby database) without affecting the datafiles or redo logs. You can return the standby database to manual or managed recovery mode at any time, without having to shut it down. Figure 1–4 shows a standby database in read-only mode.

Figure 1–4 Standby Database in Read-Only Mode



In a managed standby environment, the standby site continues to receive redo logs archived by the primary database and the control file continues to be updated with their records. Consequently, archiving continues to the standby site even though the standby database does not perform recovery while in read-only mode.

A read-only standby database is useful when you want to decrease the number of queries to the primary database. For example, if specific tablespaces in a primary database change infrequently but are accessed frequently, you can direct those queries to the standby database so the primary database does not become overloaded with read requests.

See Also: To learn how to open a standby database in read-only mode, see Opening a Standby Database in Read-Only Mode on page 3-17.

Failover to a Standby Database

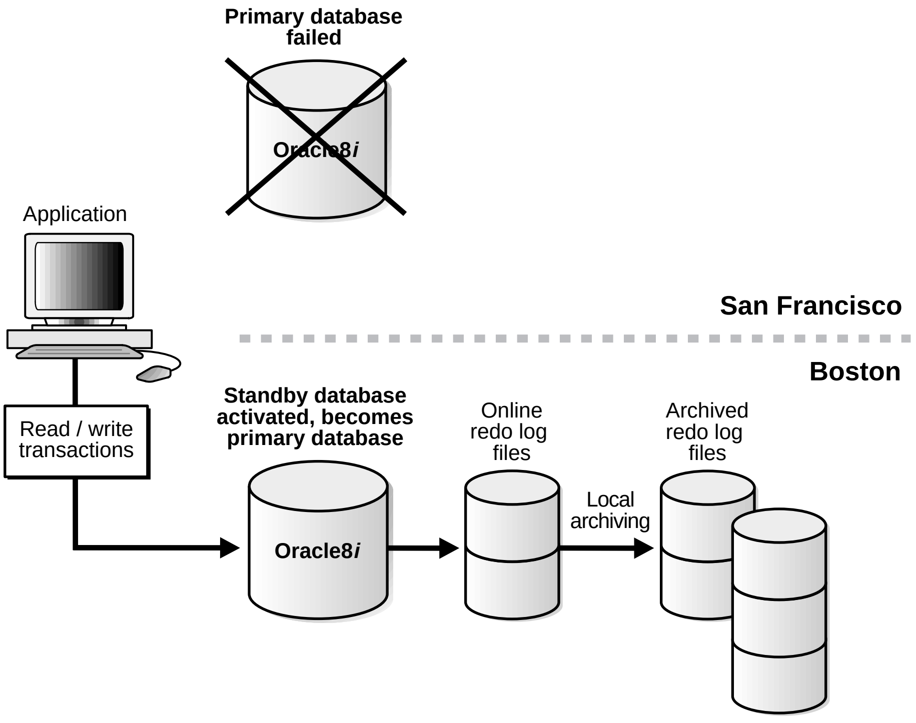
Performing a *failover* to a standby database, also known as *activation*, occurs when you issue the following SQL statement:

```
ALTER DATABASE ACTIVATE STANDBY DATABASE;
```

You can issue this statement only when the standby database is mounted.

Figure 1–5 depicts a failover operation from a primary database in San Francisco to a standby database in managed recovery mode in Boston.

Figure 1–5 Failover to a Standby Database



After you activate the standby database, it ceases to be a standby database and becomes a fully functional primary database. At this point, you can open the database in read / write or read-only mode and make changes or issue queries as usual.

CAUTION: Activating a standby database is a permanent operation. You cannot undo the activation and return the database to its former role as a standby database.

See Also: To learn how to perform failover to a standby database, see Activating a Standby Database on page 3-20.

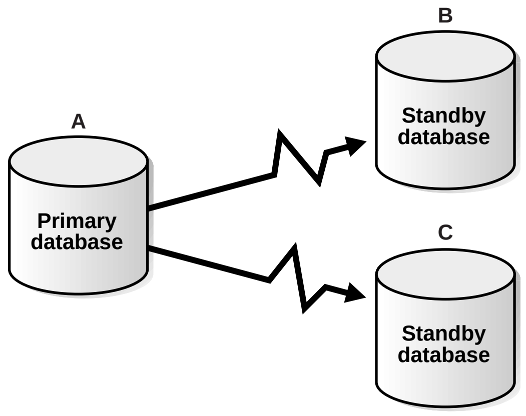
Consequences of Failover

Failover permanently transforms a standby database into a primary database. Because standby activation is a unidirectional operation, you cannot return the new primary database to any of the standby modes. In other words, you cannot perform a failover and then undo it.

CAUTION: Activating a standby database resets the online logs of the standby database. After activation, the archived logs from the standby database and the primary database are incompatible.

Another consequence of failover is that any other standby databases that were supporting the original primary database are now invalid as standby databases to the new, activated primary database. For example, assume primary database A supports standby databases B and C, as illustrated in Figure 1–6.

Figure 1–6 Primary Database with Multiple Standby Databases



If you perform a failover from A to B, then C does not function as standby database to the newly activated B. Because B's redo logs are reset, you cannot apply archived redo logs from B to C.

In some situations, maintaining multiple standby databases can lessen the repercussions of a failover. For example, assume the preceding scenario, with standby databases B and C supporting primary database A. The following events occur:

1. A’s machine suffers a media failure.
2. You activate standby database B. Users now access B as the primary database.
3. You quickly fix the media problem on A’s machine.
4. You shut down B, then restart A.
5. Users now access A as the primary database again. C continues to function as a standby database for A, while B is invalidated.

One consequence of this scenario is that any changes made to B while it briefly served as the primary database cannot be applied through archived redo logs. Note that you can generate a report of these changes by using the LogMiner utility and then reenter the changes manually into database A.

Testing the Standby Database Without Performing Failover

Because failover to a standby database destroys its standby functionality, perform this operation only when absolutely necessary. If you want to test the standby database, do not activate it—open it in read-only mode instead. By opening in read-only mode, you can query the standby database to ensure that it is correctly updating the datafiles with the redo logs received from the primary database.

Re-Creating the Original Primary Database After Failover

If you activate a standby database and then solve the problem at the original primary site that necessitated the failover operation, you have the option of re-creating the primary database on the original primary site. Perform the following steps, assuming the original primary site was on node A and the activated standby site is on node B:

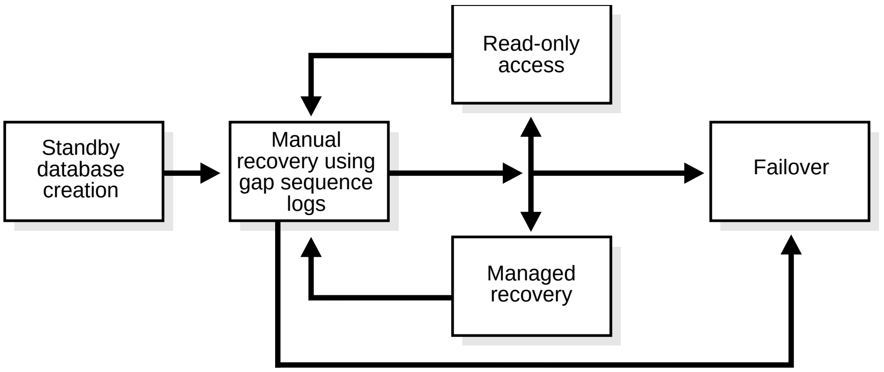
1. Make a consistent backup of the activated standby database on node B.
2. Restore the backup created on node B to node A.
3. Shut down the activated standby database on node B.
4. Open the restored database on A. It is now the primary database.
5. Make a backup of the database on node A.
6. Use the backup of A to re-create the standby database on node B.

See Also: To learn how to re-create a standby database, see Scenario 8: Re-Creating a Standby Database on page 5-44.

Standby Database Life Cycle

Most standby database implementations use managed recovery. The life cycle of a standby database intended for managed recovery is illustrated in Figure 1–7.

Figure 1–7 Standby Database Life Cycle


Read-only access
Standby database creation
Manual recovery using gap sequence logs
Failover
Managed recovery

The four stages illustrated in this diagram are:

1. Standby Database Creation
2. Manual Recovery Using Logs in the Gap Sequence
3. Managed Recovery and Read-Only Access Cycle
4. Failover to the Standby Database

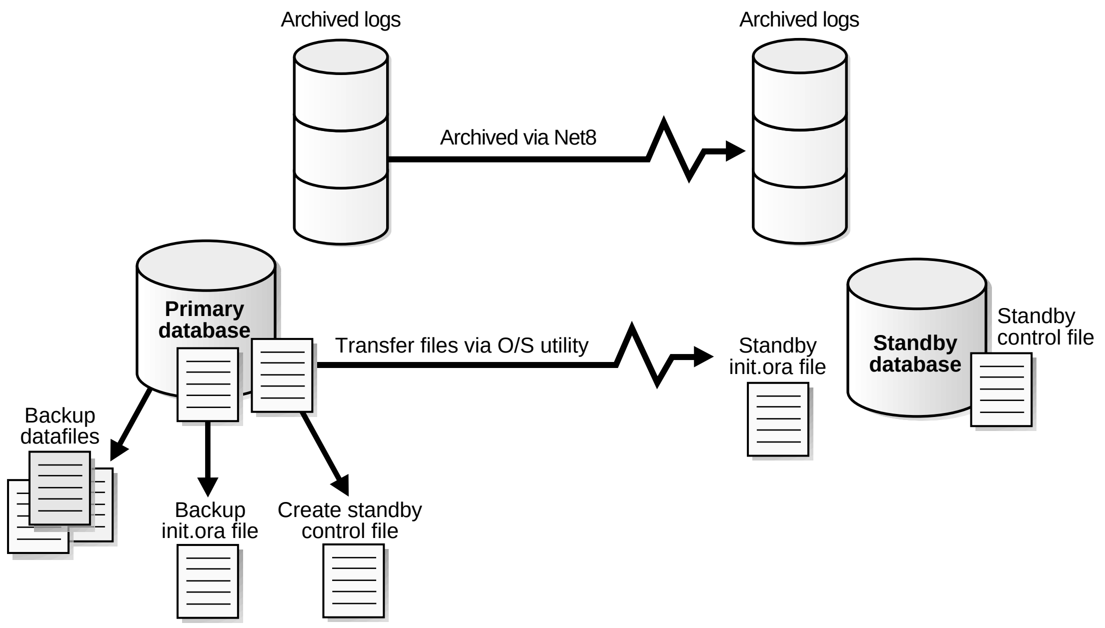
Standby Database Creation

In this stage, you must construct the standby database from backups of the primary database control files and datafiles, and then prepare it for managed recovery. This stage, illustrated in Figure 1–8, involves the following basic steps:

1. Make a backup of the primary datafiles (or access a previous backup) and create the standby control file.
2. Transfer the standby datafiles and control file to the standby site.
3. Configure Net8 so that you can connect to the standby service name.
4. Configure the primary and standby initialization parameter files.
5. Initiate automatic archiving on the primary site.

6. Start the standby instance without mounting it.

Figure 1–8 Standby Database Creation



See Also: This procedure is explained in complete detail in Chapter 2. For additional information, see Scenario 1: Creating a Standby Database on the Same Host on page 5-2 and Scenario 2: Creating a Standby Database on a Remote Host on page 5-14.

Manual Recovery Using Logs in the Gap Sequence

A *gap sequence* is created whenever you are unable to apply the next archived redo log generated by the primary database to the standby database. For example, the primary database archives log 100 to the standby site, but the standby control file has no knowledge of any logs after log sequence 89, because the standby control file was created when the most recent log archived by the primary database was 89. The gap sequence in this case spans logs 90 to 99.

To be able to begin managed recovery, you must first manually apply logs in the gap sequence to the standby database. After you have performed this manual recovery, you can issue the RECOVER MANAGED STANDBY DATABASE statement, at which point Oracle applies subsequent logs to the standby database automatically.

Typical Causes of Gap Sequences

Most commonly, gap sequences occur in the following situations:

- Creation of the standby database
- Shutdown of the standby database when the primary database is open
- A network failure that prevents archiving to the standby site

In the first two situations, the primary database can archive redo logs to the standby site, but the standby database control file is unaware of logs archived while it was not mounted. Whenever the primary database archives to the primary site but the standby control file does not contain records of logs that are necessary for recovery of the database, a gap sequence is created.

In the third situation, the primary database continues to archive locally, but is prevented from archiving to the standby site by a network failure. Archived logs accumulate at the primary site, but the standby control file does not know about them. Consequently, you must transfer the accumulated logs manually and then apply them in a manual recovery operation before managed recovery can begin.

The Gap Sequence Cycle

Because a gap sequence can occur whenever the primary database is archiving logs that the standby control file is not informed about, you can occasionally go through a *gap sequence cycle*. This cycle occurs whenever you must exit either managed recovery mode or read-only mode to perform manual recovery using logs in a gap sequence. After you have completed manual recovery of all logs in the sequence, you can return to managed recovery or read-only mode.

> **See Also:** Detailed procedures for resolving the gap sequence problem are described in Resolving a Gap Sequence Before Initiating Managed Recovery on page 3-5. Various scenarios in Chapter 5 also describe the procedure for resolving gap sequences.

Managed Recovery and Read-Only Access Cycle

In most scenarios, you run the database primarily in managed recovery mode or primarily in read-only mode. During managed recovery, the standby site receives logs from the primary database and the standby recovery process applies them automatically. In read-only mode, the standby site receives logs from the primary database, but the standby recovery process does not apply them.

You can easily switch between managed recovery mode and read-only mode. Because the standby control file continues to be updated about incoming logs when it is in read-only mode, you do not have to perform manual recovery before returning to managed recovery mode.

> **See Also:** Detailed procedures for performing managed recovery are in Placing the Standby Database in Managed Recovery Mode on page 3-13. Detailed procedures for opening the standby database in read-only mode are in Opening a Standby Database in Read-Only Mode on page 3-17.

Failover to the Standby Database

You can activate the standby database at any time, so long as the database is mounted. Typically, you perform a failover operation while the standby database is running in managed recovery mode. It can occur, however, that the primary database fails while you are performing manual recovery using logs in the gap sequence. In this case, you can activate the standby database even though it does not contain the latest changes made to the primary database.

After failover, you can back up the new primary database and start the standby life cycle over again by creating a new standby database.

> **See Also:** Detailed procedures for performing failover are described in Activating a Standby Database on page 3-20.

Configuration of the Standby Database Environment

A standby database environment refers to the physical configuration of a primary database with one or more associated standby databases. This section describes the main factors affecting configuration of the standby database environment:

- Number of Standby Databases
- Method of Transferring Archived Redo Logs to the Standby Site
- Location and Directory Structure of Primary and Standby Sites

Number of Standby Databases

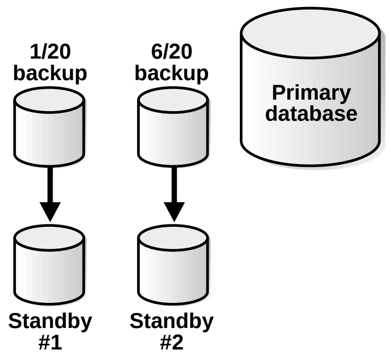
Although a standby database can be synchronized with one and only one primary database, a single primary database can support a theoretically unlimited number of standby databases. These standby databases are separate and independent, and can reside on multiple machines or on a single machine.

Using Backups for Standby Creation

Each standby database must be created from a backup of the primary database. This backup can be either consistent or inconsistent, open or closed.

You can also use a single backup of the primary database to create an unlimited number of standby databases, although the various standby databases in the environment do not have to be created from the same backup. For example, you can create one standby database from a backup of the primary database taken on January 20 and create another standby database from the backup taken on June 20 (see Figure 1–9). So long as you have the archived redo logs required to perform complete recovery of a backup, then it can serve as the basis for a standby database.

Figure 1–9 Creating Standby Databases Using Different Backups



Limits Imposed by Automatic Archival to Standby Sites

In a managed standby environment, the primary database can automatically archive to a maximum of four standby sites. Consequently, you can simultaneously run a maximum of four standby databases in managed recovery mode in any given standby database environment.

Although automatic archival limits the number of standby databases that you can maintain in managed recovery mode, it does not limit the number of standby databases that you can maintain in *manual* recovery mode. For example, you can configure the primary database to archive automatically to four standby sites, then use operating system commands to copy archived redo logs to five additional standby sites. Consequently, there is no theoretical limit to the number of standby databases that can support a given primary database.

Method of Transferring Archived Redo Logs to the Standby Site

One crucial aspect of any standby database environment is the transfer of archived redo logs from the primary site to the standby site. You have two options for transferring the logs:

- Configure the primary database to archive automatically to the standby site
- Transfer the logs manually

If you archive automatically to a standby site, then you are operating in a *managed recovery environment*.

Managed Recovery Environment

If you choose to implement a managed recovery environment, then you must connect the primary and standby databases through Net8. To configure the primary initialization parameter file for automatic archival to a standby site, you must specify a service name. To specify a service name, you must configure network files such as `tnsnames.ora`, `listener.ora`, and possibly `names.ora`.

When a primary database archives to a standby service, Oracle automatically transfers the archived redo logs through Net8 to a directory on the standby site. As the primary database archives each log, Oracle automatically transfers the new log to the standby site.

Independence of Automatic Archival and Managed Recovery Note that the primary database can continue to archive to the standby site *even if* the standby database is not in standby recovery mode, but only if the standby instance is started. The mechanism for recovery of a standby database is independent of the mechanism for automatic transfer of archived redo logs to the standby site. Consequently, you can take a standby database out of managed recovery mode and temporarily place it in read-only mode. While the standby database is in read-only mode, archived redo logs continue to accumulate at the standby site.

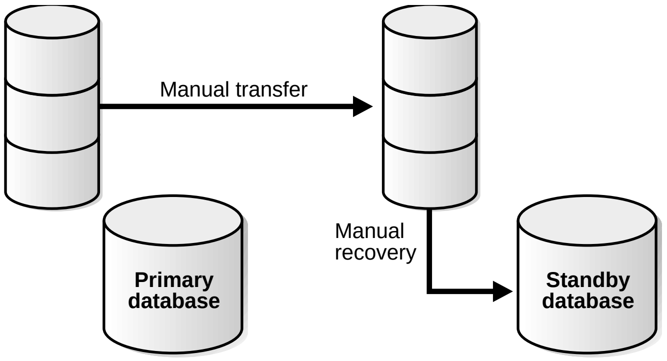
Optional Manual Transfer of Archived Redo Logs Even if you configure the primary database to archive automatically to the standby site, you can still transfer the logs manually if the occasion requires it, but only if you have specified the standby site as an *optional* destination (see Specifying Mandatory and Optional Archive Destinations on page 2-17).

For example, assume that a problem with the Net8 configuration prevents the transfer of archived logs to the standby site. The primary database continues to archive locally, so you can transfer the logs manually using operating system commands, then perform manual recovery at the standby site to synchronize the standby database.

Non-Managed Recovery Environment

In a non-managed recovery environment, you do not configure the primary database to archive automatically to the standby site and so must transfer the archived redo logs by hand. Because the primary database is not archiving automatically to the standby site, you do not need to maintain a Net8 connection. For example, you can use an operating system utility such as UNIX **cp** or **ftp** to transfer the archived redo logs to the standby site, connect to the standby database using operating system authentication, then recover the database manually (see Figure 1–10).

Figure 1–10 Non-Managed Recovery Environment



Note: Managed and non-managed recovery environments are not mutually exclusive. For example, you can configure the primary database to archive automatically to one standby location through Net8, then manually transfer archived logs from the primary site to a different, non-network standby site.

Location and Directory Structure of Primary and Standby Sites

One crucial aspect of the standby database environment is the number and configuration of the machines involved. Of particular importance are whether:

- The primary and standby databases reside on the same host or on different hosts
- The primary and standby sites have identical or different directory structures

Number and Location of Machines

Just as there is no theoretical limit to the number of standby databases associated with a given primary database, there is no theoretical limit to the number of machines on which the standby databases reside. For example, you can locate a standby database:

- On the same host as the primary database
- On a different host in the same data center
- On a different host in a different data center, but in the same metropolitan area
- On a different host in a data center in a geographically remote location, for example, on a different continent

The location of hosts involved in the standby database environment has obvious implications for a disaster recovery strategy. For example, if the primary host in a data center is destroyed, then you cannot perform failover to a standby database unless it resides on a different host, which may or may not be in the same data center. In a worst case scenario, if the data center is completely destroyed, then you cannot perform a failover to a standby database unless the standby database is located on a different machine in a remote location.

See Also: Scenario 5: Deciding Which Standby Database to Fail Over to in a Multiple Standby Database Configuration on page 5-30 addresses the location of the primary and standby databases.

Directory Structure of Standby Sites

The directory structure of the various standby sites is important because it determines the path names for the standby datafiles and redo logs. Note that a standby site on the same host as the primary database necessarily uses a different directory structure; otherwise, the standby database attempts to overwrite the primary database files.

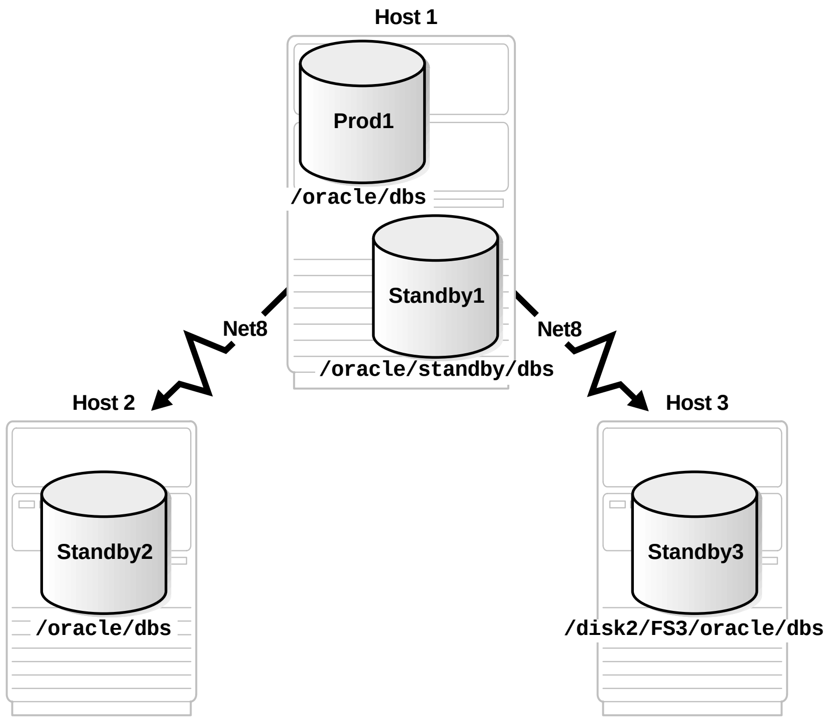
In general, use the same path names for the standby files if at all possible. This option prevents you from having to set filename conversion parameters or manually rename standby files. Nevertheless, if you need to use a host with a different directory structure or place the standby and primary databases on the same host, you can do so with a minimum of extra administration.

Configuration Options

You have three basic configuration options, which are illustrated in Figure 1–11:

- A standby database on the same host as the primary database
- A standby database on a separate host that uses the same directory structure as the primary host
- A standby database on a separate host that uses a different directory structure from the primary host

Figure 1–11 Possible Standby Configurations



The following table describes possible configurations of primary and standby databases and the consequences of each:

Standby Host	Directory Structure	Consequences
Same as primary host	Differs from primary host (necessarily)	■ You must set the LOCK_NAME_SPACE parameter. ■ You must rename primary database filenames in the standby database control file (see Renaming Primary Filenames in the Standby Control File on page 2-21). ■ Some operating systems do not permit two instances with the same name to run on the same machine. Refer to your platform-specific documentation for more information. ■ The standby database does not protect against disaster.
Separate	Same as primary host	■ You do not need to rename primary database filenames in the standby control file, although you can still do so if you want a new naming scheme (for example, to spread the files among different disks). ■ Using separate physical media for your databases safeguards your primary data.
Separate	Differs from primary host	■ You must rename primary database filenames in the standby database control file (see Renaming Primary Filenames in the Standby Control File on page 2-21). ■ Using separate physical media for your databases safeguards your primary data.

Although these configurations are mutually exclusive, that is, a given standby database must use only one configuration, you can run multiple standby databases simultaneously for a given primary database. Consequently, you can implement any combination of configurations. For example, you can maintain one standby database on the same host as the primary database, another standby database on a separate host in the same data center, and a third standby database on a separate host on the other side of the world. You can run each standby in managed recovery mode, manual recovery mode, or read-only mode.

Besides increasing disaster protection, maintaining more than one standby database gives you more flexibility at the standby database management level. For example, you can shut down one standby database to upgrade hardware while maintaining disaster protection by running other standby databases. Also, you can delay the application of archived redo logs to one standby database but not to another, so that if the primary database is corrupted, you can activate the standby database that has not yet become corrupted.

A standby database provides excellent protection against user errors, for example, accidentally truncating a table. By delaying the application of an archived redo log to the standby database, you can open the standby database in read-only mode and export the missing data. You can then import the missing data back into the primary database. When the correct data is in the primary database, you can continue to recover the standby database. This procedure propagates both the user error and the correction of the user error, so you can avoid activating the standby database.

Standby Database Maintenance

Although a managed standby environment is mostly automated, it is not completely automated. In various situations, you may need to perform status checks or maintenance on the standby database. The most common maintenance operations are:

- Checking the Status of Archived Redo Logs
- Backing Up the Standby Database
- Responding to Physical Changes in the Primary Database

Checking the Status of Archived Redo Logs

To ensure that the standby database is functioning properly, you may want to see which logs have been archived to the standby site and which logs have been applied to the standby database. You have these options:

- Use the V$LOG_HISTORY view on the standby database to see the most recently applied log.
- Analyze a standby trace file, which you create using the LOG_ARCHIVE_TRACE parameter in the standby initialization parameter file, to obtain information on logs received at the standby site.

> **See Also:** For more information about analyzing archived log behavior, see Troubleshooting the Standby Database Configuration on page 2-33.

Backing Up the Standby Database

If needed, you can back up your standby database, but not while the database is in manual or managed recovery mode. You must take the standby database out of managed recovery mode, make the backups, then resume managed recovery. You can make the backups when the database is shut down or when it is in read-only mode.

See Also: For more information about making backups, see Backing Up the Standby Database on page 4-16.

Responding to Physical Changes in the Primary Database

Changes to the primary database structure always affect a standby database. In cases such as the following, the standby database is updated automatically through applied redo:

- Taking a primary tablespace online or offline
- Changing tablespace status from read / write and read-only
- Renaming a datafile
- Dropping a tablespace

In cases such as the following, however, you must perform maintenance on the standby database:

- Adding a datafile to the primary database
- Creating a tablespace on the primary database

In cases such as the following, you must re-create the standby database entirely:

- Restoring the backup control file on the primary database
- Opening the primary database with the RESETLOGS option

See Also: For more information about maintenance issues relating to structural changes on the primary database, see Responding to Events That Affect the Standby Database on page 4-9. See Scenario 3: Accommodating Physical Changes in the Primary Database on page 5-21 for a description of procedures you should follow when a physical change is made in the primary database.

Standby Database Statements

Several different SQL and SQL*Plus statements use a STANDBY keyword to specify operations on a standby database. Other SQL statements do not include standby-specific syntax, but are useful for performing operations on a standby database. Table 1–1 describes relevant statements.

Table 1–1 Standby Database Statements

Statement	Description
`ALTER DATABASE ACTIVATE STANDBY DATABASE`	Activates the standby database, which effectively destroys the standby functionality of the database. See Activating a Standby Database on page 3-20.
`ALTER DATABASE CREATE STANDBY CONTROLFILE AS '`*`filename`*`'`	Creates a standby control file. Issue this statement at the primary database. See Creating the Standby Control File on page 2-8.
`ALTER DATABASE MOUNT STANDBY DATABASE`	Mounts a standby database.
`ALTER DATABASE OPEN READ ONLY`	Opens the standby database in read-only mode. See Opening a Standby Database in Read-Only Mode on page 3-17.
`RECOVER [FROM '`*`/dir`*`'] STANDBY DATABASE`	Initiates manual recovery of the standby database. You can direct Oracle to search a nondefault directory for the archived redo logs. See Placing the Standby Database in Manual Recovery Mode on page 3-3.
`RECOVER MANAGED STANDBY DATABASE [TIMEOUT `*`integer`*`]`	Initiates managed recovery of the standby database. Use the TIMEOUT option to cause Oracle to exit recovery after waiting *integer* minutes for the log to arrive. See Placing the Standby Database in Managed Recovery Mode on page 3-13.
`RECOVER MANAGED STANDBY DATABASE CANCEL`	Causes Oracle to wait for managed recovery to finish writing the current log before canceling recovery. See Canceling Managed Recovery on page 3-16.
`RECOVER MANAGED STANDBY DATABASE CANCEL IMMEDIATE`	Cancels managed recovery either before reading another block from the redo log or before opening the next redo log file, whichever comes first. See Canceling Managed Recovery on page 3-16.

Table 1–1 (Cont.) Standby Database Statements

Statement	**Description**
`STARTUP NOMOUNT pfile='`*`initSID.ora`*`'`	Starts the standby instance without mounting the control file. You must execute this statement before mounting the standby database.

2 Preparing a Standby Database

This chapter describes how to set up a standby database in preparation for manual or managed recovery. It includes the following topics:

Assessing the Environment Before Standby Database Creation

After you have decided to implement a standby database, you should address several important issues relating to the environment in which the standby database will reside. How you answer the questions depends on the purpose of the standby database: protection against data loss or corruption, or supplemental reporting.

- How Should the Standby Database Be Updated?
- Should the Standby Database Reside on a Separate Host?
- How Should You Configure the Online Redo Logs in the Primary Database?
- How Should You Connect to the Standby Database After Failover?

How Should the Standby Database Be Updated?

The first question is whether the standby database will run in a managed standby environment (see Configuration Options on page 1-2). In this environment, the primary database automatically archives redo logs to the standby site so long as the standby instance is started.

Whether you implement a managed standby environment depends on your situation. The following table offers suggestions:

Purpose	Standby Environment
Disaster protection	Managed
Protection against data corruption	Managed with a time lag, or non-managed
Supplemental reporting on a data warehouse or database that does not change frequently	Managed or non-managed
Supplemental reporting on a data warehouse or database that changes frequently	Managed

If you choose a managed standby environment, then you can run the standby database in managed recovery mode so that it applies archived redo logs automatically. Whether you plan to use the database for reporting or protection against data loss, this option requires the least user intervention. If you choose a non-managed environment, then you must transfer the logs manually and perform manual recovery of the standby database.

Should the Standby Database Reside on a Separate Host?

If you can, it is almost always better to place a standby database on a separate host, for the following reasons:

- If you do not, the standby database cannot protect against disaster.
- Placing the standby database on the same host as the primary database creates system overhead.
- You can use the same directory structure on both hosts, thereby eliminating the requirement to rename datafiles and online redo logs in the standby control file. Different directory structures for the primary and standby sites necessitate filename conversions.

> **See Also:** To learn about renaming primary files in the standby control file, see Renaming Primary Filenames in the Standby Control File on page 2-21 and Manually Renaming Standby Files Not Captured by Conversion Parameters on page 2-31.

How Should You Configure the Online Redo Logs in the Primary Database?

Both the size of the online redo logs and the frequency with which they switch affect the generation of archived redo logs at the primary site. Table 2–1 describes some consequences of various online redo log configurations.

Table 2–1 Consequences of Online Redo Log Configuration

If the online redo logs are	Then
Small	The logs switch more frequently. If you maintain a managed standby environment, frequent switches can cause: ■ Significant network traffic caused by continual log transfer. ■ Difficulties for manual recovery of a standby database that cannot begin managed recovery because of a gap sequence. For a discussion of gap sequences, see Resolving a Gap Sequence Before Initiating Managed Recovery on page 3-5. ■ Additional protection against data loss, because fewer transactions are contained in each archived redo log.

Table 2–1 (Cont.) Consequences of Online Redo Log Configuration

If the online redo logs are	Then
Large	The redo logs switch less frequently, but more transactions are stored in each file. This situation can: ■ Increase the transfer time of a single log, creating a possible network bottleneck. ■ Increase the amount of potential data loss if the primary database fails before it can archive its redo logs.
Numerous	The primary database does not have to reuse redo logs as often as it does when you maintain the minimum of two. Because Oracle is not forced to archive at every log switch, a disaster can wipe out multiple logs before the primary database archives them to the standby site.

How Should You Connect to the Standby Database After Failover?

If the primary database fails, users should be able to connect to the activated standby database seamlessly. For example, one simple solution is to configure the client `tnsnames.ora` files or Oracle Names server to contain an entry for the standby service name. After failover, users can connect to the database using the standby service name instead of the primary service name.

Other failover solutions include:

- The Net8 failover facility
- Server IP / DNS configurations
- Application rerouting

When deciding on a method, weigh the simplicity of the solution against the need for a seamless failover.

Setting Up a Standby Database: Basic Tasks

After you have decided to create a standby database and determined where to place it, you can begin preparing it. The procedure of standby database creation differs depending on whether you plan to set up a managed standby environment or a non-managed standby environment.

This section contains the following topics:

- Preparing a Standby Database for Managed Recovery: Basic Tasks

- Preparing a Standby Database for Manual Recovery: Basic Tasks

See Also: For a complete overview of what a standby database is and how it works, see Chapter 1. For an explanation of important terms such as *managed standby environment*, see Concepts and Terminology on page 1-5.

Preparing a Standby Database for Managed Recovery: Basic Tasks

Setting up a standby database for managed recovery requires you to perform a series of different tasks. After you have completed the preparation and initiated managed recovery, the standby database automatically and continuously applies redo logs as they are received from the primary database.

Table 2–2 summarizes the basic tasks for setting up a standby database and synchronizing it so it is ready to begin managed recovery.

Table 2–2 Task List: Preparing for Managed Recovery

	Task	Procedure
1	Either make a new backup of the primary database datafiles or access an old backup.	Creating the Standby Datafiles on page 2-7
2	Connect to the primary database and create the standby control file.	Creating the Standby Control File on page 2-8
3	Copy the backup datafiles and control file from the primary site to the standby site.	Transferring Files to the Standby Site on page 2-9
4	Create a service name for the standby database.	Configuring the tnsnames.ora File on the Primary Site on page 2-12
5	Configure the listener on the standby site so that it can receive the archived redo logs from the primary site.	Configuring the listener.ora File on the Standby Site on page 2-13
6	Set the initialization parameters for the primary database.	Configuring the Primary Database Initialization Parameter File on page 2-15
7	Create the standby initialization parameter file and set the initialization parameters for the standby database. Depending on your configuration, you may need to set filename conversion parameters.	Configuring the Standby Database Initialization Parameter File on page 2-19
8	Start the standby instance and mount the standby database.	Starting the Standby Instance on page 2-31

Table 2–2 (Cont.) Task List: Preparing for Managed Recovery

	Task	Procedure
9	Manually change the names of the primary datafiles and redo logs in the standby control file for all files *not* automatically renamed using DB_FILE_NAME_CONVERT and LOG_FILE_NAME_CONVERT in step 7. If step 7 renamed all files, skip this step.	Manually Renaming Standby Files Not Captured by Conversion Parameters on page 2-31
10	Manually enable initialization parameter changes on the primary database so that it can initiate archival to the standby site.	Enabling Changes to the Initialization Parameter Settings on page 2-32

Preparing a Standby Database for Manual Recovery: Basic Tasks

Table 2–3 illustrates the basic tasks for setting up a standby database in preparation for manual recovery. This procedure assumes that you plan to connect to the standby database through Net8. If you do not wish to use Net8 to connect to the standby database, skip tasks 4 and 5.

Table 2–3 Task List: Preparing for Manual Recovery

	Task	Procedure
1	Either make a new backup of the primary datafiles or access an old backup.	Creating the Standby Datafiles on page 2-7
2	Connect to the primary database and create the standby control file.	Creating the Standby Control File on page 2-8
3	Copy the backup datafiles and standby control file from the primary site to the standby site.	Transferring Files to the Standby Site on page 2-9
4	If you want to create a Net8 connection to the standby database, create a service name.	Configuring the tnsnames.ora File on the Primary Site on page 2-12
5	If you want to create a Net8 connection to the standby database, configure the listener on the standby site so that it can receive requests for connections to the standby instance.	Configuring the listener.ora File on the Standby Site on page 2-13

Table 2–3 (Cont.) Task List: Preparing for Manual Recovery

	Task	**Procedure**
6	Create the standby initialization parameter file on the standby site and set the initialization parameters for the standby database. Optionally, set DB_FILE_NAME_CONVERT and LOG_FILE_NAME_CONVERT to automatically rename primary files in the standby control file.	Configuring the Standby Database Initialization Parameter File on page 2-19
7	Start the standby instance and mount the standby database.	Starting the Standby Instance on page 2-31
8	While connected to the standby database, manually change the names of the primary datafiles and redo logs in the standby control file for all files *not* automatically renamed using DB_FILE_NAME_CONVERT and LOG_FILE_NAME_CONVERT in step 6. If step 6 renamed all files, skip this step.	Manually Renaming Standby Files Not Captured by Conversion Parameters on page 2-31

Creating the Standby Database Files

You can create a standby database on the same host as your primary database or on a separate host. If you create your standby database on the same host, follow the creation procedure carefully when creating the standby database files so that you do not overwrite files on the primary database.

The creation of the standby database files occurs in three stages:

1. Creating the Standby Datafiles
2. Creating the Standby Control File
3. Transferring Files to the Standby Site

Creating the Standby Datafiles

First, make backups of your primary database datafiles. You create the standby datafiles from these backups.

You can use any backup of the primary database so long as you have archived redo logs to completely recover the database. The backup can be old or new, consistent or inconsistent. Hot backups have the advantage of allowing you to keep the database open while performing the backup. Nevertheless, you may prefer to make a new closed, consistent backup to prevent the application of a large number of archived redo logs.

To make a consistent, whole database backup to serve as the basis for the standby database:

1. Start a SQL*Plus session on your primary database and query the V$DATAFILE view to obtain a list of the primary datafiles. For example, enter:

   ```
   SQL> SELECT name FROM v$datafile;
   NAME
   ----------------------------------------------------------------------------
   /oracle/dbs/tbs_01.f
   /oracle/dbs/tbs_02.f
   /oracle/dbs/tbs_03.f
   /oracle/dbs2/tbs_11.f
   /oracle/dbs2/tbs_12.f
   /oracle/dbs3/tbs_21.f
   /oracle/dbs3/tbs_22.f
   7 rows selected.
   ```

2. Shut down the primary database cleanly:

   ```
   SQL> SHUTDOWN IMMEDIATE;
   ```

3. Make a consistent backup of the datafiles from your primary database using an operating system utility. For example, to copy all of the datafiles into the `/backup` temporary directory, enter:

   ```
   % cp /oracle/dbs/*.f /backup
   % cp /oracle/dbs2/*.f /backup
   % cp /oracle/dbs3/*.f /backup
   ```

4. Reopen the primary database. For example, enter:

   ```
   SQL> STARTUP pfile=initPROD1.ora;
   ```

   > **See Also:** To learn how to make operating system backups, see the *Oracle8i Backup and Recovery Guide*.

Creating the Standby Control File

After you have created the backups that will be used as the standby datafiles, you can create the standby database control file. The control file must have been created at a time later than the latest timestamp for the backup datafiles.

Note: You cannot use a single control file for both the primary and standby databases. The standby instance is independent from the primary instance and so requires exclusive possession of its database files.

To create the standby database control file:

1. Ensure that the primary database is in ARCHIVELOG mode and that archiving is enabled. Either issue the ARCHIVE LOG LIST statement or query the V$DATABASE view:

   ```
   SQL> CONNECT sys/change_on_install@prod1
   SQL> ARCHIVE LOG LIST
   Database log mode              Archive Mode
   Automatic archival             Enabled
   Archive destination            /vobs/oracle/work/arc_dest/arc
   Oldest online log sequence     821
   Next log sequence to archive   822
   Current log sequence           822
   ```

2. Connect to the primary database and create the control file for your standby database. For example, if you want to create the standby control file as `oracle/dbs/stbycf.f` on the primary site, enter the following:

   ```
   SQL> ALTER DATABASE CREATE STANDBY CONTROLFILE AS '/oracle/dbs/stbycf.f';
   ```

   Note that the filename for the created standby control file *must* be different from the filename of the current control file of the primary database.

   See Also: For ALTER DATABASE syntax, see the *Oracle8i SQL Reference*.

Transferring Files to the Standby Site

After you have successfully created the standby datafiles and control file, transfer the files to the standby site using an operating system utility. For example, if the standby site and primary site are on the same host, you can use the UNIX **`cp`** command to transfer files; if they are on separate hosts, you can use **`ftp`**.

If the standby database is on	Then you
A separate host with the same directory structure as the primary database	Can use the same path names for the standby files as the primary files. In this way, you do not have to rename the primary datafiles in the standby control file.
The same host as the primary database, or the standby database is on a separate host with a different directory structure	Must rename the primary datafiles in the standby control file after copying them to the standby site. You can: ■ Set the filename conversion initialization parameters (see Renaming Primary Filenames in the Standby Control File on page 2-21). ■ Rename the files manually using ALTER DATABASE statements (see Manually Renaming Standby Files Not Captured by Conversion Parameters on page 2-31). ■ Use a combination of conversion parameters and manual renames.

To transfer datafiles and the control file to the standby site:

Transfer the created control file and datafile backups to the standby site using operating system commands or utilities. Use an appropriate method for transferring binary files.

1. Transfer the control file first, because this transfer takes the least time. For example, enter the following:

   ```
   % cp /backup/db.cf /standby/oracle/dbs/db.cf
   ```

2. Transfer the backup datafiles. For example, enter:

   ```
   % cp /backup/*.df /standby/oracle/dbs
   ```

3. Transfer all available archived redo logs to the standby site. For example, enter:

   ```
   % cp /arc_dest/*.arc /standby/arc_dest
   ```

Configuring Network Files for Primary and Standby Databases

Network configuration varies greatly from system to system. If you do *not* plan to implement a managed standby environment in which the primary site archives to a standby site, you do not need Net8. You can connect to the standby database using operating system authentication. To implement a managed standby environment, however, you must create a service name for the standby instance. Consequently, you must use Net8.

In a managed standby environment, you must configure the `listener.ora` file for the standby site so that the standby site can receive archived logs from the primary site. If you do not use a name server, you will have to configure a `tnsnames.ora` file for the primary site. You may also have to configure additional network files such as `SQL*Net.ora` and `names.ora`. For an account of Net8 configuration, see the *Net8 Administrator's Guide*.

The information in the primary site `tnsnames.ora` file must correspond to the information in the standby site `listener.ora` file. The following table indicates which parameters must be identical and gives sample settings. Values that must be the same in both the `tnsnames.ora` and `listener.ora` files are in bold:

Value	Sample tnsnames.ora Setting	Sample listener.ora Setting
PORT	sbdb = (DESCRIPTION= (ADDRESS=(PROTOCOL=tcp) (**PORT=1512**)(HOST=hst1)) (CONNECT_DATA = (SID=stby1)))	LISTENER = (ADDRESS_LIST= (ADDRESS=(PROTOCOL=tcp) (**PORT=1512**)(HOST=hst1)))
HOST	sbdb = (DESCRIPTION= (ADDRESS=(PROTOCOL=tcp) (PORT=1512)(**HOST=hst1**)) (CONNECT_DATA = (SID=stby1)))	LISTENER = (ADDRESS_LIST= (ADDRESS=(PROTOCOL=tcp) (PORT=1512)(**HOST=hst1**)))
SID (tnsnames) = SID_NAME (listener)	sbdb = (DESCRIPTION= (ADDRESS=(PROTOCOL=tcp) (PORT=1512)(HOST=hst)) (CONNECT_DATA = (**SID=stby1**)))	SID_LIST_LISTENER = (SID_LIST= (SID_DESC= (**SID_ NAME=stby1**)(ORACLE_ HOME=/oracle)))
PROTOCOL	sbdb = (DESCRIPTION= (ADDRESS=(**PROTOCOL=tcp**) (PORT=1512)(HOST=hst1)) (CONNECT_DATA = (SID=stby1)))	LISTENER = (ADDRESS_LIST= (ADDRESS=(**PROTOCOL=tcp**) (PORT=1512)(HOST=hst1)))
KEY	sbdb = (DESCRIPTION= (ADDRESS=(PROTOCOL=ipc) (**KEY=foo**))(CONNECT_ DATA=(SID=stby1)))	LISTENER = (ADDRESS_LIST= (ADDRESS = (PROTOCOL=ipc) (**KEY=foo**)))

Following are generic procedures for modifying network files in a typical configuration. This section contains the following topics:

- Configuring the tnsnames.ora File on the Primary Site
- Configuring the listener.ora File on the Standby Site

Configuring the tnsnames.ora File on the Primary Site

If you want to archive redo logs to a standby site, and you are not using an Oracle Names server, you must configure the `tnsnames.ora` file on the primary site. This file contains the service name of the standby instance.

To modify the tnsnames.ora file for a TCP connection:

1. Start a text editor and open the `tnsnames.ora` file, which is typically located in `$ORACLE_HOME/network/admin`.
2. Add the following entry to the `tnsnames.ora` file, substituting appropriate values for *standby_service_name*, *port_number*, *host_name*, and *standby_sid.* Make sure the SID matches the SID_NAME of the `listener.ora` file and the PORT values are the same in the two files:

```
standby_service_name = (DESCRIPTION=
    (ADDRESS=(PROTOCOL=tcp)(PORT=port_number)(HOST=host_name))
     (CONNECT_DATA=(SID=standby_sid))
)
```

   For example, for standby database STANDBY1 on host REMOTE2 you can enter the following:

```
standby1 = (DESCRIPTION=
    (ADDRESS=(PROTOCOL=tcp)(PORT=1512)(HOST=remote2))
     (CONNECT_DATA=(SID=standby1))
)
```

To modify the tnsnames.ora file for an IPC connection:

1. Start a text editor and open the `tnsnames.ora` file, which is typically located in `$ORACLE_HOME/network/admin`.
2. Add the following entry to the `tnsnames.ora` file, substituting appropriate values for *standby_service_name*, *key_handle*, and *standby_sid*. Make sure the SID matches the SID_NAME of the `listener.ora` file and the KEY values are the same in the two files:

```
standby_service_name = (DESCRIPTION=
    (ADDRESS=(PROTOCOL=ipc)(KEY=key_handle))
     (CONNECT_DATA=(SID=standby_sid))
)
```

   For example, for standby database STANDBY1 you might enter the following:

```
standby1 = (DESCRIPTION=
      (ADDRESS=(PROTOCOL=ipc)(KEY=stby))
       (CONNECT_DATA=(SID=standby1))
    )
```

See Also: For information about the `tnsnames.ora` file or network configuration parameters, see the *Net8 Administrator's Guide*.

Configuring the listener.ora File on the Standby Site

In order to archive redo logs to a standby site, you must configure the `listener.ora` file on the standby site. The listener receives network connection requests from client applications and reroutes them to the standby server.

To modify the listener.ora file for a TCP connection:

1. Start a text editor and open the `listener.ora` file, which is typically located in `$ORACLE_HOME/network/admin`.
2. Add the listener to the file, substituting appropriate values for *port_number* and *host_name*. Note that you can create a new listener and call it anything you want, or add a new address to a previously existing listener:

   ```
   STANDBY_LISTENER = (ADDRESS_LIST=
    (ADDRESS=(PROTOCOL=tcp)(PORT=port_number)(HOST=host_name))
   )
   ```

   For example, for a standby host called REMOTE2 you might enter:

   ```
   STANDBY_LISTENER = (ADDRESS_LIST=
    (ADDRESS=(PROTOCOL=tcp)(PORT=1512)(HOST=remote2))
   )
   ```

3. Add the following entry to the SID list for the listener, substituting appropriate values for *standby_sid_name* and *oracle_home*:

   ```
   SID_LIST_STANDBY_LISTENER = (SID_LIST=
    (SID_DESC=(SID_NAME=standby_sid_name)(ORACLE_HOME=/oracle_home))
   )
   ```

   For example, for a standby database called STANDBY1 you might enter:

   ```
   SID_LIST_STANDBY_LISTENER = (SID_LIST=
    (SID_DESC=(SID_NAME=standby1)(ORACLE_HOME=/oracle))
   )
   ```

4. If the listener is already started, enter `stop` and then enter `start`. If it is not started, use an operating system command to start it. For example, on a UNIX system enter:

```
% lsnrctl

LSNRCTL for Solaris: Version 8.1.5.0.0 - Production on 23-MAR-99 12:04:10

(c) Copyright 1998 Oracle Corporation.  All rights reserved.

Welcome to LSNRCTL, type "help" for information.

LSNRCTL> stop
```

5. Ensure that the service is enabled by issuing the `status` command:

```
LSNRCTL> status
```

   If the information for the standby service is configured correctly, you should see it among the list of the valid service names:

```
standby1               has 1 service handler(s)
```

To modify the listener.ora file for an IPC connection:

1. Start a text editor and open the `listener.ora` file, which is typically located in `$ORACLE_HOME/network/admin`.
2. Add the listener to the file, substituting appropriate values for *key_handle*. Note that you can create a new listener and call it anything you want, or add a new address to a previously existing listener:

```
STANDBY_LISTENER = (ADDRESS_LIST=
   (ADDRESS = (PROTOCOL=ipc) (KEY=key_handle))
)
```

   For example, for a standby host called REMOTE2 you might enter:

```
STANDBY_LISTENER = (ADDRESS_LIST=
   (ADDRESS = (PROTOCOL=ipc) (KEY=stby))
)
```

3. Add the following entry to the SID list for the listener, substituting appropriate values for *standby_sid_name* and *oracle_home*:

```
SID_LIST_STANDBY_LISTENER = (SID_LIST=
 (SID_DESC = (SID_NAME=standby_sid_name)(ORACLE_HOME=oracle_home))
)
```

For example, for a standby database called STANDBY1 you might enter:

```
SID_LIST_STANDBY_LISTENER = (SID_LIST=
 (SID_DESC = (SID_NAME=standby1)(ORACLE_HOME=/oracle))
)
```

4. If the listener is already started, enter `stop` and then enter `start`. If it is not started, use an operating system command to start it. For example, on a UNIX system enter:

```
% lsnrctl

LSNRCTL for Solaris: Version 8.1.5.0.0 - Production on 23-MAR-99 12:04:10

(c) Copyright 1998 Oracle Corporation.  All rights reserved.

Welcome to LSNRCTL, type "help" for information.

LSNRCTL> stop
```

5. Ensure that the service is enabled by issuing the `status` command:

```
LSNRCTL> status
```

If the information for the standby service is configured correctly, you should see it among the list of the valid service names:

```
standby1               has 1 service handler(s)
```

Configuring the Primary Database Initialization Parameter File

If you do *not* plan to implement a managed standby environment, you should not have to change the initialization parameter file of the primary system at all. You must transfer and apply the archived redo logs to the standby database manually, so you do not need to alter the archiving destinations.

If you *do* plan to implement a managed standby environment, you need to add new archiving destinations to the initialization parameter file of the primary system. This section assumes that you plan to implement a managed standby environment.

For the primary database to archive to a local or remote standby database location, the following must be true:

- The LOG_ARCHIVE_DEST_*n* parameters (where *n* is an integer from 1 to 5) in the primary initialization parameter file are correctly defined using the SERVICE attribute. The SERVICE attribute must be used for all standby database locations, whether local or remote.
- The `tnsnames.ora` file on the primary site and the `listener.ora` file on the standby site have the correct corresponding entries.
- The listener is started on the standby site.
- The standby instance is started on the standby site.

This section contains these topics:

See Also: To learn how to manage archived redo logs, see the chapter on archived redo logs in the *Oracle8i Administrator's Guide*. For more information about Oracle networking options, see the *Net8 Administrator's Guide*. For an overview of the archiver process, see *Oracle8i Concepts*.

Specifying Archive Destinations

Specify up to five destinations for your primary database archived logs by setting the LOG_ARCHIVE_DEST_*n* initialization parameter (where *n* is an integer from 1 to 5). Each numerically suffixed parameter uniquely identifies an individual destination, as shown in the following example:

```
# first local archiving destination
LOG_ARCHIVE_DEST_1 = 'LOCATION=/oracle/arc/'

# second local archiving destination
LOG_ARCHIVE_DEST_2 = 'LOCATION=/oracle/arc2/'

# third standby archiving destination
LOG_ARCHIVE_DEST_3 = 'SERVICE=stby'
```

If you wish to archive redo logs to a standby database in managed recovery mode, you must use LOG_ARCHIVE_DEST_*n* in conjunction with the SERVICE keyword.

Note that at least one archiving destination must be a local directory; that is, a non-standby site.

When using LOG_ARCHIVE_DEST_*n*, specify the archiving location using these keywords:

Keyword	Indicates	Example
LOCATION	A local directory	LOG_ARCHIVE_DEST_1= 'LOCATION=/arc_dest/'
SERVICE	Archival through a Net8 service name	LOG_ARCHIVE_DEST_2 = 'SERVICE=standby1'

When using the LOCATION keyword, specify a valid path name for your operating system. When you specify SERVICE, Oracle translates the net service name through the `tnsnames.ora` file to a connect descriptor. The descriptor contains the information necessary for connecting to the standby database.

See Also: For a detailed account of LOG_ARCHIVE_DEST_*n* and the archiving process, see the chapter on archived redo logs in the *Oracle8i Administrator's Guide*.

Specifying Mandatory and Optional Archive Destinations

Using the LOG_ARCHIVE_DEST_*n* parameters, you can specify whether a destination has the attributes OPTIONAL (default) or MANDATORY. For example, you can set the parameter as follows:

```
LOG_ARCHIVE_DEST_3 = 'SERVICE=standby1 MANDATORY'
```

Oracle Corporation recommends specifying the local directory destination as MANDATORY.

The LOG_ARCHIVE_MIN_SUCCEED_DEST=*m* parameter (where *m* is an integer from 1 to 5) specifies the number of destinations that must archive successfully before LGWR can overwrite the online redo logs. All MANDATORY destinations and non-standby OPTIONAL destinations contribute to satisfying the LOG_ARCHIVE_MIN_SUCCEED_DEST=*m* count. For example, you can set the parameter as follows:

```
LOG_ARCHIVE_MIN_SUCCEED_DEST = 2   # Oracle must archive to at least two
                                   # locations before overwriting the online
                                   # redo logs.
```

See Also: For a detailed account of the OPTIONAL and MANDATORY keywords, see the chapter on archived redo logs in the *Oracle8i Administrator's Guide*.

Enabling Archive Destination States

The LOG_ARCHIVE_DEST_STATE_*n* (where *n* is an integer from 1 to 5) initialization parameter specifies the *state* of the destination specified by its corresponding LOG_ARCHIVE_DEST_*n* parameter (where *n* is the same integer). For example, the LOG_ARCHIVE_DEST_STATE_3 parameter specifies the state of the LOG_ARCHIVE_DEST_3 archiving destination.

The archiving destination parameters can have two values: ENABLE and DEFER. ENABLE indicates that Oracle can archive to the destination, whereas DEFER indicates that it should not.

For example, you can set the parameter as follows:

```
LOG_ARCHIVE_DEST_STATE_2 = ENABLE
LOG_ARCHIVE_DEST_2 = 'SERVICE=stby1'
```

See Also: For a detailed account of the archive destination states, see the chapter on archived redo logs in the *Oracle8i Administrator's Guide*. For a description of the LOG_ARCHIVE_DEST_STATE_*n* parameter, see the *Oracle8i Reference*.

Configuring Oracle to Re-Archive to a Failed Destination

Use the LOG_ARCHIVE_DEST_*n* parameters to determine whether and when the archiver process attempts to re-archive to a failed destination following an error. The REOPEN=*s* (where *s* is an integer) option specifies the minimum number of seconds before the archiver process should try to reaccess a failed destination. Note that REOPEN applies to all errors, not just OPEN errors.

If you specify REOPEN, the keyword has a default value of 300 seconds. If you do not specify REOPEN, it has the value of 0, which is the same as turning off the option. Note that if the REOPEN option is turned off, the archiver process will never attempt to reaccess a destination following an error.

You cannot use REOPEN to specify a limit on the number of attempts to reconnect and transfer archived logs. The REOPEN attempt either succeeds or fails, in which case the REOPEN information is reset. By default, the managed recovery operation waits indefinitely for a requested archived redo log; it terminates only through a CANCEL command, a shutdown, or a crash.

For example, you can set the parameter as follows to specify a reopen time of 60 seconds:

```
LOG_ARCHIVE_DEST_2 = 'SERVICE=standby2 OPTIONAL REOPEN=60'
```

> **See Also:** For a detailed account of how to use the REOPEN option, see the chapter on archived redo logs in the *Oracle8i Administrator's Guide*.

Configuring the Standby Database Initialization Parameter File

Once you have configured the primary database initialization parameter file, you can duplicate the file for use by the standby database. The procedure for creating the standby initialization parameter file is as follows:

1. Copy the initialization parameter file for the primary database using an operating system utility.
2. Edit the initialization parameter file for use by the standby database.
3. Transfer the initialization parameter file to the standby site using an appropriate operating system utility.

This section contains the following topics:

- Configuring Standby Initialization Parameters: General Considerations
- Renaming Primary Filenames in the Standby Control File
- Specifying Filenames for the Standby Database Archived Redo Logs
- Configuring the Standby Initialization Parameter File: Typical Settings

Configuring Standby Initialization Parameters: General Considerations

As a rule, most initialization parameters at the primary and standby databases should be identical, although some initialization parameters such as CONTROL_FILES and DB_FILE_NAME_CONVERT need to be changed. Differences in initialization parameters other than those described in the following table can cause performance degradation at the standby database and, in some cases, halt standby database operations. Only change parameter values when required for the functioning of the standby database or for filename conversions.

The initialization parameters in Table 2–4 play a key role in the configuration of the standby database.

Table 2–4 Configuring Standby Database Initialization Parameters

Parameter	Guideline
COMPATIBLE	Always set the same at the primary and standby databases. If different, you may not be able to apply the logs from your primary database to the standby database.
DB_NAME	Always set to the same value as DB_NAME value in the primary database.
CONTROL_FILES	Always set to a different value from CONTROL_FILES in primary database. The names of the control files for the standby database must exist at the standby site.
DB_FILE_NAME_ CONVERT	Set to distinguish standby datafile filenames from primary datafile filenames, for example, when both databases reside on the same host, or a separate standby host uses different path names from the primary host. Because the standby control file is a copy of the primary control file, you must convert the standby filenames if the standby database is on the same host as the primary database or on a separate host with different path names. See also Renaming Primary Filenames in the Standby Control File on page 2-21.
LOCK_NAME_SPACE	Always set this value when the standby and primary databases share a host. Specifies the name space that the distributed lock manager uses to create lock names.
LOG_ARCHIVE_DEST	Specifies the location of the archived logs for a standby database in manual recovery mode. When performing manual recovery on the standby database, Oracle relies on either LOG_ARCHIVE_DEST or a user-defined filename to locate the logs. This parameter is also relevant for managed recovery. If a log is missing at the standby site and managed recovery halts, you must issue RECOVER STANDBY DATABASE to initiate manual recovery, which causes Oracle to look in LOG_ ARCHIVE_DEST for the logs by default. Typically, you set STANDBY_ARCHIVE_ DEST and LOG_ARCHIVE_DEST to the same value for managed recovery.

Table 2–4 (Cont.) Configuring Standby Database Initialization Parameters

Parameter	Guideline
LOG_ARCHIVE_ TRACE	Optionally, set this parameter to an integer value to see the progression of the archiving of redo logs to the standby site. Oracle writes an audit trail of the archived logs received from the primary database into a trace file. The parameter controls output generated by the archiving process (ARC*n* and foreground processes) on the primary database and the RFS process of the standby database. For a description of the possible integer values for this parameter and their meanings, see Determining Which Archived Logs Have Been Received by the Standby Site on page 4-5.
LOG_FILE_NAME_ CONVERT	Set to make your standby redo log filenames distinguishable from primary database redo log filenames. The parameter value converts the filename of a new log file on the primary database to the filename of a log file on the standby database. Adding a log file to the primary database necessitates adding a corresponding file to the standby database. When the standby database is updated, this parameter is used to convert the log file name on the primary database to the log file name on the standby database. The file must exist and be writable on the standby database or the recovery process will halt with an error.
STANDBY_ARCHIVE_ DEST	Used solely by a standby database in managed recovery mode to determine the location for the archived logs received from the primary database. Managed recovery mode uses this value along with LOG_ARCHIVE_FORMAT to generate the fully qualified standby database log filenames and stores the filenames in the standby control file. Managed recovery uses this data to drive recovery. For managed recovery, set STANDBY_ARCHIVE_DEST and LOG_ARCHIVE_DEST to the same value. If manual recovery is required because of a gap sequence, copy the missing log to the same location as the other logs and recover manually. You can then place the standby database back into managed recovery mode.

See Also: For more information on these initialization parameters, see the *Oracle8i Reference*.

Renaming Primary Filenames in the Standby Control File

If the standby site uses the same directory structure as the primary site, then you do not have to rename the primary filenames in the standby control file. If the primary and standby databases occupy the same node, however, or if the primary and standby sites use different directory structures, then you must rename the primary files in the standby control file so that the archived logs can be applied to the standby datafiles.

You can set initialization parameters so that your standby database automatically converts datafile and archived redo log filenames based on data in the standby database control file. If you cannot rename all primary filenames automatically using these parameters, then you must rename them manually (see Manually Renaming Standby Files Not Captured by Conversion Parameters on page 2-31).

Note: Note that if you do not set the LOCK_NAME_SPACE parameters differently when the standby and primary databases share a node, you will receive an `ORA-1102` error.

The initialization parameters in Table 2–5 perform automatic filename conversions.

Table 2–5 Filename Conversion

Parameter	Function
DB_FILE_NAME_CONVERT	Converts primary database datafile filenames to standby datafile filenames, for example, from `tbs_*` to `standbytbs_*`.
LOG_FILE_NAME_CONVERT	Converts primary database redo log filenames to standby database redo log filenames, for example, from `log_*` to `standbylog_*`.

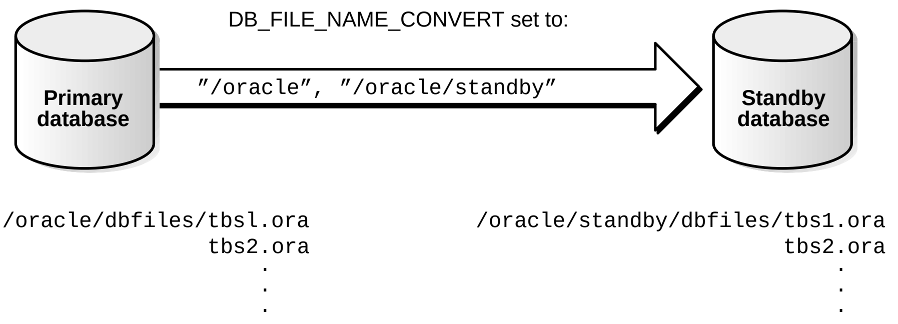
Use DB_FILE_NAME_CONVERT to convert the filename of a datafile on the primary database to a filename on the standby database; use LOG_FILE_NAME_CONVERT to convert the filename of a new redo log on the primary database to a filename on the standby database. Adding a datafile or log to the primary database necessitates adding a corresponding file to the standby database.

When the standby database is updated, DB_FILE_NAME_CONVERT is used to convert the datafile name on the primary database to a datafile name on the standby database. The file must exist and be writable on the standby database or the recovery process will halt with an error.

The DB_FILE_NAME_CONVERT and LOG_FILE_NAME_CONVERT parameters must have two strings. The first string is a sequence of characters to be looked for in a primary database filename. If that sequence of characters is matched, it is replaced by the second string to construct the standby database filename.

Figure 2–1 shows how the filename conversion parameters work.

Figure 2–1 Setting Filename Conversion Parameters


DB_FILE_NAME_CONVERT set to:
"/oracle", "/oracle/standby"
Primary database
Standby database
/oracle/dbfiles/tbsl.ora
tbs2.ora
/oracle/standby/dbfiles/tbs1.ora
tbs2.ora

If you execute the following statements, then the conversion parameters do not apply to the affected files:

- ALTER TABLESPACE ADD DATAFILE
- ALTER TABLESPACE RENAME DATAFILE
- ALTER DATABASE ADD LOGFILE
- ALTER DATABASE RENAME FILE
- ALTER DATABASE CREATE DATAFILE ... AS

> **See Also:** To learn how to add datafiles to the standby database, see Adding Tablespaces or Datafiles to the Primary Database on page 4-10.

Specifying Filenames for the Standby Database Archived Redo Logs

The naming conventions for archived redo logs on the standby site depend on whether the standby database is part of a managed standby environment. This section contains the following topics:

- Specifying Log Filenames for Managed Recovery
- Specifying Log Filenames for Manual Recovery

Specifying Log Filenames for Managed Recovery

When in a managed standby environment, the standby database uses the STANDBY_ARCHIVE_DEST parameter in the standby initialization parameter file to determine the directory in which to store the archived redo logs. Oracle uses this value in conjunction with LOG_ARCHIVE_FORMAT to generate the archived log filenames on the standby site.

Parameter	Indicates	Example
STANDBY_ARCHIVE_DEST	Directory in which to place logs	STANDBY_ARCHIVE_DEST= /arc_dest/
LOG_ARCHIVE_FORMAT	Format for filenames of logs	LOG_ARCHIVE_FORMAT = "log_%t_%s.arc" **Note:** The %s corresponds to the sequence number. The %t, which is required for OPS configurations, corresponds to the thread.

Oracle stores the fully qualified filenames in the standby control file. Managed recovery uses this information to perform the recovery operation. Access this information through the V$ARCHIVED_LOG view:

```
SQL> SELECT name FROM v$archived_log;
NAME
--------------------------------------------------------------------------------
/arc_dest/log_1_771.arc
/arc_dest/log_1_772.arc
/arc_dest/log_1_773.arc
/arc_dest/log_1_774.arc
/arc_dest/log_1_775.arc
```

Specifying Log Filenames for Manual Recovery

With the exception of RECOVER MANAGED STANDBY DATABASE, the RECOVER STANDBY DATABASE statements rely on one of the following to provide the location of the archived files:

- The LOG_ARCHIVE_DEST value
- A user-specified filename

Even if you run a standby database in managed recovery mode, manual recovery may sometimes be necessary; for example, when an archived redo log is absent from the standby site. This situation can occur when a network error prevents one or more archived redo logs from being transferred by the primary database.

Issuing the RECOVER STANDBY DATABASE statement manually in these circumstances requires you to use the LOG_ARCHIVE_DEST parameter to locate the necessary archived redo log. For a standby database that is normally in managed recovery mode, Oracle Corporation recommends setting STANDBY_

ARCHIVE_DEST and LOG_ARCHIVE_DEST to the same value. In this way, a standby database can access the same set of archived redo logs whether in managed or manual recovery mode.

Configuring the Standby Initialization Parameter File: Typical Settings

How you configure the initialization parameter file for the standby database depends on several interrelated factors:

- Whether the primary and standby databases are on the same host
- Whether the directory structure for the standby site is the same as the directory structure for the primary site
- Whether you plan to run the standby database in managed recovery mode or manual recovery mode
- Whether you maintain more than one standby database

Table 2–6 illustrates which parameters you should set depending on the standby database configuration.

Table 2–6 Setting Standby Database Initialization Parameters

Standby Host	Directory Structure	Recovery Mode	Standby Initialization Parameters
Same as primary host	Different from primary host	Manual	COMPATIBLE CONTROL_FILES DB_FILE_NAME_CONVERT LOCK_NAME_SPACE LOG_ARCHIVE_DEST LOG_ARCHIVE_TRACE LOG_FILE_NAME_CONVERT
Same as primary host	Different from primary host	Managed	Same as for manual recovery, except set STANDBY_ARCHIVE_DEST to the same value as LOG_ARCHIVE_DEST
Separate from primary host	Same as primary host	Manual	COMPATIBLE CONTROL_FILES LOG_ARCHIVE_DEST LOG_ARCHIVE_TRACE

Table 2–6 (Cont.) Setting Standby Database Initialization Parameters

Standby Host	Directory Structure	Recovery Mode	Standby Initialization Parameters
Separate from primary host	Same as primary host	Managed	Same as for manual recovery, except set STANDBY_ARCHIVE_DEST to the same value as LOG_ARCHIVE_DEST
Separate from primary host	Different from primary host	Manual	COMPATIBLE CONTROL_FILES DB_FILE_NAME_CONVERT LOG_ARCHIVE_DEST LOG_ARCHIVE_TRACE LOG_FILE_NAME_CONVERT
Separate from primary host	Different from primary host	Managed	Same as for manual recovery, except set STANDBY_ARCHIVE_DEST to the same value as LOG_ARCHIVE_DEST

Example: Managed Standby Database on Same Host as Primary Database Following are some sample initialization parameter settings for a managed standby database that resides on the same host as primary database PROD1.

```
COMPATIBLE = 8.1.6
CONTROL_FILES = /oracle/work/standby/dbs/cf1.f
DB_FILE_NAME_CONVERT = /oracle/dbs/tbs, /oracle/work/standby/dbs/standby
DB_NAME = prod1
LOCK_NAME_SPACE = stby  # required for a standby on the same host as the primary
LOG_ARCHIVE_DEST = /oracle/work/standby/dbs/arc/
LOG_ARCHIVE_FORMAT = arc_%t_%s.ar
LOG_ARCHIVE_TRACE = 8
LOG_FILE_NAME_CONVERT = /oracle/dbs/t1, /oracle/work/standby/dbs/standby_t1
STANDBY_ARCHIVE_DEST = /oracle/work/standby/dbs/arc/
```

Example: Managed Standby Database on Different Host from Primary Database Following are some sample initialization parameter settings for a managed standby database that resides on a different host from primary database PROD1, but uses the same directory structure.

```
COMPATIBLE = 8.1.6
CONTROL_FILES = /oracle/work/standby/dbs/cf1.f
DB_NAME = prod1
```

```
LOG_ARCHIVE_DEST = /oracle/work/standby/arc_dest/
LOG_ARCHIVE_FORMAT = arc_%t_%s.ar
LOG_ARCHIVE_TRACE = 6
STANDBY_ARCHIVE_DEST = /oracle/work/standby/arc_dest/
```

Example: Manual Standby Database on Different Host from Primary Database Following are some sample initialization parameter settings for a standby database in manual recovery mode that resides on a different host from primary database PROD1, and also uses a different directory structure.

```
COMPATIBLE = 8.1.6
CONTROL_FILES = /oracle/work/standby/cf1.f
DB_FILE_NAME_CONVERT = /oracle, /oracle/work/standby
DB_NAME = prod1
LOG_ARCHIVE_DEST = /oracle/work/arc_dest/
LOG_ARCHIVE_FORMAT = arc_%t_%s.ar
LOG_ARCHIVE_TRACE = 34
LOG_FILE_NAME_CONVERT = /oracle/dbs/t1, /oracle/work/standby/dbs/standby_t1
```

Configuring Parameter and Network Files: Scenario

This scenario assumes the following:

- The primary database PROD1 is on host LOCAL.
- Three standby databases exist: STANDBY1, STANDBY2, and STANDBY3.
- STANDBY1 is on local host LOCAL, STANDBY2 is on a remote host REMOTE2, and STANDBY3 is on remote host REMOTE3.
- Host REMOTE2 has the same directory structure as LOCAL, while REMOTE3 uses a different directory structure from LOCAL.

Settings for PROD1 Initialization Parameter File Following are sample settings for LOG_ARCHIVE_DEST_1, LOG_ARCHIVE_DEST_2, LOG_ARCHIVE_DEST_3, and LOG_ARCHIVE_DEST_4 in the initialization parameter file for the primary database PROD1:

```
# This example specifies a local archiving destination and enables the
# destination.

LOG_ARCHIVE_DEST_1 = 'LOCATION=/arc_dest/'
LOG_ARCHIVE_DEST_STATE_1 = ENABLE

# This example specifies net service name "standby1", makes archiving mandatory,
# and enables the destination.

# A REOPEN value of 5 indicates that if the LOG_ARCHIVE_DEST_2 location
# encounters an error during archival of a redo log file, the destination
# remains inactive until the archival of a redo file is about to begin and 5
# seconds has elapsed. At that time, Oracle again attempts the archival to LOG_
# ARCHIVE_DEST_2.

# If Oracle encounters an error when archiving to a mandatory destination, the
# destination is inactive for the duration of the archival of the current redo
# log file. The destination may be reactivated (based on the REOPEN attribute)
# at the start of the archival of another redo log.

LOG_ARCHIVE_DEST_2 = 'SERVICE=standby1 MANDATORY REOPEN=5'
LOG_ARCHIVE_DEST_STATE_2 = ENABLE

# Specifies net service name "standby2", makes archiving optional, and makes
# Oracle retry archiving after 5 seconds should an error occur. The destination
# is enabled.

LOG_ARCHIVE_DEST_3 = 'SERVICE=standby2 OPTIONAL REOPEN=5'
LOG_ARCHIVE_DEST_STATE_3 = ENABLE

# Specifies net service name "standby3", makes archiving optional, and makes
# Oracle retry archiving after 60 seconds should an error occur. The
# destination is deferred, however, which means that Oracle has currently
# disabled archiving to this destination.

LOG_ARCHIVE_DEST_4 = 'SERVICE=standby3 OPTIONAL REOPEN=60'
LOG_ARCHIVE_DEST_STATE_4 = DEFER
```

TNS Settings Following are settings in the primary host `tnsnames.ora` file for connecting to the standby databases STANDBY1, STANDBY2, and STANDBY3 in the preceding example:

```
# The standby1 database is on the same node as the primary database. Connection
# is made through the IPC protocol.
standby1 = (DESCRIPTION=
             (ADDRESS=
                (PROTOCOL=ipc)
                (KEY=local_standby))
             (CONNECT_DATA=
                (SID=stby1)
                (SERVER=DEDICATED)))
```

```
# The standby2 database is on a different node from the primary database.
# Connection is made through the TCP protocol.
standby2 = (DESCRIPTION=
             (ADDRESS=
                (PROTOCOL=tcp)
                (HOST=remote2)
                (PORT=1512))
             (CONNECT_DATA=
                (SID=stby2)
                (GLOBAL_NAME=standby2)
                (SERVER=DEDICATED)))

# The standby3 database is on a different node from the primary database.
# Connection is made through the TCP protocol.
standby3 = (DESCRIPTION=
             (ADDRESS=
                (PROTOCOL=tcp)
                (HOST=remote3)
                (PORT=1523))
             (CONNECT_DATA=
                (SID=stby3)
                (GLOBAL_NAME=standby3)
                (SERVER=DEDICATED)))
```

Listener Settings Following are the settings in the `listener.ora` files for the standby databases STANDBY1, STANDBY2, and STANDBY3:

```
# The listener settings for standby1 on host local
LISTENER = (ADDRESS_LIST=
   (ADDRESS=
   (PROTOCOL=ipc)
   (KEY=local_standby)))

SID_LIST_LISTENER = (SID_LIST=
 (SID_DESC=(SID_NAME=stby1)(ORACLE_HOME=/oracle))

# The listener settings for standby2 on the remote host remote2
LISTENER = (ADDRESS_LIST=
 (ADDRESS=
   (PROTOCOL=tcp)
   (HOST=remote2)
   (PORT=1512)))

SID_LIST_LISTENER = (SID_LIST=
 (SID_DESC=(SID_NAME=stby2)(ORACLE_HOME=/oracle))
```

```
# The listener settings for standby3 on the remote host remote3
LISTENER = (ADDRESS_LIST=
 (ADDRESS=
   (PROTOCOL=tcp)
   (HOST=remote3)
   (PORT=1523)))

SID_LIST_LISTENER = (SID_LIST=
 (SID_DESC=(SID_NAME=stby3)(ORACLE_HOME=/fs3/oracle))
```

Settings for STANDBY1, STANDBY2, and STANDBY3 Initialization Parameter Files Following are settings in the initialization parameter files for the standby databases STANDBY1, STANDBY2, and STANDBY3 in the preceding example. These settings determine the filenames on the standby database for the archived redo logs:

```
# The init.ora values for the standby1 database, which is on the same host as
# the primary database
STANDBY_ARCHIVE_DEST = /oracle/standby/arc/
LOG_ARCHIVE_DEST = /oracle/standby/arc/ # the location is the same as
                                        # STANDBY_ARCHIVE_DEST
LOG_ARCHIVE_FORMAT = log%t_%s.arc
LOG_ARCHIVE_TRACE = 16
LOCK_NAME_SPACE = foo_bar
DB_FILE_NAME_CONVERT = /oracle/dbs, /oracle/standby/dbs
LOG_FILE_NAME_CONVERT = /oracle/dbs, /oracle/standby/dbs

# The init.ora values for the standby2 database, which is on
# host remote2. Host remote2 uses the same directory structure as host local.
STANDBY_ARCHIVE_DEST = /oracle/standby/arc/
LOG_ARCHIVE_DEST = /oracle/standby/arc/ # the location is the same as
                                        # STANDBY_ARCHIVE_DEST
LOG_ARCHIVE_FORMAT = log%t_%s.arc
LOG_ARCHIVE_TRACE = 16

# The init.ora values for the standby3 database, which is on host remote3. Host
# remote3 uses a different directory structure from host local.
STANDBY_ARCHIVE_DEST = /fs3/arc_dest/
LOG_ARCHIVE_DEST = /fs3/arc_dest/   # the location is the same as
                                    # STANDBY_ARCHIVE_DEST
LOG_ARCHIVE_FORMAT = log%t_%s.arc
DBS_FILE_NAME_CONVERT = /oracle, /fs3/oracle
LOG_FILE_NAME_CONVERT = /oracle, /fs3/oracle
```

Starting the Standby Instance in Preparation for Recovery

The final stage of standby database preparation is starting the standby database instance so you can begin manual or managed recovery. This stage involves three tasks:

1. Starting the Standby Instance
2. Manually Renaming Standby Files Not Captured by Conversion Parameters
3. Enabling Changes to the Initialization Parameter Settings

Starting the Standby Instance

After all necessary parameter and network files have been configured, you can start the standby instance. Note that if the instance is not started, the standby database cannot receive archived redo logs that are automatically transmitted to the standby site by the primary database.

To start the standby instance:

1. Use SQL*Plus to connect to the standby database instance. For example, enter:

   ```
   SQL> CONNECT sys/change_on_install@standby1
   ```

2. Start the Oracle instance at the standby database without mounting the database. For example, enter:

   ```
   SQL> STARTUP NOMOUNT pfile=initSTANDBY.ora;
   ```

3. Mount the standby database:

   ```
   SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
   ```

Manually Renaming Standby Files Not Captured by Conversion Parameters

Sometimes all of the primary datafiles and redo log files cannot be renamed in the standby control file by conversion parameters. For example, assume that your database has the following datafiles, which you want to rename as shown in the following table:

Primary Filename	Standby Filename
/oracle/dbs/df1.f	/standby/df1.f
/oracle/dbs/df2.f	/standby/df2.f

Primary Filename	Standby Filename
/data/df3.f	/standby/df3.f

You can set DB_FILE_NAME_CONVERT as follows to convert the filenames for the first two datafiles:

```
DB_FILE_NAME_CONVERT = '/oracle/dbs', '/standby'
```

Nevertheless, this parameter will not capture the renaming of `/data/df3.f`. You must rename this datafile manually in the standby database control file by issuing a SQL statement as follows:

```
SQL> ALTER DATABASE RENAME FILE '/data/df3.f' to '/standby/df3.f';
```

To rename a datafile manually:

1. Using SQL*Plus, start the standby instance (if it is not already started) and then mount the database:

   ```
   SQL> STARTUP NOMOUNT pfile=initSTANDBY1.ora;
   SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
   ```

2. Issue an ALTER DATABASE statement for each datafile requiring renaming, where *old_name* is the old name of the datafile as recorded in the control file and *new_name* is the new name of the datafile that will be recorded in the standby control file:

   ```
   SQL> ALTER DATABASE RENAME FILE 'old_name' TO 'new_name';
   ```

When you manually rename all of the datafiles that are not captured by the DB_FILE_NAME_CONVERT parameter, the standby database control file can correctly interpret the log stream during the recovery process.

Enabling Changes to the Initialization Parameter Settings

If you configured the primary initialization parameter file to archive to the standby site, you should enable these new parameter settings *after* starting the standby instance and the listener on the standby site.

You can make changes to the primary database initialization parameter file while the database is open, but the changes only take effect when the instance is restarted. If the database is open and you want to avoid restarting it, enable the parameter changes dynamically using ALTER SYSTEM statements.

For example, assume that you made the following changes to the initialization parameter file while the database was open:

```
LOG_ARCHIVE_DEST_1="LOCATION=/arc_dest/ MANDATORY REOPEN=2";
LOG_ARCHIVE_DEST_2="SERVICE=stby1 MANDATORY REOPEN=2";
LOG_ARCHIVE_DEST_STATE_1=ENABLE;
LOG_ARCHIVE_DEST_STATE_2=ENABLE;
LOG_ARCHIVE_MIN_SUCCEED_DEST=2;
```

You can then connect to the primary database using SQL*Plus and issue ALTER SYSTEM statements as follows to enable these settings:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_1="LOCATION=/arc_dest/ MANDATORY REOPEN=2";
ALTER SYSTEM SET LOG_ARCHIVE_DEST_2="SERVICE=stby1 MANDATORY REOPEN=2';
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_1=ENABLE;
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;
ALTER SYSTEM SET LOG_ARCHIVE_MIN_SUCCEED_DEST=2;
```

Once you have enabled these changes, the primary database can start archiving redo logs to a standby database.

Troubleshooting the Standby Database Configuration

If you encounter a problem during standby database preparation, it will probably be one of the following:

- The Standby Site Does Not Receive Logs Archived by the Primary Database
- You Cannot Mount the Standby Database

The Standby Site Does Not Receive Logs Archived by the Primary Database

If the standby site is not receiving the logs, the first thing you should do is obtain information about the archiving status of the primary database by querying the V$ARCHIVE_DEST view. Check especially for error messages. For example, enter:

```
SQL> SELECT dest_id "ID",
  2> status "DB_status",
  3> destination "Archive_dest",
  4> error "Error"
  5> FROM v$archive_dest;

ID DB_status Archive_dest                   Error
-- --------- ------------------------------
--------------------------------------
```

```
 1  VALID     /vobs/oracle/work/arc_dest/arc
 2  ERROR     standby1         ORA-16012: Archivelog standby database
identifier mismatch
 3  INACTIVE
 4  INACTIVE
 5  INACTIVE
5 rows selected.
```

If the output of the query does not help you, check the following list of possible issues. If any of the following conditions is not met, the primary database will fail to archive to the standby site:

- The service name for the standby instance is not configured correctly in the `tnsnames.ora` file at the primary site.
- The service name listed in the LOG_ARCHIVE_DEST_*n* parameter of the primary initialization parameter file is incorrect.
- The LOG_ARCHIVE_DEST_STATE_*n* parameter specifying the state of the standby archiving destination has the value DEFER.
- The `listener.ora` file has not been configured correctly at the standby site.
- The listener is not started.
- The standby instance is not started.
- You have added a standby archiving destination to the primary initialization parameter file, but have not yet enabled the change.
- You used an invalid backup as the basis for the standby database (for example, you used a backup from the wrong database, or did not create the standby control file using the correct method).

You Cannot Mount the Standby Database

If any of the following conditions is not met, you cannot mount the standby database:

- The standby instance is not started in NOMOUNT mode. You must first start the instance and *then* mount the database.
- The standby control file was not created with the ALTER DATABASE CREATE STANDBY CONTROLFILE ... statement. You cannot use the following types of control file backups:
  - An RMAN-created backup

- An operating system-created backup
- A backup created using an ALTER DATABASE statement *without* the STANDBY option

3 Managing a Standby Database

This chapter describes how to manage a standby database. It includes the following topics:

Choosing Standby Database Modes

As explained in Standby Database Modes on page 1-7, you can run the standby database in the following mutually exclusive modes:

- Managed recovery mode
- Manual recovery mode
- Read-only mode

Before running in these modes, you must first start the standby instance and then mount the database. The sections in this chapter describe the procedures for initiating the various modes as well as for performing failover to a standby database.

Which Modes Are Typical in a Standby Environment?

Typically, you create a standby database for one or more of the following reasons:

- Protection against the total destruction of the primary database
- Protection against application corruption of the primary database
- Supplemental reporting of data contained in the primary database

If you want maximum protection against data loss or corruption, then maintain the standby database in managed recovery mode in a managed standby environment. In this setup, the primary database archives logs to the standby site, and the standby database automatically applies these logs.

Note: You may need to apply archived logs manually to the standby database before managed recovery. To learn why, see Resolving a Gap Sequence Before Initiating Managed Recovery on page 3-5.

If you want to use the standby database for reporting purposes, then open it in read-only mode in a managed standby environment. Oracle cannot apply archived redo logs to the standby database when it is in this mode, but you can still execute queries on the database. The primary database continues to archive to the standby site so long as the standby instance is started.

You can easily switch back and forth between managed recovery mode and read-only mode. In most implementations of a managed standby environment, you need to perform this switch at various times to either:

- Update a standby database used primarily for reporting
- Check that data is correctly applied to a database that is used primarily for disaster protection

See Also: To learn how to initiate managed recovery, see Placing the Standby Database in Managed Recovery Mode on page 3-13. To learn how to open a standby database in read-only mode, see Opening a Standby Database in Read-Only Mode on page 3-17.

When Is Manual Recovery Required?

Manual recovery mode is required in the following cases:

Managed Standby Environment?	Reason for Manual Recovery
Yes	You must resolve a gap sequence; that is, you must put the standby database into a state in which you can initiate managed recovery (see Resolving a Gap Sequence Before Initiating Managed Recovery on page 3-5).
No	Managed recovery is not an option (see Non-Managed Recovery Environment on page 1-22), so you must manually transfer logs to the standby site and manually apply them to the standby database.

Consequently, even if you implement a managed standby environment, you may occasionally need to perform manual recovery on the standby database.

See Also: To learn how to perform manual recovery of a standby database, see Placing the Standby Database in Manual Recovery Mode on page 3-3.

Placing the Standby Database in Manual Recovery Mode

After you have started and mounted the standby database, you can place it in manual recovery mode. To keep the standby database current, you must manually apply archived redo logs from the primary database to the standby database.

Archived logs arrive at the standby site in one of the following ways:

- The primary database automatically archives the logs (only if you implement a managed standby environment).

- You manually transfer logs using an operating system utility or some other means.

The standby database assumes that the archived log file group is in the location specified by either of the following parameters in the standby initialization parameter file:

- First valid disk location specified by LOG_ARCHIVE_DEST_*n* (where *n* is an integer from 1 to 5)
- Location specified by LOG_ARCHIVE_DEST

If the archived logs are not in the location specified in the initialization parameter file, you can specify an alternative location using the FROM option of the RECOVER statement.

To place the standby database in manual recovery mode:

1. Use SQL*Plus to connect to the standby instance and then start the Oracle instance at the standby database. For example, enter:

   ```
   STARTUP NOMOUNT pfile=initSTANDBY.ora
   ```

2. Mount the standby database:

   ```
   ALTER DATABASE MOUNT STANDBY DATABASE;
   ```

3. If Oracle is not archiving logs automatically to the standby site, then manually transfer the logs to the desired location on the standby host using an appropriate operating system utility for transferring binary data. For example, enter:

   ```
   % cp /oracle/arc_dest/*.arc /standby/arc_dest
   ```

4. Issue a RECOVER statement to place the standby database in manual recovery mode.

   Note: Specify the FROM *'location'* option only if the archived log group is *not* in the location specified by the LOG_ARCHIVE_DEST_*n* (where *n* is an integer from 1 to 5) or LOG_ARCHIVE_DEST parameter in the standby initialization parameter file.

   For example, execute one of the following statements:

   ```
   RECOVER STANDBY DATABASE # uses location for logs specified in
   ```

```
                                        # initialization parameter file
RECOVER FROM '/logs' STANDBY DATABASE # specifies nondefault location
```

As Oracle generates archived redo logs, you must continually transfer and apply them to the standby database to keep it current.

Resolving a Gap Sequence Before Initiating Managed Recovery

A *gap sequence* is a range of archived redo logs needed by the standby database before it can enter managed recovery mode. A standby database is able to begin managed recovery when you can apply the next archived log generated by the primary database to the standby database in managed recovery mode. This section contains the following topics:

- What Causes Gap Sequences?
- Determining Whether a Gap Sequence Exists
- Transmitting the Logs in the Gap Sequence to the Standby Site
- Applying the Logs in the Gap Sequence to the Standby Database

What Causes Gap Sequences?

A gap sequence can occur whenever the primary database archives a log but the log is not transferred to the standby site. Because the standby database requires the sequential application of redo logs, a missing log prevents managed recovery from applying subsquent logs.

Gap sequences can occur in the following situations:

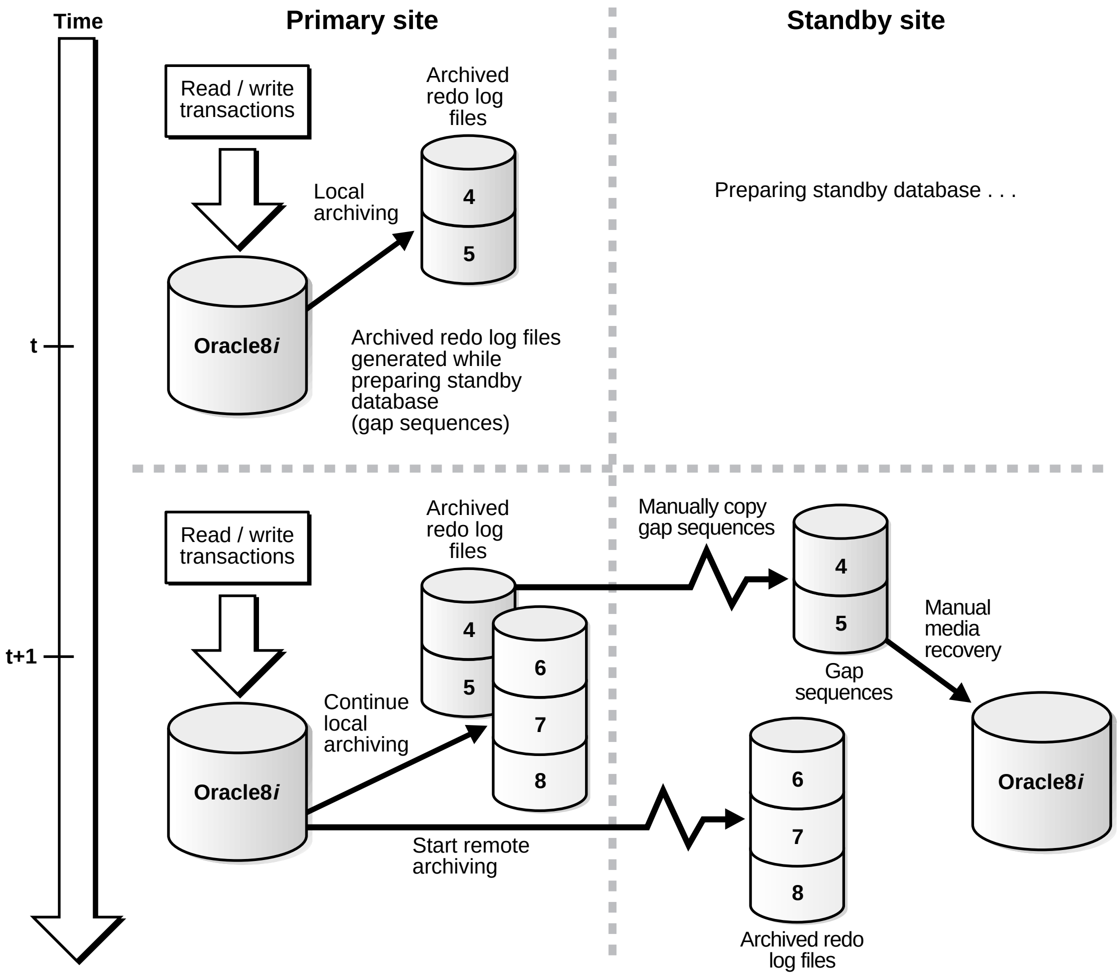
- Creation of the Standby Database
- Shutdown of the Standby Database When the Primary Database Is Open
- Network Failure Preventing the Transfer of Logs to the Standby Site

Creation of the Standby Database

One example of a gap sequence occurs when you create the standby database from an old backup. For example, if the standby database is made from a backup that contains changes through log 100, and the primary database currently contains changes through log 150, then the standby database requires that you manually apply logs 101 to 150 before managed recovery can begin.

Another typical example of a gap sequence occurs when you generate the standby database from a hot backup of an open database. For example, assume the scenario illustrated in Figure 3–1.

Figure 3–1 Manual Recovery of Archived Logs in a Gap Sequence



The following steps occur:

1. You take a hot backup of database PRIMARY.

2. At time *t*, while you are busy configuring the network files, PRIMARY archives logs 4 and 5.
3. At time *t + 1*, you start the standby instance.
4. PRIMARY archives logs 6, 7, and 8 to both the primary site and the standby site.

Archived logs 4 and 5 are now part of a gap sequence; that is, you must apply them manually to the standby database before managed recovery can apply archived logs 6, 7, and 8 to the standby database.

Shutdown of the Standby Database When the Primary Database Is Open

You may be required to shut down the standby database to resolve maintenance issues. For example, you must shut down the standby database in the following scenarios:

- You change a control file parameter, for example, MAXDATAFILE, in the primary database.
- You issue an ALTER DATABASE CLEAR UNARCHIVED LOGFILES statement.

CAUTION: Performing a RESETLOGS operation on the primary database sinvalidates the standby database. If you reset the logs on the primary database, you must rebuild the standby database.

To avoid creating gap sequences, follow these rules:

- Start the standby databases and listeners *before* starting the primary database.
- Shut down the primary database *before* shutting down the standby database.

If you violate either of these two rules, then the standby database is down while the primary database is open and archiving. Consequently, Oracle can create a gap sequence. When you restart the standby database later, you must synchronize the standby database manually with the primary database before you can initiate managed recovery.

Note: If the standby site is specified as MANDATORY in one of the LOG_ARCHIVE_DEST_*n* parameters of the primary initialization parameter file (see Specifying Mandatory and Optional Archive Destinations on page 2-17), dynamically change it to OPTIONAL before shutting down the standby database. Otherwise, the primary database eventually stalls because it cannot archive its online redo logs.

Network Failure Preventing the Transfer of Logs to the Standby Site

If you maintain a managed standby environment, and the network goes down, the primary database may continue to archive to disk but be unable to archive to the standby site. In this situation, archived logs accumulate as usual on the primary site, but the standby instance is unaware of them.

To prevent this problem, you can specify that the standby destination have mandatory status. If the archiving destination is mandatory, then the primary database will not archive any logs until it is able to archive to the standby site. For example, you can set the following in the primary initialization parameter file to make STANDBY1 a mandatory archiving destination:

```
LOG_ARCHIVE_DEST_2 = 'SERVICE=standby1 MANDATORY'
```

One consequence of this configuration is that unless the network problem is fixed, the primary database eventually stalls because it cannot switch into an unarchived online redo log. This problem is exacerbated if you maintain only two online redo logs in your primary database.

See Also: For a detailed account of the significance of the OPTIONAL and MANDATORY options for standby archival, see the chapter on archived redo logs in the *Oracle8i Administrator's Guide*. See Scenario 7: Recovering After a Network Failure on page 5-40 for additional information.

Determining Whether a Gap Sequence Exists

To determine whether there is a gap sequence, execute the SQL script in the following procedure. If there is a gap sequence, the output of the query specifies the thread number and log sequence number of all logs in the gap sequence. If there is *no* gap sequence for a given thread, the query returns either no rows or an identical number in the LowSeq# and HighSeq# columns.

To identify the logs in the gap sequence:

1. Start SQL*Plus and mount the standby database:

   ```
   SQL> CONNECT sys/sys_pwd@standby1 AS SYSDBA
   SQL> STARTUP NOMOUNT pfile=/oracle/admin/pfile/init.ora
   SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
   ```

2. Copy the following SELECT statement into a script and then run the script on the standby database:

   ```
   SELECT high.thread#, "LowGap#", "HighGap#"
   FROM
        (
        SELECT thread#, MIN(sequence#)-1 "HighGap#"
        FROM
        (
            SELECT a.thread#, a.sequence#
            FROM
            (
                SELECT *
                FROM v$archived_log
            ) a,
            (
                SELECT thread#, MAX(next_change#)gap1
                FROM v$log_history
                GROUP BY thread#
            ) b
            WHERE a.thread# = b.thread#
            AND a.next_change# > gap1
        )
        GROUP BY thread#
    ) high,

    (
        SELECT thread#, MIN(sequence#) "LowGap#"
        FROM
        (
            SELECT thread#, sequence#
            FROM v$log_history, v$datafile
            WHERE checkpoint_change# <= next_change#
            AND checkpoint_change# >= first_change#
        )
        GROUP BY thread#
    ) low
    WHERE low.thread# = high.thread#;
   ```

3. Examine the output of the query to determine the gap sequence. For example, the output may look like:

   ```
   SQL> @gap
   THREAD#    LowSeq#    HighSeq#
   ---------- ---------- ----------
            1 460        463
            2 202        204
            3 100        100
   ```

   Not every thread has a gap sequence. As this example illustrates, the LowSeq# and HighSeq# for thread 3 are identical, so no gap sequence exists for this thread.

   You must apply the logs in the gap sequence for each thread to the standby database to prepare it for managed recovery.

   See Also: To learn how to perform manual recovery, see Placing the Standby Database in Manual Recovery Mode on page 3-3.

Transmitting the Logs in the Gap Sequence to the Standby Site

After you have obtained the log sequence numbers of the logs in the gap sequence, you can obtain their filenames by querying the V$ARCHIVED_LOG view on the primary site (see Determining Which Archived Logs Have Been Received by the Standby Site on page 4-5). The archived log filenames on the standby site are generated by the STANDBY_ARCHIVE_DEST and LOG_ARCHIVE_FORMAT parameters in the standby initialization parameter file.

If the standby database is on the same host as the primary database, or the standby database is on a remote host with a different directory structure from the primary database, the filenames for the logs on the standby site cannot be the same as the filenames of the logs archived by the primary database. Before transmitting the archived logs to the standby site, determine the correct filenames for the logs at the standby site.

To transmit logs in a gap sequence to the standby site:

1. Review the list of gap sequence logs that you obtained earlier. For example, assume you have the following gap sequence:

   ```
   THREAD#    LowSeq#    HighSeq#
   ---------- ---------- ----------
            1 460        463
            2 202        204
   ```

```
        3   100        100
```

Note that no gap sequence exists for thread 3, so you only need to transmit logs from threads 1 and 2.

2. Determine the filenames of the logs in the gap sequence that were archived by the primary database. For example, after connecting to the primary database using SQL*Plus, issue a SQL query to obtain the name of a log in each thread:

```
SQL> CONNECT sys/sys_pwd@primary
SQL> SELECT name
  2> FROM v$archived_log
  3> WHERE sequence# in (460, 202);

NAME
----------------------------------------------------------------------------
/primary/thread1_dest/arcr_1_460.arc
/primary/thread2_dest/arcr_2_202.arc
2 rows selected.
```

3. Review the settings for STANDBY_ARCHIVE_DEST and LOG_ARCHIVE_FORMAT in the standby initialization parameter file. For example, you discover the following:

```
STANDBY_ARCHIVE_DEST = /standby/arc_dest/
LOG_ARCHIVE_FORMAT = log_%t_%s.arc
```

These parameter settings determine the filenames of the archived redo logs at the standby site.

4. Transfer the gap sequence logs from the primary site to the standby site, renaming them according to values for STANDBY_ARCHIVE_DEST and LOG_ARCHIVE_FORMAT. For example, enter:

```
% cp /primary/thread1_dest/arcr_1_460.arc /standby/arc_dest/log_1_460.arc
% cp /primary/thread1_dest/arcr_1_461.arc /standby/arc_dest/log_1_461.arc
% cp /primary/thread1_dest/arcr_1_462.arc /standby/arc_dest/log_1_462.arc
% cp /primary/thread1_dest/arcr_1_463.arc /standby/arc_dest/log_1_463.arc

% cp /primary/thread1_dest/arcr_2_202.arc /standby/arc_dest/log_2_202.arc
% cp /primary/thread1_dest/arcr_2_203.arc /standby/arc_dest/log_2_203.arc
% cp /primary/thread1_dest/arcr_2_204.arc /standby/arc_dest/log_2_204.arc
```

5. If the LOG_ARCHIVE_DEST and STANDBY_ARCHIVE_DEST parameter values are *not* the same, then copy the gap sequence logs from the STANDBY_

ARCHIVE_DEST directory to the LOG_ARCHIVE_DEST directory. If these parameter values *are* the same, then you do not need to perform this step.

For example, assume the following standby initialization parameter settings:

```
STANDBY_ARCHIVE_DEST = /standby/arc_dest/
LOG_ARCHIVE_DEST = /log_dest/
```

Because the parameter values are different, copy the archived logs to the LOG_ARCHIVE_DEST location:

```
% cp /standby/arc_dest/* /log_dest/
```

When you initiate manual recovery, Oracle looks at the LOG_ARCHIVE_DEST value to determine the location of the logs.

Now that all required logs are in the STANDBY_ARCHIVE_DEST directory, you can proceed to the next stage: applying the gap sequence logs to the standby database.

Applying the Logs in the Gap Sequence to the Standby Database

After you have transmitted the logs in the gap sequence to the standby site, you can apply them using the RECOVER AUTOMATIC statement. Once applied, you can go on to place the standby database in managed recovery mode.

To apply the archived redo logs in the gap sequence:

1. Start SQL*Plus and mount the standby database (if it is not already mounted). For example, enter:

```
SQL> CONNECT sys/sys_pwd@standby1 AS SYSDBA
SQL> STARTUP NOMOUNT pfile=/oracle/admin/pfile/initSTBY.ora
SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
```

2. Recover the database using the AUTOMATIC option:

```
SQL> RECOVER AUTOMATIC STANDBY DATABASE
```

After recovering the available logs, Oracle prompts for the name of a log that does not exist. The reason is that the recovery process does not know about the logs archived to the standby site by the primary database. For example, you might see:

```
ORA-00308: cannot open archived log '/oracle/standby/standby_logs/arcr_1_
540.arc'
```

```
ORA-27037: unable to obtain file status
SVR4 Error: 2: No such file or directory
Additional information: 3
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
```

3. Cancel recovery after Oracle has applied the available logs by executing the following statement (or typing CTRL+C):

   ```
   SQL> CANCEL
   Media recovery cancelled.
   ```

   Note that the following error messages are acceptable after recovery cancellation and do not indicate a problem:

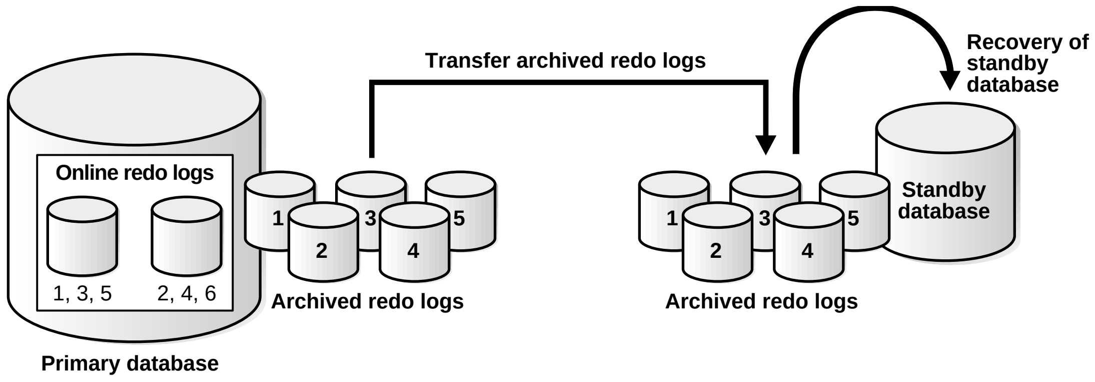
   ```
   ORA-01547: warning: RECOVER succeeded but OPEN RESETLOGS would get error
   below
   ORA-01194: file 1 needs more recovery to be consistent
   ORA-01110: data file 1: 'some_filename'
   ORA-01112: media recovery not started
   ```

You are now able to initiate managed recovery.

Placing the Standby Database in Managed Recovery Mode

If you implement a managed standby environment, you can automate archiving to either a local or remote host. Oracle keeps the standby database synchronized with the primary database by waiting for archived logs from the primary database and then automatically applying them to the standby database, as shown in Figure 3–2. This feature eliminates the need to interactively provide the recovery process with the filenames of the archived logs.

Figure 3–2 Transmitting and Applying Archived Redo Logs to a Standby Database



Initiating Managed Recovery Mode

Enable managed recovery using the following SQL*Plus statement:

```
RECOVER MANAGED STANDBY DATABASE
```

You can use the TIMEOUT option of the RECOVER statement to specify an optional *timeout interval*. In this case, the managed recovery operation waits the specified number of minutes for Oracle to write the requested archived log entry to the standby control file's directory.

If Oracle times out because it cannot find the required next log entry in the standby control file, the system issues an appropriate message and exits managed recovery mode. By default, the managed recovery operation waits indefinitely for the next archived log; it terminates only through a CANCEL statement (or CTRL+C key combination), a shutdown, or a crash.

To place the standby database in managed recovery mode:

1. After connecting to the standby database, start the standby database without mounting it, specifying a parameter file if necessary:

   ```
   STARTUP NOMOUNT pfile=initSTANDBY.ora
   ```

2. Mount the database:

   ```
   ALTER DATABASE MOUNT STANDBY DATABASE;
   ```

3. Ensure that the standby database is synchronized; that is, no gap sequence exists. See Determining Whether a Gap Sequence Exists on page 3-8 to determine whether there is a gap sequence. If there is, manually synchronize the database. If there is not, proceed to the next step.

4. Place the standby database in managed recovery mode:

   ```
   RECOVER MANAGED STANDBY DATABASE
   ```

   Note: The RECOVER MANAGED STANDBY DATABASE statement does not permit a FROM '*location*' option.

   If you want to use the optional timeout option, add TIMEOUT *integer* to the command syntax:

   ```
   RECOVER MANAGED STANDBY DATABASE TIMEOUT 60
   ```

Oracle now begins managed recovery. As the primary database archives redo logs to the standby site, the standby database automatically applies them.

Note: After you execute the RECOVER statement, the prompt sits on the line following the RECOVER statement; this is the expected behavior.

To quickly test whether managed recovery is applying the transferred archived redo logs:

Several methods exist for determining the status of archived redo logs, as explained in Troubleshooting the Standby Database Configuration on page 2-33. The following procedure forces archiving at the primary site, which allows you to query a view on the standby database to determine whether the archived log was applied.

1. Connect to the primary database and make sure it is open.

   ```
   % sqlplus sys/sys_pwd@primary
   SQL> SELECT status FROM v$instance;

   STATUS
   -------
   OPEN
   ```

2. Archive the current online redo log:

   ```
   ALTER SYSTEM ARCHIVE LOG CURRENT;
   ```

```
System altered.
```

3. Query the primary database to determine the most recently archived redo log:

```
SQL> SELECT max(sequence#) FROM v$log_history;

MAX(SEQUENCE#)
--------------
           541
```

4. Create a new session on the standby instance and query the V$LOG_HISTORY view:

```
% sqlplus sys/sys_pwd@standby1
SQL> SELECT max(sequence#) FROM v$log_history;

MAX(SEQUENCE#)
--------------
           541
```

The sequence number should be the same as the number on the primary site. If it is not, wait a short time for Oracle to finish receiving and applying the log and try again. If Oracle still does not apply the log, see Troubleshooting the Standby Database Configuration on page 2-33 for ways of obtaining troubleshooting information.

Canceling Managed Recovery

Cancel the managed recovery operation at any time by issuing either of the following SQL*Plus statements:

```
RECOVER MANAGED STANDBY DATABASE CANCEL
RECOVER MANAGED STANDBY DATABASE CANCEL IMMEDIATE
```

The first statement waits for the managed recovery operation to finish with the current redo log before terminating recovery. If you use the IMMEDIATE option, however, Oracle stops the managed recovery operation either before reading another block from the redo log or before opening the next redo log file, whichever comes first. Note the following scenarios:

If you cancel recovery	Then
Before recovery opens the next redo log	CANCEL IMMEDIATE is equivalent to CANCEL.

If you cancel recovery	Then
While the standby database is processing a redo log	CANCEL IMMEDIATE leaves the database in an inconsistent state. Oracle does not allow a database to be opened in an inconsistent state, although you can still initiate manual or managed recovery.

Opening a Standby Database in Read-Only Mode

The read-only mode allows users to open and query a standby database without the potential for online data modifications. This functionality enables you to reduce system overhead on the primary database by using the standby database for reporting purposes. Also, you can periodically open the standby database in read-only mode to ensure that a managed standby database is being updated properly.

This section contains the following topics:

- Considering Whether to Run in Read-Only Mode
- Receiving Archived Redo Logs While in Read-Only Mode
- Opening the Database in Read-Only Mode
- Creating Temporary Tablespaces

Considering Whether to Run in Read-Only Mode

When determining whether to run the standby database in read-only mode, consider the following:

- If you maintain the standby database primarily for disaster protection, then you should not rely too heavily on your standby database as a source of information. If a disaster does occur, then you will have to activate the standby database quickly and immediately cease all user activity.
- Using a standby database for queries makes it unavailable for managed recovery. At some point, you need to run a recovery operation against the standby database to resynchronize it with the primary database. This action limits the role of the standby database as a disaster recovery database.

If you need the standby database both for disaster prevention and reporting, then you can maintain multiple standby databases, some read-only and some in managed recovery mode. You will need to resynchronize the read-only database, but the recovery mode databases give you protection against disaster.

Receiving Archived Redo Logs While in Read-Only Mode

While the standby database is in read-only mode, the site can still receive archived redo logs from the primary site. Nevertheless, Oracle does not apply these logs automatically, as in managed recovery. Consequently, a read-only standby database is not synchronized with the primary database at the archive level. You should not activate the standby database in a failover situation unless all archived redo logs have been applied.

Opening the Database in Read-Only Mode

The following states are possible for a standby database:

- Shutdown
- Manual recovery mode
- Managed recovery mode
- Read-only mode

You can move from any of the first three states into read-only mode (and back again) using the following procedures.

To open the standby database in read-only mode when the database is shut down:

1. Use SQL*Plus to start the Oracle instance for the standby database without mounting it:

   ```
   SQL> STARTUP NOMOUNT pfile=initSTANDBY.ora
   ```

2. Mount the standby database:

   ```
   SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
   ```

3. Open the database in read-only mode:

   ```
   SQL> ALTER DATABASE OPEN READ ONLY;
   ```

To open the standby database in read-only mode when in manual recovery mode:

1. Cancel the recovery by entering the following (terminate the flow of archived redo logs to get the prompt):

   ```
   SQL> RECOVER CANCEL
   ```

2. Open the database in read-only mode:

```
SQL> ALTER DATABASE OPEN READ ONLY;
```

To open the standby database in read-only mode when in managed recovery mode:

1. Start a SQL*Plus session and execute the following statement:

   ```
   SQL> RECOVER MANAGED STANDBY DATABASE CANCEL
   ```

2. Open the database in read-only mode:

   ```
   SQL> ALTER DATABASE OPEN READ ONLY;
   ```

To move the standby database from read-only mode back to managed recovery mode:

1. Terminate all active user sessions on the standby database.
2. Issue the following statement:

   ```
   SQL> RECOVER MANAGED STANDBY DATABASE # you can also set the TIMEOUT option
   ```

To move the standby database from read-only mode back to manual recovery mode:

1. Terminate all active user sessions on the standby database.
2. Issue the following statement:

   ```
   SQL> RECOVER STANDBY DATABASE # you can also set the TIMEOUT option
   ```

Creating Temporary Tablespaces

In order to perform queries on a read-only standby database, the Oracle database server needs to be able to perform on-disk sorting operations. You cannot allocate space for sorting operations in tablespaces that cause Oracle to write to the data dictionary, however, for these modifications cause the standby control file to diverge from the primary control file.

Temporary tablespaces allow you to add tempfile entries in read-only mode for the purposes of making queries. You can then perform on-disk sorting operations in an Oracle8*i* read-only database without affecting dictionary files or generating redo entries.

Note the following requirements for creating temporary tablespaces:

- The tablespaces must be temporary, locally managed, and contain only tempfiles.

- User-level allocations and permissions to use the locally managed temporary tablespaces must be in place on the primary database. You cannot change these settings on the standby database.

You should also follow these guidelines:

- Minimize data validation time on the standby database so that you can switch into managed or manual recovery mode when necessary.
- Minimize runtimes for reports.
- Implement desired optimizations on the primary database only.

> **See Also:** For more information about using tempfiles and temporary tablespaces, see the *Oracle8i Administrator's Guide*.

To create a temporary tablespace for use on a read-only standby database:

1. Open the standby database in read-only mode using the relevant procedure described in Opening the Database in Read-Only Mode on page 3-18.
2. Create a temporary tablespace. For example, enter:

```
SQL> CREATE TEMPORARY TABLESPACE tbs_1 TEMPFILE 'file_1.f'
     EXTENT MANAGEMENT LOCAL UNIFORM SIZE 16M;
```

> **See Also:** For CREATE TEMPORARY TABLESPACE syntax, see the *Oracle8i SQL Reference*.

Activating a Standby Database

You should not activate or perform a *failover* to the standby database except in an emergency. After it is activated, the standby database becomes a normal production database and loses its standby functionality. You cannot undo the activation and return the database to standby mode.

> **Note:** You should not activate a standby database to test whether it is being updated correctly. Open it in read-only mode instead.

Depending on the nature of the emergency, you may not have access to your primary database files. If you do have access, then you should attempt to archive the current online redo log on the primary database manually, and then transfer and apply all available archived redo logs to the standby database.

To attempt to salvage the primary database redo logs:

1. If possible, archive the current online redo log on the primary database:

   ```
   SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
   ```

2. If you do *not* maintain a managed standby environment, manually transfer to the standby site all available archived logs that have not yet been applied to the standby database. Use an appropriate operating system utility for transferring binary data. For example, enter:

   ```
   % cp /oracle/arc_dest/*.arc /standby/arc_dest
   ```

3. If the standby database *is* currently in managed recovery mode, cancel recovery:

   ```
   SQL> CANCEL
   ```

   If the standby database is *not* currently in managed recovery mode, put it in a mounted state. For example, if the standby instance is not started, enter:

   ```
   SQL> STARTUP NOMOUNT pfile=initSTANDBY.ora
   SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
   ```

4. Perform manual recovery on the standby database *before* activating it, optionally specifying the FROM '*location*' option (see Placing the Standby Database in Manual Recovery Mode on page 3-3). For example, enter one of the following statements:

   ```
   SQL> RECOVER AUTOMATIC STANDBY DATABASE # uses location for logs listed in
   initialization parameter file
   SQL> RECOVER AUTOMATIC FROM '/logs' STANDBY DATABASE # specifies nondefault
   location
   ```

Following this procedure rolls forward the standby database to the time immediately before the failure of the primary database. You can apply any redo log other than the current redo log to the standby database. If you have lost your noncurrent online redo logs and they have not been archived, then activate the standby database without recovering the transactions from the unarchived redo logs of the primary database.

Activating the standby database automatically resets the online redo logs. Note that the redo logs from the standby database and primary database are now incompatible. You cannot apply archived redo logs from the original primary database to the activated standby database or vice versa. Also, the standby database is not mounted when activated; therefore, the tables and views do not contain useful information immediately after activation.

To activate a standby database:

1. Ensure that your standby database is mounted in EXCLUSIVE mode by executing the following query:

   ```
   SQL> SELECT name,value FROM v$parameter WHERE name='parallel_server';
   NAME                   VALUE
   -----------------------------------------------------------------
   parallel_server          FALSE
   1 row selected.
   ```

   If the value is TRUE, then the database is *not* mounted exclusively; if the value is FALSE, then the database is mounted exclusively.

2. Activate the standby database:

   ```
   SQL> ALTER DATABASE ACTIVATE STANDBY DATABASE;
   ```

   Note that this command resets the online redo logs.

3. Shut down the standby instances:

   ```
   SQL> SHUTDOWN IMMEDIATE
   ```

4. As soon as possible, back up your new production database. At this point, the former standby database is now your production database. This task, while not required, is a recommended safety measure because you cannot recover changes made after activation without a backup.

5. Start the new production instance in read / write or read-only mode:

   ```
   SQL> STARTUP MOUNT
   SQL> ALTER DATABASE READ ONLY; # opens the database in read-only mode
   SQL> ALTER DATABASE READ WRITE; # opens the database in read/write mode
   ```

Note: After you activate the standby database, you lose all transactions from unarchived logs at your original production database. Because the standby redo logs are reset at activation, you cannot apply logs archived before activation.

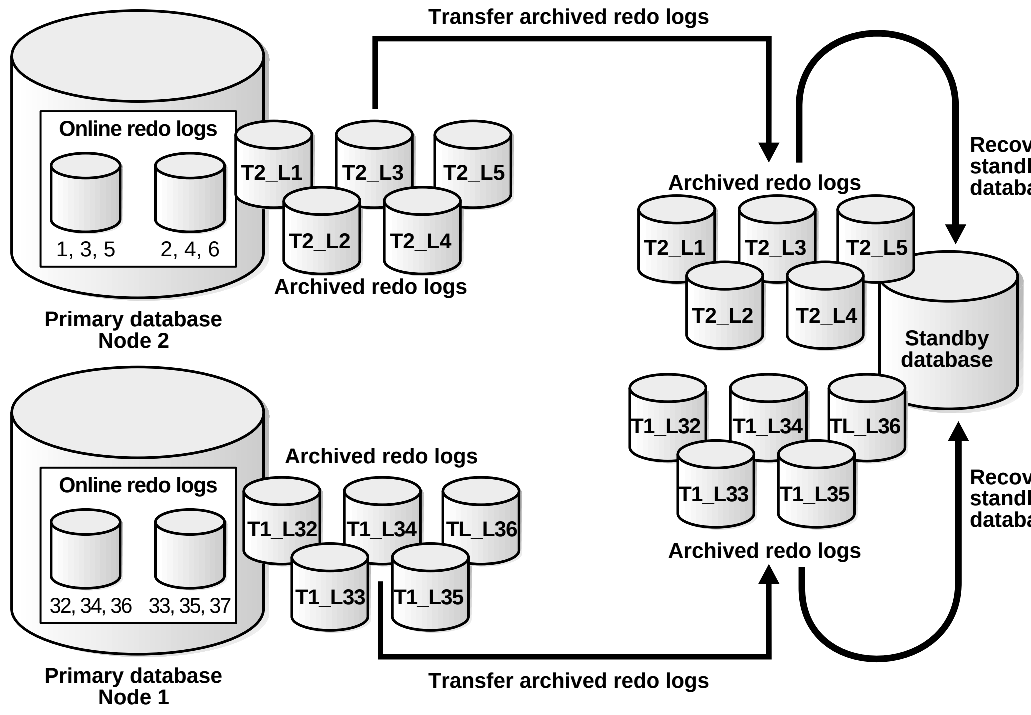
Using a Standby Database in an Oracle Parallel Server Configuration

You can use a standby database in conjunction with the Oracle Parallel Server (OPS) option. The following table describes the possible combinations of nodes in the primary and standby databases:

	Single-Instance Standby Database	**Multi-Instance Standby Database**
Single-Instance Primary Database	Yes	Yes (for read-only queries)
Multi-Instance Primary Database	Yes	Yes

In each scenario, each node of the primary database transmits its own thread of archived redo logs to the standby database. For example, Figure 3–3 illustrates an Oracle Parallel Server database with two nodes transmitting redo logs to a single-instance standby database.

Figure 3–3   Transmitting Archived Logs from a Multi-Instance Primary Database



In this case, node 1 of the primary database transmits logs 1, 2, 3, 4, 5 while node 2 transmits logs 32, 33, 34, 35, 36. If the standby database is in managed recovery mode, it automatically determines the correct order in which to apply the archived redo logs.

If both your primary and standby databases are in an Oracle Parallel Server configuration, and the standby database is in managed recovery mode, then a single node of the standby database applies all sets of logs transmitted by the primary nodes. In this case, the standby nodes that are *not* applying redo cannot be in read-only mode while managed recovery is in progress; in most cases, the non-recovery nodes should be shut down, although they can also be mounted.

> **See Also:** For information about configuring a database for Oracle Parallel Server, see the *Oracle8i Parallel Server Setup and Configuration Guide.*

4 Performing Maintenance on a Standby Database

This chapter describes how to perform typical maintenance operations on a standby database. It includes the following topics:

- Monitoring Events That Affect the Standby Database
- Responding to Events That Affect the Standby Database
- Backing Up the Standby Database

Monitoring Events That Affect the Standby Database

To prevent possible problems, you should be aware of events that affect a standby database and learn how to monitor them. Most changes to a primary database are automatically propagated to a standby database through archived redo logs and so require no user intervention. Nevertheless, some changes to a primary database require manual intervention at the standby site.

This section contains the following topics:

- Monitoring the Primary and Standby Databases
- Determining Which Archived Logs Have Been Received by the Standby Site
- Determining Which Logs Have Been Applied to the Standby Database

Monitoring the Primary and Standby Databases

Table 4–1 indicates whether a command is normally propagated or requires extra administrative efforts to be fully propagated. It also describes how to respond to these events.

Table 4–1 Propagating a Command

Primary Database Event	Detection at Primary Site	Detection at Standby Site	Response
Archiving errors	▪ ERROR.V$ARCHIVE_DESTINATION ▪ `alert.log` ▪ ARCHIVED.V$LOG ▪ Archiving trace files	remote file server (RFS) process trace file	Create scripts to push or pull archived redo logs if errors occur or if performance is degraded. See Adding Tablespaces or Datafiles to the Primary Database on page 4-10.
Thread events	▪ `alert.log` ▪ V$THREAD	`alert.log`	Thread events are automatically propagated through archived logs, so no extra action is necessary.
Redo log changes	▪ `alert.log` ▪ V$LOG and STATUS.V$LOGFILE	N/A	Redo log changes do not affect standby database unless a redo log is cleared or lost. In these cases, you must rebuild the standby database. See Creating the Standby Database Files on page 2-7. Pre-clear the logs on the standby database with the `ALTER DATABASE CLEAR LOGFILE` statement. See Clearing Online Redo Logs on page 4-16.
Issue CREATE CONTROLFILE	`alert.log`	Database functions normally until it encounters redo depending on any parameter changes.	Re-create the standby control file (see Refreshing the Standby Database Control File on page 4-15). Re-create the standby database if the primary database is opened RESETLOGS.
Media recovery performed	`alert.log`	N/A	Re-create the standby database if the RESETLOGS option is utilized.
Tablespace status changes (made read/write or read-only, placed online or offline)	▪ DBA_TABLESPACES ▪ `alert.log`	▪ Verify that all datafiles are online. ▪ V$RECOVER_FILE	Status changes are automatically propagated, so no response is necessary. Datafiles remain online.

Table 4–1 (Cont.) Propagating a Command

Primary Database Event	Detection at Primary Site	Detection at Standby Site	Response
Add datafile or create tablespace	■ DBA_DATA_FILES ■ `alert.log`	■ `ORA-283`, `ORA-1670`, `ORA-1157`, `ORA-1110` ■ Standby recovery stops.	Manually create datafile and restart recovery. See Adding Tablespaces or Datafiles to the Primary Database on page 4-10.
Drop tablespace	■ DBA_DATA_FILES ■ `alert.log`	`alert.log`	Remove datafile from operating system.
Tablespace or datafile taken offline, or datafile is dropped offline	■ V$RECOVER_FILE ■ `alert.log` The tablespace or datafile requires recovery when you attempt to bring it online.	■ Verify that all datafiles are online. ■ V$RECOVER_FILE	Datafiles remain online. The tablespace or datafile is fine after standby database activation.
Rename datafile	`alert.log`	N/A	N/A
Unlogged or unrecoverable operations	■ Direct loader invalidates block range redo entry in online redo log. Check V$DATAFILE. ■ V$DATABASE	`alert.log`. File blocks are invalidated unless they are in the future of redo, in which case they are not touched.	Unlogged changes are not propagated to the standby database. If you want to apply these changes, see Performing Direct Path Operations on page 4-13.
Recovery progress	`alert.log`	■ V$RECOVER_LOG ■ `alert.log`	Make sure the standby database is not following behind the primary database.
Autoextend a datafile	`alert.log`	May cause operation to fail on standby database because it lacks disk space.	Ensure that there is enough disk space for the expanded datafile.
Issue OPEN RESETLOGS or CLEAR UNARCHIVED LOGFILES statements	`alert.log`	Standby database is invalidated.	Rebuild the standby database. See Creating the Standby Database Files on page 2-7.
Change initialization parameter	`alert.log`	May cause failure because of redo depending on the changed parameter.	Dynamically change the standby parameter or shut down the standby database and edit the initialization parameter file.

Determining Which Archived Logs Have Been Received by the Standby Site

The following table lists two methods for gaining information about archived redo logs received by the standby site:

Method	Advantages	Disadvantages
Access V$ARCHIVED_LOG view on the standby database.	■ Is easily accessible. ■ Does not require setting additional parameters.	■ Lists minimal information. ■ Does not record archiving errors.
Set archive tracing on primary and standby sites through the LOG_ARCHIVE_TRACE initialization parameter.	■ Allows you to control the level of detail in the trace output. ■ Gives extensive information if desired. ■ Records archiving errors as well as successes.	Requires setting an initialization parameter and interpreting trace output.

Accessing the V$ARCHIVED_LOG View

The simplest way to determine the most recent archived log received by the standby site is to query the V$ARCHIVED_LOG view. This view is only useful after the standby site has started receiving logs, because before that time the view is populated by old archived log records generated from the primary control file. For example, you can execute the following script (sample output included):

```
col name format a20
col thread# format 999
col sequence# format 999
col first_change# format 999999
col next_change# format 999999

SELECT thread#, sequence# AS "SEQ#", name, first_change# AS "FIRSTSCN",
       next_change# AS "NEXTSCN",archived, deleted,completion_time AS "TIME"
FROM   v$archived_log
/

SQL> @archived_script

THREAD#       SEQ# NAME                   FIRSTSCN    NEXTSCN ARC DEL TIME
------- ---------- -------------------- ---------- ---------- --- --- ---------
      1        947 /arc_dest/arc_1_947       33113      33249 YES NO  23-JUN-99
```

Setting Archive Tracing

To see the progression of the archiving of redo logs to the standby site, set the LOG_ARCHIVE_TRACE parameter in the primary and standby initialization parameter files.

LOG_ARCHIVE_TRACE on	Causes Oracle to write	In trace file
Primary database	Audit trail of archiving process activity (ARC*n* and foreground processes) on primary database	Whose filename is specified in the USER_DUMP_DEST initialization parameter
Standby database	Audit trail of RFS process activity relating to archived redo logs on the standby database	Whose filename is specified in the USER_DUMP_DEST initialization parameter

Determining the Location of the Trace Files The trace files for a database are located in the directory specified by the USER_DUMP_DEST parameter in the initialization parameter file. Connect to the primary and standby instances using SQL*Plus and issue a SHOW statement to determine the location:

```
SQL> SHOW PARAMETER user_dump_dest
NAME                                 TYPE    VALUE
------------------------------------ ------- ------------------------------
user_dump_dest                       string  ?/rdbms/log
```

Setting the Log Trace Parameter The format for the archiving trace parameter is as follows, where *trace_level* is an integer:

```
LOG_ARCHIVE_TRACE=trace_level
```

To enable, disable, or modify the LOG_ARCHIVE_TRACE parameter in a primary database, do one of the following:

- Shut down the primary database, modify the initialization parameter file, and restart the database.
- Issue ALTER SYSTEM SET LOG_ARCHIVE_TRACE = *integer* while the database is open or mounted.

To enable, disable, or modify the LOG_ARCHIVE_TRACE parameter in a standby database in read-only or recovery mode, issue the following SQL statement:

```
ALTER SYSTEM SET ...;
```

If managed recovery is active, then issue the ALTER SYSTEM statement from a different standby session so that it affects trace output generated by the remote file service (RFS) when the next archived log is received from the primary database. For example, enter:

```
SQL> ALTER SYSTEM SET log_archive_trace=32;
```

Choosing an Integer Value The integer values for the LOG_ARCHIVE_TRACE parameter represent levels of tracing data. In general, the higher the level, the more detailed the information. The following integer levels are available:

Level	Meaning
0	Disable archivelog tracing - default setting.
1	Track archival of REDO log file.
2	Track archival status per archivelog destination.
4	Track archival operational phase.
8	Track archivelog destination activity.
16	Track detailed archivelog destination activity.
32	Track archivelog destination parameter modifications.
64	Track ARCn process state activity.

You can combine tracing levels by setting the value of the LOG_ARCHIVE_TRACE parameter to the sum of the individual levels. For example, setting the parameter to 3 generates level 1 and level 2 trace output.

Following are examples of the ARC0 trace data generated on the primary site by the archival of redo log 387 to two different destinations: the service STANDBY1 and the local directory `/vobs/oracle/dbs`.

Note: The level numbers do not appear in the actual trace output: they are shown here for clarification only.

```
Level   Corresponding entry content (sample)
-----   --------------------------------
( 1)    ARC0: Begin archiving log# 1 seq# 387 thrd# 1
( 4)    ARC0: VALIDATE
( 4)    ARC0: PREPARE
( 4)    ARC0: INITIALIZE
( 4)    ARC0: SPOOL
( 8)    ARC0: Creating archive destination 2 : 'standby1'
(16)    ARC0:  Issuing standby Create archive destination at 'standby1'
( 8)    ARC0: Creating archive destination 1 : '/vobs/oracle/dbs/d1arc1_387.dbf'
(16)    ARC0:  Archiving block 1 count 1 to : 'standby1'
(16)    ARC0:  Issuing standby Archive of block 1 count 1 to 'standby1'
(16)    ARC0:  Archiving block 1 count 1 to :  '/vobs/oracle/dbs/d1arc1_387.dbf'
( 8)    ARC0: Closing archive destination 2  : standby1
(16)    ARC0:  Issuing standby Close archive destination at 'standby1'
( 8)    ARC0: Closing archive destination 1  :  /vobs/oracle/dbs/d1arc1_387.dbf
( 4)    ARC0: FINISH
( 2)    ARC0: Archival success destination 2 : 'standby1'
( 2)    ARC0: Archival success destination 1 : '/vobs/oracle/dbs/d1arc1_387.dbf'
( 4)    ARC0: COMPLETE, all destinations archived
(16)    ARC0: ArchivedLog entry added: /vobs/oracle/dbs/d1arc1_387.dbf
(16)    ARC0: ArchivedLog entry added: standby1
( 4)    ARC0: ARCHIVED
( 1)    ARC0: Completed archiving log# 1 seq# 387 thrd# 1

(32)  Propagating archive 0 destination version 0 to version 2
         Propagating archive 0 state version 0 to version 2
         Propagating archive 1 destination version 0 to version 2
         Propagating archive 1 state version 0 to version 2
         Propagating archive 2 destination version 0 to version 1
         Propagating archive 2 state version 0 to version 1
         Propagating archive 3 destination version 0 to version 1
         Propagating archive 3 state version 0 to version 1
         Propagating archive 4 destination version 0 to version 1
         Propagating archive 4 state version 0 to version 1

(64) ARCH: changing ARC0 KCRRNOARCH->KCRRSCHED
        ARCH: STARTING ARCH PROCESSES
        ARCH: changing ARC0 KCRRSCHED->KCRRSTART
        ARCH: invoking ARC0
        ARC0: changing ARC0 KCRRSTART->KCRRACTIVE
        ARCH: Initializing ARC0
        ARCH: ARC0 invoked
        ARCH: STARTING ARCH PROCESSES COMPLETE
        ARC0 started with pid=8
        ARC0: Archival started
```

Following is the trace data generated by the RFS process on the standby site as it receives archived log 387 in directory `/stby` and applies it to the standby database:

```
level    trace output (sample)
----    ------------------
( 4)      RFS: Startup received from ARCH pid 9272
( 4)      RFS: Notifier
( 4)      RFS: Attaching to standby instance
( 1)      RFS: Begin archive log# 2 seq# 387 thrd# 1
(32)      Propagating archive 5 destination version 0 to version 2
(32)      Propagating archive 5 state version 0 to version 1
( 8)      RFS: Creating archive destination file: /stby/parc1_387.dbf
(16)      RFS:  Archiving block 1 count 11
( 1)      RFS: Completed archive log# 2 seq# 387 thrd# 1
( 8)      RFS: Closing archive destination file: /stby/parc1_387.dbf
(16)      RFS: ArchivedLog entry added: /stby/parc1_387.dbf
( 1)      RFS: Archivelog seq# 387 thrd# 1 available 04/02/99 09:40:53
( 4)      RFS: Detaching from standby instance
( 4)      RFS: Shutdown received from ARCH pid 9272
```

Determining Which Logs Have Been Applied to the Standby Database

Query the V$LOG_HISTORY view on the standby database, which records the latest log sequence number that has been applied. For example, issue the following query:

```
SQL> SELECT thread#, max(sequence#) AS "LAST_APPLIED_LOG"
  2> FROM   v$log_history
  3> GROUP BY thread#;

THREAD# LAST_APPLIED_LOG
------- ----------------
      1              967
```

In this example, the archived redo log with log sequence number 967 is the most recently applied log.

Note: V$LOG is not updated during recovery.

Responding to Events That Affect the Standby Database

Typically, physical changes to the primary database require a manual response on the standby database. This section contains the following topics:

- Adding Tablespaces or Datafiles to the Primary Database
- Renaming Datafiles on the Primary Database

- Adding or Dropping Redo Logs on the Primary Database
- Resetting or Clearing Unarchived Redo Logs on the Primary Database
- Altering the Primary Database Control File
- Taking Datafiles in the Standby Database Offline
- Performing Direct Path Operations
- Refreshing the Standby Database Control File
- Clearing Online Redo Logs

Adding Tablespaces or Datafiles to the Primary Database

Adding a tablespace or datafile to the primary database generates redo that, when applied at the standby database, automatically adds the datafile name to the standby control file. If the standby database locates the file with the filename specified in the control file, then recovery continues. If the standby database is unable to locate a file with the filename specified in the control file, then recovery terminates.

Perform one of the following procedures to create a new datafile in the primary database and update the standby database. Note that if you do not want the new datafile in the standby database, you can take the datafile offline manually using the following syntax:

```
SQL> ALTER DATABASE DATAFILE 'filename' OFFLINE DROP;
```

To add a tablespace or datafile to the primary database and create the datafile in the standby database:

1. Create a tablespace on the primary database as usual. For example, to create new datafile `t_db2.f` in tablespace `tbs_2`, issue:

   ```
   SQL> CREATE TABLESPACE tbs_2 DATAFILE 't_db2.f' SIZE 2M;
   ```

2. If the standby database is shut down, start the standby instance without mounting it. For example, enter:

   ```
   SQL> STARTUP NOMOUNT pfile=/private1/stby/initSTANDBY.ora
   ```

   If the standby database is currently in managed recovery mode, skip to step 4.

3. Mount the standby database, then place it in managed recovery mode:

   ```
   SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
   ```

```
SQL> RECOVER MANAGED STANDBY DATABASE;
```

4. Switch redo logs on the primary database to initiate redo archival to the standby database:

   ```
   SQL> ALTER SYSTEM SWITCH LOGFILE;
   ```

   If the recovery process on the standby database tries to apply the redo containing the CREATE TABLESPACE statement, it stops because the new datafile does not exist on the standby site.

5. Either wait for the standby database to cancel recovery because it cannot find the new datafile, or manually cancel managed recovery:

   ```
   SQL> RECOVER MANAGED STANDBY DATABASE CANCEL;
   ```

   Note that CREATE TABLESPACE redo adds the new filename to the standby control file. The following `alert.log` entry is generated:

   ```
   WARNING! Recovering datafile 2 from a fuzzy file. If not the current file it
   might be an online backup taken without entering the begin backup command.
   Successfully added datafile 2 to media recovery
   Datafile #2: '/private1/stby/t_db2.f'
   ```

6. Create the datafile on the standby database. For example, issue:

   ```
   SQL> ALTER DATABASE CREATE DATAFILE '/private1/stby/t_db2.f'
                                    AS '/private1/stby/t_db2.f';
   ```

7. Place the standby database in managed recovery mode:

   ```
   SQL> RECOVER MANAGED STANDBY DATABASE;
   ```

Continue normal processing on the primary database. The primary and standby databases are now synchronized.

> **See Also:** For more information on offline datafile alterations, see Taking Datafiles in the Standby Database Offline on page 4-13.

Renaming Datafiles on the Primary Database

Datafile renames on your primary database do not take effect at the standby database until you refresh the standby database control file. To keep the datafiles at the primary and standby databases synchronized when you rename primary database datafiles, perform analogous operations on the standby database.

Adding or Dropping Redo Logs on the Primary Database

You can add redo log file groups or members to the primary database without affecting the standby database. Similarly, you can drop log file groups or members from the primary database without affecting your standby database. Enabling and disabling of threads at the primary database has no effect on the standby database.

Consider whether to keep the online redo log configuration the same at the primary and standby databases. Although differences in the online redo log configuration between the primary and standby databases do not affect the standby database functionality, they do affect the performance of the standby database after activation. For example, if the primary database has 10 redo logs and the standby database has 2, and you then activate the standby database so that it functions as the new primary database, the new primary database is forced to archive more frequently than the old primary database.

To prevent problems after standby activations, Oracle Corporation recommends keeping the online redo log configuration the same at the primary and standby databases. Note that when you enable a log file thread with the ALTER DATABASE ENABLE THREAD statement at the primary database, you must create a new control file for your standby database before activating it. See Refreshing the Standby Database Control File on page 4-15 for procedures.

Resetting or Clearing Unarchived Redo Logs on the Primary Database

If you clear log files at the primary database by issuing the ALTER DATABASE CLEAR UNARCHIVED LOGFILE statement, or open the primary database using the RESETLOGS option, you invalidate the standby database. Because both of these operations reset the primary log sequence number to 1, you must re-create the standby database in order to be able to apply archived logs generated by the primary database. See Creating the Standby Database Files on page 2-7 for the procedure. See Scenario 8: Re-Creating a Standby Database on page 5-44 for additional information.

Altering the Primary Database Control File

If you use the CREATE CONTROLFILE statement at the primary database to perform any of the following operations, you may invalidate the control file for the standby database:

- Change the maximum number of redo log file groups or members.
- Change the maximum number of instances that can concurrently mount and open the database.

Using the CREATE CONTROLFILE statement with the RESETLOGS option on your primary database will force the next open of the primary database to reset the online logs, thereby invalidating the standby database.

If you have invalidated the control file for the standby database, re-create the file using the procedures in Refreshing the Standby Database Control File on page 4-15.

Taking Datafiles in the Standby Database Offline

You can take standby database datafiles offline as a means to support a subset of your primary database's datafiles. For example, you may decide not to recover the primary database's temporary tablespaces on the standby database.

Take the datafiles offline using the following statement on the standby database:

```
ALTER DATABASE DATAFILE 'filename' OFFLINE DROP;
```

If you execute this statement, then the tablespace containing the offline files must be dropped after opening the standby database.

Performing Direct Path Operations

When you perform a direct load originating from any of the following, the performance improvement applies *only* to the primary database (there is no corresponding recovery process performance improvement on the standby database):

- Direct path load
- CREATE TABLE through subquery
- CREATE INDEX on the primary database

The standby database recovery process continues to sequentially read and apply the redo information generated by the unrecoverable direct load.

Propagating UNRECOVERABLE Processes Manually

Primary database processes using the UNRECOVERABLE option are not propagated to the standby database because these processes do not appear in the archived redo logs. If you perform an UNRECOVERABLE operation at the primary database and then recover the standby database, you do not receive error messages during recovery; instead, Oracle writes error messages in the standby database `alert.log` file. The following error message is displayed:

```
26040, 00000, "Data block was loaded using the NOLOGGING option\n"
```

```
//* Cause: Trying to access data in block that was loaded without
//*        redo generation using the NOLOGGING/UNRECOVERABLE option
//* Action: Drop the object containing the block.
```

Although the error message recommends dropping the object that contains the block, *do not perform this operation*. Instead, perform any *one* of the following tasks:

- Take the affected datafiles offline in the standby database and drop the tablespace after activation (see Taking Datafiles in the Standby Database Offline on page 4-13).
- Re-create the standby database from a new database backup (see Creating the Standby Database Files on page 2-7).
- Back up the affected tablespace and archive the current logs in the primary database, transfer the datafiles to the standby database, and resume standby recovery. This is the same procedure that you would perform to guarantee ordinary database recoverability after an UNRECOVERABLE operation.

> **See Also:** For more details, see Taking Datafiles in the Standby Database Offline on page 4-13 and Scenario 4: Recovering After the NOLOGGING Clause Was Specified on page 5-28.

Determining Whether a Backup Is Required After UNRECOVERABLE Operations

If you have performed UNRECOVERABLE operations on your primary database, determine whether a new backup is required.

To determine whether a new backup is necessary:

1. Query the V$DATAFILE view on the primary database to determine the system change number (SCN) or time at which Oracle generated the most recent invalidation redo data.
2. Issue the following SQL statement on the primary database to determine whether you need to perform another backup:

   ```
   SELECT unrecoverable_change#,
          to_char(unrecoverable_time, 'mm-dd-yyyy hh:mi:ss')
   FROM   v$datafile;
   ```

3. If the query in the previous step reports an unrecoverable time for a datafile that is more recent than the time when the datafile was last backed up, then make another backup of the datafile in question.

See Also: For more information about the V$DATAFILE view, see the *Oracle8i Reference*.

Refreshing the Standby Database Control File

The following steps describe how to refresh, or create a copy, of changes you have made to the primary database control file. Refresh the standby database control file after making major structural changes to the primary database, such as adding or dropping files.

To refresh the standby database control file:

1. Start a SQL*Plus session on the standby instance and issue the CANCEL statement on the standby database to halt its recovery process.

   ```
   SQL> RECOVER CANCEL  # for manual recovery mode
   SQL> RECOVER MANAGED STANDBY DATABASE CANCEL   # for managed recovery mode
   ```

2. Shut down the standby instances:

   ```
   SQL> SHUTDOWN IMMEDIATE
   ```

3. Start a SQL*Plus session on the production instance and create the control file for the standby database:

   ```
   SQL> ALTER DATABASE CREATE STANDBY CONTROLFILE AS 'filename';
   ```

4. Transfer the standby control file and archived log files to the standby site using an operating system utility appropriate for binary files.
5. Connect to the standby instance and mount (but do not open) the standby database:

   ```
   SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
   ```

6. Restart the recovery process on the standby database:

   ```
   SQL> RECOVER STANDBY DATABASE  # recovers using location for logs
                                  # specified in initialization parameter file
   SQL> RECOVER FROM 'location' STANDBY DATABASE # recovers from specified
                                                 # location
   ```

Clearing Online Redo Logs

After creating the standby database, you can clear standby database online redo logs to optimize performance by issuing the following statement, where *integer* refers to the number of the log group:

```
ALTER DATABASE CLEAR LOGFILE GROUP integer;
```

This statement optimizes standby activation because it is no longer necessary for Oracle to *zero* the logs at activation. Zeroing involves writing zeros to the entire contents of the redo log and then setting a new header to make the redo log look like it was when it was created. Zeroing occurs during a RESETLOGS operation.

If you clear the logs manually, Oracle realizes at activation that the logs already have zeros and skips the zeroing step. This optimization is important because it can take a long time to write zeros into all of the online logs. If you prefer not to perform this operation during maintenance, Oracle clears the online logs automatically during activation.

Backing Up the Standby Database

If necessary, you can back up your standby database, but not while the database is in manual or managed recovery mode. You must take the standby database out of managed recovery mode, make the backups, then resume managed recovery. You can make the backups when the database is shut down or in read-only mode.

The following table lists some advantages and disadvantages of these methods:

Method	**Advantages**	**Disadvantages**
Shut down the standby database.	Can back up the database after performing other maintenance operations requiring database shutdown.	The primary database may create a gap sequence because the standby database is not receiving archived logs. If you create a gap sequence, you must perform manual recovery before you can place the standby database in managed recovery mode.
Place the standby database in read-only mode.	The standby site continues to receive archived logs from the primary database so no gap sequence is generated.	

To back up tablespaces on a standby database when it is in read-only mode:

1. Start a SQL*Plus session on the standby database and take the database out of managed or manual recovery mode:

   ```
   RECOVER MANAGED STANDBY DATABASE CANCEL    # for managed recovery
   RECOVER CANCEL                             # for manual recovery
   ```

2. Open the database in read-only mode:

   ```
   ALTER DATABASE OPEN READ ONLY
   ```

3. Take backups of some tablespaces using operating system utilities. You should not back up the standby control file.

   Minimize the time that the database is down. For example, to back up datafiles `tbs11.f`, `tbs12.f`, and `tbs13.f` in tablespace TBS_1 on UNIX you might enter:

   ```
   % cp /disk1/oracle/dbs/tbs11.f /disk2/backup/tbs11.bk
   % cp /disk1/oracle/dbs/tbs12.f /disk2/backup/tbs12.bk
   % cp /disk1/oracle/dbs/tbs13.f /disk2/backup/tbs13.bk
   ```

4. Terminate all active user sessions on the standby database.
5. Place the database in manual or managed recovery mode:

   ```
   RECOVER MANAGED STANDBY DATABASE     # for managed recovery
   RECOVER STANDBY DATABASE             # for manual recovery
   ```

6. Back up the control file on the primary database using an operating system utility. You must back up the primary database control file, not the standby database control file.
7. Repeat the preceding steps until you have backed up each tablespace in the database.

To back up tablespaces on a standby database when it is shut down:

1. Start a SQL*Plus session on the standby database and take the database out of managed or manual recovery mode:

   ```
   RECOVER MANAGED STANDBY DATABASE CANCEL    # for managed recovery
   RECOVER CANCEL                             # for manual recovery
   ```

2. Shut down the database:

   ```
   SHUTDOWN IMMEDIATE
   ```

3. Make cold backups of some tablespaces using operating system utilities. Minimize the time that the database is down. For example, to back up datafiles `tbs11.f`, `tbs12.f`, and `tbs13.f` in tablespace TBS_1 on UNIX you might enter:

```
% cp /disk1/oracle/dbs/tbs11.f /disk2/backup/tbs11.bk
% cp /disk1/oracle/dbs/tbs12.f /disk2/backup/tbs12.bk
% cp /disk1/oracle/dbs/tbs13.f /disk2/backup/tbs13.bk
```

4. Use SQL*Plus to start the Oracle instance at the standby database without mounting it, specifying a parameter file if necessary:

```
STARTUP NOMOUNT pfile = initSTANDBY.ora
```

5. Mount the database:

```
ALTER DATABASE MOUNT STANDBY DATABASE
```

6. Place the database in manual or managed recovery mode:

```
RECOVER MANAGED STANDBY DATABASE     # for managed recovery
RECOVER STANDBY DATABASE             # for manual recovery
```

7. Repeat the preceding steps until you have backed up each tablespace in the database.

5

Standby Database Scenarios

This chapter describes the following standby database scenarios:

Scenario 1: Creating a Standby Database on the Same Host

This scenario describes the creation of a standby database STANDBY1 on the same host as the primary database PROD1. The host is a UNIX machine with three file systems, each mounted on a separate disk configuration on a different controller. By placing the standby database on a different file system from the primary database, you protect the primary database from a hard disk failure. By running the same-host standby database in managed recovery mode, you can keep it continuously up-to-date.

After you set up the standby database on the local host, you plan to create a standby database on a remote host for total disaster protection. In this way, even if all disks of the primary database crash or are destroyed in a disaster, you can fail over to the remote standby database and keep the database open.

Step 1: Plan the Standby Database.

Because the host uses three file systems, each on its own set of disks with its own controller, you decide to maintain the primary database files on the first file system, the standby database files on the second file system, and the ORACLE_HOME binaries on the third file system. If the primary database disks fail, you can switch to the standby database; if the ORACLE_HOME disks fail, you can switch to the remote standby database.

To host the standby database on the same machine as the primary database, you must set the following parameters in the standby database initialization parameter file:

- CONTROL_FILES
- LOCK_NAME_SPACE
- DB_FILE_NAME_CONVERT
- LOG_FILE_NAME_CONVERT

Fortunately, most (but not all) of the primary database datafiles and redo log files are in the same directory and are named consistently. You will have to rename some of the files manually using ALTER DATABASE statements.

Because the primary database is shut down every Sunday for an hour for maintenance, you decide to use that time to make a cold, consistent backup. You can then restart the database while you make the necessary configurations for the standby database.

Step 2: Create the Standby Database.

The next step in the procedure is to create the backup that will form the basis for the standby database. You know that you can use either an inconsistent or consistent backup, but because the database must go down every Sunday for maintenance, you decide to make a consistent backup then and use it for the standby database.

1. Determine the database files.

   On Sunday, before shutting down the primary database, you query the database to determine which datafiles it contains:

   ```
   SQL> SELECT name FROM v$datafile;
   NAME
   --------------------------------------------------------------------------------
   /fs1/dbs/tbs_01.f
   /fs1/dbs/tbs_02.f
   /fs1/dbs/tbs_11.f
   /fs1/dbs/tbs_12.f
   /fs1/dbs/tbs_21.f
   /fs1/dbs/tbs_22.f
   /fs1/dbs/tbs_13.f
   /fs1/dbs/tbs_23.f
   /fs1/dbs/tbs_24.f
   /fs1/dbs/tbs_31.f
   /fs1/dbs/tbs_32.f
   /fs1/dbs/tbs_41.f
   /fs1/dbs2/tbs_42.f
   /fs1/dbs2/tbs_51.f
   /fs1/dbs2/tbs_52.f
   /fs1/dbs2/tbs_03.f
   /fs1/dbs3/tbs_14.f
   /fs1/dbs3/tbs_25.f
   /fs1/dbs3/tbs_33.f
   /fs1/dbs3/tbs_43.f
   /fs1/dbs3/tbs_53.f
   21 rows selected.
   ```

2. Back up the datafiles.

   After determining which datafiles are in the database, you shut down the database with the IMMEDIATE option:

   ```
   SQL> SHUTDOWN IMMEDIATE;
   ```

   At this point, you decide to back up all of the primary datafiles to a temporary directory as follows:

   ```
   % cp /fs1/dbs/* /fs1/temp
   ```

```
% cp /fs1/dbs2/* /fs1/temp
% cp /fs1/dbs3/* /fs1/temp
```

You perform some other routine maintenance operations and then restart the database as follows:

```
SQL> STARTUP PFILE=initPROD1.ora;
```

3. Create the standby database control file.

   After a few minutes, you create the standby database control file in the same directory in which you stored the consistent backup:

   ```
   SQL> ALTER DATABASE CREATE STANDBY CONTROLFILE AS '/fs1/temp/stbycf.f';
   ```

4. Transfer files to the standby file system.

   After you have successfully created the standby database control file, you can copy the datafiles and the standby database control file from the primary file system to the standby file system.

   Because the transferring of datafiles can take a long time, you first copy the control file, begin copying the datafiles, and then proceed to other tasks (such as network configuration). For example, enter the following at the UNIX command shell:

   ```
   % cp /fs1/temp/stbycf.f /fs2/dbs/cf1.f
   % cp /fs1/temp/tbs* /fs2/dbs
   ```

Step 3: Configure the Network Files.

In order to run a standby database in a managed standby environment, you must configure a Net8 connection between the primary and standby databases so that you can archive the redo logs to the standby service.

You use the IPC protocol to connect the primary database to the standby database because both databases are on the same host. Because you do not maintain an Oracle Names server, you must create both a `tnsnames.ora` entry for the primary database and a `listener.ora` entry for the standby database.

1. Configure the `tnsnames.ora` file.

   Your next step is to open the `tnsnames.ora` file in a text editor:

   ```
   % vi /fs3/oracle/network/admin/tnsnames.ora
   ```

   Currently, only one service name entry exists in the file, a TCP/IP connection to the PROD1 database:

```
prod1 = (DESCRIPTION=
        (ADDRESS=(PROTOCOL=tcp)(PORT=1512)(HOST=dlsun183))
        (CONNECT_DATA=(SID=prod1))
)
```

To define an IPC connection between the primary and the standby database, you add an entry with the following format:

```
standby_service_name = (DESCRIPTION=
                           (ADDRESS=(PROTOCOL=ipc) (KEY=keyhandle))
                           (CONNECT_DATA=(SID=standby_sid)))
```

Substitute appropriate values for standby_service_name, keyhandle, and standby_sid, as the following example shows:

```
standby1 = (DESCRIPTION=
              (ADDRESS=(PROTOCOL=ipc) (KEY=kstdby1))
              (CONNECT_DATA=(SID=stdby1)))
```

2. Configure the `listener.ora` file.

   Your next step is to open the `listener.ora` file, which is located on file system /fs3:

```
% vi /fs3/oracle/network/admin/listener.ora
```

   You discover the following list of addresses (where on the host the listener is listening) and SIDs (which connections the listener is listening for):

```
LISTENER = (ADDRESS_LIST=
 (ADDRESS=(PROTOCOL=tcp)(PORT=1512)(HOST=dlsun183))
)
SID_LIST_LISTENER = (SID_LIST=
 (SID_DESC=(SID_NAME=PROD1)(ORACLE_HOME=/fs3/oracle))
)
```

   Currently, the listener is listening on port 1512 of the host dlsun183 for database PROD1.

   You need to edit the `listener.ora` file and add two entries with the following format:

```
STANDBY_LISTENER = (ADDRESS_LIST=(ADDRESS=(PROTOCOL=ipc)
                        (KEY=keyhandle)))

SID_LIST_STANDBY_LISTENER = (SID_LIST=
                  (SID_DESC=(SID_NAME=standby_sid)(ORACLE_HOME=/oracle_home)))
```

The `listener.ora` file is typically located in the $ORACLE_HOME/network/admin directory on the standby site. Substitute appropriate values for keyhandle, standby_sid, and oracle_home as the following example shows:

```
STBY1_LISTENER = (ADDRESS_LIST=(ADDRESS=(PROTOCOL=ipc)   # same node as primary
                   (KEY=kstdby1)))   # ORACLE_SID standby instance is started with

SID_LIST_STBY1_LISTENER = (SID_LIST=
                  (SID_DESC=(SID_NAME=stdby1)(ORACLE_HOME=/vobs/oracle)))
```

Now that you have edited the `listener.ora` file, you must restart the listener so that it recognizes the changes in the file:

```
% lsnrctl
LSNRCTL for Solaris: Version 8.1.5.0.0 - Production on 05-APR-99 11:39:41

(c) Copyright 1998 Oracle Corporation.  All rights reserved.

Welcome to LSNRCTL, type "help" for information.

LSNRCTL> start stby1_listener
```

As an alternative to the steps outlined in this section, you can use the Net8 Assistant graphical user interface to configure the network files. For additional information, see the *Net8 Administrator's Guide*.

Step 4: Configure the Primary Database Parameter File.

Now that you have configured the network files, you can edit the primary database initialization parameter file. The primary database is now up and running, so these changes will only be enabled if you restart the instance or issue ALTER SESSION or ALTER SYSTEM statements.

The only changes you need to make to the file involve archiving to the standby service. Currently, the primary database parameter file looks as follows:

```
db_name=prod1
control_files=(/fs1/dbs/cf1.f,/fs1/dbs/cf2.f)
compatible=8.1.6
log_archive_start = TRUE
log_archive_dest_1 = 'LOCATION=/fs1/arc_dest/ MANDATORY REOPEN=60'
log_archive_dest_state_1 = ENABLE
log_archive_format = log_%t_%s.arc
audit_trail=FALSE
o7_dictionary_accessibility=FALSE
global_names=FALSE
db_domain=regress.rdbms.dev.us.oracle.com
```

```
remote_login_passwordfile = exclusive

# default parameters for instance 1
processes=30
sessions=30
transactions=21
transactions_per_rollback_segment=21
distributed_transactions=10
db_block_buffers=1000
db_files=200
shared_pool_size=10000000
```

1. Specify standby archive destinations.

   Currently, you archive to only one location: a local directory. Because you want to maintain the local standby database in managed recovery mode, you must specify a new archiving location using a service name.

   Open the primary database initialization parameter file with a text editor and examine the current archiving location and format:

   ```
   log_archive_dest_1 = 'LOCATION=/fs1/arc_dest/ MANDATORY REOPEN=60'
   log_archive_dest_state_1 = ENABLE
   log_archive_format = log_%t_%s.arc
   ```

Parameter/Option	Meaning
LOG_ARCHIVE_DEST_1	Indicates an archiving destination.
LOCATION	Indicates a local directory.
LOG_ARCHIVE_DEST_STATE_1	Indicates the state of the LOG_ARCHIVE_DEST_1 archiving destination.
ENABLE	Indicates that Oracle can archive to the destination.
LOG_ARCHIVE_FORMAT	Indicates the format for filenames of log files.

2. Because you want to archive to the standby database with service STANDBY1, you edit the file, adding the following entries:

   ```
   log_archive_dest_2 = 'SERVICE=standby1 OPTIONAL REOPEN=180'
   log_archive_dest_state_2 = ENABLE
   ```

Parameter/Option	Meaning
LOG_ARCHIVE_DEST_2	Indicates a new archiving destination. LOG_ ARCHIVE_DEST_1 is already reserved for local archiving to `/fs1/arc_dest/`.
SERVICE	Indicates the service name of the standby database.
OPTIONAL	Indicates that Oracle can reuse online redo logs even if this destination fails.
REOPEN	Indicates how many seconds the archiving process waits before reattempting to archive to a previously failed destination.
LOG_ARCHIVE_DEST_STATE_2	Indicates the state of the LOG_ARCHIVE_DEST_ 2 archiving destination.
ENABLE	Indicates that Oracle can archive to the destination.

After editing the primary database initialization parameter file, create a copy for use by the standby database:

```
% cp /fs1/temp/initPROD1.ora /fs3/oracle/dbs/initSTANDBY1.ora
```

If the primary database initialization parameter file contains the IFILE parameter, you also need to copy the file referred to by the IFILE parameter to the standby site and, if necessary, make appropriate changes to it.

Step 5: Configure the Standby Database Parameter File.

You know that the initialization parameters shown in Table 5–1 play a key role in the standby database recovery process, and decide to edit them.

Table 5–1 Configuring Standby Database Initialization Parameters

Parameter	Setting
COMPATIBLE	This parameter must be the same at the primary and standby databases. Because it is already set to 8.1.6 in the primary database parameter file, you can leave the standby setting as it is.
CONTROL_FILES	This parameter must be different between the primary and standby databases. You decide to locate the control files in the `/fs2/dbs` directory.
DB_FILE_NAME_CONVERT	Set when you want to make your standby datafile filenames distinguishable from your primary database filenames. Most (but not all) of the datafiles are in the `/fs1/dbs` directory. You set this parameter to `/fs2/dbs` to convert the files in the `/dbs` subdirectory automatically; the others you will convert using ALTER DATABASE RENAME FILE statements.
DB_FILES	DB_FILES must be the same at both databases so that you allow the same number of files at the standby database as you allow at the primary database. Consequently, you leave this parameter alone. An instance cannot mount a database unless DB_FILES is equal to or greater than MAXDATAFILES.
DB_NAME	This value should be the same as the DB_NAME value in the production database parameter file. Consequently, you leave this parameter alone.
LOCK_NAME_SPACE	Specifies the name space that the distributed lock manager (DLM) uses to generate lock names. Set this value if the standby and primary databases share the same host. You decide to set the name to STANDBY1.
LOG_ARCHIVE_DEST_1	This parameter specifies the location of the archived redo logs. You must use this directory when performing manual recovery. You decide to set the value to `/fs2/arc_dest/`.
LOG_FILE_NAME_CONVERT	Set when you want to make your standby redo log filenames distinguishable from your primary database redo log filenames. Because your primary redo logs are located in `/fs1/dbs`, you decide to locate the standby logs in `/fs2/dbs`.
STANDBY_ARCHIVE_DEST	Oracle uses this value to create the name of the logs received from the primary site. You decide to set it to `/fs2/stdby/`.

Edit the standby database parameter file as follows (with edited values in bold):

```
db_name = prod1                #The same as PRMYinit.ora
control_files = (/fs2/dbs/cf1.f)
compatible = 8.1.6
log_archive_start = TRUE
log_archive_dest_1='LOCATION=/fs2/arc_dest/'
log_archive_dest_state_1 = ENABLE
log_archive_format = log_%t_%s.arc
```

```
standby_archive_dest = /fs2/stdby/
db_file_name_convert = ('/fs1/dbs','/fs2/dbs')
log_file_name_convert = ('/fs1/dbs','/fs2/dbs')
lock_name_space = standby1
audit_trail=FALSE
o7_dictionary_accessibility=FALSE
global_names=FALSE
db_domain=regress.rdbms.dev.us.oracle.com
remote_login_passwordfile = exclusive

# default parameters for instance 1
processes=30
sessions=30
transactions=21
transactions_per_rollback_segment=21
distributed_transactions=10
db_block_buffers=1000
db_files=200
shared_pool_size=10000000
```

Step 6: Start the Standby Database in Preparation for Managed Recovery.

Now that you have configured all network and parameter files, you can enable archiving to the standby database.

1. Set the ORACLE_SID environment variable to the same value as the SID parameter in the `tnsnames.ora` file on the primary site and the `listener.ora` file on the standby site as follows:

   ```
   % setenv ORACLE_SID stdby1
   ```

2. Start the instance.

   First, you start the standby database instance without mounting the standby database control file, as the following example shows:

   ```
   SQL> CONNECT sys/change_on_install@standby1
   SQL> STARTUP NOMOUNT PFILE=/fs3/oracle/dbs/initSTANDBY1.ora;
   SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
   ```

3. Manually rename datafiles.

   Next, write a SQL script to rename the datafiles not captured by the DB_FILE_NAME_CONVERT parameter:

   ```
   ALTER DATABASE RENAME FILE /fs1/dbs2/tbs_42.f,
                              /fs1/dbs2/tbs_51.f,
                              /fs1/dbs2/tbs_52.f,
                              /fs1/dbs2/tbs_03.f,
   ```

```
                          /fs1/dbs3/tbs_14.f,
                          /fs1/dbs3/tbs_25.f,
                          /fs1/dbs3/tbs_33.f,
                          /fs1/dbs3/tbs_43.f,
                          /fs1/dbs3/tbs_53.f,
TO
                          /fs2/dbs/tbs_42.f,
                          /fs2/dbs/tbs_51.f,
                          /fs2/dbs/tbs_52.f,
                          /fs2/dbs/tbs_03.f,
                          /fs2/dbs/tbs_14.f,
                          /fs2/dbs/tbs_25.f,
                          /fs2/dbs/tbs_33.f,
                          /fs2/dbs/tbs_43.f,
                          /fs2/dbs/tbs_53.f
/
```

4. Enable changes to the primary database parameter file.

   Finally, you enable the changes you made to the primary database parameter file so that the standby database can begin receiving archived redo logs:

```
SQL> CONNECT sys/change_on_install@prod1 as sysdba
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_2 = ’SERVICE=standby1 OPTIONAL REOPEN=180’;
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2 = ENABLE;
```

Step 7: Identify the Logs in the Gap Sequence.

Because you have enabled the changes to the primary database parameter file, the primary database is now able to archive to the standby service name. Before you can perform managed recovery, however, you must synchronize the standby database by applying those logs containing changes made after the primary database backup, but before the first log received by the standby database.

Write the following SQL script and run it on the standby database:

```
SELECT high.thread#, "LowGap#", "HighGap#"
FROM
(     SELECT thread#, MIN(sequence#)-1 "HighGap#"
      FROM
      (     SELECT a.thread#, a.sequence#
            FROM
            (     SELECT *
                  FROM v$archived_log
            ) a,
            (     SELECT thread#, MAX(sequence#)gap1
                  FROM v$log_history
                  GROUP BY thread#
            ) b
            WHERE a.thread# = b.thread#
            AND a.sequence# > gap1
      )
      GROUP BY thread#
) high,

(     SELECT thread#, MIN(sequence#) "LowGap#"
      FROM
      (     SELECT thread#, sequence#
            FROM v$log_history, v$datafile
            WHERE checkpoint_change# <= next_change#
            AND checkpoint_change# >= first_change#
      )
      GROUP BY thread#
) low
WHERE low.thread# = high.thread#;
```

The output of the query is as follows:

```
SQL> @gap
THREAD#    LowGap#    HighGap#
---------- ---------- ----------
       1   250        252
```

Hence, you must apply log sequence 250, 251, and 252 to synchronize the standby database before initiating managed recovery.

Step 8: Copy the Logs in the Gap Sequence to the Standby File System.

The archived log filenames generated by gap sequence queries on the standby database are generated by the LOG_ARCHIVE_DEST_1 and LOG_ARCHIVE_FORMAT parameters in the initialization parameter file. Before transmitting the logs from the primary site to the standby site, you must determine the correct filenames.

First, you determine the filenames of the logs in the gap that were archived by the primary database. After connecting to the primary database using SQL*Plus, issue the following SQL query to obtain the names:

```
SQL> CONNECT sys/change_on_install@prod1
SQL> SELECT name FROM v$archived_log WHERE sequence# IN (250,251,252);

NAME
--------------------------------------------------------------------------------
/fs1/arc_dest/log_1_250.arc
/fs1/arc_dest/log_1_251.arc
/fs1/arc_dest/log_1_252.arc
```

The gap sequence in this case consists of log_1_250.arc, log_1_251.arc, and log_1_252.arc. The settings for LOG_ARCHIVE_DEST_1 and LOG_ARCHIVE_FORMAT in the standby database parameter file are as follows:

```
LOG_ARCHIVE_DEST_1 = 'LOCATION=/fs2/arc_dest/'
LOG_ARCHIVE_FORMAT = log_%t_%s.arc
```

You move the gap sequence logs from the primary file system to the standby file system, renaming them according to values for the LOG_ARCHIVE_DEST_1 and LOG_ARCHIVE_FORMAT initialization parameters at the standby site:

```
% cp /fs1/arc_dest/log_1_250.arc /fs2/arc_dest/log_1_250.arc
% cp /fs1/arc_dest/log_1_251.arc /fs2/arc_dest/log_1_251.arc
% cp /fs1/arc_dest/log_1_252.arc /fs2/arc_dest/log_1_252.arc
```

Step 9: Apply the Logs in the Gap Sequence to the Standby Database.

Now you can apply the gap sequence logs using the RECOVER AUTOMATIC STANDBY DATABASE statement. This statement uses the values of the LOG_ARCHIVE_DEST_1 and LOG_ARCHIVE_FORMAT parameters to construct the target filename. Once the gap sequence logs have been applied, the standby database is synchronized with the primary database and can be placed in managed recovery mode.

While connected to the standby database in SQL*Plus, recover the database using the AUTOMATIC option:

```
SQL> RECOVER AUTOMATIC STANDBY DATABASE;
ORA-00279: change 35083 generated at 08/16/1999 14:08:37 needed for thread 2
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
CANCEL
Media recovery cancelled.
SQL>
```

After recovering the gap sequence logs, Oracle prompts you for the name of a log that does not exist. The reason is that the recovery process does not know about the logs archived to the STANDBY service by the primary database. Cancel recovery at this point:

```
SQL> CANCEL
```

Step 10: Place the Standby Database in Managed Recovery Mode.

You can now enable managed recovery using the RECOVER MANAGED STANDBY DATABASE statement. You decide to use the TIMEOUT option of the RECOVER statement to specify a time interval of 20 minutes so that Oracle waits the specified number of minutes to write the requested archived log entry to the directory of the standby database control file. If the requested archived log entry is not written to the standby database control file directory within the specified time interval, the recovery operation is canceled.

While connected to the standby database using SQL*Plus, place the standby database in managed recovery mode:

```
SQL> RECOVER MANAGED STANDBY DATABASE TIMEOUT 20;
```

Oracle now begins managed recovery. As the primary database archives redo logs to the standby site, the standby database automatically applies them.

Scenario 2: Creating a Standby Database on a Remote Host

This scenario describes the creation of a standby database STANDBY1 on a remote host. The following assumptions are being made:

- You can perform a consistent backup.
- TCP/IP is used to connect to primary and standby databases.
- The standby database is part of a managed recovery environment.
- Th remote host name is STBYHOST.
- `PRMYinit.ora` is the initialization parameter file for the primary database.
- `STBYinit.ora` is the initialization parameter file for the standby database.

Step 1: Back Up the Primary Database Datafiles.

Create the backup that will form the basis for the standby database.

1. Query the primary database to determine the datafiles. Invoke SQL*Plus and query the V$DATAFILE view to obtain a list of the primary database datafiles, as the following example shows:

   ```
   SQL> SELECT name FROM v$datafile;
   NAME
   -----------------------------------------------------------------------
   /vobs/oracle/dbs/dbf_1.f
   1 row selected.
   ```

2. Shut down the primary database to make a consistent backup of the datafiles:

   ```
   SQL> SHUTDOWN IMMEDIATE;
   ```

3. Copy the primary database datafiles to a temporary location (/backup), as the following example shows:

   ```
   % cp /vobs/oracle/dbs/dbf_1.f /backup
   ```

4. Reopen the primary database as follows:

   ```
   SQL> STARTUP PFILE=PRMYinit.ora;
   ```

Step 2: Create the Standby Database Control File.

1. Before you create the standby database control file, ensure that the primary database is in ARCHIVELOG mode and that automatic archival is enabled. Issue the ARCHIVE LOG LIST statement:

   ```
   SQL> ARCHIVE LOG LIST
   ```

   If the output from the ARCHIVE LOG LIST statement displays "No Archive Mode," perform the following steps:

   a. Shut down the primary database as follows:

      ```
      SQL> SHUTDOWN IMMEDIATE;
      ```

   b. Start and mount the primary database instance without opening it:

      ```
      SQL> STARTUP MOUNT PFILE=PRMYinit.ora;
      ```

   c. Set the log archive mode as follows:

      ```
      SQL> ALTER DATABASE ARCHIVELOG;
      ```

   d. Open the primary database:

      ```
      SQL> ALTER DATABASE OPEN;
      ```

2. Create the standby database control file by issuing the following statement:

   ```
   SQL> ALTER DATABASE CREATE STANDBY CONTROLFILE AS '/backup/stbycf.f'
   ```

   The standby database control file and the primary database datafiles are in the same temporary location at the primary site to make copying to the standby site easier.

Step 3: Transfer the Datafiles and Control File to the Standby Site.

Copy the primary database datafiles and the standby control file from the temporary location at the primary site to the standby site, as the following example shows:

```
% rcp /backup/* STBYHOST:/fs2/oracle/stdby
```

Step 4: Configure the Network Files.

This scenario assumes that the TCP/IP network protocol is used to connect to the primary and the standby databases. This step involves editing the following parameter files:

- `tnsnames.ora` file on the primary site
- `listener.ora` file on the standby site

1. Configure the `tnsnames.ora` file.

   You need to edit the `tnsnames.ora` file and add an entry with the following format:

   ```
   standby_service_name = (DESCRIPTION=
      (ADDRESS=(PROTOCOL=tcp) (PORT=port_number)(HOST=host_name))
      (CONNECT_DATA=(SID=standby_sid)))
   ```

   The `tnsnames.ora` file is typically located in the $ORACLE_HOME/network/admin directory on the primary site. Substitute appropriate values for standby_service_name, port_number, host_name, and standby_sid, as the following example shows:

   ```
   standby1 = (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)
            (PORT=5112)(HOST=STBYHOST))
            (CONNECT_DATA=(SID=stdby1)))
   ```

2. Configure the `listener.ora` file.

   You need to edit the `listener.ora` file and add two entries with the following format:

   ```
   STANDBY_LISTENER = (ADDRESS_LIST=(ADDRESS=(PROTOCOL=tcp)
                                    (PORT=port_number)(HOST=host_name)))

   SID_LIST_STANDBY_LISTENER = (SID_LIST=
                     (SID_DESC=(SID_NAME=standby_sid)(ORACLE_HOME=/oracle_home)))
   ```

   The `listener.ora` file is typically located in the $ORACLE_HOME/network/admin directory on the standby site. Substitute appropriate values for port_number, host_name, standby_sid, and oracle_home, as the following example shows:

```
STDBY1_LISTENER = (ADDRESS_LIST=(ADDRESS=(PROTOCOL=tcp)
                         (PORT=5112)(HOST=STBYHOST)))

SID_LIST_STDBY1_LISTENER = (SID_LIST=
          (SID_DESC=(SID_NAME=stdby1)(ORACLE_HOME=/oracle)))
```

Make sure the SID_NAME in the `listener.ora` file matches the SID in the `tnsnames.ora` file. Also, make sure the PORT and HOST values are the same in the two files. Note that you have the option of creating a new listener or adding a new address to an existing listener.

Step 5: Start the Listener on the Standby Site.

The two entries that you added in step 4 define a listener to listen for connections to standby database stdby1 on port 5112. In this step, you must start the `stdby1_listener` listener. For example:

```
% lsnrctl start stdby1_listener;
```

Step 6: Configure the Standby Initialization Parameter File.

1. Copy the primary database initialization parameter file from the primary site to the standby site. From the standby site, issue a command similar to the following:

```
% rcp /vobs/oracle/dbs/PRMYinit.ora STBYHOST:/fs2/oracle/stdby/STBYinit.ora
```

2. Edit the standby initialization parameter file (`STBYinit.ora`). Edit the following parameters:

Parameter	Value
CONTROL_FILES	stbycf.f
STANDBY_ARCHIVE_DEST	/fs2/oracle/stdby/
LOG_ARCHIVE_DEST_1	/fs2/oracle/stdby/
LOG_ARCHIVE_FORMAT	stdby_%t_%s
DB_FILE_NAME_CONVERT	('/vobs/oracle/dbs','/fs2/oracle/stdby')

Parameter	Value
LOG_FILE_NAME_ CONVERT	('/vobs/oracle/dbs','/fs2/oracle/stdby')
LOG_ARCHIVE_START	FALSE

The `STBYinit.ora` file looks as follows:

```
#
#parameter file STBYinit.ora
#

db_name=prod1                  #The same as PRMYinit.ora

# The following two parameters have been changed from PRMYinit.ora
control_files=/fs2/oracle/stdby/stbycf.f
lock_name_space=stdby1

# The following parameters are the same as PRMYinit.ora
audit_trail=FALSE
o7_dictionary_accessibility=FALSE
global_names=FALSE
db_domain=regress.rdbms.dev.us.oracle.com
commit_point_strength=1

processes=30
sessions=30
transactions=21
transactions_per_rollback_segment=21
distributed_transactions=10
db_block_buffers=100
shared_pool_size=4000000
ifile=/vobs/oracle/work/tkinit.ora # Verify that file exists on the standby site
                                   # and that the file specification is valid

# specific parameters for standby database
log_archive_format = stdby_%t_%s.arc
standby_archive_dest=/fs2/oracle/stdby/
log_archive_dest_1='LOCATION=/fs2/oracle/stdby/'
db_file_name_convert=('/vobs/oracle/dbs','/fs2/oracle/stdby')
log_file_name_convert=('/vobs/oracle/dbs','/fs2/oracle/stdby')
log_archive_start=FALSE
log_archive_trace=127
```

Step 7: Copy the Standby Initialization Parameter File.

1. Make a copy of the `STBYinit.ora` file by issuing the following command:

   ```
   % cp STBYinit.ora Failover.ora
   ```

   Edit `Failover.ora` so if you fail over to the stdby1 standby database, you can use the `Failover.ora` file as the initialization parameter file for the new primary database. Make sure you use appropriate values for the LOG_ARCHIVE_DEST_*n* parameters.

2. Edit the `tnsnames.ora` file on the standby site in case failover to the standby database occurs. See step 4 for information on how to configure the `tnsnames.ora` file.

Step 8: Start the Standby Database.

Start the standby database to enable archiving.

1. Set the ORACLE_SID environment variable to the same value as the SID parameter in the `tnsnames.ora` file on the primary site and the `listener.ora` file on the standby site as follows:

   ```
   % setenv ORACLE_SID stdby1
   ```

2. Start SQL*Plus:

   ```
   SQL> CONNECT sys/sys_password as sysdba
   ```

3. Start the standby database instance without mounting the database:

   ```
   SQL> STARTUP NOMOUNT PFILE=STBYinit.ora;
   ```

4. Mount the standby database:

   ```
   SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
   ```

Step 9: Configure the Primary Initialization Parameter File.

1. Specify the archive destination by adding the following entry to the `PRMYinit.ora` file:

   ```
   LOG_ARCHIVE_DEST_2 = 'SERVICE=standby1 MANDATORY REOPEN=60'
   ```

2. Enable the archive destination state by adding the following entry to the `PRMYinit.ora` file:

   ```
   LOG_ARCHIVE_DEST_STATE_2 = ENABLE
   ```

3. Issue the following statements to ensure that the initialization parameters you have set in this step take effect:

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='SERVICE=standby1 MANDATORY REOPEN=60';
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;
```

Step 10: Apply the Logs in the Gap Sequence.

1. On the primary database, archive the current redo log as follows:

```
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

2. On the standby database, run the following SQL script to identify the archived redo logs in the gap sequence:

```
SELECT high.thread#, "LowGap#", "HighGap#"
FROM
(
    SELECT thread#, MIN(sequence#)-1 "HighGap#"
    FROM
    (
        SELECT a.thread#, a.sequence#
        FROM
        (
            SELECT *
            FROM v$archived_log
        ) a,
        (
            SELECT thread#, MAX(next_change#)gap1
            FROM v$log_history
            GROUP BY thread#
        ) b
        WHERE a.thread# = b.thread#
        AND a.next_change# > gap1
    )
    GROUP BY thread#
) high,

(
    SELECT thread#, MIN(sequence#) "LowGap#"
    FROM
    (
        SELECT thread#, sequence#
        FROM v$log_history, v$datafile
        WHERE checkpoint_change# <= next_change#
        AND checkpoint_change# >= first_change#
    )
    GROUP BY thread#
) low
WHERE low.thread# = high.thread#;


     THREAD#    LowGap#    HighGap#
    ---------- ---------- ----------
             1         90         92
```

3. On the primary database, obtain the filenames of the logs in the gap sequence by performing a query on the V$ARCHIVED_LOG view as follows:

```
SELECT name FROM v$archived_log
WHERE thread#=1 AND sequence#<=92 AND sequence#>=90;

NAME
-----------------------------------------
/vobs/oracle/dbs/r_1_90.arc
/vobs/oracle/dbs/r_1_91.arc
/vobs/oracle/dbs/r_1_92.arc
```

4. Transfer the logs in the gap sequence from the primary database to the standby database as follows:

```
% rcp /vobs/oracle/dbs/r_1_90.arc STBYHOST:/fs2/oracle/stdby/stdby_1_90.arc
% rcp /vobs/oracle/dbs/r_1_91.arc STBYHOST:/fs2/oracle/stdby/stdby_1_91.arc
% rcp /vobs/oracle/dbs/r_1_92.arc STBYHOST:/fs2/oracle/stdby/stdby_1_92.arc
```

5. On the standby database, issue the following SQL statement to manually apply the logs in the gap sequence:

```
SQL> RECOVER STANDBY DATABASE;
```

Step 11: Place the Standby Database in Managed Recovery Mode.

On the standby database, enable managed recovery by issuing the following SQL statement:

```
SQL> RECOVER MANAGED STANDBY DATABASE;
```

Scenario 3: Accommodating Physical Changes in the Primary Database

This scenario describes the procedures you should follow when a physical change is made in the primary database. The following topics are covered:

- Adding a Datafile to the Primary Database
- Renaming a Datafile in the Primary Database

- Deleting a Datafile or Tablespace in the Primary Database
- Adding or Removing Online Redo Logs
- Altering Control Files
- Refreshing the Standby Database Control File
- Physical Changes That Require You to Rebuild the Standby Database

Adding a Datafile to the Primary Database

To maintain consistency when you add a datafile to the primary database, you must add a corresponding datafile to the standby database. Otherwise, changes in the online redo logs that relate to the new datafile in the primary database will not be applied to the standby database.

If	Then
You create a new datafile in the primary database	Create a new datafile in the standby database.
You transfer an existing datafile from another database to the primary database	Copy the datafile from the primary database to the standby database. When you copy the datafile, you preserve the contents of the datafile.

Creating a New Datafile in the Standby Database

1. In the primary database, assume you create a datafile as follows:

   ```
   SQL> CREATE TABLESPACE tbs_4 DATAFILE 'tbs_4.f' SIZE 2M;
   ```

2. Start the standby database:

   ```
   SQL> STARTUP NOMOUNT PFILE=STBYinit.ora;
   ```

3. On the primary site, force a log switch as follows:

   ```
   SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
   ```

4. Place the standby database in managed recovery mode:

   ```
   SQL> RECOVER MANAGED STANDBY DATABASE;
   ```

   This SQL statement causes Oracle to stop applying archived redo logs because a datafile on the primary site does not exist on the standby site. Messages similar to the following are displayed when you try to archive the redo logs:

```
ORA-00283: recovery session canceled due to errors
ORA-01157: cannot identify/lock data file 4 - see DBWR trace file
ORA-01110: data file 4: '/vobs/oracle/dbs/stdby/tbs_4.f'
```

The error messages indicate that the datafile has been added to the standby database control file, but the datafile has not been created yet.

5. On the standby site, create the new datafile by issuing the following statement:

```
SQL> ALTER DATABASE CREATE DATAFILE '/vobs/oracle/dbs/stdby/tbs_4.f'
      AS '/vobs/oracle/dbs/stdby/tbs_4.f';
```

6. On the standby site, resume applying archived redo logs by issuing the following statement:

```
SQL> RECOVER MANAGED STANDBY DATABASE;
```

Copying a Datafile from the Primary Database

In the primary database, assume you have transferred a datafile from another database to the primary database.

1. In the primary database, back up the new datafile:

```
SQL> ALTER TABLESPACE tbs_4 BEGIN BACKUP;
```

2. Copy the datafile to a temporary directory:

```
% cp tbs_4.f /backup
```

3. In the primary database, issue the following statement:

```
SQL> ALTER TABLESPACE tbs_4 END BACKUP;
```

4. Copy the `tbs_4.f` datafile from the backup temporary directory to the standby site.

```
% cp /backup/tbs_4.f STBYHOST:/vobs/oracle/dbs/stdby
```

5. On the standby site, resume applying archived redo logs by issuing the following statement:

```
SQL> RECOVER MANAGED STANDBY DATABASE;
```

Refreshing the Standby Database Control File After You Add a Datafile to the Primary Database

After you add a datafile to the primary database and you either create or copy a corresponding datafile to the standby database, the status field in the V$DATAFILE view will contain the "RECOVER" value. The "RECOVER" value indicates that the datafile needs to be recovered. The "RECOVER" value is not accurate in this situation. You need to refresh the standby database control file to get an accurate value in the status field of the V$DATAFILE view. See Refreshing the Standby Database Control File on page 5-26 for additional information.

Note: While refreshing the standby database control file will give you an accurate value, it may invalidate managed recovery. If managed recovery is invalidated, your only option is manual recovery.

Renaming a Datafile in the Primary Database

When you rename one or more datafiles in the primary database, you also need to rename the corresponding datafiles in the standby database.

1. Rename the datafile at the primary site:

```
SQL> ALTER TABLESPACE tbs_4 OFFLINE;
SQL> ALTER TABLESPACE tbs_4 RENAME DATAFILE ’tbs_4.f’ TO ’tbs_x.f’;
SQL> ALTER DATABASE RECOVER TABLESPACE tbs_4;
SQL> ALTER TABLESPACE tbs_4 ONLINE;
```

2. At the primary site, create the standby database control file by issuing the following statement:

```
SQL> ALTER DATABASE CREATE STANDBY CONTROLFILE AS ’stbycf.f’;
```

3. Ensure that the standby database has applied all of the online redo logs by issuing the following statement:

```
SQL> RECOVER AUTOMATIC STANDBY DATABASE;
```

4. Shut down the standby database with the IMMEDIATE option:

```
SQL> SHUTDOWN IMMEDIATE;
```

5. Copy the standby database control file from the primary site to the standby site, overwriting the control file that exists on the standby site:

```
% rcp stbycf.f STBYHOST:/fs2/oracle/stdby/
```

6. Rename the datafile at the standby site:

```
% mv tbs_4.f tbs_x.f
```

7. Start the standby database instance without mounting the database:

```
SQL> STARTUP NOMOUNT PFILE=STBYinit.ora;
```

8. Mount the standby database:

   ```
   SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
   ```

9. Identify and apply the logs in the gap sequence.
10. On the standby database, enable managed recovery by issuing the following statement:

    ```
    SQL> RECOVER MANAGED STANDBY DATABASE;
    ```

If you do not rename the corresponding file at the standby site, and then attempt to refresh the standby database control file, the standby database will attempt to use the renamed datafile, but it will not find the renamed datafile. Consequently, you will see error messages similar to the following:

```
ORA-00283: recovery session canceled due to errors
ORA-01157: cannot identify/lock data file 4 - see DBWR trace file
ORA-01110: data file 4: '/vobs/oracle/dbs/stdby/tbs_x.f'
```

Deleting a Datafile or Tablespace in the Primary Database

When you delete one or more datafiles or tablespaces in the primary database, you also need to delete the corresponding datafiles or tablespaces in the standby database. You also need to refresh the standby database control file.

1. Delete the datafile at the primary site as follows:

   ```
   SQL> DROP TABLESPACE tbs_4;
   % rm tbs_4.f
   ```

2. Refresh the standby database control file. See Refreshing the Standby Database Control File on page 5-26 for the steps.
3. Delete the corresponding datafile at the standby site as follows:

   ```
   % rm tbs_4.f
   ```

Adding or Removing Online Redo Logs

One method of tuning available to the DBA is changing the size and number of online redo logs. Consequently, when you add or remove an online redo log at the primary site, it is important that you refresh the standby database control file.

1. Add or remove an online redo log as follows:

```
SQL> ALTER DATABASE ADD LOGFILE ’prmy3.log’ SIZE 100K;
```

or

```
SQL> ALTER DATABASE DROP LOGFILE 'prmy3.log’;
```

2. Refresh the standby database control file. See Refreshing the Standby Database Control File on page 5-26 for the steps.

Altering Control Files

If you use the CREATE CONTROLFILE statement in the primary database to change one or more database parameters, you need to refresh the standby database control file. Some of the parameters you can change with this statement are the maximum number of redo log file groups, redo log file members, archived redo log files, data files, or instances that can concurrently have the database mounted and open.

1. On the primary database, alter the control file as follows:

```
SQL> CREATE CONTROLFILE REUSE DATABASE stby1 NORESETLOGS
    LOGFILE  'prmy1.log' SIZE 100K, 'prmy2.log' SIZE 100K
    DATAFILE 'dbf_1.f' SIZE 10M MAXLOGFILES 52 ARCHIVELOG;
```

2. Refresh the standby database control file. See Refreshing the Standby Database Control File on page 5-26 for the steps.

If you use the RESETLOGS clause of the CREATE CONTROLFILE statement, you will invalidate the standby database. Once the standby database is invalidated, your only option is to re-create the standby database.

Refreshing the Standby Database Control File

In some cases, when you make a physical change to the primary database, in addition to making the corresponding change to the standby database, you also need to refresh the standby database control file. The online redo logs do not record changes made to the primary database control file. This section describes the cases where you need to refresh the standby database control file and the steps to follow.

You need to refresh the standby database control file whenever you:

- Rename a datafile
- Delete a datafile
- Delete a tablespace

- Add one or more online redo logs
- Remove one or more online redo logs
- Alter control files

Perform the following steps to keep the standby database control file synchronized with the primary database control file:

1. At the primary site, create the standby database control file by issuing the following statement:

   ```
   SQL> ALTER DATABASE CREATE STANDBY CONTROLFILE AS ’stbycf.f’;
   ```

2. If the standby database is in managed recovery mode, you need to cancel recovery by issuing the following statement:

   ```
   SQL> RECOVER MANAGED STANDBY DATABASE CANCEL;
   ```

3. Shut down the standby database with the IMMEDIATE option:

   ```
   SQL> SHUTDOWN IMMEDIATE;
   ```

4. Copy the standby database control file from the primary site to the standby site, overwriting the control file that exists on the standby site:

   ```
   % rcp stbycf.f STBYHOST:/fs2/oracle/stdby/
   ```

5. Start the standby database instance without mounting the database:

   ```
   SQL> STARTUP NOMOUNT PFILE=STBYinit.ora;
   ```

6. Mount the standby database:

   ```
   SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
   ```

7. Identify and apply the logs in the gap sequence.
8. On the standby database, enable managed recovery by issuing the following statement:

   ```
   SQL> RECOVER MANAGED STANDBY DATABASE;
   ```

   You may get the following error messages when you try to enable managed recovery:

   ```
   ORA-00308: cannot open archived log 'standby1'
   ORA-27037: unable to obtain file status
   SVR4 Error: 2: No such file or directory
   Additional information: 3
   ORA-01547: warning: RECOVER succeeded but OPEN RESETLOGS would get error below
   ```

```
ORA-01152: file 1 was not restored from a sufficiently old backup
ORA-01110: data file 1: '/vobs/oracle/dbs/stdby/tbs_1.f'
```

If you get the ORA-00308 error, cancel recovery by issuing the following statement:

```
SQL> CANCEL
```

These error messages are issued when one or more logs in the gap sequence have not been successfully applied. If you receive these errors, repeat steps 7 and 8.

Physical Changes That Require You to Rebuild the Standby Database

Some physical changes you make to the primary database can invalidate the standby database. Once a standby database is invalidated, your only option is to rebuild it.

The following clauses of the ALTER DATABASE statement invalidate the standby database:

- CLEAR UNARCHIVED LOGFILE
- OPEN RESETLOGS

Scenario 4: Recovering After the NOLOGGING Clause Was Specified

In some SQL statements, the user has the option of specifying the NOLOGGING clause, which indicates that the database operation is not logged in the redo log file. Even though the user specifies the NOLOGGING clause, a redo log record is still written to the redo log. However, when the redo log file is transferred to the standby site and applied to the standby database, a portion of the datafile is unusable and marked as being unrecoverable. When you either activate the standby database, or open the standby database with the read-only option, and attempt to read the range of blocks that are marked as "UNRECOVERABLE," you will see error messages similar to the following:

```
ORA-01578: ORACLE data block corrupted (file # 1, block # 2521)
ORA-01110: data file 1: '/vobs/oracle/dbs/stdby/tbs_1.f'
ORA-26040: Data block was loaded using the NOLOGGING option
```

In order to recover after the NOLOGGING clause was specified, you need to copy the datafile that contains the unjournaled data from the primary site to the standby site. Perform the following steps:

1. Determine which datafiles should be copied.

   a. Issue the following query in the primary database:

   ```
   SQL> SELECT name, unrecoverable_change# FROM v$datafile;
   NAME                                                    UNRECOVERA
   ------------------------------------------------------- ----------
   /vobs/oracle/dbs/tbs_1.f                                      5216
   /vobs/oracle/dbs/tbs_2.f                                         0
   /vobs/oracle/dbs/tbs_3.f                                         0
   /vobs/oracle/dbs/tbs_4.f                                         0
   4 rows selected.
   ```

   b. Issue the following query in the standby database:

   ```
   SQL> SELECT name, unrecoverable_change# FROM v$datafile;
   NAME                                                    UNRECOVERA
   ------------------------------------------------------- ----------
   /vobs/oracle/dbs/stdby/tbs_1.f                                5186
   /vobs/oracle/dbs/stdby/tbs_2.f                                   0
   /vobs/oracle/dbs/stdby/tbs_3.f                                   0
   /vobs/oracle/dbs/stdby/tbs_4.f                                   0
   4 rows selected.
   ```

   c. Compare the query results from the primary and the standby databases.

   Compare the value of the unrecoverable_change# column in both query results. If the value of the unrecoverable_change# column in the primary database is greater than the same column in the standby database, then the datafile needs to be copied from the primary site to the standby site.

   In this example, the value of the unrecoverable_change# in the primary database for the `tbs_1.f` datafile is greater, so you need to copy the `tbs_1.f` datafile to the standby site.

2. On the primary site, back up the datafile that you need to copy to the standby site as follows:

   ```
   SQL> ALTER TABLESPACE system BEGIN BACKUP;
   SQL> EXIT;
   % cp tbs_1.f /backup
   SQL> ALTER TABLESPACE system END BACKUP;
   ```

3. Create a new standby database control file.

   In the primary database, issue the following statement:

   ```
   SQL> ALTER DATABASE CREATE STANDBY CONTROLFILE AS '/backup/stbycf.f;
   ```

4. Shut down the standby database.

   In the standby database, issue the following statement:

   ```
   SQL> SHUTDOWN IMMEDIATE;
   ```

5. Copy the datafile and the standby database control file from the primary site to the standby site as follows:

   ```
   % rcp /backup/* STBYHOST:/fs2/oracle/stdby/
   ```

6. Start the standby database instance without mounting the database:

   ```
   SQL> STARTUP NOMOUNT PFILE=STBYinit.ora;
   ```

7. Mount the standby database:

   ```
   SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
   ```

8. Identify and apply the logs in the gap sequence.
9. On the standby database, enable managed recovery by issuing the following statement:

   ```
   SQL> RECOVER MANAGED STANDBY DATABASE;
   ```

   You may get the following error messages when you try to enable managed recovery:

   ```
   ORA-00308: cannot open archived log 'standby1'
   ORA-27037: unable to obtain file status
   SVR4 Error: 2: No such file or directory
   Additional information: 3
   ORA-01547: warning: RECOVER succeeded but OPEN RESETLOGS would get error below
   ORA-01152: file 1 was not restored from a sufficiently old backup
   ORA-01110: data file 1: '/vobs/oracle/dbs/stdby/tbs_1.f'
   ```

   If you get the `ORA-00308` error, cancel recovery by issuing the following statement:

   ```
   SQL> CANCEL
   ```

   These error messages are issued when one or more logs in the gap sequence have not been successfully applied. If you receive these errors, repeat steps 8 and 9.

Scenario 5: Deciding Which Standby Database to Fail Over to in a

Multiple Standby Database Configuration

Every standby database is associated with one and only one primary database. A single primary database can, however, support multiple standby databases. This scenario identifies the kind of information you need in order to decide which of the multiple standby databases to activate.

One of the important things to consider in a multiple standby database configuration is whether the archive destination is mandatory or optional. The following table lists an advantage and disadvantage for each destination:

Destination	Advantage	Disadvantage

MANDATORY	All archived redo log files are archived to the mandatory archive destination. After you apply the archived redo log files at the standby site, you can ensure that the standby database is up-to-date. Furthermore, you can activate the standby database as the new production database with minimum loss of data.	In some cases, (such as network failure), the archived redo log files cannot reach the mandatory archive destination, causing the archiving of the redo log file to stop. In the worst case, if all of the online redo log files are full, and cannot be archived, the primary database instance will stop working.
		
		You can issue the following SQL query to determine whether the primary database stopped because it was not able to switch to an online redo log:
		
		SELECT decode(count(*),0,'NO','YES') "switch_possible"
		
		FROM v$log
		
		WHERE archived='YES';
		
		If the output from the query displays "Yes," a log switch is possible; if the output displays "No," a log switch is not possible.

Destination	Advantage	Disadvantage
OPTIONAL	The primary database continues to operate normally when archival of the redo logs to the optional archive destination at the standby site is interrupted.	In some cases, (such as network failure), the archived redo log files cannot reach the optional archive destination, causing the archiving of the redo log files to stop. There is the potential for multiple gap sequences in the standby database. The gaps must be resolved and the redo logs must be transferred manually from the primary site to the standby site before automatic archival can resume.

Consider the following recommendations in a multiple standby database configuration:

- Specify at least one remote standby database as a mandatory archive destination. Ideally, choose the standby database with the most stable network link and the most stable system configuration.
- Specify a local archive destination as mandatory. This allows the primary database to archive locally in the case of a network failure.

Suppose the primary database is located in San Francisco and supports five standby databases as follows:

Standby	Location	Type	Description
1	local directory	Mandatory	Local copy of the archived redo logs.
2	San Francisco	Mandatory	Fail over to this standby database when there is physical damage at the primary site. This standby site is connected to the primary site by a local area network.
3	Boston	Optional	Fail over to this standby database when a disaster occurs that affects San Francisco.
4	Los Angeles	Optional	This standby site receives archived redo logs, but does not apply them. See Scenario 9: Standby Database with No Ongoing Recovery on page 5-46 for a description of this type of configuration.

Standby	Location	Type	Description
5	San Francisco	Optional	This standby site receives archived redo logs, and applies them after an 8-hour time lag. See Scenario 10: Standby Database with a Time Lag on page 5-51 for a description of this type of configuration.

Assume that a disaster occurs in San Francisco where the primary site is located, and the primary host is damaged. One of the standby databases must be activated. You cannot assume that the database administrator (DBA) who set up the multiple standby database configuration is available to decide which standby database to fail over to. Therefore, it is imperative to have a disaster recovery plan at each standby site, as well as at the primary site. Each member of the disaster recovery team needs to know about the disaster recovery plan and be aware of the procedures to follow. This scenario identifies the kind of information that the person who is making the decision would need when deciding which standby database to activate. One method of conveying information to the disaster recovery team is to include a ReadMe file at each standby site.

The ReadMe file at each site should describe how to:

- Log on to the local database server as a DBA.
- Log on to each system where the standby databases are located.

  There may be firewalls between systems. The ReadMe file should include instructions for going through the firewalls.
- Log on to other database servers as a DBA.
- Identify the most up-to-date standby database.
- Activate the standby database.
- Configure network settings to ensure that client applications access the newly activated database instead of the original primary database.

The following example shows the contents of a sample ReadMe file:

```
----------------Standby Database Disaster Recovery ReadMe File----------------

Warning:
********************************************************************************
Perform the steps in this procedure only if you are responsible for failing over
to a standby database after the primary database fails.
```

```
If you perform the steps outlined in this file unnecessarily, you may corrupt
the entire database system.
********************************************************************************

Multiple Standby Database Configuration:

No.     Location       Type     IP Address
--- --------------- --------- --------------
 1   San Francisco   Primary   128.1.124.25
 2   San Francisco   Standby   128.1.124.157
 3   Boston          Standby   136.132.1.55
 4   Los Angeles     Standby   145.23.82.16
 5   San Francisco   Standby   128.1.135.24

You are in system No. 3, which is located in Boston.

Perform the following steps to fail over to the most up-to-date and available
standby database:

1.  Log on to the local standby database as a DBA.

    a)  Log on with the following user name and password:

               username: Standby3
               password: zkc722Khn

    b)  Invoke SQL*Plus as follows:

        % sqlplus

    c)  Connect as the DBA as follows:

        CONNECT sys/s23LsdIc AS SYSDBA

2.  Connect to as many remote systems as possible. You can connect to a maximum
    of four systems. System 4 does not have a firewall, so you can connect to it
    directly. Systems 1, 2, and 5 share the same firewall host.  You need to go
    to the firewall host first and then connect to each system.  The IP address
    for the firewall host is 128.1.1.100.  Use the following user name and
    password:
               username: Disaster
               password: 82lhsIW32

3.  Log on to as many remote systems as possible with the following user names
```

and passwords:

Login information:

No.	Location	IP Address	username	password
1	San Francisco	128.1.124.25	Oracle8i	sdd290Ec
2	San Francisco	128.1.124.157	Standby2	ei23nJHb
3		(L o c a l)		
4	Los Angeles	145.23.82.16	Standby4	23HHoe2a
5	San Francisco	128.1.135.24	Standby5	snc#$dnc

4. Invoke SQL*Plus on each remote system you are able to log on to as follows:

```
% sqlplus
```

5. Connect to each remote database as follows:

```
CONNECT sys/password AS SYSDBA
```

The DBA passwords for each location are:

No.	Location	Password
1	San Francisco	x2dwlsd91
2	San Francisco	a239s1DAq
3	(L o c a l)	
4	Los Angeles	owKL(@as23
5	San Francisco	sad_KS13x

6. If you are able to log on to System 1, invoke SQL*Plus and issue the following statements:

```
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP PFILE=PRMYinit.ora;
```

Note: If you are able to execute the STARTUP statement successfully, the primary database has not been damaged. Do not continue with this procedure.

7. Issue the following SQL statements on each standby database (including the one on this machine) that you were able to connect to:

```
SQL> RECOVER MANAGED STANDBY DATABASE CANCEL;
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP NOMOUNT PFILE=STBYinit.ora;
SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
SQL> RECOVER AUTOMATIC STANDBY DATABASE;
       *** press the Return key for each of the prompts
SQL> SELECT THREAD#, MAX(SEQUENCE#) FROM V$LOG_HISTORY GROUP BY THREAD#;

Compare the query results of each standby database. Activate the
standby database with the largest sequence number.

8.  Fail over to the standby database with the largest sequence number.

    On the standby database with the largest sequence number, invoke SQL*Plus
    and issue the following SQL statements:

    SQL> ALTER DATABASE ACTIVATE STANDBY DATABASE;
    SQL> SHUTDOWN IMMEDIATE;
    SQL> STARTUP PFILE=Failover.ora;

------------End of Standby Database Disaster Recovery ReadMe File-------------
```

Scenario 6: Configuring Client Application Failover

When a standby database is activated, it becomes a production database, and is no longer capable of serving as a standby database. Client applications need to redirect their connections from the original primary database to the newly activated production database. This scenario describes the following ways to set up client failover:

- Local TNS Configuration
- Oracle Names Server Configuration
- Transparent Application Failover (TAF) Configuration
- Manual Network Configuration

Local TNS Configuration

The `tnsnames.ora` file can be configured for multiple addresses. In a local TNS configuration, at least one of the addresses should be the address of a standby site. Modify the `tnsnames.ora` file at each client site to ensure that an address for a standby site has been supplied. The `tnsnames.ora` file is typically located in the $ORACLE_HOME/network/admin directory. You can assign multiple addresses to one TNS name and use the FAILOVER option. See the *Net8 Administrator's Guide* for details about how to set multiple addresses and how to use the FAILOVER option. The following example shows an entry that has an address for a standby site in addition to the address for the primary site:

```
ProductDB = (   DESCRIPTION=
                (FAILOVER=ON)
                (LOAD_BALANCE=OFF)
                (ADDRESS=(PROTOCOL=tcp)(PORT=1521)(HOST=PRMYHOST.foo.com))
                (ADDRESS=(PROTOCOL=tcp)(PORT=1521)(HOST=STBYHOST.foo.com))
                (CONNECT_DATA=(SID=db1))
            )
```

In this example, the primary database is located at PRMYHOST.foo.com, and the standby database is located at STBYHOST.foo.com. When the client application connects to ProductDB, it tries to send a connection request to PRMYHOST.foo.com first. If there is no response, the client application tries to send another connection request to STBYHOST.foo.com. When the primary database is down and the standby database is activated, the client application can connect to the new production database automatically.

If the primary database fails after the connection has been established, the client application will not automatically direct the remaining request to the newly activated production database. You must establish the connection to the ProductDB database again.

Oracle Names Server Configuration

If you are using an Oracle Names server, you can change the TNS name settings on the server. You can assign multiple addresses to one TNS name and use the FAILOVER option. See the *Net8 Administrator's Guide* for details about how to set multiple addresses and how to use the FAILOVER option on an Oracle Names server. The format for setting up the TNS name on the Oracle Names server is the same as the format for the local `tnsnames.ora` file. Therefore, the example in Local TNS Configuration also applies to the Oracle Names server configuration.

Transparent Application Failover (TAF) Configuration

The following configurations require that the client reconnect to the database server when the primary database fails over to the standby database:

- Local TNS configuration
- Oracle Names server configuration

- DNS configuration

However, if you are using an Oracle Call Interface (OCI) client, you can use transparent application failover (TAF). TAF is the ability of applications to automatically reconnect to the database if the connection fails. If the client application is not involved in a database transaction, then users may not notice the failure of the primary database server. See *Oracle8i Designing and Tuning for Performance* for details on how to configure TAF. The following example shows address information for the ProductDB database and the Standby1 database:

```
ProductDB=( DESCRIPTION=
             (ADDRESS=(PROTOCOL=tcp)(PORT=1521)(HOST=PRMYHOST.foo.com))
             (CONNECT_DATA=(SID=db1)(FAILOVER_MODE=(BACKUP=Standby1)
                                                  (TYPE=session)
                                                  (METHOD=basic)))
           )
Standby1 =( DESCRIPTION=
             (ADDRESS=(PROTOCOL=tcp)(PORT=1521)(HOST=STBYHOST.foo.com))
             (CONNECT_DATA=(SID=db1))
           )
```

Sequence of events:

1. Client application is connected to the ProductDB database.
2. The primary database in PRMYHOST.foo.com fails.
3. The standby database is activated as the new production database.
4. When the client application fails to connect to PRMYHOST.foo.com, it uses the database specified with the BACKUP parameter in the FAILOVER_MODE clause, and automatically connects to STBYHOST.foo.com.

Manual Network Configuration

Instead of setting the configurations so that the client can automatically fail over to the new production database, you can always choose to manually modify the network settings after the standby database is activated.

You can modify the local `tnsnames.ora` file. Redirect the TNS name pointing to the original production database to the newly activated production database. For example, assume the original TNS setting is as follows:

```
ProductDB=( DESCRIPTION=
                (ADDRESS=(PROTOCOL=tcp)(PORT=1521)(HOST=PRMYHOST.foo.com))
                 (CONNECT_DATA=(SID=db1))
```

```
          )
```

When the primary database on PRMYHOST.foo.com fails, and the standby database in STBYHOST.foo.com is activated, causing it to become the new production database, you need to edit the `tnsnames.ora` file and change the entry to the following:

```
ProductDB=( DESCRIPTION=
               (ADDRESS=(PROTOCOL=tcp)(PORT=1521)(HOST=STBYHOST.foo.com))
                (CONNECT_DATA=(SID=db1))
          )
```

You do not need to change your client application; subsequent connections to the production database will be sent to the new production database on STBYHOST .foo.com.

If you are using an Oracle Names server, make a similar change for the corresponding entry. All clients using this Oracle Names server will send their subsequent connections to the new production database on STBYHOST.foo.com.

You can also change the settings on the DNS server. Change the settings for the domain name, which is used by the clients to locate the production database.

For example, assume the following:

Item	**Value**
Domain name	ProductDB.foo.com
`tnsnames.ora` entry	ProductDB=(DESCRIPTION= (ADDRESS=(PROTOCOL=tcp) (PORT=1521) (HOST=ProductDB.foo.com)) (CONNECT_DATA=(SID=db1))
DNS server entry for domain name	ProductDB.foo.com IN A 136.1.23.15
IP address of STBYHOST.foo.com	128.3.151.63

Change the DNS server entry for the domain name in the DNS server to the following:

```
ProductDB.foo.com   IN  A   128.3.151.63
```

After you change the DNS settings, not all clients know about the change immediately. The old DNS settings may be cached somewhere, causing some clients to continue to use the old settings. The old settings need to be replaced with the new settings.

Scenario 7: Recovering After a Network Failure

When a standby database is in managed recovery mode, the standby database automatically applies archived redo logs as it receives them from the primary database. When the network goes down, automatic archival from the primary database to the standby database stops.

If the standby database is specified as an optional archive destination, then the primary database continues to operate normally.

When the network is up and running again, automatic archival of the archived redo logs from the primary database to the standby database resumes. However, if the standby database is specified as an optional archive destination, and a log switch occurred at the primary site, the standby database has a gap sequence for the time when the network was down. The standby database cannot apply any of the archived redo logs at the standby site until the gap sequence is resolved.

If the standby database is specified as a mandatory archive destination, then the primary database will not archive any redo logs until the network failure is resolved and the primary database is able to archive to the standby site.

The primary database may eventually stall if the network problem is not fixed in a timely manner because the primary database will not be able to switch to an online redo log that has not been archived. You can issue the following SQL query to determine whether the primary database stalled because it was not able to switch to an online redo log:

```
SELECT decode(count(*),0,'NO','YES') "switch_possible"
FROM v$log
WHERE archived='YES';
```

If the output from the query displays "Yes," a log switch is possible; if the output displays "No," a log switch is not possible.

This scenario describes how to recover after a network failure.

1. Identify the network failure.

The V$ARCHIVE_DEST view contains the network error and identifies which standby database cannot be reached. On the primary database, issue the following SQL statement:

```
SQL> SELECT * FROM V$ARCHIVE_DEST
DEST_ID    STATUS    BINDING   NAME_SP TARGET  REOPEN_SEC DESTINATION
FAIL_DATE FAIL_SEQUE FAIL_BLOCK ERROR
---------- --------- --------- ------- ------- ----------
--------------------------------------------------------------------------------
---------
---------- ----------
--------------------------------------------------------------------------------
         1 VALID     MANDATORY SYSTEM  PRIMARY          0 /vobs/oracle/dbs
0          0
         2 ERROR     MANDATORY SYSTEM  STANDBY          5 standby1
26-JUL-99         61          0 ORA-12541: TNS:no listener
         3 INACTIVE  OPTIONAL  SYSTEM  PRIMARY          0
0          0
         4 INACTIVE  OPTIONAL  SYSTEM  PRIMARY          0
0          0
         5 INACTIVE  OPTIONAL  SYSTEM  PRIMARY          0
0          0
```

The query results show there are errors archiving to the standby1 standby database.

2. Try to resolve the network failure.

   The query results in step 1 display the error as `TNS:no listener`. You should check whether the listener on the standby site is started. If the listener is stopped, then start it.

3. Prevent the primary database from stalling, if possible.

   If you are not able to solve the network problem quickly, and if the standby database is specified as a mandatory destination, you should try to prevent the database from stalling by doing one of the following:

   - Disable the mandatory archive destination:

     ```
     SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2 = DEFER;
     ```

     When the network problem is resolved, you can enable the archive destination again:

     ```
     SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2 = ENABLE;
     ```

   - Change the archive destination from mandatory to optional:

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_2 = ’SERVICE=standby1 OPTIONAL
REOPEN=60’;
```

When the network problem is resolved, you can change the archive destination from optional back to mandatory:

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_2 = ’SERVICE=standby1 MANDATORY
REOPEN=60’;
```

In some cases, you may not be the person responsible for correcting the problem. You can periodically query the V$ARCHIVE_DEST view to see if the problem has been resolved.

4. On the primary database, archive the current redo log:

   ```
   SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
   ```

5. Identify and apply the logs in the gap sequence.

   When the network is up and running again, and new archived logs are transferred from the primary to the standby database, they will not be applied to the standby database until the logs in the gap sequence have been applied manually.

   On the standby database, run the following SQL script to identify the archived redo logs in the gap sequence:

   ```
   SELECT high.thread#, "LowGap#", "HighGap#"
   FROM
   (
       SELECT thread#, MIN(sequence#)-1 "HighGap#"
       FROM
       (
           SELECT a.thread#, a.sequence#
           FROM
           (
               SELECT *
               FROM v$archived_log
           ) a,
           (
               SELECT thread#, MAX(next_change#)gap1
               FROM v$log_history
               GROUP BY thread#
           ) b
           WHERE a.thread# = b.thread#
           AND a.next_change# > gap1
       )
       GROUP BY thread#
   ) high,
   ```

```
(
    SELECT thread#, MIN(sequence#) "LowGap#"
    FROM
    (
        SELECT thread#, sequence#
        FROM v$log_history, v$datafile
        WHERE checkpoint_change# <= next_change#
        AND checkpoint_change# >= first_change#
    )
    GROUP BY thread#
) low
WHERE low.thread# = high.thread#;


     THREAD#    LowGap#    HighGap#
    ---------- ---------- ----------
             1         90         92
```

6. On the primary database, obtain the filenames of the logs in the gap sequence by performing a query on the V$ARCHIVED_LOG view as follows:

```
SELECT name FROM v$archived_log
WHERE thread#=1 AND sequence#<=92 AND sequence#>=90;

NAME
-----------------------------------------
/vobs/oracle/dbs/r_1_90.arc
/vobs/oracle/dbs/r_1_91.arc
/vobs/oracle/dbs/r_1_92.arc
```

7. Transfer the logs in the gap sequence from the primary database to the standby database as follows:

```
% rcp /vobs/oracle/dbs/r_1_90.arc STBYHOST:/fs2/oracle/stdby/stdby_1_90.arc
% rcp /vobs/oracle/dbs/r_1_91.arc STBYHOST:/fs2/oracle/stdby/stdby_1_91.arc
% rcp /vobs/oracle/dbs/r_1_92.arc STBYHOST:/fs2/oracle/stdby/stdby_1_92.arc
```

8. On the standby database, issue the following statement to manually apply the gap sequence:

```
SQL> RECOVER AUTOMATIC STANDBY DATABASE;
```

9. Repeat steps 5 through 8 until there are no more gaps.

   A network failure can cause multiple gaps in the standby database. The gaps must be resolved one at a time.

It is important to specify a local directory at the primary site as a mandatory archive destination, so that all of the archived redo logs reside on the same system as the primary database. When the primary system is unreachable, and the primary database is part of a multiple standby database configuration, you can try to identify the archived redo logs at the other standby sites.

Scenario 8: Re-Creating a Standby Database

This scenario describes the case where you have failed over to a standby database and have begun using it as a normal production database. After a period of time, you decide you want to fail back to the original production system and make the production database the standby database again. The general steps are:

1. Create a standby database at the original primary site.
2. Fail over to the standby database at the original primary site.
3. Create a new standby database at the original standby site.

The detailed steps follow:

1. Copy the original standby initialization parameter file from the original standby site to the original primary site as follows:

   ```
   % rcp STBYinit.ora PRMYHOST:fallback.ora
   ```

   The `fallback.ora` file will become the standby initialization parameter file for the standby database at the original primary site.

2. On the original primary site, configure the `fallback.ora` file. You need to modify the following parameters:

Parameter	Value
LOCK_NAME_SPACE	fallback
LOG_ARCHIVE_FORMAT	r_%t_%s.arc
STANDBY_ARCHIVE_DEST	/vobs/oracle/dbs/
LOG_ARCHIVE_DEST_1	/vobs/oracle/dbs/
DB_FILE_NAME_ CONVERT	('/fs2/oracle/stdby','/vobs/oracle/dbs')
LOG_FILE_NAME_ CONVERT	('/fs2/oracle/stdby','/vobs/oracle/dbs')

The `fallback.ora` file looks as follows:

```
#
#parameter file fallback.ora
#

db_name=prod1              #The same as PRMYinit.ora

control_files=/fs2/oracle/stdby/stbycf.f
lock_name_space=fallback;

audit_trail=FALSE
o7_dictionary_accessibility=FALSE
global_names=FALSE
db_domain=regress.rdbms.dev.us.oracle.com
commit_point_strength=1

processes=30
sessions=30
transactions=21
transactions_per_rollback_segment=21
distributed_transactions=10
db_block_buffers=100
shared_pool_size=4000000
ifile=/vobs/oracle/work/tkinit.ora

# specific parameters for standby database
log_archive_format = r_%t_%s.arc
standby_archive_dest=/vobs/oracle/dbs/
log_archive_dest_1='LOCATION=/vobs/oracle/dbs/'
log_archive_dest_state_1 = ENABLE
db_file_name_convert=('/fs2/oracle/stdby','/vobs/oracle/dbs')
log_file_name_convert=('/fs2/oracle/stdby','/vobs/oracle/dbs')
log_archive_start=FALSE
log_archive_trace=127
```

3. On the original primary site, create or modify the primary initialization parameter file.

   You need to supply appropriate values for the LOG_ARCHIVE_DEST_1 and LOG_ARCHIVE_DEST_STATE_1 parameters.

   Note: Step 7 of Scenario 2: Creating a Standby Database on a Remote Host suggested that you make a copy of the standby database initialization parameter file. If you made a copy, then you can modify the copy in this step.

4. On the original standby site, back up the production database data files.
5. On the original standby site, create the standby database control file.

6. Copy the production database data files and the standby database control file from the original standby site to the original primary site.
7. Archive the current online redo log and shut down the production database to prevent further modifications as follows:

   ```
   SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
   SQL> SHUTDOWN IMMEDIATE;
   ```

8. Copy the archived redo logs that were generated after the production database data files were backed up from the original standby site to the original primary site, as follows:

   ```
   % rcp /fs2/oracle/stdby/stdby_1_102.arc PRMYHOST:/vobs/oracle/dbs/r_1_102.arc
   % rcp /fs2/oracle/stdby/stdby_1_103.arc PRMYHOST:/vobs/oracle/dbs/r_1_103.arc
   ```

9. On the original primary site, start and mount the standby database:

   ```
   SQL> CONNECT sys/sys_password AS SYSDBA
   SQL> STARTUP NOMOUNT PFILE=fallback.ora;
   SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
   ```

10. Apply the logs in the gap sequence:

    ```
    SQL> RECOVER AUTOMATIC STANDBY DATABASE;
    ```

11. Activate the standby database:

    ```
    SQL> ALTER DATABASE ACTIVATE STANDBY DATABASE;
    SQL> SHUTDOWN IMMEDIATE;
    ```

12. Start the production database at the original primary site:

    ```
    SQL> STARTUP PFILE=PRMYinit.ora;
    ```

13. Configure the network settings to enable client applications to access the production database.
14. Create a new standby database at the original standby site. See Scenario 2: Creating a Standby Database on a Remote Host on page 5-14 for the steps to follow to create a standby database.

Scenario 9: Standby Database with No Ongoing Recovery

This scenario describes what the DBA needs to do when archived online redo logs are automatically transferred to the standby site, but they are not automatically applied to the standby database.

For example, suppose a standby database is set up on a host running a Web server. The workload for the host is very heavy, and you do not want to add workload to the host by automatically applying redo logs as they arrive from the primary site. You decide to let the host receive archived redo logs without applying them. Thus, the system resources for applying the redo logs will be saved. Later, if you decide to open the standby database as a read-only database to perform some queries, or activate the standby database as a production database, you can stop the Web server and apply all of the redo logs at one time.

This scenario assumes that you already have a standby database set up. Steps 1 through 10 in Scenario 2: Creating a Standby Database on a Remote Host on page 5-14 describe how to set up a standby database.

The listener for the standby database should be started. The standby database should be started and mounted, but not in recovery mode. (Thus, the time and resources for applying redo logs will be saved.)

This scenario is similar to Scenario 2: Creating a Standby Database on a Remote Host. In Scenario 2, the standby database is in managed recovery mode. When a standby database is in managed recovery mode, it automatically applies archived redo logs received from the primary site.

Managing a Standby Database with No Ongoing Recovery

When the archived redo logs are not immediately applied to the standby database, there is no ongoing recovery at the standby site. This section describes the tasks a DBA needs to consider when managing a standby database with no ongoing recovery. This section covers the following topics:

- Adding a Datafile to the Primary Database
- Responding to a Change in the Primary Database Control File
- Responding When the NOLOGGING Clause Is Specified
- Responding to Network Problems

Adding a Datafile to the Primary Database

When a datafile is added to the primary site, you need to decide what action to take on the standby site. You have the following options:

- Copy the new datafile from the primary site to the standby site when the datafile is added.
- Do not copy the new datafile from the primary site.

When you attempt to archive the redo logs, messages similar to the following are displayed:

```
ORA-00283: recovery session canceled due to errors
ORA-01157: cannot identify/lock data file 4 - see DBWR trace file
ORA-01110: data file 4: '/vobs/oracle/dbs/stdby/tbs_4.f'
```

Create the correponding new datafile by issuing the following statement:

```
SQL> ALTER DATABASE CREATE DATAFILE '/vobs/oracle/dbs/stdby/tbs_4.f' AS
  '/vobs/oracle/dbs/stdby/tbs_4.f';
```

Responding to a Change in the Primary Database Control File

When the control file on the primary site has been altered, if you want to account for it on the standby site, you should apply the redo logs at the standby site by issuing the following statement:

```
SQL> RECOVER AUTOMATIC STANDBY DATABASE;
```

After you apply the redo logs at the standby site, refresh the standby database control file. See Refreshing the Standby Database Control File on page 5-26 for the steps.

Responding When the NOLOGGING Clause Is Specified

In order to recover after the NOLOGGING clause was specified at the primary site, you must do the following:

1. Apply the redo logs at the standby site by issuing the following statement:

   ```
   SQL> RECOVER AUTOMATIC STANDBY DATABASE;
   ```

2. Copy the affected datafiles and standby database control file to the standby site.

See Scenario 4: Recovering After the NOLOGGING Clause Was Specified on page 5-28 for the procedure to follow.

If you can afford to lose the changes incurred by specifying NOLOGGING at the primary site, you may choose to do nothing.

Responding to Network Problems

If this standby site is mandatory, you do not need to do anything at the standby site. If the standby site is optional, you should resolve the gaps immediately by applying the redo logs. It is very important to resolve the gaps in a timely manner. Otherwise, you run the risk of not being able to access the archived redo logs when the primary site is unreachable.

Activating a Standby Database with No Ongoing Recovery

Before you activate the standby database, you should apply all applicable redo logs. You must resolve any gaps in the redo log sequence before you activate the standby database, as outlined in the following steps:

1. Run the following SQL script to identify the archived redo logs in the gap sequence:

```
SELECT high.thread#, "LowGap#", "HighGap#"
FROM
(
    SELECT thread#, MIN(sequence#)-1 "HighGap#"
    FROM
    (
        SELECT a.thread#, a.sequence#
        FROM
        (
            SELECT *
            FROM v$archived_log
        ) a,
        (
            SELECT thread#, MAX(sequence#)gap1
            FROM v$log_history
            GROUP BY thread#
        ) b
        WHERE a.thread# = b.thread#
        AND a.sequence# > gap1
    )
    GROUP BY thread#
) high,

(
    SELECT thread#, MIN(sequence#) "LowGap#"
    FROM
    (
        SELECT thread#, sequence#
        FROM v$log_history, v$datafile
        WHERE checkpoint_change# <= next_change#
        AND checkpoint_change# >= first_change#
    )
    GROUP BY thread#
) low
WHERE low.thread# = high.thread#;


THREAD#    LowGap#    HighGap#
---------- ---------- ----------
         1         90         92
```

The gap sequence is the LowGap# to the HighGap#. In this example, the gap sequence is 90, 91, and 92 for thread 1. If no gap sequence is selected in this step, go to step 5.

2. Identify the archived redo logs that need to be transferred from the primary site to the standby site.

   Obtain the filenames of the logs in the gap sequence by performing a query on the V$ARCHIVED_LOG view as follows:

```
SELECT name FROM v$archived_log
WHERE thread#=1 AND sequence#<=92 AND sequence#>=90;

NAME
-----------------------------------------
/vobs/oracle/dbs/r_1_90.arc
/vobs/oracle/dbs/r_1_91.arc
/vobs/oracle/dbs/r_1_92.arc
```

3. Transfer the logs in the gap sequence from the primary database to the standby database as follows:

```
% rcp /vobs/oracle/dbs/r_1_90.arc STBYHOST:/fs2/oracle/stdby/stdby_1_90.arc
% rcp /vobs/oracle/dbs/r_1_91.arc STBYHOST:/fs2/oracle/stdby/stdby_1_91.arc
% rcp /vobs/oracle/dbs/r_1_92.arc STBYHOST:/fs2/oracle/stdby/stdby_1_92.arc
```

   It is important to specify a local directory at the primary site as a mandatory archive destination so that all of the archived redo logs reside on the same system as the primary database. When the primary system is unreachable, and the primary database is part of a multiple standby database configuration, you can try to identify the archived redo logs at the other standby sites.

4. On the standby database, issue the following statement to manually apply the logs in the gap sequence:

```
SQL> RECOVER AUTOMATIC STANDBY DATABASE;
```

   Repeat steps 1 through 4 until there are no more gaps.

5. Activate the standby database by issuing the following statement:

```
SQL> ALTER DATABASE ACTIVATE STANDBY DATABASE;
```

6. Shut down the standby database instance as follows:

```
SQL> SHUTDOWN IMMEDIATE;
```

7. Start the new production instance.

   If necessary, build the parameter file for the new production instance. You can build it from the parameter file for the standby database. Then, you can issue the following statement at the standby database:

   ```
   SQL> STARTUP PFILE=FailOver.ora;
   ```

Scenario 10: Standby Database with a Time Lag

In managed recovery mode, the standby database automatically applies redo logs when they arrive from the primary database. But in some cases, you may not want the logs to be applied immediately, because you want to create a time lag between the archiving of a redo log at the primary site and the application of the log at the standby site. A time lag can protect against the transfer of corrupted or erroneous data from the primary site to the standby site.

For example, suppose you run a batch job every night on the primary database. Unfortunately, you accidently ran the batch job twice and you did not realize the mistake until the batch job completed for the second time. Ideally, you need to roll back the database to the point in time before the batch job began. A primary database that has a standby database with a time lag (for example, 8 hours) could help you to recover. You could activate the standby database with the time lag and use it as the new production database.

To create a standby database with a time lag, use manual recovery mode instead of managed recovery mode. The online redo log files can still be automatically transferred, just as with managed recovery mode. But, the log files are not immediately applied to the standby database. You can run a script periodically to check whether each redo log is old enough to meet your desired time lag. The script will move the redo logs that are old enough to the manual recovery directory and then, manually apply them.

This scenario use a 4-hour time lag in the examples and covers the following topics:

- Creating a Standby Database with a Time Lag
- Managing a Standby Database with a Time Lag
- Rolling Back the Database to a Specified Time
- Bypassing the Time Lag and Activating the Standby Database

Readers of this scenario are assumed to be familiar with the procedures for creating a typical standby database. The details have been omitted from the steps outlined in this scenario. Refer to Scenario 2: Creating a Standby Database on a Remote Host on page 5-14 for details of normal standby database setup.

Creating a Standby Database with a Time Lag

Perform the following steps to create a standby database with a time lag:

1. Back up the datafiles and create the standby database control file at the primary site.
2. Transfer the datafiles and the standby database control file to the standby site.
3. Configure the `tnsnames.ora` and `listener.ora` network files.
4. Start the listener on the standby site.
5. Configure the standby initialization parameter file.

   Edit the STANDBY_ARCHIVE_DEST and LOG_ARCHIVE_DEST_1 parameters.

   The LOG_ARCHIVE_DEST_1 parameter specifies the location of the archived redo logs. You must use this directory when performing manual recovery. You must use the directory specified by the STANDBY_ARCHIVE_DEST parameter when performing managed recovery.

   For example, assume you have set these parameters to the following values:

   ```
   STANDBY_ARCHIVE_DEST=/fs2/oracle/stdby_log/
   LOG_ARCHIVE_DEST_1    ='LOCATION=/fs2/oracle/stdby/'
   ```

6. Mount the standby database. Do not open it or put it in managed recovery mode.
7. Configure the primary initialization parameter file.
8. Identify and apply the logs in the gap sequence.
9. Write a script that you can periodically run to check the log files in the managed recovery directory and move the log files that have a specified timestamp to the manual recovery directory. If new redo logs are being moved, start the manual recovery mode and apply the newly moved redo logs.

   The following PERL script performs what is outlined in this step:

   ```
   #!/usr/local/bin/perl

   #How many hours the standby database should lag behind the primary database
   $LAG_HOUR = 4;
   ```

```
#The manual recovery directory
$DEST_DIR = '/fs2/oracle/stdby/';

#The flag for whether there are new logs to be applied.
$needApply = 0;

#Check the managed recovery directory
while ( </fs2/oracle/stby_log/*.arc> ) {
    # Get the timestamp of the file
    $file_time = (stat($_))[9];
    # See if the file is "old enough"
    if ( time-$file_time > $LAG_HOUR*60*60 ) {
        print  "mv $_ $DEST_DIR\n";
        system "mv $_ $DEST_DIR";
        $needApply = 1;
     }
}
#If redo logs were moved in this round, apply them
if ( $needApply == 1 ) {
    system "/usr/Lagged_Standby/ApplyLog";
}
```

The SHELL script (/usr/Lagged_Standby/ApplyLog) used to apply the redo logs consists of the following:

```
sqlplus internal << EOF

    recover automatic standby database;
    cancel
    exit

EOF
```

10. Refer to your platform-specific documentation for information on how to create a job that is triggered at specific times throughout the day.

For example, in UNIX, you can write a CRON job file. Issue the **man crontab** command at the UNIX command shell to get help on how to write a CRON job file.

You should decide how frequently to schedule the job. The more frequently the job runs, the more granularity you can get, and the less volume of logs need to be applied at a time. The minimum granularity is one redo log file.

For example, if you want the job to run every 10 minutes, you can write the following CRON job file, (suppose the script file written in step 9 is /usr/Lagged_Standby/lag4.pl):

```
0,10,20,30,40,50 * * * * /usr/Lagged_Standby/lag4.pl
```

Suppose the preceding CRON job file is /usr/Lagged_Standby/jobs.cron. You can schedule the CRON job by issuing the following command:

```
% crontab /usr/Lagged_Standby/jobs.cron
```

The `lag4.pl` script will run every 10 minutes. The exact time lag will appear after the specified time (in this example, it is 4 hours).

Managing a Standby Database with a Time Lag

The runtime scenario of a standby database with a time lag is slightly different from a standby database with no time lag, because the application of the online redo logs lags behind.

As the DBA, you need to keep the following tasks in mind when managing a standby database with a time lag:

- When a datafile is added to the primary site, you need to copy the new datafile to the standby site.
- Before you refresh the standby database control file, you must apply all of the archived redo logs that have been transferred to the standby site, but have not been applied to the standby database. Perform the following steps:

  1. Manually move all archived redo logs to the manual recovery directory, as the following example shows:

     ```
     % mv /fs2/oracle/stdby_log/*.arc /fs2/oracle/standby/
     ```

  2. Apply all of the archived redo logs as follows:

     ```
     SQL> RECOVER AUTOMATIC STANDBY DATABASE;
     ```

  3. Refresh the standby database control file.

  This way, the standby database control file will be updated, but the lag disappears. You need to wait for the specified time lag (for example, 4 hours) to reinforce the lag.

- To recover after the NOLOGGING clause was specified at the primary site, you must apply all of the archived redo logs first. If you can afford to lose the changes incurred by specifying NOLOGGING at the primary site, you can choose to do nothing.

  1. Manually move all archived redo logs to the manual recovery directory. For example:

```
% mv /fs2/oracle/stdby_log/*.arc /fs2/oracle/standby/
```

2. Apply all of the archived redo logs:

```
SQL> RECOVER AUTOMATIC STANDBY DATABASE;
```

3. Copy the affected datafiles and standby database control file.

This way, the standby database control file will be updated, but the lag disappears. You need to wait for the specified time lag (for example, 4 hours) to reinforce the lag.

- When there are network problems:

  If this standby site is mandatory, you do not need to do anything at the standby site. If the standby site is optional, you can resolve the gaps immediately by performing the following steps:

  1. Manually move all archived redo logs to the manual recovery directory. For example:

```
% mv /fs2/oracle/stdby_log/*.arc /fs2/oracle/standby/
```

  2. Apply all of the archived redo logs:

```
SQL> RECOVER AUTOMATIC STANDBY DATABASE;
```

  3. Identify and apply the logs in the gap sequence.

  This way, the standby database control file will be updated, but the lag disappears. You need to wait for the specified time lag (for example, 4 hours) to reinforce the lag.

Rolling Back the Database to a Specified Time

In the case stated at the beginning of this scenario, you may want to take advantage of the time lag, and get a production database whose status is a specified time (for example, 4 hours) before the current production database. You can activate the appropriate time-lagged standby database as follows:

1. Activate the standby database by issuing the following statement:

```
SQL> ALTER DATABASE ACTIVATE STANDBY DATABASE;
```

2. Shut down the standby database instance:

```
SQL> SHUTDOWN IMMEDIATE;
```

3. Start the new production instance:

```
SQL> STARTUP PFILE=Failover.ora;
```

Bypassing the Time Lag and Activating the Standby Database

If you do not want to take advantage of the time lag, you can activate the standby database as a normal standby database with no time lag as follows:

1. Manually move all archived redo logs to the manual recovery directory. For example:

```
% mv /fs2/oracle/stdby_log/*.arc /fs2/oracle/standby/
```

2. Apply all of the archived redo logs:

```
SQL> RECOVER AUTOMATIC STANDBY DATABASE;
```

3. Activate the standby database by issuing the following statement:

```
SQL> ALTER DATABASE ACTIVATE STANDBY DATABASE;
```

4. Shut down the standby database instance:

```
SQL> SHUTDOWN IMMEDIATE;
```

5. Start the new production instance:

```
SQL> STARTUP PFILE=Failover.ora;
```

Index