Oracle9*i*

Data Guard Concepts and Administration

Release 1 (9.0.1)

June 2001

Part No. A88808-01

ORACLE

Oracle9*i* Data Guard Concepts and Administration, Release 1 (9.0.1)

Part No. A88808-01

Contents

2 Creating the Standby Database Environment

3 Log Transport Services

4 Log Apply Services

6 Data Guard Scenarios

List of Examples

List of Figures

List of Tables

Send Us Your Comments

Oracle9*i* Data Guard Concepts and Administration, Release 1 (9.0.1)

Part No. A88808-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this document. 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?

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

- Electronic mail: nedc-doc_us@oracle.com
- FAX: 603-897-3825 Attn: Oracle9*i* Data Guard Documentation
- Postal service:
  Oracle Corporation
  Oracle9*i* Data Guard Documentation
  One Oracle Drive
  Nashua, NH 03062-2804
  U.S.A.

If you would like a reply, please give your name, address, telephone number, and (optionally) electronic mail address.

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

Preface

Standby databases are the most frequently used and most effective disaster recovery solution for Oracle databases. Oracle9*i* Data Guard enhances Oracle8*i* Standby Databases to provide database recovery that meets the essential disaster recovery solution for Oracle databases, and also survives mistakes, corruptions, and disasters such as earthquakes and hurricanes.

This guide describes Oracle9*i* Data Guard concepts and helps you configure and implement standby databases that can take over production operations if your primary database becomes unusable.

This preface contains these topics:

Audience

Oracle9i Data Guard Concepts and Administration is intended for database administrators (DBAs) who administer the backup, restore, and recovery operations of an Oracle database system.

To use this document, you should be familiar with relational database concepts and basic backup and recovery administration. You should also be familiar with the operating system environment under which you are running Oracle.

Organization

This document contains:

Part I, "Concepts and Administration"

Chapter 1, "Oracle9i Data Guard Concepts"

This chapter offers a general overview of the Oracle9*i* Data Guard architecture and environment.

Chapter 2, "Creating the Standby Database Environment"

This chapter explains how to assess and configure a Data Guard environment, set up the standby database, create the standby database files, and configure network files for both the primary and standby databases. It also describes the initialization parameters related to the Data Guard environments.

Chapter 3, "Log Transport Services"

This chapter introduces log transport services. It provides procedures and guidelines for configuring log transport services on a primary and standby database. It also provides guidelines and procedures for configuring data availability modes, network tuning for log transport services, and log transport services monitoring.

Chapter 4, "Log Apply Services"

This chapter introduces log apply services. It provides guidelines for managing a standby database in managed recovery mode and read-only mode.

Chapter 5, "Managing the Data Guard Environment"

This chapter introduces role management services. It provides information on database role transitions, as well as monitoring and responding to events that affect the database role.

Chapter 6, "Data Guard Scenarios"

This chapter describes common database scenarios such as creating, recovering, failing over, switching over, configuring, and backing up standby and primary databases.

Part II, "Reference"

Chapter 7, "Initialization Parameters"

This reference chapter describes initialization parameters for each Oracle instance, including the primary database and each standby database in the Data Guard environment.

Chapter 8, "LOG_ARCHIVE_DEST_n Parameters Attributes"

This reference chapter provides syntax and examples for the attributes of the `LOG_ARCHIVE_DEST_`*n* initialization parameters.

Chapter 9, "SQL Statements"

This reference chapter provides SQL statements that are useful for performing operations on a standby database.

Chapter 10, "Fixed Views"

This reference chapter lists fixed views that contain useful information for monitoring the Data Guard environment. It summarizes the columns contained in each view and provides a description for each column.

Part III, "Appendixes and Glossary"

Appendix A, "Troubleshooting the Standby Database"

This appendix discusses troubleshooting for the standby database.

Appendix B, "Manual Recovery"

This appendix describes managing a standby database in manual recovery mode. It provides instructions for manually resolving archive gaps and renaming standby files not captured by conversion parameters.

Appendix C, "Log Writer Asynchronous Network I/O"

This appendix describes how to achieve asynchronous network I/O when using the log writer process (LGWR) to transmit primary database online redo log modifications to standby databases.

Appendix D, "Standby Database Real Application Cluster Support"

This appendix describes the primary and standby database configurations in a Real Application Cluster environment.

Glossary

Related Documentation

Every reader of *Oracle9i Data Guard Concepts and Administration* is presumed to have read:

- The beginning of the *Oracle9i Database Concepts* manual, which provides an overview of the concepts and terminology related to the Oracle database server and a foundation for the more detailed information in this guide. The rest of the *Oracle9i Database Concepts* manual explains the Oracle architecture and features in detail.
- The chapters in the *Oracle9i Database 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 guides:

- *Oracle9i Database Administrator's Guide*
- *Oracle9i Data Guard Broker*
- *Oracle9i SQL Reference*
- *Oracle9i Database Reference*
- *Oracle9i User-Managed Backup and Recovery Guide*
- *Oracle9i Recovery Manager User's Guide*
- *Oracle Net Services Administrator's Guide*

- *SQL*Plus User's Guide and Reference*

In addition to the Data Guard product, you might want to read about the following Oracle products and features that provide disaster recovery and high data availability solutions:

- Oracle Real Application Clusters
- Oracle Transparent Application Failover
- Oracle Flashback Query
- Oracle LogMiner SQL-based Log Analyzer
- Oracle Point-in-Time Recovery
- Oracle Advanced Replication
- Oracle Advanced Queuing
- Hardware Geo-mirroring

In North America, printed documentation is available for sale in the Oracle Store at

`http://oraclestore.oracle.com/`

Customers in Europe, the Middle East, and Africa (EMEA) can purchase documentation from

`http://www.oraclebookshop.com/`

Other customers can contact their Oracle representative to purchase printed documentation.

To download free release notes, installation documentation, white papers, or other collateral, please visit the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at

`http://technet.oracle.com/membership/index.htm`

If you already have a username and password for OTN, then you can go directly to the documentation section of the OTN Web site at

`http://technet.oracle.com/docs/index.htm`

Conventions

This section describes the conventions used in the text and code examples of this documentation set. It describes:

- Conventions in Text
- Conventions in Code Examples

Conventions in Text

We use various conventions in text to help you more quickly identify special terms. The following table describes those conventions and provides examples of their use.

Convention	Meaning	Example
Bold	Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.	When you specify this clause, you create an **index-organized table.**
Italics	Italic typeface indicates book titles or emphasis.	*Oracle9i Database Concepts* Ensure that the recovery catalog and target database do *not* reside on the same disk.
`UPPERCASE monospace (fixed-width font)`	Uppercase monospace typeface indicates elements supplied by the system. Such elements include parameters, privileges, datatypes, RMAN keywords, SQL keywords, SQL*Plus or utility commands, packages and methods, as well as system-supplied column names, database objects and structures, usernames, and roles.	You can specify this clause only for a `NUMBER` column. You can back up the database using the `BACKUP` command. Query the `TABLE_NAME` column in the `USER_TABLES` data dictionary view. Use the `DBMS_STATS.GENERATE_STATS` procedure.

Convention	Meaning	Example
`lowercase monospace (fixed-width font)`	Lowercase monospace typeface indicates executables, filenames, directory names, and sample user-supplied elements. Such elements include computer and database names, net service names, and connect identifiers, as well as user-supplied database objects and structures, column names, packages and classes, usernames and roles, program units, and parameter values. **Note:** Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	Enter `sqlplus` to open SQL*Plus. The `department_id`, `department_name`, and `location_id` columns are in the `hr.departments` table. Set the `QUERY_REWRITE_ENABLED` initialization parameter to `true`. Connect as `oe` user. The `JRepUtil` class implements these methods.
`lowercase monospace (fixed-width font) italic`	Lowercase monospace italic font represents placeholders or variables.	You can specify the *`parallel_clause`*. Run *`Uold_release.SQL`* where *`old_release`* refers to the release you installed prior to upgrading.

Conventions in Code Examples

Code examples illustrate SQL, PL/SQL, SQL*Plus, or other command-line statements. They are displayed in a monospace (fixed-width) font and separated from normal text as shown in this example:

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

The following table describes typographic conventions used in code examples and provides examples of their use.

Convention	Meaning	Example
[ ]	Brackets enclose one or more optional items. Do not enter the brackets.	`DECIMAL (digits [ , precision ])`
{ }	Braces enclose two or more items, one of which is required. Do not enter the braces.	`{ENABLE | DISABLE}`

Convention	Meaning	Example
|	A vertical bar represents a choice of two or more options within brackets or braces. Enter one of the options. Do not enter the vertical bar.	{ENABLE | DISABLE} [COMPRESS | NOCOMPRESS]
...	Horizontal ellipsis points indicate either: ■ That we have omitted parts of the code that are not directly related to the example ■ That you can repeat a portion of the code	CREATE TABLE ... AS subquery; SELECT col1, col2, ... , col*n* FROM employees;
. . .	Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example.	
Other notation	You must enter symbols other than brackets, braces, vertical bars, and ellipsis points as shown.	acctbal NUMBER(11,2); acct CONSTANT NUMBER(4) := 3;
Italics	Italicized text indicates placeholders or variables for which you must supply particular values.	CONNECT SYSTEM/*system_password* DB_NAME = *database_name*
UPPERCASE	Uppercase typeface indicates elements supplied by the system. We show these terms in uppercase in order to distinguish them from terms you define. Unless terms appear in brackets, enter them in the order and with the spelling shown. However, because these terms are not case sensitive, you can enter them in lowercase.	SELECT last_name, employee_id FROM employees; SELECT * FROM USER_TABLES; DROP TABLE hr.employees;
lowercase	Lowercase typeface indicates programmatic elements that you supply. For example, lowercase indicates names of tables, columns, or files. **Note:** Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	SELECT last_name, employee_id FROM employees; sqlplus hr/hr CREATE USER mjones IDENTIFIED BY ty3MU9;

Documentation Accessibility

Oracle's goal is to make our products, services, and supporting documentation accessible to the disabled community with good usability. To that end, our

documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For additional information, visit the Oracle Accessibility Program Web site at

`http://www.oracle.com/accessibility/`

JAWS, a Windows screen reader, may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, JAWS may not always read a line of text that consists solely of a bracket or brace.

What's New in Data Guard?

This section describes new features of Oracle9*i* Data Guard and provides pointers to additional information. New feature information from Oracle8*i* releases is retained to help those users migrating to the current release.

The following sections describe the new features in Data Guard:

- Oracle9i Release 1 (9.0.1) New Features in Data Guard
- Oracle8i New Features in Data Guard

Oracle9*i* Release 1 (9.0.1) New Features in Data Guard

The features and enhancements described in this section were added to Data Guard in Oracle9*i*.

- **Oracle9*i* Data Guard**

  Oracle8*i* Standby Database has been renamed to Oracle9*i* Data Guard.

- **Oracle9*i* Data Guard Broker**

  The broker is a new management framework that simplifies the configuration and control of a Data Guard configuration, and adds the ability to monitor the configuration.

  The broker provides two new interfaces:

  – **Oracle9*i* Data Guard Manager (GUI)**

    Oracle9*i* Data Guard Manager is a graphical user interface that provides easy configuration of a two-site Data Guard environment, the most typical Data Guard configuration.

  – **Data Guard command-line interface**

    The Data Guard command-line interface allows you to control and monitor a Data Guard configuration directly from the command-line prompt or from within scripts.

    See Also: *Oracle9i Data Guard Broker*

- **No Data Loss**

  With the **no-data-loss** feature, the primary database will not acknowledge primary database modifications until they have been confirmed as being available on at least one standby database. Data is protected when primary database modifications occur while there is connectivity to the standby database. Data is unprotected when primary database modifications occur while connectivity to the standby database is not available.

  The database administrator (DBA) can place the database into one of the following modes:

  – Guaranteed protection

  – Instant protection

  – Rapid protection

- Delayed protection

  See Also: Section 3.6

- **New syntax added to the `ALTER DATABASE` statement to support the no-data-loss feature is:**
  - `SET STANDBY DATABASE {PROTECTED | UNPROTECTED}`
  - `ADD [STANDBY] LOGFILE TO [THREAD` *`integer`*`] [GROUP` *`integer`*`]` *`filespec`*
  - `ADD [STANDBY] LOGFILE MEMBER '`*`filename`*`' [REUSE] TO GROUP` *`integer`*

  See Also: Chapter 9, "SQL Statements"

- **No data divergence**

  Data divergence occurs when the primary database is modified, but the modifications are not available to be applied to a corresponding standby database. Subsequent failover of the primary database to the standby database would result in the loss of previously committed transactions.

  Note that no data divergence is different from no data loss.

  Data divergence is prohibited through the concept of *protected* data. In this sense, the primary database's data is protected by the existence of one or more synchronized standby databases. The failure of the last standby database causes the primary database instance to shut down to avoid data divergence.

  Data integration is responsible for reapplying diverged (unprotected) data to the standby databases.

  See Also: Section 3.6.3.5

- **Database switchover**

  The new database switchover feature provides the database administrator (DBA) with the ability to switch the role of the primary database to one of the available standby databases. The chosen standby database becomes the primary database. The original primary database then becomes a standby database. There is no need to reinstantiate any of the databases in the switchover operation. There is no data divergence between the original and the new primary database after the successful completion of the database switchover.

See Also: Chapter 5, "Managing the Data Guard Environment"

- **Archive gaps are automatically detected and transmitted**

  Archive gaps are detected and transmitted automatically. This feature is built into log transport services and log apply services. The log transport services archive gap detection is always enabled in Oracle9*i* on the primary database. To enable the log apply services archive gap detection on the standby database, the DBA must define the values of two new initialization parameters: `FAL_CLIENT` and `FAL_SERVER`. Oracle Corporation recommends that you always set the log apply services archive gap detection. The log transport services archive gap detection is a best-attempt-effort and does not guarantee complete archive gap detection in some cases. The log apply services archive gap detection does guarantee complete archive gap detection and transmittal for the managed recovery process.

  See Also: Section 4.5

- **Adding new datafiles**

  You can add new datafiles to the primary database without having to manually add the corresponding datafile to the standby databases. The new datafile is created and added to the standby databases if the DBA has set up the `STANDBY_FILE_MANAGEMENT` and `DB_FILE_NAME_CONVERT` initialization parameters.

  See Also: Section 4.6 and Section 5.8.1

- **Background managed recovery mode**

  You can now create a background process to perform managed recovery. This frees the foreground terminal session that you used to execute the managed recovery statement.

  See Also: Section 4.3.2

- **Parallel recovery**

  You can execute parallel recovery using the SQL `RECOVER MANAGED STANDBY DATABASE` statement. This allows faster recovery on your standby databases.

  See Also: Section 4.4.8

- **More archive destinations**

You can specify up to 10 archive destinations in Oracle9*i*. Prior versions allowed only up to 5 archive destinations.

See Also: Section 3.3.1

- **Incremental change to `LOG_ARCHIVE_DEST_`*n***

  The DBA can incrementally modify individual attributes of the `LOG_ARCHIVE_DEST_`*n* initialization parameters, where *n* is a value from 1 to 10.

  See Also: Section 3.3.1

- **Standby redo logs are introduced**

  You can control where the standby database stores the redo data received from the primary site. This data can be stored on the standby site using either standby redo logs or archived redo logs.

  See Also: Section 3.6.3.4

- **Archiver process (ARC*n*) available on standby databases**

  The ARC*n* process is used on standby databases to archive standby redo logs. It uses the `LOG_ARCHIVE_DEST_1` initialization parameter and is limited to local archival operations.

  See Also: Section 3.3.1

- **Changes to archiving online redo logs**

  In prior releases, the current redo log had to be full before you could archive it to the standby site. In Oracle9*i*, it is possible to:

  – Archive the current redo log. In previous releases, you needed to perform a log switch first.

  – Archive an online redo log based on the SCN (system change number) value when the database is mounted, but not open.

  – Archive an online redo log when a backup control file is being used. In previous releases, a current control file was required.

  See Also: Chapter 3, "Log Transport Services"

- **New control options**

  The following control options have been added to this release:

  - `DELAY`

    Specifies an absolute apply delay interval to the managed recovery operation. The managed recovery operation waits the specified number of minutes before applying the archived redo logs.

  - `DISCONNECT`

    Starts managed recovery in background mode.

  - `EXPIRE`

    Specifies the number of minutes after which the managed recovery operation automatically terminates, relative to the current time. The managed recovery operation terminates at the end of the current archived redo log that is being processed.

  - `FINISH`

    Completes managed recovery in preparation for a failover from the primary database to the standby database.

  - `NEXT`

    Directs the managed recovery operation to apply a specified number of archived redo logs as soon as possible after the log transport services have archived them.

  - `NODELAY`

    Directs the managed recovery operation to apply the archived redo logs as soon as possible after log transport services have archived them.

    See Also: Section 4.4

- **Standby Real Application Clusters support**

  Oracle9*i* provides the ability to perform true database archival from a primary database to a standby database when both databases reside on a Real Application Cluster. This allows you to separate the log transport services processing from the log apply services processing on the standby database, thereby improving overall primary and standby database performance.

  See Also: Appendix D, "Standby Database Real Application Cluster Support"

- **Archive log repository**

  Oracle9*i* introduces the archive log repository, which is a standalone standby control file. This type of destination allows off-site archival of redo log files.

  See Also: Section 1.4 and Section 3.2.2

- **Relationship defined between an archived redo log and an archive destination**

  The DBA can now determine the following by joining the `V$ARCHIVED_LOG` and `V$ARCHIVE_DEST` fixed views:

  – Archive destination information for an archived redo log

  – All archived redo logs generated from a destination

  See Also: Chapter 10, "Fixed Views"

- **New initialization parameters**

  The following initialization parameters have been added to this release for each Oracle instance:

  – `REMOTE_ARCHIVE_ENABLE`

  – `FAL_CLIENT`

  – `FAL_SERVER`

  – `STANDBY_FILE_MANAGEMENT`

  – `ARCHIVE_LAG_TARGET`

See Also: Chapter 7, "Initialization Parameters"

- **New attributes for `LOG_ARCHIVE_DEST_`*n* include:**
  - `ARCH | LGWR`
  - `[NO]AFFIRM`
  - `[NO]ALTERNATE`
  - `[NO]DELAY`
  - `[NO]DEPENDENCY`
  - `[NO]MAX_FAILURE`
  - `[NO]QUOTA_SIZE`
  - `[NO]QUOTA_USED`
  - `[NO]REGISTER | REGISTER [=registration location]`
  - `NOREOPEN`
  - `SYNC | ASYNC`

  See Also: Section 3.3.1 and Chapter 8, "LOG_ARCHIVE_DEST_n Parameters Attributes"

- **New range of values and keyword have been added to the `LOG_ARCHIVE_DEST_STATE_`*n* initialization parameters**
  - The variable *n* can be a value from 1 to 10 (versus 1 to 5 in Oracle8*i*).
  - The new keyword, `ALTERNATE`, has been added.

  See Also: Section 3.3.1.2

- **One to ten destinations (versus one to five in Oracle8*i*) must archive successfully before the log writer process (LGWR) can overwrite the online redo logs**

  See Also: Section 3.4.2.4

- **New tracing levels have been added to the `LOG_ARCHIVE_TRACE` initialization parameter**

- Tracing level of 128 allows you to track fetch archive log (FAL) server process activity.
- Tracing level 256 will be available in a future release.
- Tracing level 512 allows you to track asynchronous log writer activity.

  See Also: Section 4.9.5

- **New clauses have been added to the `ALTER DATABASE` statement:**
  - `ACTIVATE [PHYSICAL] STANDBY DATABASE [SKIP [STANDBY LOGFILE]]`
  - `ADD [STANDBY] LOGFILE TO [THREAD` *`integer`*`] [GROUP` *`integer`*`]` *`filespec`*
  - `ADD [STANDBY] LOGFILE MEMBER '`*`filename`*`' [REUSE] TO '`*`logfile-descriptor`*`'`
  - `COMMIT TO SWITCHOVER TO [PHYSICAL] {PRIMARY | STANDBY} [[NO]WAIT]`
  - `REGISTER [PHYSICAL] LOGFILE` *`filespec`*
  - `SET STANDBY DATABASE {PROTECTED | UNPROTECTED}`

  See Also: Chapter 9, "SQL Statements"

- **New keywords have been added to the `RECOVER MANAGED STANDBY DATABASE` statement**
  - `NODELAY`
  - `CANCEL [IMMEDIATE] [NOWAIT]`
  - `[DISCONNECT [FROM SESSION]]`
  - `[FINISH [NOWAIT]]`
  - `[PARALLEL [`*`integer`*`]]`
  - `NEXT`
  - `EXPIRE`
  - `DELAY`

See Also: Chapter 9, "SQL Statements"

- **New fixed views have been added:**
  - V$ARCHIVE_DEST_STATUS
  - V$ARCHIVE_GAP
  - V$MANAGED_STANDBY
  - V$STANDBY_LOG

    See Also: Chapter 10, "Fixed Views"

- **New columns have been added to existing fixed views:**
  - V$ARCHIVE_DEST view:
    - AFFIRM
    - ALTERNATE
    - ARCHIVER
    - ASYNC_BLOCKS
    - DELAY_MINS
    - DEPENDENCY
    - FAILURE_COUNT
    - LOG_SEQUENCE
    - MANIFEST
    - MAX_FAILURE
    - MOUNTID
    - PROCESS
    - QUOTA_SIZE
    - QUOTA_USED
    - REGISTER
    - SCHEDULE
    - TRANSMIT_MODE
    - TYPE

    - New values, ALTERNATE and FULL, have been added to STATUS column
- V$ARCHIVED_LOG view:
    - APPLIED
    - BACKUP_COUNT
    - COMPLETION_TIME
    - CREATOR
    - DELETED
    - DEST_ID
    - DICTIONARY_BEGIN
    - DICTIONARY_END
    - REGISTRAR
    - STANDBY_DEST
    - STATUS
    - END_OF_REDO
    - ARCHIVAL_THREAD#
- V$LOG view:
    - A new value, INVALIDATED, has been added to STATUS column
- V$LOGFILE view:
    - TYPE
- V$DATABASE view:
    - ACTIVATION#
    - ARCHIVELOG_CHANGE#
    - DATABASE_ROLE
    - REMOTE_ARCHIVE
    - STANDBY_MODE
    - SWITCHOVER_STATUS
- V$ARCHIVE_DEST_STATUS view:

* STANDBY_LOGFILE_COUNT
* STANDBY_LOGFILE_ACTIVE

See Also: Chapter 10, "Fixed Views"

Oracle8*i* New Features in Data Guard

The features and enhancements described in this section were added to the standby database in Oracle8*i*.

- **Read-only mode**

  The read-only mode option can be used to make a standby database available for queries and reporting, even while redo logs are being archived from the primary database site.

  See Also: Section 4.7

- **Backing up a primary database from a standby database**

  You can use the Recovery Manager utility (RMAN) to back up the primary database using the standby database.

  See Also: Section 5.6

Part I

Concepts and Administration

This part contains the following chapters:

1

Oracle9*i* Data Guard Concepts

Many organizations today depend on continuous access to data and computing resources. The interruption of computing services due to hardware failures, software failures, or human errors can result in the interruption of database access for primary business functions. When business data is unavailable, the consequences can range from merely inconveniencing certain users, to more severe effects like harming the financial health of the organization.

Oracle9*i* Data Guard provides a complete set of data protection and disaster recovery features to help you to survive mistakes, corruptions, and disasters that can destroy a database.

This chapter explains Oracle9*i* Data Guard concepts. It includes the following topics:

1.1 Oracle9*i* Data Guard Overview

Oracle9*i* **Data Guard** works with **standby databases** to protect your data against errors, failures, and corruptions that might otherwise destroy your database. It protects critical data by automating the creation, management, and monitoring aspects of a **standby database environment.** It automates the otherwise manual process of maintaining a transactionally consistent copy of an Oracle production database for the purpose of recovering from the loss or damage of the production database.

Without Oracle9*i* Data Guard, the database administrator (DBA) must copy **archived redo logs** to the remote host and apply them so that the **primary database** and standby database remain synchronized. Oracle9*i* Data Guard automates the tasks involved in setting up and managing the Data Guard environment, including one or more standby databases, the log transport services, log apply services, role management services, and related applications.

Generally, **redo logs** are transferred from the primary database to the standby database as they are archived. However, Oracle9*i* Data Guard provides the DBA with great flexibility when defining archive destinations, archive completion requirements, I/O failure handling, and automated transmission restart capability.

For example:

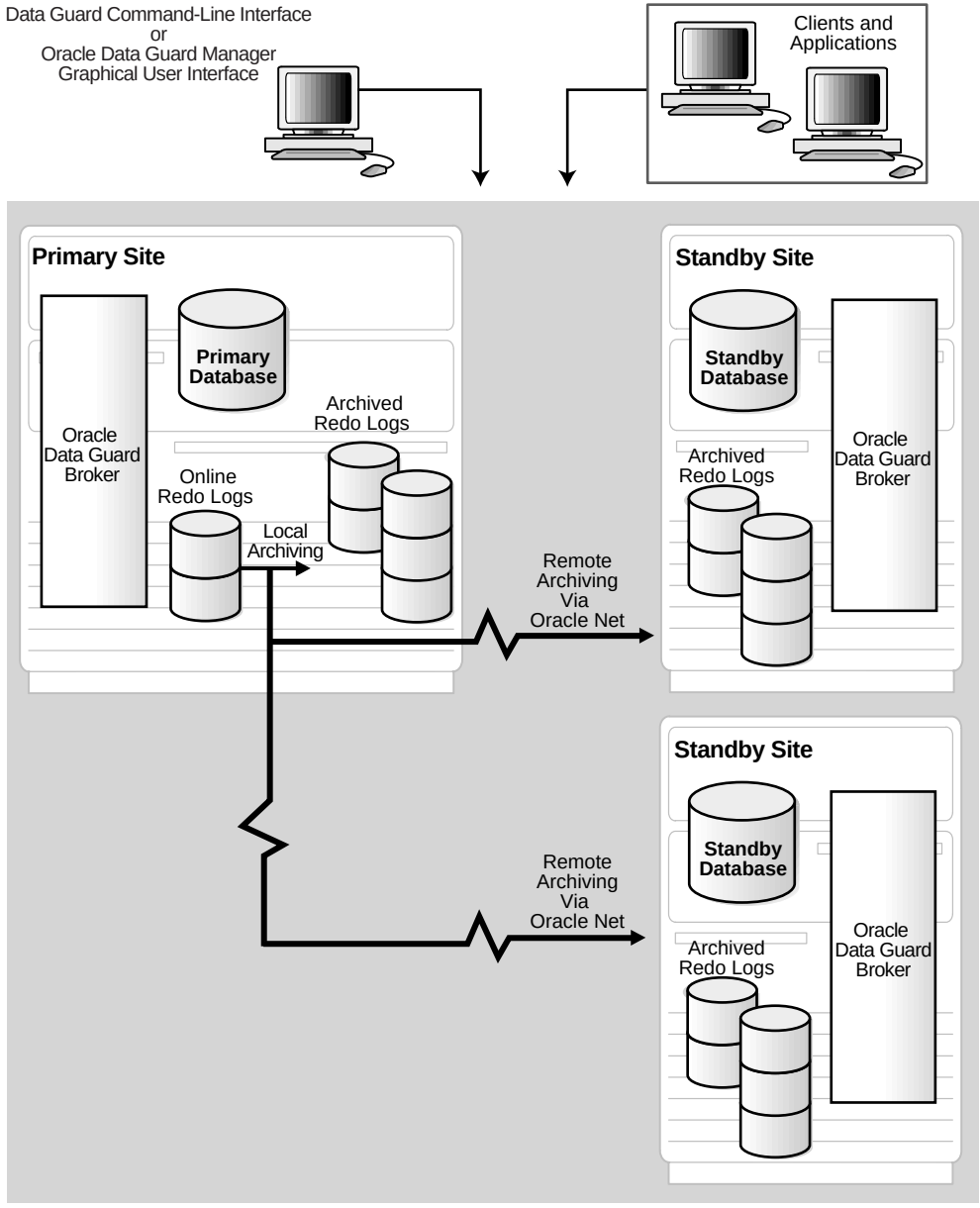
- The DBA can choose to synchronously write redo log updates directly from the primary to the standby database to provide for a comprehensive *no-data-loss* disaster-recovery solution. Should a disaster strike at the **primary site,** all redo logs necessary to preserve all transactions will be available for application at the **standby site.**
- The DBA can choose to transfer archived redo logs asynchronously to the standby site. Although this does not provide a no-data-loss solution, it minimizes any potential data loss, while providing optimal performance over greater distances and offering protection from network failures. Customization options provide the means to control the potential for data loss.
- The DBA can specify a delay for application of the redo log data once it arrives at the standby site. Immediate application means faster **failover** due to a more up-to-date standby database. However, delaying the application of redo logs

allows the DBA to fail over to the standby database and resume services before errors can propagate to the standby database.

You can perform administrative tasks using the Oracle9*i* Data Guard Manager graphical user interface, the Data Guard command-line interface, or SQL statements.

Figure 1–1 shows an example of a Data Guard configuration that allows failover to occur to either of two standby systems. If the primary site becomes unavailable, the workload can fail over to either standby site.

Figure 1–1 Oracle9i Data Guard Configuration



Oracle9*i* Data Guard helps you survive events that might otherwise destroy your database by:

- Enhancing high **availability** and disaster protection through a distributed computing configuration

  An Oracle9*i* Data Guard configuration consists of a collection of loosely connected systems, called **sites**, that combine the primary database and the **physical standby databases** into a single, easily managed disaster recovery solution. Often, the sites in a Data Guard configuration are dispersed geographically and are connected by Oracle Net.

- Providing a single, uniform management interface

  The Oracle9i Data Guard **broker** is the component that helps you to configure database resources into a single unit of failover across a Data Guard configuration. The broker provides two optional interfaces: the Oracle9*i* Data Guard Manager graphical user interface and the Oracle9*i* Data Guard command-line interface.

- Automating the creation and configuration of physical standby databases

  Use the broker to create the database and to manage the database. One site is designated a primary site, where the database is available to applications and from where Oracle Data Guard **log transport services** automatically transfer the data in the form of archived redo logs. You can configure and instantiate from one to nine additional server sites to serve as standby systems to the primary site. The standby sites accept redo logs archived from the primary site and apply changes to their copies of the database.

- Providing a single, atomic unit of failover that groups dependent database and application resources (including failover policies that you can customize). Also, support for **switchover** helps to reduce **downtime** during routine maintenance operations.

  The failover and switchover capabilities allow you to determine how the role of the primary database site should migrate in a failure. With the broker, you can define and automate the primary role migration by implementing an *n*-way failover system with replicated data across all of the server nodes.

- Safeguarding against physical corruptions

  For physical standby databases, the risk of physical corruptions is reduced, because the code path and devices where a primary-side physical corruption begins are unlikely to propagate through the archived redo logs that are transported to the standby site.

- Providing configurable redo log transport services

  You can set up the standby database to achieve the following levels of data protection:

  - Guaranteed protection
  - Instant protection
  - Rapid protection
  - Delayed protection

- Providing configurable redo log apply services

  Using **log apply services**, you can control whether you maintain the standby database in **managed recovery mode** or in open **read-only mode.**

- Providing monitoring, alert, and control mechanisms

  Besides automating the complex task of configuration, the broker automates the management and operational tasks a DBA must perform across the multiple sites in a Data Guard configuration. The broker monitors and controls all of the systems within a single standby database configuration.

Note: The primary purpose of Oracle Data Guard is to keep your data highly available against any event, including disasters; it is not intended to be used for data integrity or as a replacement for traditional backup, restore, and recovery techniques.

1.2 Operational Requirements

Note the following operational requirements for maintaining a standby database:

- The primary database must run in **ARCHIVELOG mode.**
- Use the same version and release of the operating system on the primary and standby sites. The standby site can, however, use a different directory structure.
- The hardware architecture on the primary and standby sites must be the same.
- Use the same version and release of the Oracle database server for the primary and standby databases.
- Each primary database and standby database must have its own **control file.**
- If you place your primary and standby databases on the same system, some operating systems will not allow you to mount two instances with the same

database name on the same system simultaneously. Workarounds for this situation exist for every platform.

1.3 Oracle Data Guard Architecture

The Oracle9*i* Data Guard architecture consists of the following components:

- Primary database

  A **primary database** is a production database. The primary database is 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.

  The redo logs associated with the primary database are archived to a standby destination to be applied to the standby database.

- Physical standby database

  A **physical standby database** is a database replica created from a backup of a primary database.

- Log transport services

  Log transport services control the automated transfer of archived redo logs from the primary database to one or more standby sites or **destinations.**

- Network configuration

  The primary database is connected to one or more remote standby databases through Oracle Net.

- Log apply services

  Log apply services apply the archived redo logs to the standby database.

- Role management services

  Role management services control the changing of database roles from primary to standby and back to primary. The services include switchover and switchback, as well as database failover.

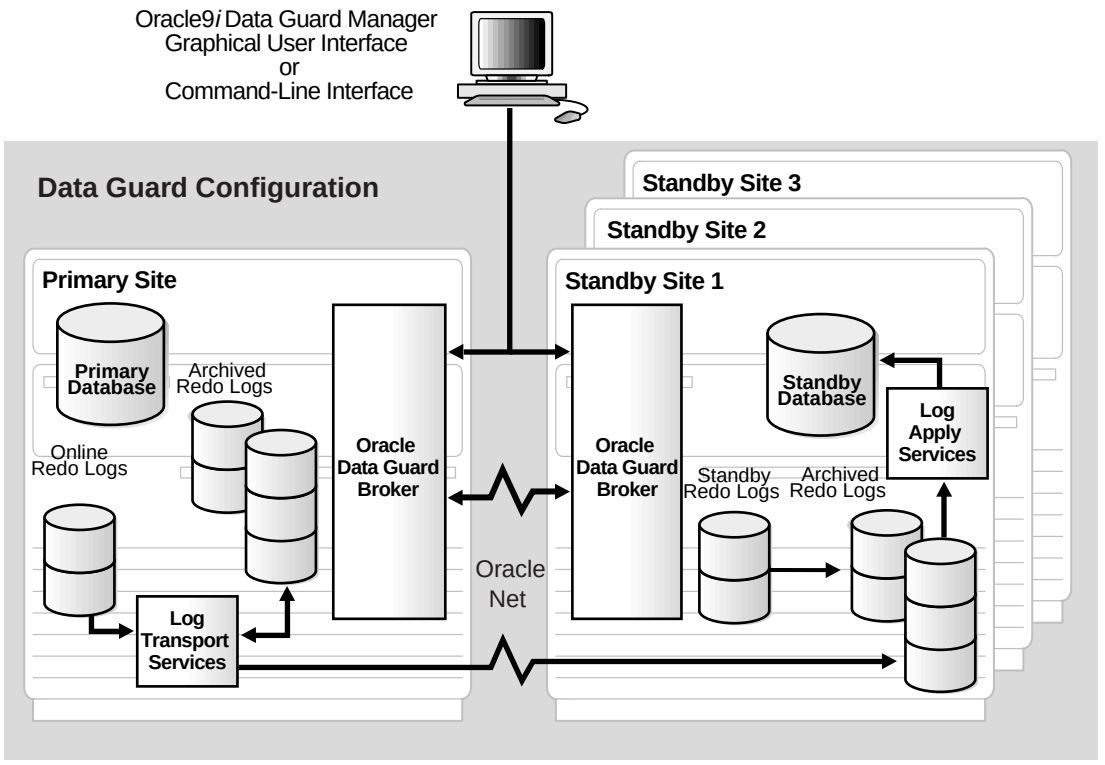
- Data Guard broker

  Data Guard **broker** is the management and monitoring component that helps you create, control, and monitor a primary database protected by one or more physical standby databases. The broker automates the previously manual process of configuration. Once it has created the Data Guard configuration, the

broker monitors the activity, health, and availability for all systems in the Data Guard configuration.

Figure 1–2 shows the Data Guard architecture.

Figure 1–2 Data Guard Architecture



1.4 Log Transport Services

The log transport services component of Oracle9*i* Data Guard controls the automated archival of archived redo logs from the primary site to one or more standby sites or destinations.

You can use log transport services to set up the following:

- Physical standby database

This is a physical copy of the primary database.

- Archive log repository

  This is a standalone standby control file. This type of destination allows off-site archival of redo logs.

- Cross-instance archival

  In a Real Application Clusters environment, each instance directs its archived redo logs to a single instance of the cluster. This instance, known as the **recovery instance**, is typically the instance where managed recovery is performed. The recovery instance typically has a tape drive available for RMAN backup and restore support.

Log transport services provide control of different log archiving mechanisms, log archiving, error handling, reporting, and re-archiving logs after a system failure.

You set up a Data Guard configuration such that the databases are running on each site:

- Designate one site as a primary site. This is where the database is available to applications and from where the data is copied.
- Designate one or more additional sites as standby sites to the primary site. Each standby system accepts redo log files transferred from the primary site and applies changes to their copies of the database.

You create a standby database initially by copying the primary database. As redo log files are generated on the primary database, log transport services automatically archive them to each standby database. Log apply services apply the redo log files to each standby database. Log transport services and log apply services allow each standby database to be synchronized with the primary database with minimum latency.

You can customize log transport services for a variety of standby database configurations.

See Also: Chapter 3, "Log Transport Services"

1.5 Log Apply Services

Log apply services maintain the standby database in either a managed recovery mode or an open read-only mode. The **managed recovery mode** automatically maintains and applies archived redo logs to maintain transactional synchronization

with the primary database, and allows transactionally consistent read-only access to the data.

In a Data Guard environment, the log apply services component coordinates its activities with the log transport services component.

See Also: Chapter 4, "Log Apply Services"

1.6 Role Management Services

Role management services provide the means to change database roles from a primary role to a standby role and back to a primary role. Database role transition includes database switchover and database switchback, as well as database failover.

Role management services operate in conjunction with the log transport services and log apply services to provide many options for the recovery of primary database modifications if the primary database is unavailable due to an unplanned shutdown. Role management services provide database failover options that may not require reinstantiation of other standby databases.

See Also: Chapter 5, "Managing the Data Guard Environment"

1.7 Physical Standby Process Architecture

The physical standby component of Oracle Data Guard uses several processes to achieve the automation necessary for disaster recovery and high availability.

On the primary site, log transport services use the following processes:

- Log writer process (LGWR)

  The **log writer process (LGWR)** collects transaction redo and updates the **online redo logs.** The LGWR process can also create local archived redo logs and transmit online redo to standby databases.

- Archiver process (ARC*n*)

  The **archiver process (ARC*n*)**, or a SQL session performing an archival operation, creates a copy of the online redo logs, either locally or remotely for standby databases.

- Fetch archive log (FAL) client

  The **fetch archive log (FAL) client** is a background Oracle database server process. The initialization parameter for the FAL client is set on the standby site. The FAL client *pulls* archived redo log files from the primary site and initiates

and requests the transfer of archived redo log files automatically when it detects an **archive gap** on the standby database.

- Fetch archive log (FAL) server

  The **fetch archive log (FAL) server** is a background Oracle database server process. The initialization parameter for the FAL server is set on the standby site. The FAL server runs on the primary database and services the fetch archive log (FAL) requests coming from the FAL client. For example, servicing a FAL request might include queuing requests (to send archived redo log files to one or more standby databases) to an Oracle database server that runs the FAL server. Multiple FAL servers can run on the same primary database at any point in time, with a separate FAL server created for each incoming FAL request.

On the standby database, log apply services use the following processes:

- Remote file server (RFS)

  The **remote file server (RFS)** process receives archived redo logs from the primary database.

- ARC*n*

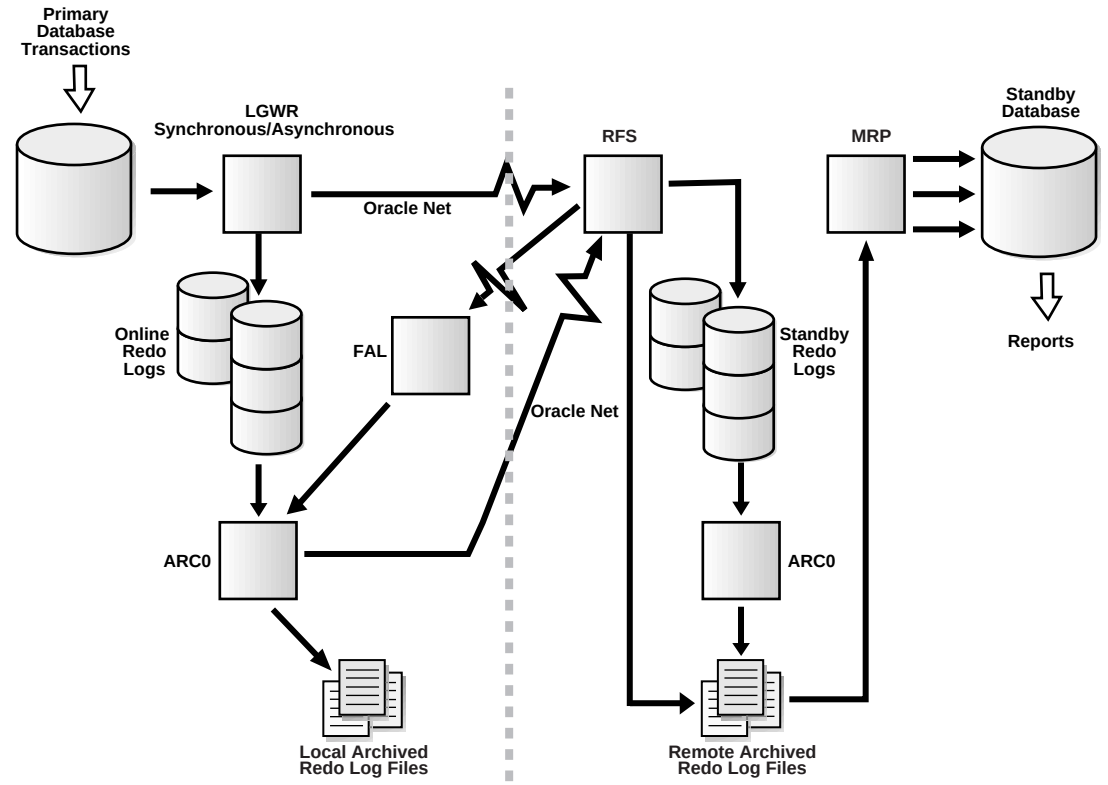
  The ARC*n* process archives the **standby redo logs** to be applied by the **managed recovery process (MRP).**

- MRP

  The MRP applies archived redo log information to the standby database.

Figure 1–3 identifies the relationships of these processes to the operations they perform and the database objects on which they operate.

Figure 1–3 Process Architecture



1.8 Data Guard Environments

You can configure standby databases in a variety of ways in a Data Guard environment, including the following:

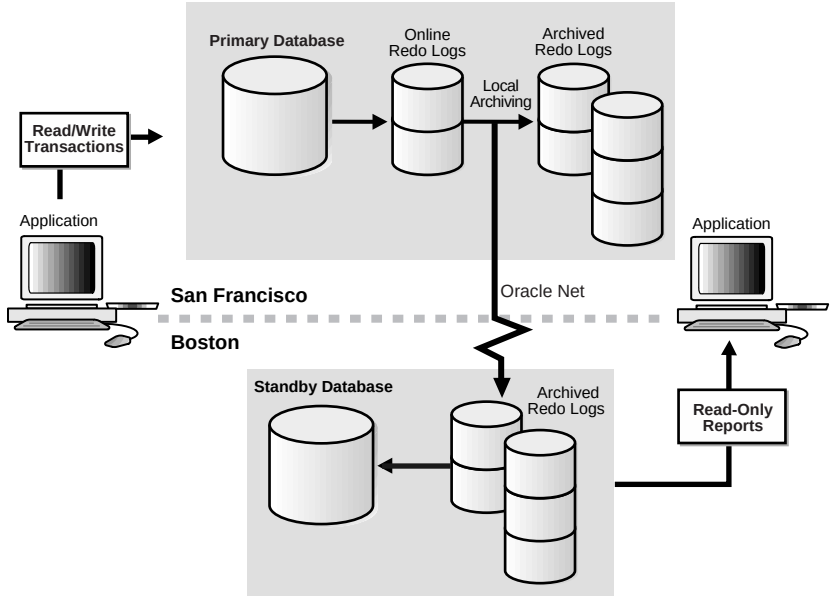
- Typical Two-Site Data Guard Environment
- Data Divergence and Data Loss
- Primary Database Protection Modes Overview
- Standby Database No-Data-Loss Failover
- Standby Database Data-Loss Failover
- Primary Database Protection Modes Description

- Data Divergence and Data Loss Summary
- Delayed Standby

1.8.1 Typical Two-Site Data Guard Environment

A typical two-site Data Guard environment consists of a primary database and a standby database on a remote host connected by a network. Figure 1–4 shows a typical two-site Data Guard environment.

Figure 1–4 Typical Two-Site Data Guard Environment



See Also: Section 6.2 for detailed information on creating a standby database on a remote site

1.8.2 Data Divergence and Data Loss

Data divergence is the temporary state of the primary database that occurs when a standby database becomes inaccessible. While in the data divergence state, the primary database is able to commit transactions whose data modifications are not immediately available on the standby database. Data divergence also occurs when you use the LGWR or ARC*n* process for asynchronous standby database archival operations.

Note: Use the LGWR process in conjunction with synchronous standby database archival operations to prevent data divergence as long as continuous network connectivity is available.

Data loss occurs when you fail over to a standby database whose corresponding primary database is in the data divergence state. To prevent primary database data divergence if network connectivity fails, use the guaranteed protection mode.

Note: When you have multiple standby databases, the primary database may have diverged relative to one standby database, but not necessarily with other standby databases. Also, the degree of divergence may differ with each standby database.

1.8.3 Primary Database Protection Modes Overview

The primary database protection modes are failure policies that dictate how to manage the primary database if a standby database becomes inaccessible. There are four primary database protection modes:

- Guaranteed protection
- Instant protection
- Rapid protection
- Delayed protection

Guaranteed protection mode dictates that the primary database modifications must always be available on at least one standby database; data divergence is prohibited

by terminating the primary instance if it loses network connectivity to the last standby database.

Instant protection mode dictates that the primary database modifications always be available on at least one standby database. Unlike guaranteed protection mode, data divergence is not prohibited by the instant protection mode. Data divergence exists for the duration that standby database connectivity is unavailable, and can be resolved when network connectivity to the primary database is reestablished.

Rapid protection mode indicates that primary database modifications are available to the standby database as soon as possible with minimal effect on primary database performance.

Delayed protection mode indicates that primary database modifications will ultimately be available on the standby site, as long as the network is active. Both the rapid and delayed protection modes allow the primary database to diverge from all standby databases, even when network connectivity is available. The degree of data divergence is equivalent to the data contained in the unarchived primary database online redo logs.

1.8.4 Standby Database No-Data-Loss Failover

On a standby database, no-data-loss failover is possible only when the corresponding primary database is operating in a data protection mode that provides no-data-divergence semantics, such as guaranteed and instant protection mode. This means that all primary database archived redo logs necessary for no-data-loss failover are available. However, it is possible that the archived redo logs have not yet been applied to the standby database. No-data-loss failover requires that all archived redo logs are first applied. If the archived redo logs are not applied, failover to a standby database will result in data loss relative to the primary database.

1.8.5 Standby Database Data-Loss Failover

Standby database data-loss failover occurs when primary database modifications have not all been applied, or when the corresponding primary database is operating in a data protection mode that allows data divergence semantics, such as rapid and delayed protection modes. This results in standby database data loss relative to the primary database. The amount of data loss can be controlled by primary database archived log destination attributes and the availability of standby redo logs at the standby database.

1.8.6 Primary Database Protection Modes Description

No data divergence means the primary database will not acknowledge primary database modifications until they have been confirmed as being available on at least one standby database. Data is protected when primary database modifications occur while there is connectivity to the standby database. Data is unprotected when primary database modifications occur while connectivity to the standby database is not available.

The primary and standby databases are synchronized by applying archived redo logs from the primary database to the standby database. Although the goal is to keep the databases identical, the transactions applied to the standby database can sometimes lag behind the primary database.

This lag may occur either because the data has not yet reached the standby site, or it may have reached the standby site, but has not yet been applied to the standby database.

If the data needed to keep the databases synchronized is not yet available on the standby site, and you must fail over, the contents of the databases will diverge, and some of the data will be lost. You can be protected from data loss by ensuring that all the data is available on the standby site. Data divergence will be eliminated once all of the data is ultimately applied.

The amount of primary database data divergence and standby database data loss can be controlled, depending on the level of protection required by your business. By weighing your requirements for availability against user demands for response time and performance, you can determine how tightly you want to synchronize the primary and standby databases and how much data you can afford to lose.

You can set up the primary database to achieve the following levels of data protection:

- Guaranteed protection

  Guaranteed protection mode indicates that primary database modifications are available to the standby database, up to the last committed transaction, and protected against data loss, even if either site fails. This mode has the greatest effect on primary database performance.

  Stock exchanges, currency exchanges, or financial institutions are examples of businesses that require guaranteed protection. When a customer requests a stock be sold at current market price, it must be traded exactly at that price, as the price fluctuates and cannot be executed again later on.

- Instant protection

Instant protection mode indicates that primary database modifications are available to the standby database, up to the last committed transaction, as long as both sites are active. This mode has less effect on primary database performance than guaranteed protection mode.

An example of a business that requires instant protection is a manufacturing plant; the risks of having no standby database for a period of time and data divergence are acceptable, as long as no data is lost if failover is necessary.

The loss of standby database network connectivity has the side effect of dynamically changing instant protection mode into delayed protection mode. Once network connectivity to the standby database is reestablished, instant protection mode resumes.

- Rapid protection

  Rapid protection mode indicates that primary database modifications are available to the standby database as soon as possible with minimal effect on primary database performance.

  In an order processing system, orders may be reentered if they were written manually, or if a customer calls and says he did not receive the item. Another example is an online auction, where it is more important to be online than to lose a few bids in the case of a failover.

- Delayed protection

  Delayed protection mode indicates that primary database modifications will ultimately be available on the standby site, as long as the network is active. This mode has the least effect on primary database performance.

  See Also: Section 3.6 for additional information on setting up a no-data-loss environment

1.8.7 Data Divergence and Data Loss Summary

Table 1–1 summarizes the available data protection modes and their implications for primary database data divergence and standby database switchover and failover.

Table 1–1 Summary of Data Protection Modes

Primary Database Protection Mode Name	Database Protection Policy (Status)	Archived Log Destination Attributes (Dynamic)	Standby Network Connectivity (Runtime)	Resulting Primary Database Operating State	Standby Database Switchover Result	Standby Database Failover Result
Guaranteed protection mode	PROTECTED	LGWR, SYNC, etc.	Connected	No data divergence	No data loss	No data loss
			Disconnected	Instance shutdown	Not possible	No data loss
Instant protection mode	UNPROTECTED	LGRW, SYNC, etc.	Connected	No data divergence	No data loss	No data loss
			Disconnected	Delayed protection mode	Not possible	Data loss
Rapid protection mode	UNPROTECTED	LGWR, ASYNC, etc.	Connected	Data divergence	No data loss	Data loss
			Disconnected	Delayed protection mode	Not possible	Data loss
Delayed protection mode	UNPROTECTED	ARCH	Connected	Data divergence	No data loss	Data loss
			Disconnected	Data divergence	Not possible	Data loss

1.8.8 Delayed Standby

By default, in managed recovery mode, the standby database automatically applies archived redo logs when they arrive from the primary database. But in some cases, you may want to delay applying the redo logs. A time lag can protect against the application of corrupted or erroneous data from the primary database to the standby database.

> **See Also:** Section 3.4.2.8 and Section 6.11 for detailed information on creating a standby database with a time lag

1.9 Standby Database Operational Modes

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

- Archiving online redo logs to the standby site
- Applying archived redo logs to the standby database

For example, in a Data Guard environment, log transport services automatically archive redo logs to the standby database site so long as the standby instance is started.

If the standby database is in managed recovery mode, the log apply services automatically apply logs received from the primary database. At any time you can open the standby database in read-only mode for reporting purposes.

You can operate a standby database in one of the following modes:

- Managed recovery mode

  For maximum protection against data loss or corruption, maintain the standby database in **managed recovery mode.** In this mode, the primary database archives logs to the standby site, and the standby database automatically applies these logs.

- Read-only mode

  To use the standby database for reporting purposes, open it in **read-only mode.** Log apply services 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 mounted.

Although the standby database cannot be in more than one mode at the same time, you can change between the modes. For example, you can run in managed recovery mode and then open in read-only mode.

In most implementations of a Data Guard environment, you need to change between managed recovery mode and read-only mode at various times to either:

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

1.10 Database Roles

A database can be in one of two mutually exclusive roles: primary or standby. You can change these roles dynamically as a planned transition or as the result of a failure.

For example, a primary database can change to the standby role, so that its corresponding standby database can change to the primary role. This *planned* transition occurs without having to reinstantiate either database. This is known as a *switchover operation*.

In a failure of the primary database, such as a system or software failure, you may need to change the corresponding standby database to the primary role. This *unplanned* transition may result in the loss of application data. The potential loss of data is dependent on user-defined parameters of the primary database. This type of transition requires you to instantiate a new standby database from the newly activated primary database. This is known as a *failover operation*.

The active role of a database is more than conceptual; you must also consider physical aspects of the role. For example, a database in the primary role uses a **current control file**, while a database in the standby role uses a standby control file. Using different physical files, depending on which database role is active, requires careful coordination of the initialization parameter file.

1.11 Failover and Switchover

Failover is the operation of taking the primary database offline on one site and bringing one of the standby databases online. This operation occurs due to a system or software failure.

One of the consequences of failover is that the original primary database must be reinstantiated.

With Oracle9*i*, it is possible to switch over to a standby database without requiring a resetlogs operation. **Switchover** is the process of intentionally taking database resources offline on one system and bringing them back online on another system. For example, you might use switchover to perform a **rolling upgrade** (you switch over all of the database resources to one system as you sequentially upgrade hardware on the other system).

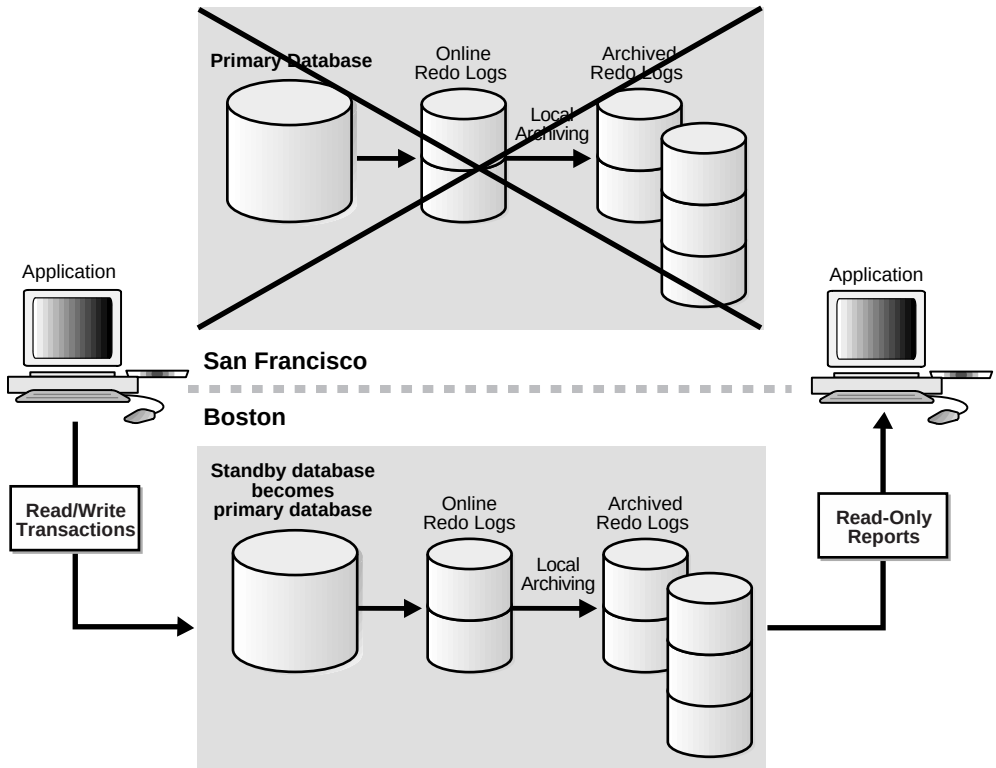
The main difference between a switchover process and a failover process is that switchover does not require reinstantiation of the database. This allows the primary database to resume its role as the standby database almost immediately. As a result, scheduled maintenance can be performed more frequently because it is less risky.

For many configurations, you can lessen the frequency of failures by a regular program of preventive maintenance tasks. The switchover capability allows you to schedule time for hardware and software chores such as the following, without interrupting processing:

- Hardware--repairs, cleaning, adjustments, upgrades, installations. Hardware failures repaired or prevented through this approach can extend the apparent **reliability** of the hardware.
- Software--upgrades, maintenance releases, patch installations, disk defragmentation, cleaning up error logs, verifying status and operation of fault management tools, and collecting data to analyze your needs for additional maintenance can be done during this preventive maintenance time.

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 fail over to 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.

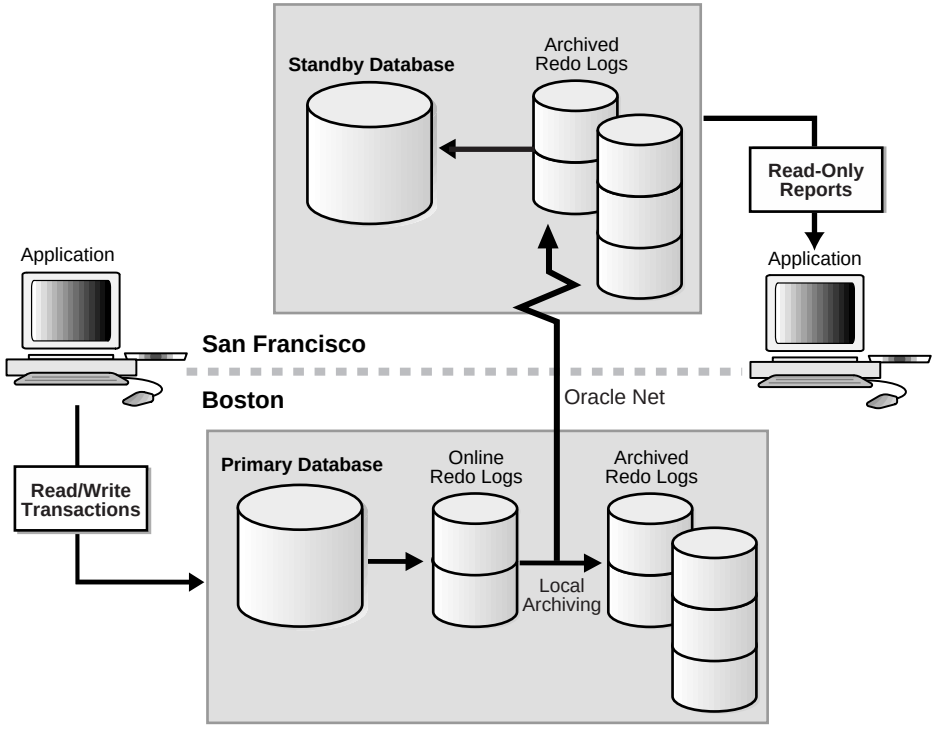
Caution: Failing over to a standby database is a permanent operation. You cannot undo the failover and return the database to its former role as a standby database.

1.11.1 Database Switchover

You can switch a database role from primary *over* to standby, and from standby *back* to primary without reinstantiating any of the databases.

Figure 1–6 depicts the environment after switchover.

Figure 1–6 Switchover to a Standby Database

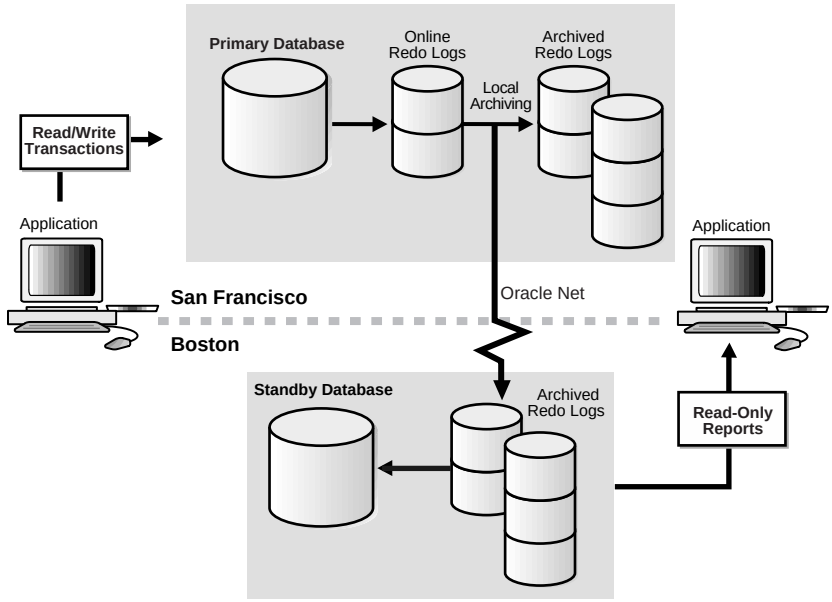


See Also: Section 5.3 for information on switching database roles and Section 6.6 for detailed steps on how to perform a switchover operation

1.11.2 Database Switchback

Once you have performed a database switchover operation, you can switch back to your original Data Guard configuration. A database **switchback** is performed using the switchover operation, but in the reverse direction. Figure 1–7 depicts the original environment after the switchback operation.

Figure 1–7 Data Guard Environment After Switchback



1.12 Data Guard Interfaces

You can use the following to configure, implement, and manage a standby database:

- SQL statements

  Several SQL 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 9–1 describes the relevant statements.

- Initialization parameters

  Several initialization parameters are used to define the Data Guard environment. Table 7–1 describes relevant initialization parameters.

- Oracle9*i* Data Guard Manager

  The Oracle9*i* Data Guard Manager graphical user interface automates many of the tasks involved in configuring and monitoring a Data Guard environment.

  See Also: *Oracle9i Data Guard Broker* and the Oracle9*i* Data Guard Manager online help

- Data Guard command-line interface

  The Data Guard command-line interface is an alternative to using the Oracle Data Guard Manager GUI. The command-line interface is useful if you want to manage a Data Guard configuration from batch programs or scripts.

  See Also: *Oracle9i Data Guard Broker*

- Recovery Manager (RMAN)

  You can use **Recovery Manager (RMAN)** to create and back up a standby database.

  See Also: *Oracle9i Recovery Manager User's Guide*

1.13 Standby Database Creation

Once you have a primary database, you can create a standby database by using one of the following methods:

- Oracle9*i* Data Guard Manager Wizard

  See Also: *Oracle9i Data Guard Broker* and the Oracle9*i* Data Guard Manager online help for information on the Data Guard Manager GUI

- Operating system commands and RMAN

  See Also: *Oracle9i Recovery Manager User's Guide* for the procedures to follow when you use RMAN to create a standby database

- Operating system commands and SQL statements

  See Also: Section 2.2, Section 6.1, and Section 6.2

2

Creating the Standby Database Environment

This chapter explains creating the standby database portion of the Data Guard environment. It includes the following topics:

- Considerations for Creating the Standby Database Environment
- Creating a Standby Database: Basic Tasks
- Creating the Standby Database Files
- Creating the Standby Initialization Parameter File

2.1 Considerations for Creating the Standby Database Environment

A **standby database environment** includes a primary database with one or more associated standby databases. This section describes the main factors affecting the Data Guard configuration:

- Number of Standby Databases
- Typical Modes for a Standby Database in the Data Guard Environment
- Method of Archiving Redo Logs to the Standby Site
- Method of Applying Redo Logs on the Standby Site
- Location and Directory Structure of Primary and Standby Sites
- Advantages of Using Multiple Standby Databases

2.1.1 Number of Standby Databases

Although a standby database can be synchronized with one and only one primary database, a single primary database can support up to nine standby databases. These standby databases are separate and independent, and can reside on multiple systems or on a single system.

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

2.1.2 Typical Modes for a Standby Database in the Data Guard 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

For maximum protection against data loss or corruption, maintain the standby database in managed recovery mode in a Data Guard 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 manually copy the archived logs to the standby database. To learn why and how to avoid the need to, see Section 4.5.

To use the standby database for reporting purposes, open it in read-only mode in a Data Guard environment. Log apply services 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 change between managed recovery mode and read-only mode. In most implementations of a Data Guard environment, you perform this change 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: Section 4.3 to learn how to initiate managed recovery mode. See Section 4.7 to learn how to open a standby database in read-only mode.

2.1.3 Method of Archiving Redo Logs to the Standby Site

One crucial aspect of any Data Guard environment is **archiving** the redo logs from the primary site to the standby site. You have two options for archiving the redo logs:

- Configure the primary database to archive automatically to the standby site using log transport services
- Archive the logs manually using operating system commands

When a primary database archives to a standby site, log transport services automatically archive the online redo logs through Oracle Net to a directory on the standby site. As redo logs are generated on the primary database, log transport services automatically archive them and log apply services apply them to each standby database. This allows standby databases to remain synchronized with the primary database. Log transport services help minimize the latency between the primary and standby databases by continuously capturing the modifications from the archived redo logs.

See Also: Chapter 3, "Log Transport Services" and Chapter 4, "Log Apply Services"

2.1.3.1 Independence of Automatic Archival and Managed Recovery

The primary database can continue to archive to the standby site even if the standby database is not in managed recovery mode, but only if the standby instance is mounted. The mechanism for recovery of a standby database is independent of the mechanism for automatic archival of 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.

2.1.3.2 Optional Manual Copy of Archived Redo Logs

Even if you configure the primary database to archive automatically to the standby site, you can still copy the completed archived redo logs manually if the occasion requires it, but only if you have specified the standby site as an optional destination.

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

See Also: Section 3.3.1

2.1.3.3 Functions of Log Transport Services

Log transport services provide the following functions:

- Control of different log archiving mechanisms
- Automatic log archiving
- Error handling and reporting
- Automatic archive gap detection and retrieval of lost logs
- Guaranteed no-data-loss data protection
- Selectable log transport modes that balance data protection needs with availability and performance
- Increased DBA and operator productivity (by eliminating the need for operator intervention)
- Ability to fetch archived logs from any number of servers, which reduces the risk that log archiving will stop because of data communications failures, out-of-space conditions, or other reasons
- Ability to configure a standby database to receive but not apply archived logs, which avoids most of the storage and processing expense of another fully configured standby database

2.1.4 Method of Applying Redo Logs on the Standby Site

The log apply services component of the Data Guard environment is responsible for maintaining the standby database in either a managed recovery mode or in open read-only mode. It coordinates activities with the log transport services.

2.1.4.1 Functions of Log Apply Services

Log apply services provide the following functions:

- Automatic application of archived logs

- Automatic archive gap recovery
- Enhanced monitoring capability
- Automatic delayed application of archived logs
- Partial archived log recovery (recovery from primary database failure)

2.1.5 Location and Directory Structure of Primary and Standby Sites

One crucial aspect of the Data Guard environment is the number and configuration of the systems involved. Of particular importance are whether:

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

2.1.5.1 Number and Location of Standby Databases

You can locate a standby database:

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

The location of sites involved in the Data Guard environment has obvious implications for a disaster recovery strategy. For example, if the primary site in a data center is destroyed, then you cannot perform failover to a standby database unless it resides on a different site, 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 system in a remote location.

See Also: Section 6.5 for a disaster recovery scenario

2.1.5.2 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. If you have a standby database on the same site as the primary database, you must use a different directory structure; otherwise, the standby database attempts to overwrite the primary database files.

Use the same path names for the standby files if possible. This option eliminates the need to set filename conversion parameters. Nevertheless, if you need to use a site with a different directory structure or place the standby and primary databases on the same site, you can do so with a minimum of extra administration.

2.1.5.3 Configuration Options

The three basic configuration options are illustrated in Figure 2–1. These include:

- A standby database on the same site as the primary database that uses a different directory structure than the primary site (`Standby1`)
- A standby database on a separate site that uses the same directory structure as the primary site (`Standby2`)
- A standby database on a separate site that uses a different directory structure than the primary site (`Standby3`)

Figure 2–1 Possible Standby Configurations

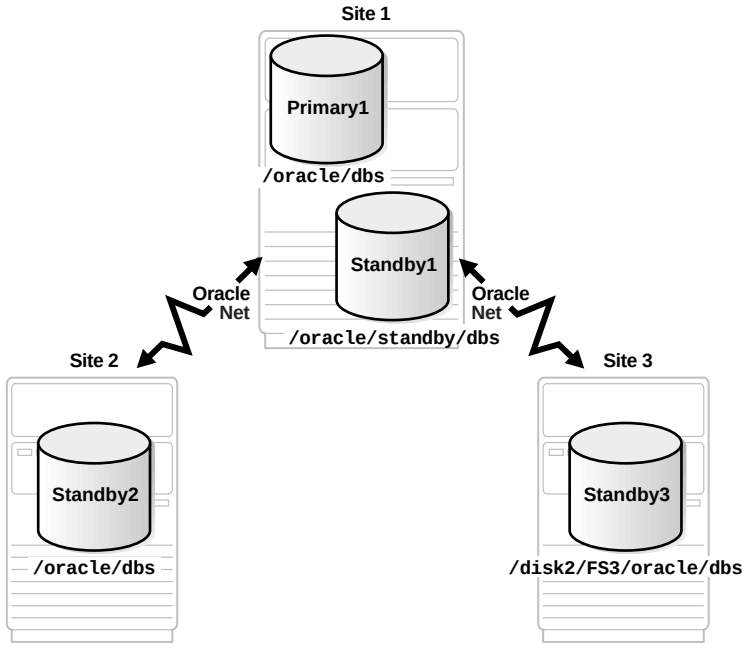


Table 2–1 describes possible configurations of primary and standby databases and the consequences of each.

Table 2–1 Primary and Standby Database Configurations

Standby Site	Directory Structure	Consequences
Same as primary site	Different than primary site (required)	■ You must set the LOCK_NAME_SPACE initialization parameter. ■ You must rename primary database datafiles in the standby database control file. You can either manually rename the datafiles (see Section B.4) or set up the DB_FILE_NAME_CONVERT initialization parameter on the standby database to automatically rename the datafiles (see Section 4.6). ■ Some operating systems do not permit two instances with the same name to run on the same system. Refer to your platform-specific documentation for more information. ■ The standby database does not protect against disaster.
Separate site	Same as primary site	■ You do not need to rename primary database filenames in the standby database 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 site	Different than primary site	■ You must rename primary database datafiles in the standby database control file. You can either manually rename the datafile (see Section B.4) or set up the DB_FILE_NAME_CONVERT initialization parameter on the standby database to automatically rename the datafiles (see Section 4.6). ■ Using separate physical media for your databases safeguards your primary data.

2.1.6 Advantages of Using Multiple Standby Databases

A given standby database must use only one configuration. However, 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 site as the primary database, another standby database on a separate site in the same data center, and a third standby database on a separate site on the other side of the world. You can run each standby database in managed 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 fail over to the standby database that has not yet become corrupted.

2.2 Creating a Standby Database: 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. Figure 2–2 shows creation of a standby database.

Figure 2–2 Standby Database Creation

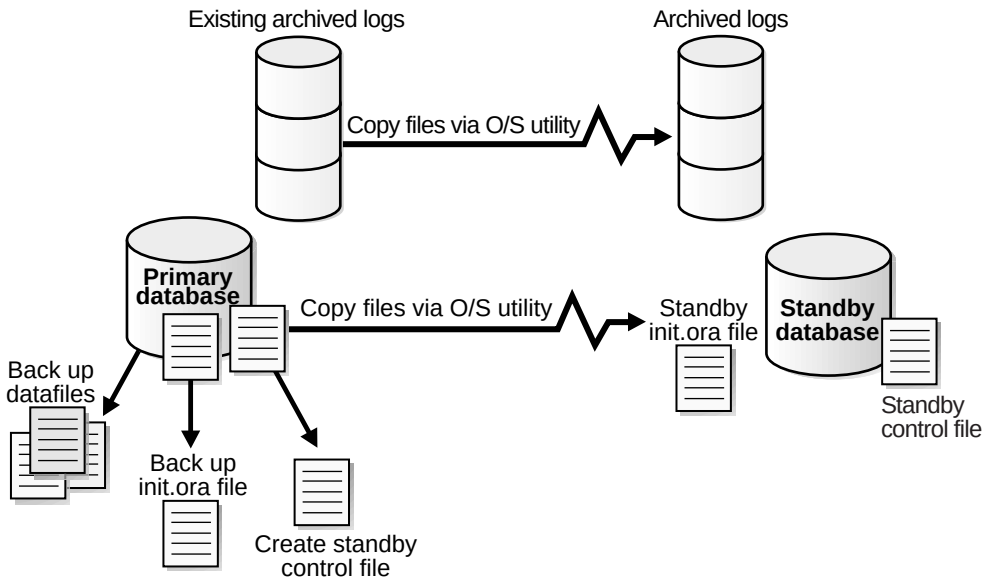


Table 2–2 summarizes the basic tasks for setting up a standby database and synchronizing it so that it is ready to begin managed recovery.

Note: You can do these steps automatically with the Create Configuration Wizard shipped with the Oracle9*i* Data Guard Manager.

Table 2–2 Task List: Preparing for Managed Recovery

Step	Task	See ...
1	Either make a new backup of the primary database datafiles or access an old backup.	Section 2.3.2
2	Ensure the primary database is in ARCHIVELOG mode.	Section 2.3.3
3	Connect to the primary database and create the standby control file.	Section 2.3.3
4	Copy the backup datafiles and control file from the primary site to the standby site.	Section 2.3.4
5	Set the initialization parameters for the primary database.	Section 3.5.3
6	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.	Section 3.5.1 and Section 4.6
7	Start the standby instance and mount the standby database.	Section 4.3.1
8	Create standby redo log files, if necessary.	Section 3.6.3.4
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.	Section B.4
10	Manually enable initialization parameter changes on the primary database so that it can initiate archival to the standby site.	Section 3.5 and Section 4.6

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

Step	Task	See ...
11	Use the Oracle Net Manager to configure a **listener** on the standby database. If you plan to manage this standby database using the Data Guard broker, you must configure the listener to use the TCP/IP protocol and statically register the standby database service using its SID.	*Oracle Net Services Administrator's Guide* for information about configuring and administering the listener
12	Use the Oracle Net Manager to create a net service name that the standby database can use to connect to the primary database. The net service name must resolve to a connect descriptor that uses the same protocol, host address, port, and SID that you specified when you configured the listener on the primary database site. If you are unsure what values to use for these parameters, run the Oracle Net Manager on the primary database site to display the listener configuration.	*Oracle Net Services Administrator's Guide* for information about configuring naming methods
13	Use the Oracle Net Manager to create a net service name that the primary database can use to connect to the standby database. The net service name must resolve to a connect descriptor that uses the same protocol, host address, port, and SID that you specified when you configured the listener on the standby database site. If you are unsure what values to use for these parameters, run the Oracle Net Manager on the standby database site to display the listener configuration.	*Oracle Net Services Administrator's Guide* for information about configuring naming methods

2.3 Creating the Standby Database Files

You can create a standby database on the same site as your primary database or on a separate site. If you create your standby database on the same site, 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 four stages:

1. Using Backups for Standby Creation

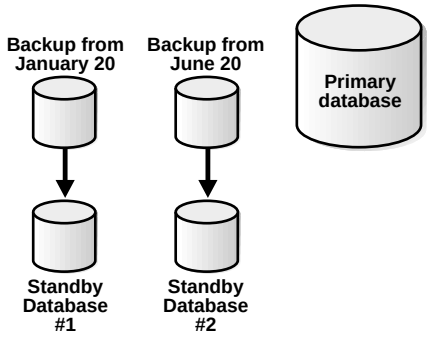
2. Creating the Standby Datafiles
3. Creating the Standby Control File
4. Copying Files to the Standby Site

2.3.1 Using Backups for Standby Creation

Each standby database must be created from a backup of the primary database.

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. Figure 2–3 shows that 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. So long as you have the archived redo logs required to perform complete recovery of a backup, it can serve as the basis for a standby database.

Figure 2–3 Creating Standby Databases Using Different Backups



2.3.2 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** (or **open backups**) allow 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 session on your primary database and query the V$DATAFILE fixed view to obtain a list of the primary datafiles. For example, enter:

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

2. Shut down the primary database cleanly:

```
SQL> SHUTDOWN;
```

3. Make a consistent backup of the datafiles from your primary database using the Recovery Manager utility (RMAN) or an operating system utility.

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

```
SQL> STARTUP pfile=initPRIMARY1.ora;
```

> **See Also:** *Oracle9i User-Managed Backup and Recovery Guide* to learn how to make operating system backups and *Oracle9i Recovery Manager Reference* for more information on the STARTUP statement

2.3.3 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 be created after the latest time stamp 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 SQL*Plus ARCHIVE LOG LIST command or query the V$DATABASE view. Take the following steps:

   a. Start and mount the primary database without opening it. For example:

   ```
   SQL> STARTUP MOUNT PFILE=initPRIMARY1.ora
   ```

   b. Issue the SQL*Plus ARCHIVE LOG LIST command to determine if the database is in ARCHIVELOG mode. For example:

   ```
   SQL> ARCHIVE LOG LIST;
   Database log mode              No Archive Mode
   Automatic archival             Disabled
   Archive destination            /oracle/dbs/arch
   Oldest online log sequence     0
   Current log sequence           1
   ```

   c. If the database is not in ARCHIVELOG mode, as shown in step b, issue the following command to place the database in ARCHIVELOG mode:

   ```
   SQL> ALTER DATABASE ARCHIVELOG;
   ```

   d. You can issue the SQL*Plus ARCHIVE LOG LIST command again to verify the database has been placed in ARCHIVELOG mode. For example:

   ```
   SQL> ARCHIVE LOG LIST;
   Database log mode              Archive Mode
   Automatic archival             Disabled
   Archive destination            /oracle/dbs/arch
   Oldest online log sequence     0
   Next log sequence to archive   1
   Current log sequence           1
   ```

   To enable the automatic archival of the online redo logs, you must set LOG_ARCHIVE_START=true in the initialization parameter file. However, this does not have to be done before you create the standby control file.

See Also: *SQL*Plus User's Guide and Reference* for additional information on the `ARCHIVE LOG LIST` command and *Oracle9i Database Administrator's Guide* for additional information on the `ALTER DATABASE ARCHIVELOG` statement and the `LOG_ARCHIVE_START` initialization parameter

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

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

The filename for the created standby control file *must* be different from the filename of the **current control file** of the primary database. You can also use RMAN to create the standby database control file.

See Also: *Oracle9i SQL Reference* for additional information on the `ALTER DATABASE` statement and *Oracle9i Recovery Manager User's Guide* for additional information on RMAN

2.3.4 Copying Files to the Standby Site

After you have successfully created the standby datafiles and control file, copy the files to the standby site using an operating system utility.

If the standby database is on	Then you
A separate site 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 site as the primary database, or the standby database is on a separate site 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 Section 3.5.1 and Section 4.6. ■ Rename the files manually using `ALTER DATABASE` statements. See Section B.4. ■ Use a combination of conversion parameters and manual renames.

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

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

1. Copy the control file.
2. Copy the backup datafiles.
3. Copy all available archived redo logs to the standby site.
4. Copy the online redo logs. This is recommended for switchover and failover operations.

See Also: Section 5.3.4 and Section 5.4.7.2

Do not copy temporary tablespaces. See Section 4.8.2.1 for more information about creating temporary tablespaces.

2.4 Creating the Standby 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.

See Also: Section 3.5.1, Section 4.6, and Section 6.2.6

3

Log Transport Services

This chapter explains how to set up and use log transport services to control the automated archival of redo logs from the primary database to one or more standby sites. It includes the following topics:

3.1 Introduction to Log Transport Services

The **log transport services** component of the Data Guard environment is responsible for the automatic archival of primary database online redo logs. Once archived, these logs are known as **archived redo logs.** Log transport services provide for the management of archived redo log permissions, destinations, transmission, reception, and transmission failure resolution. In a Data Guard environment, the log transport services component coordinates its activities with the log apply services component.

See Also: Chapter 4, "Log Apply Services"

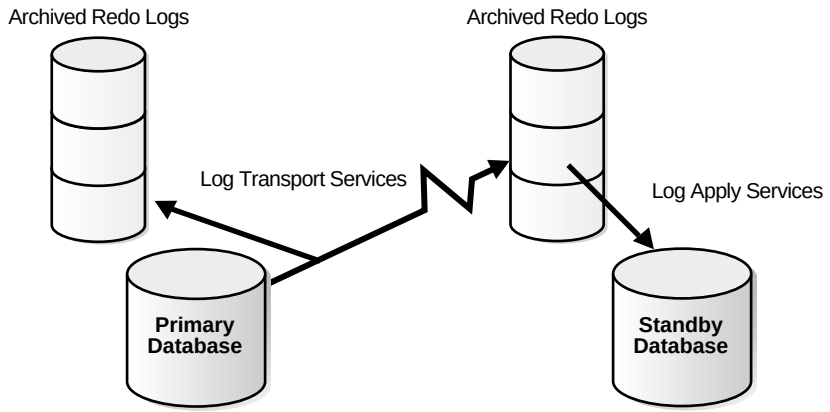
3.1.1 Background

Log transport services and log apply services automate the process of copying online redo logs to a remote node and applying them so that data from a primary database can be made available on a standby database. Log transport services provide the capability to archive the primary databases's redo log files to a maximum of ten archiving locations called **destinations.**

Log transport services provide flexibility in defining destinations and archiving completion requirements. These services also provide I/O failure handling and transmission restart capability.

Figure 3–1 shows a simple Data Guard configuration with redo logs being archived from a primary database to a local destination and a remote standby database destination using log transport services.

Figure 3–1 Archiving Redo Logs



3.1.2 Functional Overview

The archiver or **log writer process (LGWR)** can be configured to archive online redo logs to a maximum of ten destinations. These destinations are specified using database initialization parameters. One destination must be a local directory, but others can be in remote locations. A remote destination is identified by using an Oracle Net network service name, which is defined in the appropriate network

services configuration files. It is the job of the archiver or log writer process to archive redo logs to the remote sites.

Archiving redo logs to a remote destination requires uninterrupted connectivity through Oracle Net. If the destination is a standby database, it must be mounted or open in read-only mode. If a network connection to a standby database is broken, it must be reestablished to continue log archiving.

3.1.3 Log Transport Services Process Architecture

Log transport services use the following processes on the primary site:

- Log writer (LGWR)

  The log writer process (LGWR) collects transaction redo and updates the online redo logs. The log writer process can also create local archived redo logs and transmit online redo to standby databases.

- Archiver (ARC*n*)

  The archiver process (ARC*n*), or a SQL session performing an archival operation, creates a copy of the online redo logs, either locally or remotely, for standby databases.

- Fetch archive log (FAL) client

  This background Oracle database server process *pulls* archived redo log files from the primary site. The FAL client initiates and requests the automatic archiving of redo logs when it detects an archive gap on the standby database.

- Fetch archive log (FAL) server

  This background Oracle database server process runs on the primary database and services the fetch archive log (FAL) requests coming from the FAL client. For example, servicing a FAL request might include queuing requests (to send archived redo log files to one or more standby databases) to an Oracle database server that runs the FAL server. Multiple FAL servers can run on the same primary database at one time. A separate FAL server is created for each incoming FAL request.

3.2 Log Transport Services Capabilities

Log transport services capabilities allow you to determine how the archival of online redo logs will be handled in a Data Guard configuration. These capabilities can be classified into five categories:

- Permission
- Destination
- Transmission
- Reception
- Failure Resolution

3.2.1 Permission

You control whether the primary database can archive redo logs to a standby site, and whether the standby database can receive them from a primary site. Permission is set using the REMOTE_ARCHIVE_ENABLE database initialization parameter.

See Also: Section 3.4.2.1

3.2.2 Destination

Using log transport services, redo logs can be archived to any of the following destinations:

- Local

  A local file system location.

- Remote

  A remote location accessed using an Oracle Net service name. There are several types of remote destinations, as shown in the following table:

Remote Destination Types	Description
Physical standby database	An identical copy of a primary database used for disaster protection.
Archive log repository	An off-site storage location for archived redo logs. Not to be used for disaster protection.
Cross-instance archival database	Used within Real Application Clusters, another instance of the primary database. This is an archiving destination for all redo logs from the cluster and is on the node where recovery is performed.

See Also:

- Section 3.4.2 for more information about specifying destinations and destination options
- Section 5.9.1 for information about cross-instance archival

3.2.3 Transmission

You can choose the method by which redo logs are archived from the primary database to each archive destination by specifying:

- The process that will perform the archival operation

  You can specify that the log writer process or that the archiver process performs the archival operation.

- The method of transmitting redo logs over the network

  When using the log writer process for archival operations, you can specify synchronous or asynchronous network transmission of redo logs to remote destinations.

  When using the archiver process for archival operations, only synchronous transmission of redo logs to remote destinations is available.

  Using transmission options, you can configure log transport services availability modes. Availability modes give you the flexibility to specify various levels of protection for your data in the Data Guard environment.

- The method of writing archived redo logs to disk

  You can specify whether log archiving disk write I/O operations are to be performed synchronously or asynchronously.

  See Also: Section 3.6.2 for more information on log transport services data availability modes

3.2.4 Reception

Reception gives you control over determining where redo logs will be placed at remote destinations. The redo data can be stored on the standby site using either standby redo logs or archived redo logs.

Oracle9*i* introduces the concept of **standby redo logs.** Standby redo logs form a separate pool of log file groups. Standby redo log file groups are only used when a database is operating in a physical standby role.

See Also: Section 3.6.3.4 for more information about reception options

3.2.5 Failure Resolution

Failure resolution gives you control over determining what actions will occur on a primary database if log archiving from the primary database to the standby database fails. These actions include:

- Retrying the archival operation to a failed destination after a specified period of time and specifying a limit to the number of times a destination will be retried
- Specifying an alternate or substitute destination
- Specifying that archival to a destination is dependent upon the success of an archival operation to another destination
- Specifying that if connectivity to the standby database is lost, then the primary database will automatically shut down, ensuring that data in the primary database is always protected by the presence of a standby database

See Also: Section 3.4.2.5 and Section 3.6.3.5 for more information about failure resolution options

3.3 Log Transport Services Interfaces

Log transport services options are entered and changed using the following interfaces:

- Database Initialization Parameters
- SQL Interface

3.3.1 Database Initialization Parameters

You configure the primary database to perform remote archiving by setting destinations and associated states. You do this using `LOG_ARCHIVE_DEST_`*n* initialization parameters and corresponding `LOG_ARCHIVE_DEST_STATE_`*n* initialization parameters.

See Also: Chapter 8, "LOG_ARCHIVE_DEST_n Parameters Attributes"

3.3.1.1 LOG_ARCHIVE_DEST_*n* Initialization Parameters

LOG_ARCHIVE_DEST_*n* (where *n* is an integer from 1 to 10) initialization parameters allow you to specify up to ten archival destinations, including one required local destination and up to nine additional local or remote destinations. These parameters also allow you to set a number of archival options for each destination. These options are set using the attributes described in Table 3–1 and in Chapter 8.

Table 3–1 LOG_ARCHIVE_DEST_n Initialization Parameters Attributes

Attribute	Description	Capability	More Information
[NO]AFFIRM	Log transport services will ensure that redo logs are successfully archived to a destination and are immediately available for recovery operations	Transmission	See Section 3.6.3.3 and see AFFIRM and NOAFFIRM in Chapter 8
[NO]ALTERNATE	Specifies an alternate location that can be used as a destination for archival operations if archival to the associated destination fails	Failure resolution	See Section 3.4.2.5 and see ALTERNATE and NOALTERNATE in Chapter 8
ARCH	The archiver process will archive online redo logs to local and remote destinations	Transmission	See Section 3.6.3.1 and see ARCH and LGWR in Chapter 8
ASYNC=*blocks*	Network I/O operations for log transport services are to be done asynchronously. Specifies the size of the SGA network buffer to be used.	Transmission	See Section 3.6.3.2 and see SYNC and ASYNC in Chapter 8
[NO]DELAY[=*minutes*]	Specifies a time interval between archiving a redo log at a remote site and applying that archived redo log to the standby database	Reception	See Section 3.4.2.8 and see DELAY and NODELAY in Chapter 8

Table 3–1 (Cont.) LOG_ARCHIVE_DEST_n Initialization Parameters Attributes

Attribute	Description	Capability	More Information
[NO]DEPENDENCY	Archival operations to a destination are dependent upon the success or failure of archival operations to another local destination	Transmission	See Section 3.4.2.6 and see DEPENDENCY and NODEPENDENCY in Chapter 8
LGWR	The log writer process will archive current online redo logs to local and remote destinations	Transmission	See Section 3.6.3.1 and see ARCH and LGWR in Chapter 8
LOCATION	Identifies a local disk directory location where archived redo logs will be stored	Destination	See Section 3.4.2.3 and see LOCATION and SERVICE in Chapter 8
MANDATORY	Archiving to this location must succeed for log transport services operations to continue	Failure resolution	See Section 3.4.2.3 and see MANDATORY and OPTIONAL in Chapter 8
[NO]MAX_FAILURE=*count*	Sets a limit on the number of times log transport services will retry archival operations to a remote location after a communications failure	Failure resolution	See Section 3.4.2.5 and see MAX_FAILURE and NOMAX_ FAILURE in Chapter 8
OPTIONAL	Successful archiving to this location is not necessary	Failure resolution	See Section 3.4.2.3 and see MANDATORY and OPTIONAL in Chapter 8
[NO]QUOTA_SIZE=*blocks*	The maximum number of 512-byte blocks that can be archived on the specified destination	Transmission	See Section 3.4.2.9 and see QUOTA_SIZE and NOQUOTA_SIZE in Chapter 8
[NO]QUOTA_USED	Identifies the size of all of the archived redo logs currently residing on the specified destination	Transmission	See QUOTA_USED and NOQUOTA_USED in Chapter 8
[NO]REGISTER	Specifies whether or not the archival location is to be recorded in the standby database control file	Reception	See REGISTER and NOREGISTER in Chapter 8

Table 3–1 (Cont.) LOG_ARCHIVE_DEST_n Initialization Parameters Attributes

Attribute	Description	Capability	More Information
REGISTER=*location_format*	The remote archival location is to be recorded in the standby database control file	Reception	See Section 3.4.2.7 and see REGISTER=location_format in Chapter 8
[NO]REOPEN[=*seconds*]	Specifies the number of seconds that log transport services will wait before retrying an archival operation to a remote location after a communications failure	Failure resolution	See Section 3.4.2.5 and see REOPEN and NOREOPEN in Chapter 8
SERVICE	Specifies the net service name of a standby database where redo log files are to be archived	Destination	See Section 3.4.2.2 and see LOCATION and SERVICE in Chapter 8
SYNC	Network I/O operations for log transport services are to be done synchronously	Transmission	See Section 3.6.3.2 and see SYNC and ASYNC in Chapter 8

Note: The use of LOG_ARCHIVE_DEST_*n* parameters on the primary database replaces the use of LOG_ARCHIVE_DEST and LOG_ARCHIVE_DUPLEX_DEST parameters on the primary database in previous versions of the Oracle database server.

3.3.1.2 LOG_ARCHIVE_DEST_STATE_*n* Initialization Parameters

The LOG_ARCHIVE_DEST_STATE_*n* (where *n* is an integer from 1 to 10) initialization parameters specify the *state* of the corresponding destination indicated by the LOG_ARCHIVE_DEST_*n* parameters (where *n* is the same integer). For example, the LOG_ARCHIVE_DEST_STATE_3 parameter specifies the state of the LOG_ARCHIVE_DEST_3 destination.

LOG_ARCHIVE_DEST_STATE_*n* parameters attributes are described in Table 3–2.

Table 3–2 LOG_ARCHIVE_DEST_STATE_n Initialization Parameters Attributes

Attribute	Description
ENABLE	Log transport services can archive redo logs at this destination.
DEFER	Log transport services will not archive redo logs to this destination. This is an unused destination.
ALTERNATE	This destination is not enabled but will become enabled if communications to another destination fail.

3.3.1.3 Setting Destination Parameters in the Database Initialization File

Setting up log transport services requires modification of the database initialization file. When you set up log transport services parameters, you can specify attributes as:

- A single attribute on one line
- Multiple attributes on one line
- Single attributes on multiple sequential lines
- Attributes for multiple destinations on multiple lines

Example 3–1 shows how to specify a single attribute on one line.

Example 3–1 Specifying a Single Attribute on One Line

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1/d1arch'
```

Example 3–2 shows how to set multiple attributes on a single line.

Example 3–2 Specifying Multiple Attributes on One Line

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1/d1arch OPTIONAL'
```

Example 3–3 shows how to set multiple attributes incrementally on separate lines. SERVICE or LOCATION attributes must be specified on the first line.

Example 3–3 Specifying Multiple Attributes Incrementally

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1/d1arch'
LOG_ARCHIVE_DEST_1='OPTIONAL'
LOG_ARCHIVE_DEST_1='REOPEN=5'
```

Example 3–4 shows how to specify attributes for multiple destinations.

Example 3–4 Specifying Multiple Attributes for Multiple Destinations

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1/d1arch OPTIONAL'
LOG_ARCHIVE_DEST_2='SERVICE=stby REOPEN=60'
```

Attributes specified on multiple lines for a single destination must be entered sequentially. Example 3–5 shows an entry for the LOG_ARCHIVE_DEST_1 that is not allowed.

Example 3–5 Incorrect Destination Specification

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1/d1arch'
LOG_ARCHIVE_DEST_2='SERVICE=stby REOPEN=60'
LOG_ARCHIVE_DEST_1='OPTIONAL'
```

A destination specification can be redefined after another destination has been specified if the new specification includes LOCATION or SERVICE attributes. Example 3–6 shows how to replace the initial specification of LOG_ARCHIVE_ DEST_1.

Example 3–6 Replacing a Destination Specification

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1/d1arch'
LOG_ARCHIVE_DEST_2='SERVICE=stby REOPEN=60'
LOG_ARCHIVE_DEST_1='LOCATION=/disk3/d3arch MANDATORY'
```

A string containing a null value for parameter attributes will clear a previously entered destination specification. Example 3–7 shows how to clear the definition of LOG_ARCHIVE_DEST_1.

Example 3–7 Clearing a Destination Specification

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1/d1arch'
LOG_ARCHIVE_DEST_2='SERVICE=stby REOPEN=60'
LOG_ARCHIVE_DEST_1=''
```

To specify service names that use embedded characters, such as equal signs (=) and spaces, use double quotation marks (") around the service name. Example 3–8 shows the specification of a service name that includes a space.

Example 3–8 Specifying a Service Name That Includes a Space

```
LOG_ARCHIVE_DEST_6='SERVICE="stdby arch" MANDATORY'
```

Caution: Oracle Corporation does not recommend creating service names that contain spaces.

3.3.2 SQL Interface

SQL statements can be used to set up most initialization parameters for log transport services and to monitor the environment. You can use SQL statements for the following tasks:

- Viewing Current Settings of Initialization Parameters
- Changing Initialization Parameter File Settings
- Setting Log Transport Services Options Using SQL

3.3.2.1 Viewing Current Settings of Initialization Parameters

You can use SQL to query fixed views such as V$ARCHIVE_DEST to see current LOG_ARCHIVE_DEST_*n* initialization parameters settings. For example, to view current destination settings on the primary database, enter the following statement:

```
SQL> SELECT DESTINATION FROM V$ARCHIVE_DEST;
```

3.3.2.2 Changing Initialization Parameter File Settings

At runtime, LOG_ARCHIVE_DEST_*n* initialization parameters can be changed using ALTER SYSTEM and ALTER SESSION statements. You can specify the attributes in one or more strings in one statement or incrementally in separate statements.

Example 3–9 shows how to modify archive destination parameters at the system and session level.

Example 3–9 Modifying Destination Parameters at the System and Session Level

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_6='SERVICE=stby REOPEN=8';
SQL> ALTER SESSION SET LOG_ARCHIVE_DEST_6='SERVICE=stby2 REOPEN=10';
```

Unlike setting parameters in the database initialization file at startup, at runtime you can incrementally change the characteristics of a destination after modifying the settings of another destination. Example 3–10 shows how to make incremental changes to LOG_ARCHIVE_DEST_6 after LOG_ARCHIVE_DEST_7 has been defined.

Example 3–10 Adding Destination Attributes Incrementally

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_6='SERVICE=stby1 REOPEN=8';
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_7='SERVICE=stby2 NOREOPEN';
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_6='MANDATORY LGWR DELAY';
```

You can enter a null value for a destination to clear a previous definition. Example 3–11 shows how to clear the definition of LOG_ARCHIVE_DEST_6.

Example 3–11 Clearing a Destination Specification

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_6='SERVICE=stby1 REOPEN=8';
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_7='SERVICE=stby2 NOREOPEN';
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_6='';
```

To specify service names that use embedded equal signs (=) and spaces, use double quotation marks (") around the service name. Example 3–12 shows the specification of a service name that includes a space.

Example 3–12 Specifying a Service Name That Includes a Space

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_6='SERVICE="stdby arch" MANDATORY';
```

Caution: Oracle Corporation does not recommend creating service names that contain spaces.

Table 3–3 lists the attributes that can be set for LOG_ARCHIVE_DEST_*n* initialization parameters and indicates whether the attribute can be changed using an ALTER SYSTEM or ALTER SESSION statement.

Table 3–3 Changing Destination Attributes Using SQL

Attribute	ALTER SYSTEM	ALTER SESSION
[NO]AFFIRM	Yes	Yes
[NO]ALTERNATE	Yes	Yes
ARCH	Yes	Yes
ASYNC	Yes	No
[NO]DELAY	Yes	Yes
[NO]DEPENDENCY	Yes	No

Table 3–3 (Cont.) Changing Destination Attributes Using SQL

Attribute	ALTER SYSTEM	ALTER SESSION
LGWR	Yes	No
LOCATION	Yes	Yes
MANDATORY	Yes	Yes
[NO]MAX_FAILURE	Yes	No
OPTIONAL	Yes	Yes
[NO]QUOTA_SIZE	Yes	No
[NO]QUOTA_USED	Yes	No
[NO]REGISTER	Yes	Yes
REGISTER=*location_format*	Yes	Yes
[NO]REOPEN	Yes	Yes
SERVICE	Yes	Yes
SYNC	Yes	Yes

3.3.2.3 Setting Log Transport Services Options Using SQL

You can set specific options for log transport services using SQL statements. The following example sets PROTECTED mode.

```
SQL> ALTER DATABASE SET STANDBY DATABASE PROTECTED;
```

> **See Also:** *Oracle9i SQL Reference* for more information about SQL statements

3.4 Configuring Log Transport Services on the Primary Database

This section discusses what settings can be made on the primary database to use the various options of log transport services in a Data Guard environment. The following topics are included in this section:

- Configuring Log Transport Services
- Setting Up the Log Transport Services Environment

3.4.1 Configuring Log Transport Services

Considerations when configuring log transport services on the primary database include:

- Setting the Database to Run in ARCHIVELOG Mode
- Configuring Online Redo Logs in the Primary Database
- Controlling the Reuse of Archived Redo Logs
- General Considerations for Configuring Log Transport Services

3.4.1.1 Setting the Database to Run in ARCHIVELOG Mode

To use log transport services, you must set the primary database to run in **ARCHIVELOG mode.** When you run a database in ARCHIVELOG mode, you enable the archiving of the online redo logs. The database control file indicates that a group of filled online redo logs cannot be reused by the log writer process until the group is archived. A filled group is immediately available for archiving after a redo **log switch** occurs.

> **See Also:** *Oracle9i Database Administrator's Guide* for details about setting ARCHIVELOG mode

Log transport services do not require filled online redo logs to be archived to disk if you run in **NOARCHIVELOG mode.** You can set this mode at database creation or by using the SQL ALTER DATABASE statement.

> **Note:** Oracle Corporation does not recommend running in NOARCHIVELOG mode because it severely limits the possibilities for recovery of lost data.

3.4.1.2 Configuring 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. In general, the most important factor when considering the size of an online redo log should be the amount of application data that needs to be applied to a standby database during a database failover operation. The larger the online redo log, the more data that needs to be applied to a standby database to make it consistent with the primary database.

Another important consideration should be the size of the archival media. Online redo logs should be sized so that a filled online redo log group can be archived to a

single unit of offline storage media (such as a tape or disk), with the least amount of space on the medium left unused. For example, suppose only one filled online redo log group can fit on a tape, and 49% of the tape's storage capacity remains unused. In this case, it is better to decrease the size of the online redo log files slightly, so that two log groups can be archived per tape.

The best way to determine the appropriate number of online redo log files for a database instance is to test different configurations. The goal is to create the fewest groups possible without hampering the log writer process's ability to write redo log information.

In some cases, a database instance may require only two groups. In other situations, a database instance may require additional groups to guarantee that a recycled group is always available to the log writer process. During testing, the easiest way to determine if the current configuration is satisfactory is to examine the contents of the log writer process's trace file and the database's alert log. If messages indicate that the log writer process frequently must wait for a group because a checkpoint has not completed or a group has not been archived, add more groups.

Table 3–4 provides some guidelines to help in determining the number and sizes of primary database online redo logs.

Table 3–4 Guidelines for Online Redo Log Configuration

If the online redo logs are:	Advantages	Disadvantages
Small (50 megabytes)	■ Standby database lag time is minimized ■ Ideal when using the archiver process (ARC*n*) to exclusively archive online redo logs	■ Log switches occur more frequently ■ A larger number of online redo log groups may be required to allow log switches to occur smoothly ■ The large number of archived redo logs may be difficult to manage ■ Additional archiver processes may be required ■ More frequent database checkpoints may occur

Table 3–4 (Cont.) Guidelines for Online Redo Log Configuration

If the online redo logs are:	Advantages	Disadvantages
Medium (200 megabytes)	■ Ideal when using a combination of archiver and log writer processes to archive online redo logs	■ The large number of archived redo logs may be difficult to manage
Large (1000 megabytes)	■ Fewer online redo log groups are required ■ Archival occurs less frequently and is more efficient ■ Archived redo logs are easier to manage ■ Recovery is simplified because of fewer archived redo logs to apply ■ Ideal when using the log writer process (LGWR) exclusively to archive online redo logs ■ Fewer database checkpoints	■ Standby database lag time can be high ■ Increased potential for data loss when using the archiver process ■ Archived redo logs may not fit efficiently on backup media ■ Instance recovery may take longer

3.4.1.3 Controlling the Reuse of Archived Redo Logs

Set the `CONTROL_FILE_RECORD_KEEP_TIME` initialization parameter to prevent reusing archived redo logs for a specified period of days. Setting this parameter prevents the ARCHIVELOG mechanism from overwriting information in the archived redo logs. Using this parameter helps to ensure that data is made available on the standby database. The range of values for this parameter is 0 to 365 days. The default value is 7 days. If you set the parameter to 0, the redo logs are reused and overwritten immediately.

See Also: *Oracle9i Database Reference* for more details about the CONTROL_FILE_RECORD_KEEP_TIME initialization parameter

3.4.1.4 General Considerations for Configuring Log Transport Services

The Oracle database server will attempt a checkpoint at each log switch. Therefore, if the online redo log size is too small, frequent log switches will lead to frequent checkpointing and negatively affect system performance. Oracle Corporation recommends a checkpoint and log switch interval between 5 and 15 minutes during normal operations and at 5-minute intervals during peak load periods, such as during batch processing. To get started, examine the V$SYSSTAT view during a peak load period and then calculate 5 to 10 minutes of redo.

If you have remote hardware mirroring, you may want a minimum number of log groups, to ensure that no-data-loss failover can occur in case some logs are not archived. Furthermore, if some logs are not archived, they are not available to the standby database. This is another reason to avoid overly large online redo logs.

If you do not have remote hardware mirroring, the loss of a primary site will mean the loss of at least two logs of data. The sizing of the log files should factor in how much data loss your site can tolerate.

You should avoid situations where you cannot overwrite a redo log because you must wait for a checkpoint or archival of that log. In these cases, there is usually a combination of problems with redo log configuration and database tuning. Oracle Corporation recommends that you solve the tuning problem first but, if that fails to correct the problem, add more log groups or increase log sizes.

Oracle Corporation generally recommends:

- Multiplexing of online redo logs and a minimum of four log groups
- Hardware mirroring and disk striping across as many spindles as possible with approximately a 16-kilobyte stripe size or smaller of the online redo logs, especially for OLTP and batch applications

See Also: *Oracle9i Database Administrator's Guide* for more details about configuring online redo logs and online redo log groups

3.4.2 Setting Up the Log Transport Services Environment

The log transport services component of the Data Guard environment is configured primarily through the setting of database initialization parameters and options. You set parameters by editing the database initialization parameter file or using SQL statements.

See Also: *Oracle9i Database Reference* for details about creating and editing database initialization parameter files

3.4.2.1 Setting Permission to Archive Redo Logs

Permission for the archiving of online redo logs to remote destinations is specified using the REMOTE_ARCHIVE_ENABLE initialization file parameter. This parameter should be set to `true` on both the primary and standby databases in a Data Guard environment. Table 3–5 identifies the options of the REMOTE_ARCHIVE_ENABLE initialization parameter.

Table 3–5 REMOTE_ARCHIVE_ENABLE Initialization Parameter Settings

Primary Database Value	Description
`true`	The primary database is permitted to automatically archive online redo logs to remote archiving destinations, and the standby database is allowed to receive redo logs from the primary database. This is the default setting.
`false`	The primary database is not permitted to automatically archive online redo logs to remote destinations. This setting is used in manual recovery environments.

3.4.2.2 Specifying Archive Destinations for Redo Logs

In addition to setting up the primary database to run in ARCHIVELOG mode, you must configure the primary database to archive online redo logs by setting destinations and associated states. You do this using the LOG_ARCHIVE_DEST_*n* initialization parameters and corresponding LOG_ARCHIVE_DEST_STATE_*n* parameters as discussed in Section 3.3.1.

This section describes the creation of a standby database named `standby1` on a remote node named `stbyhost` based on the following assumptions:

- `prmyinit.ora` is the initialization parameter file for the primary database.
- You have backed up the primary database files.
- You have created the standby database control file.
- You have transferred the datafiles and control file to the standby site (`stbyhost`).
- You have copied the primary database initialization parameter file to `stbyhost`.

- You have used the Oracle Net Manager to configure the **listener** on `stbyhost` to use the TCP/IP protocol, and to statically register the `standby1` database service using its `SID`, so that the `standby1` standby database can be managed by the Data Guard broker.
- You have used the Oracle Net Manager to create the net service name `standby1` that can be resolved to a connect descriptor that contains the same protocol, host address, port, and `SID` that were used to statically register the `standby1` standby database with the listener on `stbyhost`.
- You have used the Listener Control utility to start the listener on `stbyhost`.

> **See Also:** *Oracle Net Services Administrator's Guide* for details about using Oracle Net networking components

To set up log transport services to archive redo logs to the standby database, make the following modifications to the primary database initialization parameter file. These modifications will take effect after the instance is restarted:

1. Specify the archive destination by adding the following entry to the `prmyinit.ora` file:

   ```
   LOG_ARCHIVE_DEST_2 = 'SERVICE=standby1 MANDATORY REOPEN=60'
   ```

   > **See Also:** LOCATION and SERVICE in Chapter 8

2. Enable the archive destination state by adding the following entry to the `prmyinit.ora` file:

   ```
   LOG_ARCHIVE_DEST_STATE_2 = ENABLE
   ```

Parameter/Option	Meaning
`LOG_ARCHIVE_DEST_2`	Specifies a remote redo log archiving destination
`SERVICE`	Specifies the Oracle Net service name of the remote destination
`MANDATORY`	Indicates that archiving to this location must succeed to continue
`REOPEN`	Indicates how many seconds the archiving process waits before reattempting to archive after a failed attempt
`LOG_ARCHIVE_DEST_STATE_2`	Indicates the state of the `LOG_ARCHIVE_DEST_2` remote destination

Parameter/Option	Meaning
ENABLE	Indicates that the primary database can archive redo logs at this destination

To avoid having to restart the instance, you can issue the following statements to ensure that the initialization parameters you have set in this step take effect immediately:

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='SERVICE=standby1 MANDATORY REOPEN=60';
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;
```

3.4.2.3 Specifying a Reuse Policy for Online Redo Logs

You can specify a policy for the reuse of online redo logs using the OPTIONAL or MANDATORY attributes of the LOG_ARCHIVE_DEST_*n* initialization parameters. The archival of log files to an OPTIONAL destination can fail, and the online redo log will be overwritten. If the archival of redo log files to a MANDATORY destination fails, online redo log files will not be overwritten.

By default, one destination is mandatory even if all destinations are designated to be optional.

Example 3–13 shows how to set a mandatory local archiving destination and enable that destination.

Example 3–13 Setting a Mandatory Archiving Destination

```
LOG_ARCHIVE_DEST_3 = 'LOCATION=/arc_dest MANDATORY'
LOG_ARCHIVE_DEST_STATE_3 = ENABLE
```

See Also: MANDATORY and OPTIONAL in Chapter 8

3.4.2.4 Setting the Number of Destinations That Must Receive Logs

The LOG_ARCHIVE_MIN_SUCCEED_DEST=*n* parameter (where *n* is an integer from 1 to 10) specifies the number of destinations that must receive redo logs successfully before the log writer process can reuse the online redo logs. All MANDATORY destinations and non-standby OPTIONAL destinations contribute to satisfying the LOG_ARCHIVE_MIN_SUCCEED_DEST=*n* count. Example 3–14 shows how to specify that redo logs must be successfully archived to two destinations before the online redo logs can be reused.

Example 3–14 Specifying a Minimum Number of Successful Destinations

```
LOG_ARCHIVE_MIN_SUCCEED_DEST=2
```

See Also: *Oracle9i Database Administrator's Guide*

3.4.2.5 Specifying Archive Failure Policies

You use attributes of the LOG_ARCHIVE_DEST_*n* initialization parameters to specify what actions are to be taken when archiving to a destination fails. You can use LOG_ARCHIVE_DEST_*n* attributes to:

- Configure log transport services to archive again to a failed destination
- Control the number of archive destination failures allowed
- Configure alternate destinations

Use the REOPEN attribute of the LOG_ARCHIVE_DEST_*n* parameters to determine whether and when the archiver process or the log writer process attempts to archive redo logs again to a failed destination following an error.

The REOPEN=*seconds* attribute specifies the minimum number of seconds that must elapse following an error before the archiving process will try again to access a failed destination. The default value is 300 seconds. The value set for the REOPEN attribute applies to all errors, not just connection failures. You can turn off the option by specifying NOREOPEN.

You can use the REOPEN attribute in conjunction with the MAX_FAILURE attribute to limit the number of attempts that will be made to reestablish communication with a failed destination. Once the specified number of attempts has been exceeded, the destination is treated as if the NOREOPEN attribute had been specified.

If you specify REOPEN for an OPTIONAL destination, it is still possible for the Oracle database server to overwrite online redo logs even if there is an error. If you specify REOPEN for a MANDATORY destination, log transport services stalls the primary database when it cannot successfully archive redo logs. When this situation occurs, consider the following options:

- Change the destination by deferring the destination, specifying the destination as optional, or changing the service.
- Specify an alternate destination.
- Disable the destination.

When you use the REOPEN attribute, note that:

- The archiver or log writer process reopens a destination only when starting an archive operation from the beginning of the log file and never during a current operation. Archival always starts the log copy from the beginning.
- If a value was specified or the default value used for the REOPEN attribute, the archiving process checks whether the time of the recorded error plus the REOPEN interval is less than the current time. If it is, the archival operation to that destination is retried.

See Also: REOPEN and NOREOPEN in Chapter 8

You can control the number of times a destination will be retried after a log archiving failure by specifying a value for the MAX_FAILURE=*count* attribute of the LOG_ARCHIVE_DEST_*n* initialization parameters.

The MAX_FAILURE attribute specifies the maximum number of contiguous archival failures that will be allowed for the particular destination. This attribute is useful for archive destinations that you want to retry after a failure, but do not want to retry indefinitely. The REOPEN attribute is required when you use the MAX_FAILURE attribute. Example 3–15 shows how to set a retry time of 5 seconds and limit retries to 3 times.

Example 3–15 Setting a Retry Time and Limit

```
LOG_ARCHIVE_DEST_1='LOCATION=/arc_dest REOPEN=5 MAX_FAILURE=3'
```

See Also: MAX_FAILURE and NOMAX_FAILURE in Chapter 8

Using the ALTERNATE attribute of the LOG_ARCHIVE_DEST_*n* parameters, you can specify alternate archive destinations. An alternate archive destination can be used when the archiving of an online redo log to a standby site fails. If archiving fails and the NOREOPEN attribute has not been specified, or the MAX_FAILURE attribute threshold has been exceeded, log transport services will attempt to archive redo logs to the alternate destination on the next archival operation.

Use the NOALTERNATE attribute to prevent the original archive destination from automatically changing to an alternate archive destination when the original archive destination fails.

Example 3–16 shows how to set the initialization parameter file so that a single, mandatory, local destination will automatically fail over to a different destination if any error occurs.

Example 3–16 Specifying an Alternate Destination

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1 MANDATORY ALTERNATE=LOG_ARCHIVE_DEST_2'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
LOG_ARCHIVE_DEST_2='LOCATION=/disk2 MANDATORY'
LOG_ARCHIVE_DEST_STATE_2=ENABLE
LOG_ARCHIVE_DEST_STATE_2=ALTERNATE
```

If the LOG_ARCHIVE_DEST_1 destination fails, the archiving process will automatically switch to the LOG_ARCHIVE_DEST_2 destination.

See Also: ALTERNATE and NOALTERNATE in Chapter 8

3.4.2.6 Specifying Dependent Destinations

Archiving redo logs to a remote database can be defined as being dependent upon the success or failure of an archival operation for another destination. The dependent destination is known as the **child destination.** The destination on which the child depends is known as the **parent destination.**

Use the DEPENDENCY attribute of the LOG_ARCHIVE_DEST_*n* initialization parameters to define a child destination. This attribute indicates that this destination depends upon the successful completion of archival operations for the parent destination.

Specifying a destination dependency can be useful in the following situations:

- The standby database and the primary database are on the same node. Therefore, the archived redo logs are implicitly accessible to the standby database.
- Clustered file systems are used to provide remote standby databases with access to the primary database archived redo logs.
- Operating system-specific network file systems are used, providing remote standby databases with access to the primary database archived redo logs.
- Mirrored disk technology is used to provide transparent networking support across geographically remote distances.
- There are multiple standby databases on the same remote node, sharing access to common archived redo logs for staggered managed recovery.

In these situations, although a physical archival operation is not required, the standby database needs to know the location of the archived redo logs on the primary site. This allows the standby database to access the archived redo logs on the primary site when they become available for managed recovery. You must

specify an archiving destination as being dependent on the success or failure of another (parent) destination. This is known as a **destination dependency.**

> **See Also:** DEPENDENCY and NODEPENDENCY in Chapter 8

3.4.2.7 Registering the Location of Archived Redo Logs in the Standby Control File

The REGISTER attribute indicates the fully qualified filename of the archived redo log at the remote destination. This information is recorded in the standby database control file. The fully qualified filename is derived from the values entered in the STANDBY_ARCHIVE_DEST and LOG_ARCHIVE_FORMAT initialization parameters. This is the implicit default setting.

For a standby destination database, this archived redo log registry serves as the manifest for the managed standby recovery operation.

The optional REGISTER=*location_format* attribute is used to specify a filename format different from the format defined by the STANDBY_ARCHIVE_DEST and LOG_ARCHIVE_FORMAT primary database initialization parameters. This is the filename that will be recorded in the corresponding remote destination control file. If this attribute value is not specified, then the setting is the same as REGISTER.

If you are using managed recovery, use the REGISTER=*location_format* attribute to specify the fully qualified filename of the archived redo log on the remote site.

> **See Also:** REGISTER and NOREGISTER and REGISTER=location_format in Chapter 8

3.4.2.8 Specifying a Time Lag for the Application of Redo Logs

By default, in managed recovery mode, the standby database automatically applies redo logs when they arrive from the primary database. But in some cases, you may want to create a time lag between the archiving of a redo log at the primary site and the applying of the redo 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.

Use the DELAY=*minutes* attribute of the LOG_ARCHIVE_DEST_*n* initialization parameters to specify a time lag for applying redo logs at the standby site. The default setting for this attribute is NODELAY. If the DELAY attribute is set with no value specified, then the value for this attribute is 30 minutes. Example 3–17 shows how to set up a destination with a time delay of 4 hours.

Example 3–17 Specifying a Time Delay for Archiving Redo Logs

```
LOG_ARCHIVE_DEST_3='SERVICE=stby1 DELAY=240 REOPEN=300'
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

See Also:

- DELAY and NODELAY in Chapter 8
- Section 6.11, "Scenario 11: Standby Database with a Time Lag"

3.4.2.9 Specifying Quotas for Archive Destinations

You can specify the amount of physical storage on a disk device to be allocated for an archiving destination using the QUOTA_SIZE attribute of the LOG_ARCHIVE_DEST_*n* initialization parameters. The QUOTA_SIZE attribute indicates the maximum number of 512-byte blocks of physical storage on a disk device that may be consumed by a destination. The value is specified in 512-byte blocks even if the physical device uses a different block size. The optional suffix values K, M, and G represent thousand, million, and billion, respectively (the value "1K" means 1,000 512-byte blocks).

A destination can be designated as being able to occupy all or some portion of the physical disk represented by the destination. For example, in a Real Application Clusters environment, a physical archived redo log disk device may be shared by two or more separate nodes (through a clustered file system, such as is available with Sun Clusters). As there is no cross-instance initialization parameter knowledge, none of the Real Application Clusters nodes is aware that the archived redo log physical disk device is shared with other instances. This leads to substantial problems when the destination disk device becomes full; the error is not detected until every instance tries to archive to the already full device. This seriously affects database availability.

The QUOTA_SIZE value does not have to be the actual number of blocks on the disk device; the value represents the portion of the disk device that can be consumed by the archived redo log destination.

See Also: QUOTA_SIZE and NOQUOTA_SIZE in Chapter 8

3.5 Configuring Log Transport Services on the Standby Database

This section describes how to set up log transport services on the standby database in preparation for switchover. The following topics are presented:

- Configuring the Standby Initialization Parameter File

- Transferring the Initialization Parameter File to the Standby Database
- Setting Up the Initialization Parameter File

3.5.1 Configuring the Standby Initialization Parameter File

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 must differ. Differences in initialization parameters other than those described in Table 3–6 can cause performance degradation at a standby database and, in some cases, halt database operations completely. Change parameter values only when it is required for the functionality of the standby database or for filename conversions.

The initialization parameters in Table 3–6 play a key role in the configuration of the standby database. For a complete list of database initialization parameters specific to the Data Guard environment, see Chapter 7.

Table 3–6 Configuring Standby Database Initialization Parameters

Parameter	Guideline	For More Information
COMPATIBLE	Always set this parameter to the same value at the primary and standby databases. If the values differ, you may not be able to archive the redo logs from your primary database to the standby database.	See *Oracle9i Database Reference*
CONTROL_FILES	Always set this parameter to a different value from the CONTROL_FILES parameter in the primary database. The names of the control files for the standby database must exist at the standby site.	See *Oracle9i Database Reference*
DB_FILE_NAME_CONVERT	Set this parameter to distinguish standby database filenames from primary database filenames. Because the standby database control file is a copy of the primary database control file, you must convert the standby database filenames if the standby database is on the same site as the primary database or on a separate site with different path names.	See Section 4.6
DB_FILES	Specifies the maximum number of database files that can be open for this database.	See *Oracle9i Database Reference*
DB_NAME	Always set to the same value as the DB_NAME parameter value in the primary database.	See *Oracle9i Database Reference*

Table 3–6 (Cont.) Configuring Standby Database Initialization Parameters

Parameter	Guideline	For More Information
FAL_CLIENT	This parameter, which is used solely by a standby database in managed recovery mode, specifies the Oracle Net service name that the FAL server should use to connect to the standby database. This parameter is set on the standby site.	See Section 4.5
FAL_SERVER	This parameter, which is used solely by a standby database in managed recovery mode, specifies the Oracle Net service name that the standby database should use to connect to the FAL server. This parameter is set on the standby site.	See Section 4.5
LOCK_NAME_SPACE	This parameter specifies the name space that the distributed lock manager (DLM) uses to generate lock names. Set this value if the standby database has the same name on the same cluster as the primary database.	See *Oracle9i Database Reference*
LOG_ARCHIVE_DEST_1	The value set in this parameter is used by the standby database ARC*n* process as the storage location for standby redo logs.	See Section 3.3.1
LOG_ARCHIVE_FORMAT	The value of this parameter is used in conjunction with the STANDBY_ARCHIVE_DEST parameter value to generate standby database log filenames.	See Section 3.6.3.4
LOG_ARCHIVE_START	Always set this parameter to `true` at the primary and standby databases in the Data Guard environment. This enables automatic archiving of filled groups each time an instance is started.	See *Oracle9i Database Administrator's Guide*
LOG_ARCHIVE_TRACE	Optionally, set this parameter to an integer value to see the progression of the archiving of redo logs at the standby site. The Oracle database server writes an audit trail of the redo logs received from the primary database into a trace file. This parameter controls output generated by the archiver process (ARC*n*), and foreground processes on the primary database and the RFS and FAL server processes on the standby database.	See Section 4.9.5

Table 3–6 (Cont.) Configuring Standby Database Initialization Parameters

Parameter	Guideline	For More Information
LOG_FILE_NAME_CONVERT	Set this parameter to make your standby redo log filenames distinguishable from primary database redo log filenames. This 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 filename on the primary database to the log filename on the standby database. The file must exist and be writable on the standby database or the recovery process will halt with an error.	See Section 4.6
REMOTE_ARCHIVE_ENABLE	Always set this parameter to `true` at the primary and standby databases in the Data Guard environment. This allows the standby database to receive redo logs for archiving from the primary database.	See Section 3.4.2
STANDBY_ARCHIVE_DEST	Used solely by a standby database in managed recovery mode to determine the archive location of online redo logs received from the primary database. The RFS process uses this value along with the LOG_ARCHIVE_FORMAT setting to generate the fully qualified standby database log filenames, and stores the filenames in the standby control file. This data is used to perform database recovery.	See Section 3.6.3.4
STANDBY_FILE_MANAGEMENT	Using this parameter, the Oracle database server automates the creation and deletion of datafiles on the standby system. The STANDBY_FILE_MANAGEMENT initialization parameter can be set to one of two values: `auto` or `manual`. The `auto` value indicates that datafile creation and deletion will be automated on the standby system; the `manual` value (the default) indicates that datafile creation and deletion will not be automated.	See Section 4.6

3.5.2 Transferring the Initialization Parameter File to the Standby Database

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.

3.5.3 Setting Up the Initialization Parameter File

Oracle Corporation suggests that you maintain two database initialization parameter files at both the primary and standby databases. This will allow you to easily change the databases from the primary role to the standby role or from the standby role to the primary role.

3.5.3.1 Primary Database Initialization Parameter Files

Example 3–18 and Example 3–19 show sections of initialization parameter files that you could maintain on the primary database. These examples show only parameters specific to log transport services. For complete examples of database initialization files, see Chapter 6, "Data Guard Scenarios".

Example 3–18 shows log transport services initialization parameters for a typical primary database. These log transport services parameter settings are used when the primary database is operating in the primary role.

Note: `LOG_ARCHIVE_DEST` and `LOG_ARCHIVE_DUPLEX_DEST` parameters have been set to null values because they are obsolete, and they are not used in the Data Guard environment.

Example 3–18 Primary Database: Primary Role Initialization Parameters

```
DB_NAME=primary1
CONTROL_FILES=primary.ctl
COMPATIBLE=9.0.1.0.0
LOG_ARCHIVE_START=true
LOG_ARCHIVE_DEST=""
LOG_ARCHIVE_DUPLEX_DEST=""
```

```
LOG_ARCHIVE_DEST_1='LOCATION=/oracle/arc/ MANDATORY REOPEN=30'
LOG_ARCHIVE_DEST_2='SERVICE=standby MANDATORY REOPEN=15'
LOG_ARCHIVE_DEST_STATE_1=enable
LOG_ARCHIVE_DEST_STATE_2=enable
LOG_ARCHIVE_FORMAT=arc%t_%s.arc
REMOTE_ARCHIVE_ENABLE=true
.
.
.
```

Example 3–19 shows log transport services parameters for an initialization parameter file to be maintained on the primary database and used if the primary database's role is changed to the standby role.

Example 3–19 Primary Database: Standby Role Initialization Parameters

```
DB_NAME=primary1
CONTROL_FILES=primary.ctl
COMPATIBLE=9.0.1.0.0
LOG_ARCHIVE_START=true
LOCK_NAME_SPACE=primary1
FAL_SERVER=standby1
FAL_CLIENT=primary1
DB_FILE_NAME_CONVERT=('/standby','/primary')
LOG_FILE_NAME_CONVERT=('/standby','/primary')
STANDBY_ARCHIVE_DEST=/oracle/arc/
LOG_ARCHIVE_DEST_1='LOCATION=/oracle/arc/'
LOG_ARCHIVE_TRACE=127
LOG_ARCHIVE_FORMAT=arc%t_%s.arc
STANDBY_FILE_MANAGEMENT=auto
REMOTE_ARCHIVE_ENABLE=true
.
.
.
```

3.5.3.2 Standby Database Initialization Parameter Files

Example 3–20 and Example 3–21 show sections of initialization parameter files that you could maintain on the standby database. These examples show only parameters specific to log transport services. For complete examples of database initialization files, see Chapter 6, "Data Guard Scenarios".

Example 3–20 shows log transport services parameters for an initialization parameter file to be maintained on the standby database when it is operating in the standby role.

Example 3–20 Standby Database: Standby Role Initialization Parameters

```
DB_NAME=primary1
CONTROL_FILES=standby.ctl
COMPATIBLE=9.0.1.0.0
LOG_ARCHIVE_START=true
LOCK_NAME_SPACE=standby1
FAL_SERVER=primary1
FAL_CLIENT=standby1
DB_FILE_NAME_CONVERT=("/primary","/standby")
LOG_FILE_NAME_CONVERT=("/primary","/standby")
STANDBY_ARCHIVE_DEST=/oracle/stby/arc
LOG_ARCHIVE_DEST_1='LOCATION=/oracle/stby/arc/'
LOG_ARCHIVE_TRACE=127
LOG_ARCHIVE_FORMAT=arc%t_%s.arc
STANDBY_FILE_MANAGEMENT=auto
REMOTE_ARCHIVE_ENABLE=true
.
.
.
```

Example 3–21 shows transport services parameters for an initialization parameter file to be maintained on the standby database and used when it is to operate in the primary role.

Example 3–21 Standby Database: Primary Role Initialization Parameters

```
DB_NAME=primary1
CONTROL_FILES=standby.ctl
COMPATIBLE=9.0.1.0.0
LOG_ARCHIVE_START=true
LOG_ARCHIVE_DEST=""
LOG_ARCHIVE_DUPLEX_DEST=""
LOG_ARCHIVE_DEST_1='LOCATION=/oracle/stby/arc/ MANDATORY REOPEN=30'
LOG_ARCHIVE_DEST_2='SERVICE=primary1 MANDATORY REOPEN=15'
LOG_ARCHIVE_DEST_STATE_1=enable
LOG_ARCHIVE_DEST_STATE_2=enable
LOG_ARCHIVE_FORMAT=arc%t_%s.arc
REMOTE_ARCHIVE_ENABLE=true
.
.
.
```

3.6 Log Transport Services Data Availability Modes

This section includes the following topics:

3.6.1 Introduction to Database Synchronization Options

The primary and standby databases are synchronized by applying redo logs from the primary database to the standby database. Although the goal is to keep the databases identical, the transactions applied to the standby database can sometimes lag behind the primary database.

This lag may occur either because the data has not yet reached the standby site, or it may have reached the standby site, but has not yet been applied to the standby database.

If the data needed to keep the databases synchronized is not yet available on the standby site, and you must fail over, the contents of the databases will diverge, and some data will be lost. To protect against potential data loss, use Data Guard features such as log data availability modes and standby redo logs.

Varying degrees of potential data loss can be configured, from no data loss to minimal data loss; each degree of potential data loss has a varying effect on primary database performance. Choose the degree of potential data loss that suits the application requirements.

3.6.1.1 No-Data-Loss Overview

The definition of **no data loss** is deceptively simple and very subtle; log transport services will not acknowledge application modifications to the primary database until the modifications are also available on a standby database. No data loss does not imply that the data modifications have been applied to the standby database, but that the data is available for log apply services to apply to the standby database.

No data loss is achieved by using standby redo logs on the standby database and setting an appropriate data availability mode.

3.6.1.2 No Data Divergence Overview

The definition of data divergence extends the concept of no data loss. **Data divergence** occurs when data modifications occur on the primary database when connectivity to the standby database is not available.

No data divergence prohibits primary database modifications when connectivity to at least one standby database is not available. In other words, the data on the primary database is protected against both loss *and* data divergence.

To prevent data divergence, Data Guard can be configured to automatically shut down the primary database instance when network connectivity to the last standby database is lost.

Note: Not all customers require this level of application data protection. The consequences of instance shutdown should be fully considered when implementing a no-data-divergence solution. Oracle Corporation highly recommends the use of multiple standby databases when you implement a no-data-divergence solution.

3.6.1.3 Minimal Data Loss Overview

In some cases, the performance of the primary database is more important than the potential loss of some data. Minimal data loss and high database performance can be achieved using the `ASYNC` and `NOAFFIRM` attributes for an archive destination. For example, using a lower block count for the `ASYNC` attribute combined with the `NOAFFIRM` attribute minimizes the amount of potential data loss and increases performance.

3.6.2 Choosing the Appropriate Data Availability Mode

Weigh your business requirements for data availability against user demands for response time and performance, to determine the appropriate data availability mode to use.

The following data availability modes can be configured in the Data Guard environment:

- Guaranteed protection

  With **guaranteed protection,** the standby database cannot diverge from the primary database at all, and no data can be lost. If a standby database is unavailable, processing automatically halts on the primary database as well.

  When operating in this mode, the log writer process transmits redo records from the primary database to the standby database, and a transaction is not committed on the primary database until it has been confirmed that the transaction data is available on at least one standby database. This has the potential to affect primary database performance, but provides the highest

degree of data availability at the standby site. Stock exchanges, currency exchanges, and financial institutions are examples of businesses that require guaranteed protection.

- Instant protection

  With **instant protection,** the standby database may temporarily diverge from the primary database, but upon failover to the standby database, the databases can be synchronized, and no data will be lost. As with guaranteed protection, the log writer process transmits redo logs from the primary database to the standby database. The transaction is not committed on the primary database until it has been confirmed that the transaction data is either available on the standby database, or that the data could not be received by the standby database. An example of a business that can use instant protection is a manufacturing plant; the risks of having no standby database for a period of time and data divergence are acceptable as long as no data is lost if failover is necessary.

- Rapid protection

  With **rapid protection,** the log writer process transmits redo logs to the standby sites. Use this mode when availability and performance on the primary database are more important than the risk of losing a small amount of data. During normal operations, processing on the primary database continues without regard to the data availability on any standby database. Rapid protection can be used if your business has some other form of backup or the loss of data would have a small impact on your business model.

  See Also: Appendix C, "Log Writer Asynchronous Network I/O"

- Delayed protection

  With **delayed protection,** the archiver process transmits the redo logs to the standby sites. This lowest level of protection has been available in releases prior to Oracle9*i*. In a failover, it is possible to lose data from one or more logs that have not yet been transmitted.

3.6.3 Configuring Log Transport Services Data Availability Modes

Log transport services provide four data availability modes, as explained in Section 3.6.2. You configure data availability modes by:

- Specifying a Redo Log Writing Process
- Specifying a Network Transmission Mode

- Specifying a Method of Writing Archived Logs to Disk
- Specifying a Redo Log Reception Option
- Setting a Failure Resolution Policy

Table 3–7 lists the availability modes and the settings required to implement each mode. You can set other variations of archive destination attributes and log transport services options; however, the following table identifies the most common settings.

Table 3–7 Configuring Log Transport Services Availability Modes

Data Availability Mode	Log Writing Process Option	Network Transmission Mode	Disk Write Option	Redo Log Reception Option	Failure Resolution Policy Option
Guaranteed protection	LGWR	SYNC	AFFIRM	Standby redo log	PROTECTED
Instant protection	LGWR	SYNC	AFFIRM	Standby redo log	UNPROTECTED
Rapid protection	LGWR	ASYNC	NOAFFIRM	Standby redo log	UNPROTECTED
Delayed protection	ARCH	SYNC	NOAFFIRM	Archived redo log	UNPROTECTED

3.6.3.1 Specifying a Redo Log Writing Process

Timely protection of application data requires use of the log writer process to propagate primary database modifications to one or more standby databases. This is achieved using the LGWR attribute of the LOG_ARCHIVE_DEST_*n* initialization parameters.

Attribute	Example	Default
{LGWR|ARCH}	LOG_ARCHIVE_DEST_3='SERVICE=stby1 LGWR'	ARCH

Choosing the LGWR attribute indicates that the log writer process (LGWR) will concurrently create the archived redo logs as the online redo log is populated. Depending on the configuration, this may require the log writer process to also transmit redo log files to remote archival destinations.

Choosing the ARCH attribute indicates that the archiver process (ARC*n*) will create archived redo logs on the primary database and also transmit redo logs for archival at specified destinations. This is the default setting.

The LGWR and ARCH attributes are mutually exclusive. Therefore, you cannot specify the two attributes for the same destination.

The LGWR attribute can be specified for individual destinations. This allows you to specify that the log writer process writes to redo logs and archives for some destinations while the archiver process archives redo logs to other destinations.

See Also: ARCH and LGWR in Chapter 8

It is possible to specify that the log writer process archive redo logs to all destinations; however, Oracle Corporation recommends that you use the log writer process for only one remote destination, due to the increased network I/O effect on the primary database.

Note: The log writer process will transmit log buffers to the archival destination only if the primary database is in ARCHIVELOG mode.

3.6.3.2 Specifying a Network Transmission Mode

When using the log writer process to archive redo logs, the DBA can specify synchronous (SYNC) or asynchronous (ASYNC) network transmission of redo logs to archiving destinations using the SYNC or ASYNC=*blocks* attributes.

The SYNC attribute indicates that all network I/O operations are to be performed synchronously, in conjunction with each write operation to the online redo log. Control is not returned to the executing application or user until the redo information is received by the standby site. This attribute has the potential to affect primary database performance adversely, but provides the highest degree of data availability at the destination site. Synchronous transmission is required for no-data-loss environments.

The ASYNC=*blocks* keyword indicates that all network I/O operations are performed asynchronously, and control is returned to the executing application or user immediately. You can specify a block count to determine the size of the SGA network buffer to be used. Block counts from 0 to 20,480 are allowed. The attribute allows the optional suffix value K to represent 1,000 (the value "1K" indicates 1,000 512-byte blocks). In general, for slower network connections, use larger block counts.

See Also: Appendix C, "Log Writer Asynchronous Network I/O"

When you use the ASYNC attribute, there are several events that cause the network I/O to be initiated:

- If the LGWR request exceeds the currently available buffer space, the existing buffer is transmitted to the standby database. The LGWR process stalls until sufficient buffer space can be reclaimed, but this should seldom occur.
- A primary database log switch forces any buffered redo data to be transmitted to the standby database.
- The primary database is shut down normally. An immediate shutdown of the primary database results in the buffered redo data being discarded. A standby database shutdown also causes the buffered redo data to be discarded.
- The primary database has no redo activity for a period of time. The duration of database inactivity cannot be user-defined.
- If the rate of redo generation exceeds the runtime network latency, then the LGWR request will be buffered if sufficient space is available. Otherwise, the existing buffer is transmitted to the standby database. The LGWR process stalls until sufficient buffer space can be reclaimed, but this should seldom occur.

Attribute	Example	Default
{SYNC|ASYNC[=*blocks*]}	LOG_ARCHIVE_DEST_3= 'SERVICE=stby1 LGWR SYNC'	SYNC

See Also: SYNC and ASYNC in Chapter 8

Table 3–8 identifies the attributes of the LOG_ARCHIVE_DEST_*n* initialization parameters that are used to specify the transmission mode.

Table 3–8 Transmission Mode Attributes

Process Attribute	Method Attribute	Description
LGWR	SYNC	The primary database log writer process will synchronously transmit the online redo log contents. Control will not be returned to the application until the data safely resides on the standby site. This mode has the best guarantee of data availability on the destination database, but the highest performance effect on the primary database.
LGWR	ASYNC=*blocks*	The primary database log writer process will asynchronously transmit the online redo log contents to the destination database. Control will be returned to the application processes immediately, even if the data has not reached the destination. Block counts up to 2048 blocks are allowed. This mode has a reasonable degree of data availability on the destination database, but much decreased performance effect on the primary database.
ARCH	SYNC	The primary database archiver process will synchronously transmit the online redo log contents. This mode has the lowest degree of data availability on the destination database, and a slightly decreased performance effect on the primary database.

3.6.3.3 Specifying a Method of Writing Archived Logs to Disk

The [NO]AFFIRM attribute of the LOG_ARCHIVE_DEST_*n* parameters is used to specify whether log archiving disk write I/O operations are to be performed synchronously or asynchronously. By default, disk write operations are performed asynchronously.

It is necessary for the primary database to receive acknowledgement of the availability of the modifications on the standby database in a no-data-loss environment. This is achieved using the SYNC and AFFIRM attributes of the LOG_ARCHIVE_DEST_*n* initialization parameters. The SYNC attribute indicates that synchronous network transmission is necessary, and the AFFIRM attribute indicates that synchronous archived redo log disk write I/O operations are necessary.

Together, these attributes ensure that primary database modifications are available on the standby database.

This attribute applies to local and remote archive destination disk I/O operations and standby redo log disk write I/O operations.

Attribute	Example	Default
[NO]AFFIRM	LOG_ARCHIVE_DEST_3= 'SERVICE=stby1 LGWR SYNC AFFIRM'	NOAFFIRM

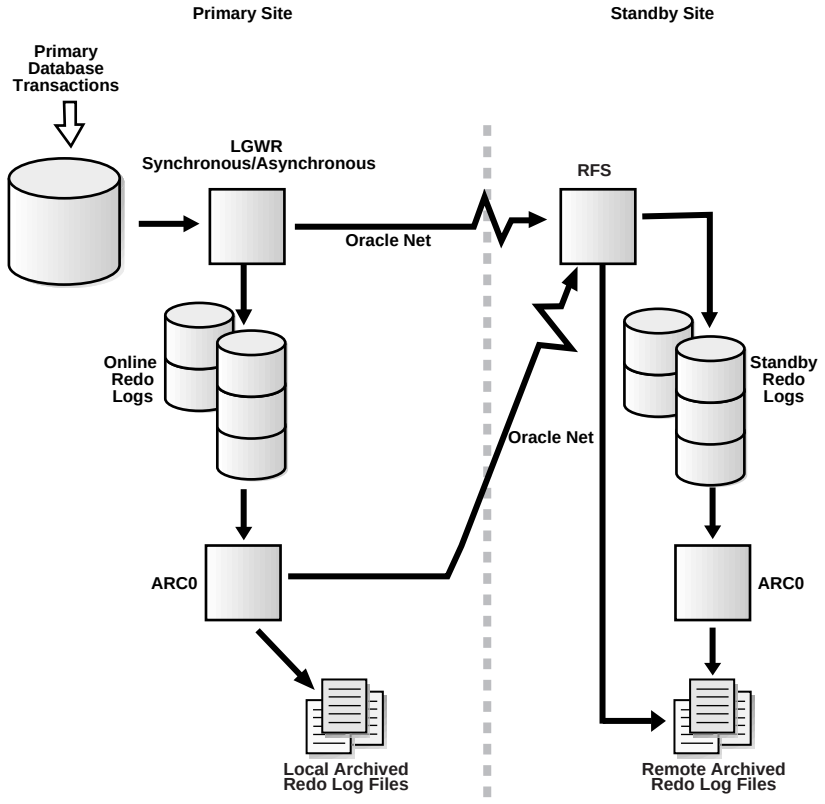
Note: The [NO]AFFIRM attribute has no effect on primary database online redo log disk I/O operations.

See Also: AFFIRM and NOAFFIRM in Chapter 8

3.6.3.4 Specifying a Redo Log Reception Option

Online redo logs transferred from the primary database are received by the remote file server process (RFS) on the standby site and can be stored as either standby redo logs or archived online redo logs as shown in Figure 3–2.

Figure 3–2 Redo Log Reception Options



Oracle9*i* Data Guard introduces the concept of **standby redo logs.** Standby redo logs form a separate pool of log file groups.

Standby redo logs provide the following advantages over archived online redo logs:

- Because standby redo logs are preallocated files, they avoid the operating system overhead of file system metadata updates common with sequential files.
- Standby redo logs can reside on raw devices, thus providing I/O write performance equal to that of the primary database.
- Standby redo logs can be duplexed using multiple members, improving reliability over archived redo logs.

- If primary database failure occurs, redo data written to standby redo logs can be fully recovered.
- Standby redo logs are required to implement a no-data-loss disaster recovery solution.

Standby redo logs are created using the `ADD STANDBY LOGFILE` clause of the `ALTER DATABASE` statement. Additional log group members can be added later to provide another level of reliability against disk failure on the standby site.

Once you have created the standby redo logs, they are automatically selected as the repository of redo data received from the primary site, if all of the following conditions apply:

- The primary database archive redo log destination uses the log writer process (LGWR) to archive online redo logs.
- The archiver process (ARC*n*) is active on the standby database.
- The size of a standby redo log exactly matches at least one of the primary database online redo logs.
- The standby redo log has been successfully archived. In other words, the standby redo logs must be empty of primary database redo data.

For example, if the primary database uses two online redo log groups whose log size is 100K and 200K, respectively, then the standby database should have standby redo log groups with those same sizes. However, it may be necessary to create additional standby log groups on the standby database, so that the archival operation has time to complete before the standby redo log is used. If the primary database is operating in protected mode, and a standby redo log cannot be allocated, the primary database instance will be shut down immediately. Therefore, be sure you allocate an adequate number of standby redo logs.

When you use Real Application Clusters, the various standby redo logs are shared among the various primary database instances. Standby redo log groups are not dedicated to a particular primary database thread.

Standby redo logs must be archived before the data can be applied to the standby database. The standby archival operation occurs automatically, even if the standby database is not in ARCHIVELOG mode. However, the archiver process must be started on the standby database. Note that the use of the archiver process (ARC*n*) is a requirement for selection of a standby redo log.

Because of this additional archival operation, using standby redo logs may cause the standby database to lag further behind the primary database than when archiving directly from the primary database to standby destinations. However, the

use of standby redo logs ultimately improves redo data availability on the standby database.

Standby redo logs created on a primary database are not used until the database assumes the standby role. However, Oracle Corporation recommends that you create standby redo logs so that the primary database can switch roles easily and quickly without additional DBA intervention.

Choosing the Number of Standby Redo Log Groups

The best way to determine the appropriate number of standby redo log groups for a database instance is to test different configurations. The minimum configuration has the same number of groups as the primary database. The optimum configuration has slightly more groups than the primary database.

In some cases, a standby database instance may require only two groups. In other situations, a database may require additional groups to guarantee that a recycled group is always available to receive redo information from the primary database. During testing, the easiest way to determine if the current standby log configuration is satisfactory is to examine the contents of the RFS process trace file and the database alert log. If messages indicate that the RFS process frequently has to wait for a group because an archival has not completed, add more standby groups.

Consider the parameters that can limit the number of standby redo log groups before setting up or altering the configuration of the standby redo log groups. The following parameters limit the number of standby redo log groups that you can add to a database:

- The `MAXLOGFILES` parameter of the `CREATE DATABASE` statement for the primary database determines the maximum number of groups of standby redo logs per standby database. The only way to override this limit is to re-create the primary database or control file.
- The `LOG_FILES` initialization parameter can temporarily decrease the maximum number of groups of standby redo logs for the duration of the current instance.
- The `MAXLOGMEMBERS` parameter of the `CREATE DATABASE` statement used for the primary database determines the maximum number of members per group. The only way to override this limit is to re-create the primary database or control file.

See Also: *Oracle9i SQL Reference*

Creating Standby Redo Log Groups

Plan the standby redo log configuration of a database and create all required groups and members of groups after you have instantiated the standby database. However, there are cases where you might want to create additional groups or members. For example, adding groups to a standby redo log configuration can correct redo log group availability problems. To create new standby redo groups and members, you must have ALTER DATABASE system privilege. A database can have as many groups as the value of MAXLOGFILES.

To create a new group of standby redo logs, use the ALTER DATABASE statement with the ADD STANDBY LOGFILE clause.

The following statement adds a new group of standby redo logs to the database:

```
SQL> ALTER DATABASE ADD STANDBY LOGFILE
  2> ('/oracle/dbs/log1c.rdo','/oracle/dbs/log2c.rdo') SIZE 500K;
```

You can also specify a number that identifies the group using the GROUP option:

```
SQL> ALTER DATABASE ADD STANDBY LOGFILE GROUP 10
  2> ('/oracle/dbs/log1c.rdo','/oracle/dbs/log2c.rdo') SIZE 500K;
```

Using group numbers can make administering standby redo log groups easier. However, the group number must be between 1 and the value of the MAXLOGFILES parameter. Do not skip redo log file group numbers (that is, do not number groups 10, 20, 30, and so on), or you will consume additional space in the standby database control file.

Adding Standby Redo Log Members to an Existing Group

In some cases, it might not be necessary to create a complete group of standby redo logs. A group could already exist, but not be complete because one or more members were dropped (for example, because of disk failure). In this case, you can add new members to an existing group.

Use fully qualified filenames of new log members to indicate where the file should be created. Otherwise, files will be created in either the default or current directory of the database server, depending upon your operating system.

To add new standby redo log group members, use the ALTER DATABASE statement with the ADD STANDBY LOGFILE MEMBER parameter. The following statement adds a new member to redo log group number 2:

```
SQL> ALTER DATABASE ADD STANDBY LOGFILE MEMBER '/oracle/dbs/log2b.rdo'
  2> TO GROUP 2;
```

> **See Also:** *Oracle9i SQL Reference* for a complete description of the ALTER DATABASE statement

Specifying Storage Locations for Archived Redo Logs and Standby Redo Logs

When archived redo logs are used, the STANDBY_ARCHIVE_DEST initialization parameter is used to specify the directory in which to store the archived redo logs. Log transport services use this value in conjunction with the LOG_ARCHIVE_FORMAT parameter to generate the archived redo log filenames on the standby site.

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

Log transport services store the fully qualified filenames in the standby control file. Log apply services use this information to perform recovery operations on the standby database. You can access this information through the V$ARCHIVED_LOG view on the standby database:

```
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
```

When standby redo logs are used, the LOG_ARCHIVE_DEST_1 initialization parameter on the standby database specifies the directory in which to store standby redo logs.

Parameter	Indicates	Example
LOG_ARCHIVE_DEST_1	The directory for storage of standby redo logs on the standby site	LOG_ARCHIVE_DEST_1= 'LOCATION=/oracle/stby/arc/'

Note: When using standby redo logs, you must enable the archiver process (ARC*n*). Oracle Corporation recommends that you always set the LOG_ARCHIVE_START initialization parameter to true on the standby database.

3.6.3.5 Setting a Failure Resolution Policy

A failure resolution policy determines what actions will occur on a primary database when the last standby destination fails to archive redo logs.

To set the highest level of data protection, guaranteed protection, place the primary database in PROTECTED mode using the SET STANDBY DATABASE clause of the ALTER DATABASE statement as shown in the following example:

```
SQL> ALTER DATABASE SET STANDBY DATABASE PROTECTED;
```

Note: This statement must be issued on the primary database.

When this statement is used, the primary database is protected against data loss and divergence. If connectivity between the primary and standby database should now be lost, the primary database will shut down. When using this level of data protection, standby databases must have two or more standby redo log groups. Also, one or more primary database archive redo log destinations must have LGWR and SYNC attributes specified. The functionality of the AFFIRM attribute is implicitly set.

You can revert to a mode that allows data divergence by placing the primary database in UNPROTECTED mode using the following statement:

```
SQL> ALTER DATABASE SET STANDBY DATABASE UNPROTECTED;
```

See Also: *Oracle9i SQL Reference* for complete ALTER DATABASE syntax

3.6.4 Comparing Network and Disk I/O Methods

The difference between the functionality of the destination disk write I/O attribute [NO]AFFIRM and the SYNC and ASYNC network transmission attributes is that the SYNC and ASYNC attributes apply only to network I/O performance when the log writer process is used. Table 3–9 shows a comparison of primary database performance to data availability on the standby database when various combinations of archive process, network I/O, and disk I/O attribute settings are used.

Table 3–9 Comparing Network and Disk I/O Methods

Archive Process Attribute Setting	Network I/O Attribute Setting	Disk I/O Attribute Setting	Primary Database Performance	Standby Database Data Availability (1=lowest availability)	Description
LGWR	ASYNC	NOAFFIRM	High	3	Redo logs are archived asynchronously on the network as well as to disk, providing high primary database performance but lower data availability on the standby database.
LGWR	ASYNC	AFFIRM	High	4	Redo logs are archived asynchronously on the network but synchronously to disk, providing high primary database performance and slightly higher data availability on the standby database than when the NOAFFIRM attribute is used.
LGWR	SYNC	NOAFFIRM	Fair	5	Redo logs are archived synchronously on the network but asynchronously to disk, providing fair primary database performance and higher data availability on the standby database.

Table 3–9 (Cont.) Comparing Network and Disk I/O Methods

Archive Process Attribute Setting	Network I/O Attribute Setting	Disk I/O Attribute Setting	Primary Database Performance	Standby Database Data Availability (1=lowest availability)	Description
LGWR	SYNC	AFFIRM	Low	6	Redo logs are archived synchronously on the network and synchronously to disk, providing low primary database performance but the highest data availability on the standby database.
ARCH	SYNC	NOAFFIRM	Fair	1	Redo logs are archived synchronously on the network and asynchronously to disk, providing fair primary database performance and the lowest data availability on the standby database.
ARCH	SYNC	AFFIRM	Low	2	Redo logs are archived synchronously on the network and to disk, providing low primary database performance and low data availability on the standby database.

3.7 Network Tuning for Log Transport Services

The process of archiving redo logs involves reading a buffer from the redo log and writing it to the archive log location. When the destination is remote, the buffer is written to the archive log location over the network using Oracle Net services.

The default archive log buffer size is 1 megabyte. The default transfer buffer size for Oracle Net is 2 kilobytes. Therefore, the archive log buffer is divided into units of approximately 2 kilobytes for transmission. These units could get further divided depending on the maximum transmission unit (MTU) of the underlying network interface.

The Oracle Net parameter that controls the transport size is session data unit (SDU). This parameter can be adjusted to reduce the number of network packets that are transmitted. This parameter allows a range of 512 bytes to 32 kilobytes.

For optimal performance, set the Oracle Net SDU parameter to 32 kilobytes for the associated SERVICE destination parameter.

The following example shows a database initialization parameter file segment that defines a remote destination netserv:

```
LOG_ARCHIVE_DEST_3='SERVICE=netserv'
SERVICE_NAMES=srvc
```

The following example shows the definition of that service name in the tnsnames.ora file:

```
netserv=(DESCRIPTION=(SDU=32768)(ADDRESS=(PROTOCOL=tcp)(HOST=host) (PORT=1521))
(CONNECT_DATA=(SERVICE_NAME=srvc)(ORACLE_HOME=/oracle)))
```

The following example shows the definition in the listener.ora file:

```
LISTENER=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=tcp)
(HOST=host)(PORT=1521))))

SID_LIST_LISTENER=(SID_LIST=(SID_DESC=(SDU=32768)(SID_NAME=sid)
(GLOBALDBNAME=srvc)(ORACLE_HOME=/oracle)))
```

If you archive to a remote site using high-latency / high-bandwidth connections, you can improve performance by increasing the TCP send and receive window sizes. Use caution, however, because this may adversely affect networked applications that do not exhibit the same characteristics as archiving. This method consumes a large amount of system resources.

See Also: *Oracle Net Services Administrator's Guide*

3.8 Log Transport Services Monitoring

This section describes manual methods of monitoring redo log archival activity for the primary database.

The Oracle9*i* Data Guard Manager graphical user interface automates many of the tasks involved in monitoring a Data Guard environment. See *Oracle9i Data Guard Broker* and the Data Guard Manager online help for more information.

3.8.1 Gathering Redo Log Archival Information

Enter the following query on the primary database to determine the current redo log sequence numbers:

```
SQL> SELECT thread#, sequence#, archived, status FROM v$log;

THREAD#   SEQUENCE#  ARC  STATUS
--------  ---------  ---  ------
       1        947  YES   ACTIVE
       1        948  NO    CURRENT
```

Enter the following query at the primary database to determine the most recently archived redo log file:

```
SQL> SELECT MAX(sequence#) FROM v$archived_log;

MAX(SEQUENCE#)
--------------
           947
```

Enter the following query at the primary database to determine the most recently archived redo log file to each of the archive destinations:

```
SQL> SELECT destination, status, archived_thread#, archived_seq#
  2> FROM v$archive_dest_status
  3> WHERE status <> 'DEFERRED' AND status <> 'INACTIVE';

DESTINATION         STATUS  ARCHIVED_THREAD#  ARCHIVED_SEQ#
------------------  ------  ----------------  -------------
/private1/prmy/lad   VALID                 1            947
standby1             VALID                 1            947
```

The most recently archived redo log file should be the same for each archive destination listed. If it is not, a status other than VALID may identify an error encountered during the archival operation to that destination.

You can issue a query at the primary database to find out if a log has not been sent to a particular site. Each archive destination has an ID number associated with it. You can query the DEST_ID column of the V$ARCHIVE_DEST fixed view on the primary database to identify archive destination IDs.

Assume the current local archive destination is 1, and one of the remote standby archive destination IDs is 2. To identify which logs have not been received by this standby destination, issue the following query:

```
SQL> SELECT local.thread#, local.sequence# FROM
  2> (SELECT thread#, sequence# FROM v$archived_log WHERE dest_id=1)
  3> local WHERE
  4> local.sequence# NOT IN
  5> (SELECT sequence# FROM v$archived_log WHERE dest_id=2 AND
  6> thread# = local.thread#);

  THREAD#  SEQUENCE#
---------  ---------
  1        12
  1        13
  1        14
```

See Also: Appendix A, "Troubleshooting the Standby Database" to learn more about monitoring the archiving status of the primary database

3.8.2 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. See Section 4.9.5 for complete details and examples.

4

Log Apply Services

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

4.1 Introduction to Log Apply Services

In a Data Guard environment, **log apply services** maintain the standby database in either a managed recovery mode or an open read-only mode. Log apply services automatically apply archived redo logs to maintain transactional synchronization with the primary database, and allow transactionally consistent read-only access to the data.

In a Data Guard environment, the log apply services component coordinates its activities with the log transport services component.

Using log apply services, you can manage the standby database in one of the following modes:

- Managed Recovery Mode

  In this setup, log transport services archive logs to the standby site, and log apply services automatically apply these logs. If you want maximum protection against data loss or corruption, then maintain the standby database in managed recovery mode in a Data Guard environment.

- Read-Only Mode

  Use read-only mode for supplemental reporting of data contained in the primary database. If you want to use the standby database for reporting purposes, then open it in read-only mode in a Data Guard environment. Log apply services 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 change between managed recovery mode and read-only mode. In most implementations of a Data Guard environment, you may want to make this change 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

The following sections in this chapter describe the procedures for initiating the various modes as well as for performing failover to a standby database.

4.2 Process Architecture

The physical standby component uses several processes to achieve the automation necessary for disaster recovery and high availability. On the standby database, log apply services use the following processes:

- Remote file server (RFS)

  The **remote file server (RFS)** process receives archived redo logs from the primary database.

- Archiver (ARC*n*)

  The ARC*n* process archives the standby redo logs to be applied by the managed recovery process (MRP).

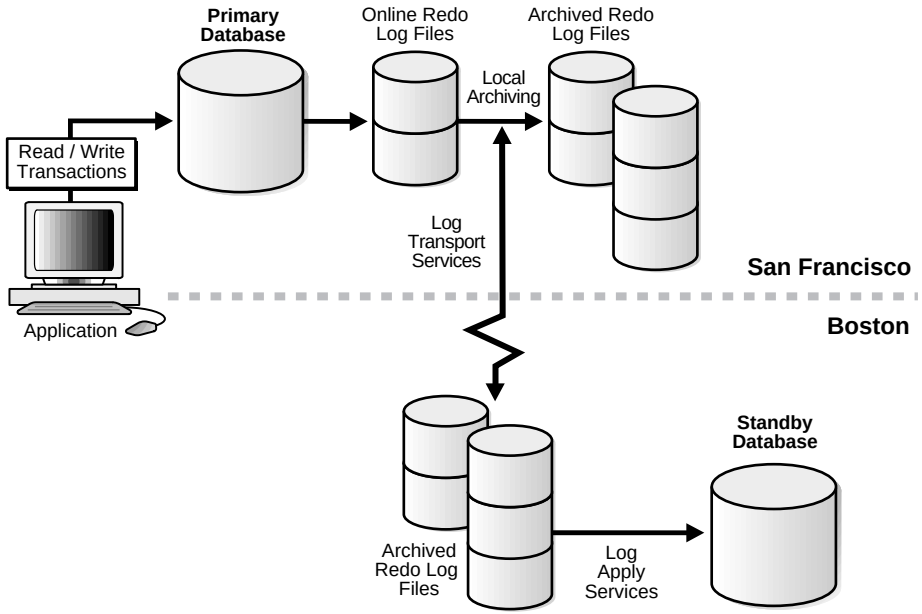
- Managed recovery process (MRP)

The MRP applies archived redo log information to the standby database.

4.3 Managed Recovery Mode

Log transport services automate archiving to a standby database. Log apply services keep 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 4–1.

Figure 4–1 Automatic Updating of a Standby Database



This section contains the following topics:

- Starting the Standby Instance in Preparation for Recovery
- Initiating Log Apply Services
- Monitoring the Recovery Process

4.3.1 Starting the Standby Instance in Preparation for Recovery

After all necessary parameter and network files have been configured, you can start the standby instance. If the instance is not started and mounted, the standby database cannot receive archived redo logs that are automatically copied to the standby site from the primary database by log transport services.

To start the standby instance:

1. Connect to the standby database instance. For example, enter:

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

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

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

   Note: Starting the database requires a NOMOUNT qualifier.

3. Mount the standby database:

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

4.3.2 Initiating Log Apply Services

Log apply services can run as a foreground session or as a background process with control options such as CANCEL, FINISH, TIMEOUT, DELAY, DISCONNECT, and PARALLEL.

You can enable managed recovery using log apply services. The following is an example of a foreground session:

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

As log transport services archive redo logs to the standby site, log apply services can automatically apply them to the standby database.

Note: After you execute the RECOVER statement, the prompt sits on the line following the RECOVER statement (unless DISCONNECT is specified); this is the expected foreground behavior.

If you want to start a detached server process and immediately return control to the user, add the DISCONNECT FROM SESSION option to the ALTER DATABASE statement. Note that this does not disconnect the current SQL session. For example:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT FROM SESSION;
```

See Also: Section 4.4 and Table 9–1

4.3.3 Monitoring the Recovery Process

To verify that you have correctly initiated log apply services, query the V$MANAGED_STANDBY fixed view on the standby database. This view monitors the progress of a managed recovery. For example:

```
SQL> SELECT process, status, thread#, sequence#, block#, blocks
  2> FROM v$managed_standby;

PROCESS STATUS       THREAD#    SEQUENCE#  BLOCK#     BLOCKS
------- ------------ ---------- ---------- ---------- ----------
MRP0    APPLYING_LOG 1          946        10         1001
```

If you did not start a detached server process, you need to execute this query from another SQL session.

To monitor activity on the standby database, query the V$ARCHIVE_DEST_STATUS fixed view.

See Also: Section 4.9

4.4 Controlling Managed Recovery Mode

To control the managed recovery operation, use the TIMEOUT, CANCEL, NODELAY, DELAY, NEXT, PARALLEL, FINISH, EXPIRE, and DISCONNECT control options.

This section contains the following topics:

- CANCEL Control Option
- DELAY Control Option
- DISCONNECT Control Option
- EXPIRE Control Option
- FINISH Control Option
- NEXT Control Option

- NODELAY Control Option
- PARALLEL Control Option
- TIMEOUT Control Option

4.4.1 CANCEL Control Option

Cancel the managed recovery operation at any time by issuing the CANCEL option of the ALTER DATABASE RECOVER MANAGED STANDBY DATABASE statement.

Format

```
CANCEL
CANCEL IMMEDIATE
CANCEL NOWAIT
```

The CANCEL option directs log apply services to stop the managed recovery operation after completely processing the current archived redo log. The managed recovery operation is terminated on an archived log boundary.

The CANCEL IMMEDIATE option, however, directs log apply services to stop the managed recovery operation either before reading another block from the archived redo log or before opening the next archived redo log, whichever occurs first. The managed recovery operation is terminated on an I/O boundary or on an archived log boundary, whichever occurs first. Stopping the recovery on an I/O boundary will leave the database in an inconsistent state and, therefore, unable to be opened. Note the following scenarios:

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

By default, the CANCEL option waits for the managed recovery operation to terminate before control is returned. If you specify the NOWAIT option, control is returned to the process that issued the CANCEL option without waiting for the managed recovery operation to terminate.

4.4.2 DELAY Control Option

Use the DELAY control option to specify an absolute apply delay interval to the managed recovery operation. The managed recovery operation waits the specified number of minutes before applying the archived redo logs. The apply delay interval begins once the archived redo logs have been selected for recovery.

The DELAY control option apply delay interval supersedes any apply delay interval specified for the standby database by the primary database's corresponding LOG_ARCHIVE_DEST_*n* initialization parameter. The DELAY control option pertains to archived redo logs being applied by this managed recovery operation.

You can use the DELAY control option when starting a managed recovery operation or to alter the mode of an active managed recovery operation. A DELAY control option value of zero (0) directs the managed recovery operation to revert to default behavior.

Format

DELAY *minutes*

In the following example, log apply services specify an absolute apply delay interval of 30 minutes to a foreground managed recovery operation:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DELAY 30;
```

4.4.3 DISCONNECT Control Option

Use the DISCONNECT control option to start managed recovery in background mode.

Format

```
DISCONNECT
DISCONNECT [FROM SESSION]
```

Using this option creates a **managed recovery process (MRP)** to perform the recovery in the background while the foreground process that issued the RECOVER statement continues performing other tasks. The optional FROM SESSION keywords can be added for clarity but do not change the behavior of the DISCONNECT option.

4.4.4 EXPIRE Control Option

Use the EXPIRE control option to specify the number of minutes after which the managed recovery operation automatically terminates, relative to the current time. The managed recovery operation terminates at the end of the current archived redo log that is being processed. This means that the value of the EXPIRE control option is the earliest amount of time that the managed recovery operation will terminate, but it could be significantly later than the specified value.

The managed recovery operation expiration is always relative to the time the statement was issued, not when the managed recovery operation was started. To cancel an existing managed recovery operation, use the CANCEL control option.

Specifying a new EXPIRE control option overrides any previously specified value. Use the value 0 to cancel a previously specified expiration value.

When you use a detached managed recovery process, the MRP makes an alert log entry when it terminates. A foreground managed recovery process simply returns control to the SQL prompt.

Format

EXPIRE *minutes*

In the following example, log apply services automatically terminates two hours from now:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE EXPIRE 240;
```

The EXPIRE control option cannot be specified in conjunction with the CANCEL control option.

4.4.5 FINISH Control Option

Use the FINISH control option to complete managed recovery in preparation for a failover from the primary database to the standby database.

Format

```
FINISH
FINISH NOWAIT
```

A managed recovery directed by the FINISH option will first apply all available archived redo logs and then recover any available standby redo logs (where a

non-finish mode recovery only applies the archived redo logs). This extra step brings the standby database up-to-date with the last committed transaction on the primary database. You can use the FINISH option when starting a managed recovery or to alter the mode of an ongoing managed recovery. If you use the FINISH option to alter the mode of an ongoing managed recovery, use the NOWAIT option to allow control to be returned to the foreground process before the recovery completes.

See Also: Section 5.4.6.1

4.4.6 NEXT Control Option

Use the NEXT control option to direct the managed recovery operation to apply a specified number of archived redo logs as soon as possible after the log transport services have archived them. Any apply delay interval specified by the DELAY attribute of the LOG_ARCHIVE_DEST_*n* initialization parameter set on the primary database is ignored by the managed recovery operation until the specified number of archived redo logs has been applied. Then, the managed recovery operation reverts to the default behavior.

You can use the NEXT control option when starting a managed recovery operation or to alter the mode of an active managed recovery operation.

Format

NEXT *count*

In the following example, log apply services direct a foreground managed recovery operation to apply the next 5 archived redo logs without delay:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE NEXT 5;
```

4.4.7 NODELAY Control Option

Use the NODELAY control option to direct the managed recovery operation to apply the archived redo logs as soon as possible after log transport services have archived them. Any apply delay interval specified by the DELAY attribute of the LOG_ARCHIVE_DEST_*n* initialization parameters on the primary database is ignored by the managed recovery operation when the NODELAY control option is specified. By default, the managed recovery operation respects any apply delay interval specified for the standby database by the primary database's corresponding LOG_ARCHIVE_DEST_*n* initialization parameter.

Format

```
NODELAY
```

In the following example, log apply services direct a foreground managed recovery operation to apply archived redo logs without delay:

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

4.4.8 PARALLEL Control Option

When running in managed recovery mode, the log apply services apply the changes generated on the primary database by many concurrent processes. Therefore, applying the archived redo logs on the standby database can take longer than the time it took to initially generate the changes on the primary database. By default, the log apply services use a single process to apply all of the archived redo logs sequentially. When using the parallel recovery option, several processes are able to apply the archived redo logs simultaneously.

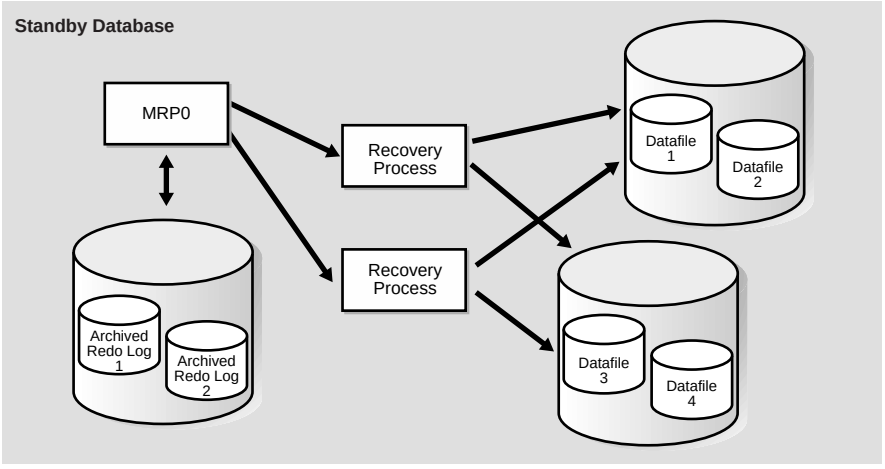
In general, using the parallel recovery option is most effective at reducing recovery time when several datafiles on several different disks are being recovered concurrently. The performance improvement from the parallel recovery option is also dependent upon whether the operating system supports asynchronous I/O. If asynchronous I/O is not supported, the parallel recovery option can dramatically reduce recovery time. If asynchronous I/O is supported, the recovery time may be only slightly reduced by using parallel recovery.

> **See Also:** Your operating system documentation to determine whether the system supports asynchronous I/O

In a typical parallel recovery situation, one process is responsible for reading and dispatching archived redo logs. This is the dedicated managed recovery server process (either the foreground SQL session or the background MRP0 process) that begins the recovery session. The managed recovery server process reading the archived redo logs enlists two or more recovery processes to apply the changes from the archived redo logs to the datafiles.

Figure 4–2 illustrates a typical parallel recovery session.

Figure 4–2 Parallel Recovery Session



Standby database recovery is a very disk-intensive activity (as opposed to a CPU-intensive activity). Therefore, the number of recovery processes needed is dependent entirely upon how many disk drives are involved in recovery. In most situations, one or two recovery processes per disk drive containing datafiles needing recovery are sufficient. In general, a minimum of eight recovery processes is needed before parallel recovery can show improvement over a serial recovery.

Format

PARALLEL *number*

In the following example, log apply services specify a parallel managed recovery operation utilizing 8 background child processes:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE PARALLEL 8;
```

4.4.9 TIMEOUT Control Option

Use the TIMEOUT control option to specify the number of minutes that log apply services wait for log transport services to complete the archival of the redo log required by the managed recovery operation. If the specified number of minutes passes without receiving the required archived redo log, the managed recovery operation is automatically terminated. By default, log apply services wait indefinitely for the next required archived redo log. The managed recovery

operation is terminated only through use of the CANCEL option, a CTRL+C key combination, or an instance shutdown.

Format

TIMEOUT *minutes*

In the following example, log apply services initiate a foreground managed recovery operation with a timeout interval of 15 minutes:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE TIMEOUT 15;
```

The TIMEOUT control option cannot be used in conjunction with the DISCONNECT control option.

Note: The TIMEOUT interval is not affected by the presence of an apply delay interval.

4.5 Archive Gap Management

An **archive gap** is a range of archived redo logs created whenever you are unable to apply the next archived redo log generated by the primary database to the standby database. For example, an archive gap occurs when the network goes down and automatic archival from the primary database to the standby database stops. 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 an archive gap for the time the network was down. The gap is automatically detected and resolved when managed recovery mode is enabled.

Setting Initialization Parameters to Automatically Resolve Archive Gaps

In previous versions, to be able to place the standby database in managed recovery mode, you would first manually apply logs in the archive gap to the standby database. After you had performed this manual recovery, you could then issue the RECOVER MANAGED STANDBY DATABASE statement, at which point log apply services would apply subsequent logs to the standby database automatically.

In Oracle9*i*, you can set initialization parameters so that log apply services automatically identify and resolve archive gaps as they occur. For log apply services to automatically identify and resolve archive gaps, you must:

1. Use the Oracle Net Manager to configure the listener on the standby site. Use the TCP/IP protocol and statically register the standby database service with the listener using the SID, so that the standby database can be managed by the Data Guard broker.
2. Use the Oracle Net Manager to create a net service name that the standby database can use to connect to the FAL server. The net service name should resolve to a connect descriptor that uses the same protocol, host address, port, and SID that you specified when you configured the listener on the FAL server site. If you are unsure what values to use for these parameters, use the Oracle Net Manager to display the listener configuration on the FAL server site.
3. Use the Oracle Net Manager to create a net service that the FAL server can use to connect to the standby database. The net service name should resolve to a connect descriptor that uses the same protocol, host address, port, and SID that you specified when you configured the listener on the standby database site. If you are unsure what values to use for these parameters, use the Oracle Net Manager to display the listener configuration on the standby database site.
4. In the initialization parameter file of the standby database, assign the net service name that you created for the standby database to the `FAL_CLIENT` initialization parameter, and assign the net service name that you created for the FAL server to the `FAL_SERVER` initialization parameter.

Log apply services automatically detect, and the FAL server process running on the primary database resolves, any gaps that may exist when you enable managed recovery with the `RECOVER MANAGED STANDBY DATABASE` statement.

If the FAL process cannot resolve an archive gap, then you must resolve it manually. You can use the `V$ARCHIVE_GAP` fixed view to manually identify archive gaps on standby databases.

> **See Also:** Section B.3 for a description of the manual steps, Section 6.10.2 for a scenario using the `V$ARCHIVE_GAP` view, and *Oracle Net Services Administrator's Guide* for information about Oracle Net

Define the `FAL_CLIENT` and `FAL_SERVER` initialization parameters in the standby database initialization parameter file:

Parameter	Function	Syntax
FAL_CLIENT	This parameter specifies the net service name that the FAL server should use to connect to the standby database.	**Syntax**: FAL_CLIENT=*Oracle Net Service Name* **Example**: FAL_CLIENT='standby1_db'
FAL_SERVER	This parameter specifies the net service name that the standby database should use to connect to the FAL server.	**Syntax**: FAL_SERVER=*Oracle Net Service Name* **Example**: FAL_SERVER='my_primary_db'

The FAL client is the component of the Oracle database server that:

- Runs on the standby database
- Detects an archive gap on the standby database
- Requests the transfer of archived redo logs from the FAL server

The FAL server is a background Oracle process that services the incoming requests from the FAL client. In most cases, the FAL server is located on a primary database. However, it can be located on another standby database.

4.6 Datafile Management

If the standby site uses the same directory naming structure as the primary site, then you do not have to rename the primary database files in the standby control file. If the primary and standby databases are located on the same site, however, or if the primary and standby sites use different directory naming structures, then you must rename the primary database files in the standby control file so that the archived redo 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 database files automatically using these parameters, then you must rename them manually.

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

The initialization parameters in Table 4–1 perform automatic filename conversions.

Table 4–1 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_*.
STANDBY_FILE_MANAGEMENT	When set to auto, this parameter automates the creation and deletion of datafile filenames on the standby site using the same filenames as the primary site.

Use the DB_FILE_NAME_CONVERT parameter to convert the filenames of one or more sets of datafiles on the primary database to filenames on the standby database; use the LOG_FILE_NAME_CONVERT parameter to convert the filename of a new redo log on the primary database to a filename on the standby database.

When the standby database is updated, the DB_FILE_NAME_CONVERT parameter 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.

Adding a datafile or log to the primary database necessitates adding a corresponding file to the standby database. Use the STANDBY_FILE_MANAGEMENT initialization parameter with the DB_FILE_NAME_CONVERT initialization parameter to automate the process of creating files with identical filenames on the standby database and primary database.

The DB_FILE_NAME_CONVERT initialization parameter allows multiple pairs of filenames to be specified. For example:

```
DB_FILE_NAME_CONVERT="/private1/prmy1/df1", "/private1/stby1/df1", \
                     "/private1/prmy1", "/private1/stby1"
STANDBY_FILE_MANAGEMENT=auto
```

Note: When you specify pairs of files, be sure to specify supersets of path names before subsets.

This section contains the following topics:

- Setting the STANDBY_FILE_MANAGEMENT Initialization Parameter
- Restrictions on ALTER DATABASE Operations

4.6.1 Setting the STANDBY_FILE_MANAGEMENT Initialization Parameter

When you set the STANDBY_FILE_MANAGEMENT initialization parameter to auto, it automatically creates the file on the standby sites using the same name that you specified on the primary site.

The STANDBY_FILE_MANAGEMENT initialization parameter works with the DB_FILE_NAME_CONVERT parameter to take care of the case where datafiles are spread across multiple directory paths on the primary database.

4.6.2 Restrictions on ALTER DATABASE Operations

You cannot rename the datafile on the standby site when the STANDBY_FILE_MANAGEMENT initialization parameter is set to auto. In addition, when you set the STANDBY_FILE_MANAGEMENT initialization parameter to auto, the following operations are no longer necessary:

- ALTER DATABASE RENAME
- ALTER DATABASE ADD/DROP LOGFILE
- ALTER DATABASE ADD/DROP LOGFILE MEMBER
- ALTER DATABASE CREATE DATAFILE AS

If you attempt to use any of these operations on the standby database, an error is returned. For example:

```
SQL> ALTER DATABASE RENAME FILE '/private1/stby1/t_db2.dbf' to 'dummy';
```

```
alter database rename file '/private1/stby1/t_db2.dbf' to 'dummy'
*
ERROR at line 1:
ORA-01511: error in renaming log/data files
ORA-01270: RENAME operation is not allowed if STANDBY_FILE_MANAGEMENT is auto
```

See Also: Section 5.8.1 to learn how to add datafiles to a database

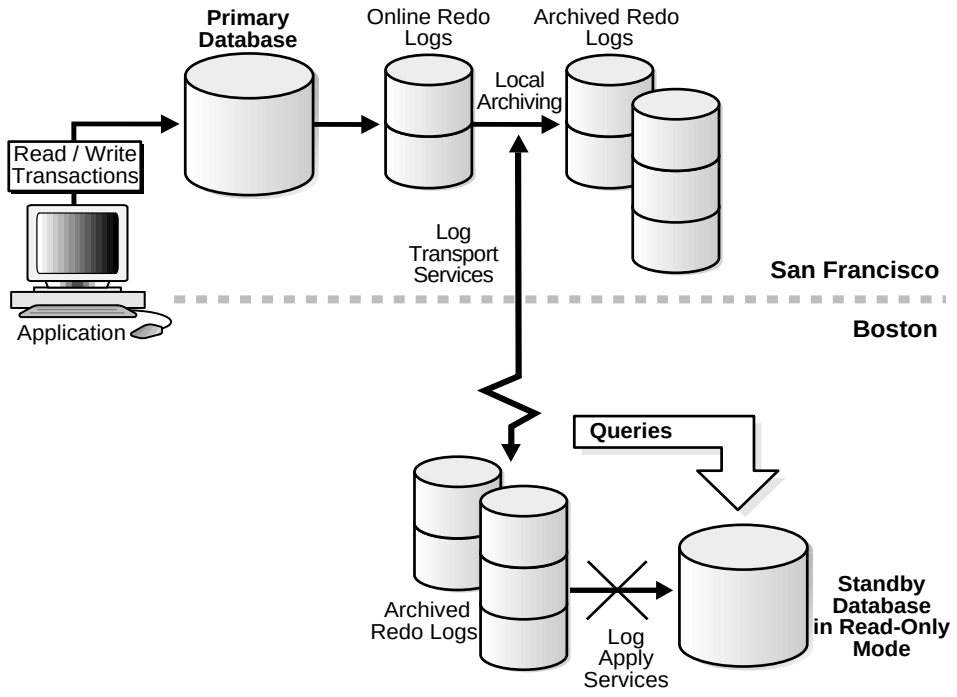
4.7 Read-Only Mode

Read-only mode allows users to open and query a standby database without the potential for online data modifications. This mode reduces system overhead on the primary database by using the standby database for reporting purposes. You can periodically open the standby database in read-only mode to:

- Run reports
- Perform ad hoc queries to ensure that log apply services are updating the standby database properly

Figure 4–3 shows a standby database in read-only mode.

Figure 4–3 Standby Database in Read-Only Mode



This section contains the following topics:

- Assessing Whether to Run in Read-Only Mode
- Placing the Database in Read-Only Mode

4.7.1 Assessing Whether to Run in Read-Only Mode

As you decide whether to run the standby database in read-only mode, consider the following:

- Having the standby database open in read-only mode available for queries makes it unavailable for managed recovery. The archived redo logs are received by the standby database site but are not applied. At some point, you need to put the standby database back in managed recovery mode and apply the archived redo logs to resynchronize the standby database 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 protection against disaster and reporting, then you can maintain multiple standby databases, some read-only and some in managed recovery mode. However, you will need to synchronize the **read-only database.** A database in managed recovery mode gives you immediate protection against disaster.

4.7.2 Placing the Database in Read-Only Mode

You can change from the shutdown mode or managed recovery mode into read-only mode (and back again) using the following procedures. The following modes are possible for a standby database:

- Started
- Mounted
- Managed recovery mode
- Read-only mode

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

1. 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 managed recovery mode:

1. Cancel log apply services:

   ```
   SQL> ALTER DATABASE 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. Restart log apply services:

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

Note: The log apply services component resumes from the time when it was last canceled.

4.8 Read-Only Mode Considerations

Before you put your standby database in read-only mode, consider the following topics:

- Receiving Archived Redo Logs While in Read-Only Mode
- Sorting While in Read-Only Mode
- Sorting Without Temporary Tablespaces

4.8.1 Receiving Archived Redo Logs While in Read-Only Mode

While the standby database is in read-only mode, the standby site can still receive archived redo logs from the primary site. However, these archived redo logs are not automatically applied to the standby database until the database is in managed recovery mode. Consequently, a read-only standby database is not synchronized with the primary database at the archive level. You should not fail over to the standby database unless all archived redo logs have been applied.

See Also: Section 3.4.2 for examples of initialization parameter settings you need to define to automatically archive from the primary site to the standby site

4.8.2 Sorting While in Read-Only Mode

To perform queries on a read-only standby database, the Oracle database server must 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.

4.8.2.1 Creating Temporary Tablespaces

Temporary tablespaces allow you to add tempfile entries in read-only mode for the purpose of making queries. You can then perform on-disk sorting operations in a 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 change to managed or **manual recovery mode** when necessary.
- Minimize runtimes for reports.
- Implement desired optimizations on the primary database only.

> **See Also:** *Oracle9i Database Administrator's Guide* for more information about using tempfiles and temporary tablespaces

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 Section 4.7.2.
2. Create a temporary tablespace. For example, enter:

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

> **See Also:** *Oracle9i SQL Reference* for information about the `CREATE TEMPORARY TABLESPACE` syntax

4.8.3 Sorting Without Temporary Tablespaces

If you attempt to sort without temporary tablespaces by executing a SQL `SELECT * FROM V$PARAMETER` statement when the database is neither mounted nor open, you will get an error. For example:

```
SQL> SELECT * FROM v$parameter;

select * from v$parameter

              *

ERROR at line 1:

ORA-01220: file based sort illegal before database is open
```

To look at the parameters when the database is not open in read/write mode:

- Set the SORT_AREA_SIZE parameter in your initialization parameter file. If you do this, you can execute the SELECT * FROM V$PARAMETER statement when the database is not mounted, is mounted, or is open. SORT_AREA_SIZE is a static parameter. This means the parameter must be specified in your initialization parameter file prior to starting the instance.
- If you do not set the SORT_AREA_SIZE parameter, you cannot select all of the columns from the V$PARAMETER view. If you need to select only a few columns, you can execute the SELECT statement in any database state.

If you can open the database in read/write mode, you can execute a SELECT * FROM V$PARAMETER statement even if the SORT_AREA_SIZE parameter is not specified in your initialization parameter file. However, this only applies to the primary database. On a standby database, opening the database in read-only mode and executing a SELECT * FROM V$PARAMETER statement fails as previously shown. For standby databases, you *must* set the SORT_AREA_SIZE parameter in your initialization parameter file if you wish to execute the SELECT * FROM V$PARAMETER statement.

4.9 Monitoring Log Apply Services

To determine the status of archived redo logs on the standby database, query the V$MANAGED_STANDBY, V$ARCHIVE_DEST_STATUS, V$ARCHIVED_LOG, and V$LOG_HISTORY fixed views. You can also monitor the standby database using the Data Guard Manager.

See Also: Appendix A, "Troubleshooting the Standby Database"

This section contains the following topics:

- Accessing the V$MANAGED_STANDBY Fixed View

- Accessing the V$ARCHIVE_DEST_STATUS Fixed View
- Accessing the V$ARCHIVED_LOG Fixed View
- Accessing the V$LOG_HISTORY Fixed View
- Setting Archive Tracing

4.9.1 Accessing the V$MANAGED_STANDBY Fixed View

Query the standby database to monitor log apply and log transport services activity at the standby site.

```
SQL> SELECT process, status, thread#, sequence#, block#, blocks
  2> FROM v$managed_standby;

PROCESS STATUS       THREAD#    SEQUENCE#  BLOCK#     BLOCKS
------- ------------ ---------- ---------- ---------- ----------
RFS     ATTACHED     1          947        72         72
MRP0    APPLYING_LOG 1          946        10         72
```

The previous query output shows an RFS process that has completed the archival of redo log file sequence number 947. The output also shows a managed recovery operation that is actively applying archived redo log sequence number 946. The recovery operation is currently recovering block number 10 of the 72-block archived redo log file.

> **See Also:** V$MANAGED_STANDBY in Chapter 10, "Fixed Views"

4.9.2 Accessing the V$ARCHIVE_DEST_STATUS Fixed View

To quickly determine the level of synchronization for the standby database, issue the following query:

```
SQL> SELECT archived_thread#, archived_seq#, applied_thread#, applied_seq#
  2> FROM v$archive_dest_status;

ARCHIVED_THREAD# ARCHIVED_SEQ# APPLIED_THREAD# APPLIED_SEQ#
---------------- ------------- --------------- ------------
1                947           1               945
```

The previous query output shows the standby database is two archived log files behind in applying the redo logs received from the primary database. This may indicate that a single recovery process is unable to keep up with the volume of archived redo logs being received. Using the PARALLEL option may be a solution.

> **See Also:** V$ARCHIVE_DEST_STATUS in Chapter 10, "Fixed Views"

4.9.3 Accessing the V$ARCHIVED_LOG Fixed View

The V$ARCHIVED_LOG fixed view on the standby database shows all the archived redo logs received from the primary database. 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 SQL*Plus script (sample output included):

```
SQL> SELECT registrar, creator, thread#, sequence#, first_change#,
  2> next_change# FROM v$archived_log;

REGISTRAR CREATOR THREAD#    SEQUENCE#  FIRST_CHANGE# NEXT_CHANGE#
--------- ------- ---------- ---------- ------------- ------------
RFS       ARCH    1          945        74651         74739
RFS       ARCH    1          946        74739         74772
RFS       ARCH    1          947        74772         74774
```

The previous query output shows three archived redo logs received from the primary database.

> **See Also:** V$ARCHIVED_LOG in Chapter 10, "Fixed Views"

4.9.4 Accessing the V$LOG_HISTORY Fixed View

The V$LOG_HISTORY fixed view on the standby database shows all the archived redo logs that have been recovered. For example:

```
SQL> SELECT thread#, sequence#, first_change#, next_change#
  2> FROM v$log_history;

THREAD#    SEQUENCE#  FIRST_CHANGE# NEXT_CHANGE#
---------- ---------- ------------- ------------
1          945        74651         74739
```

The previous query output shows that the most recently recovered archived redo log was sequence number 945.

See Also: V$LOG_HISTORY in Chapter 10, "Fixed Views"

4.9.5 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 the primary database	Whose filename is specified in the USER_DUMP_DEST initialization parameter
Standby database	Audit trail of the RFS and the ARC*n* process activity relating to archived redo logs on the standby database	Whose filename is specified in the USER_DUMP_DEST initialization parameter

4.9.5.1 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, for example:

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

4.9.5.2 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 an ALTER SYSTEM SET LOG_ARCHIVE_TRACE=*n* statement 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 a SQL statement similar to the following:

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_TRACE=15;
```

In the previous example, specifying 15 sets trace levels 1, 2, 4, and 8 as described in Section 4.9.5.3.

Issue the ALTER SYSTEM statement from a different standby session so that it affects trace output generated by the remote file service (RFS) and ARC*n* processes when the next archived log is received from the primary database. For example, enter:

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

4.9.5.3 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	Disables archived redo log tracing - default setting.
1	Tracks archival of redo log file.
2	Tracks archival status per archived redo log destination.
4	Tracks archival operational phase.
8	Tracks archived redo log destination activity.
16	Tracks detailed archived redo log destination activity.
32	Tracks archived redo log destination parameter modifications.
64	Tracks ARC*n* process state activity.
128	Tracks FAL server process activity.
256	Supported in a future release.

Level	Meaning
512	Tracks asynchronous LGWR 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 6 generates level 2 and level 4 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 /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 : '/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 :  '/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  :  /oracle/dbs/d1arc1_387.dbf
( 4)    ARC0: FINISH
( 2)    ARC0: Archival success destination 2 : 'standby1'
( 2)    ARC0: Archival success destination 1 : '/oracle/dbs/d1arc1_387.dbf'
( 4)    ARC0: COMPLETE, all destinations archived
(16)    ARC0: ArchivedLog entry added: /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
```

5

Managing the Data Guard Environment

This chapter describes how to manage the Data Guard environment. Oracle9*i* Data Guard provides the means to easily manage, manipulate, and change the standby database in many ways.

This chapter contains the following topics:

5.1 Database Roles

A database can be in one of two mutually exclusive roles: primary or standby. You can change these roles dynamically as a planned transition or you can change as a result of a database failure. This is known as **role transition.**

For example, a primary database can change to the standby role so that one of its corresponding standby databases can change to the primary role. This planned transition occurs without having to reinstantiate either database. This is known as a **switchover** operation.

In a failure of the primary database, such as a system or software failure, you may need to change one of its corresponding standby databases to the primary role. This unplanned transition may result in the loss of application data. The potential loss of data is dependent on user-defined parameters of the primary database and on the configuration of the standby databases. This type of transition often requires you to instantiate a new standby database from the newly activated primary database. This is known as a failover operation.

The active role of a database is more than conceptual, because there are physical aspects of the role that you must take into consideration. For example, a database in the primary role uses a current control file, while a database in the standby role uses a standby control file. Using different control files, depending on which database role is active, requires careful coordination of the initialization parameter file.

5.2 Database Role Transitions

Table 5–1 describes four methods for changing database roles.

Table 5–1 Database Role Transitions

Method	Description
Database Switchover	You can switch a database role from primary to standby, as well as from standby to primary, without resetting the online redo logs of the new primary database. No data is lost even when standby redo logs are not configured. No reinstantiation of the primary and standby databases is necessary.
Database Switchback	This is a database switchover, performed in reverse, that results in the original primary database resuming the primary role.
Graceful Database Failover	Database failover changes one of the standby databases into the role of primary database. A graceful failover will automatically recover some or all of the original primary database application data and therefore avoid reinstantiating other standby databases. If so, this is a graceful failover and it will be necessary to reinstantiate only the original primary database as a standby database.

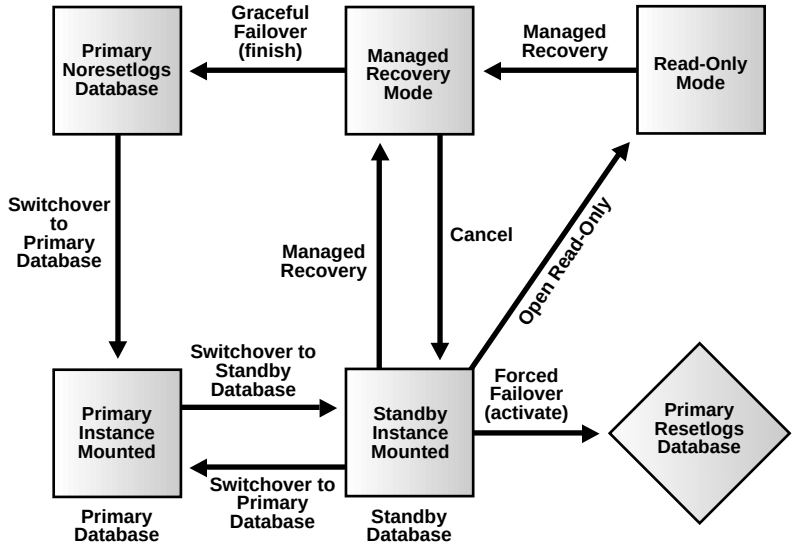
Table 5–1 (Cont.) Database Role Transitions

Method	Description
Forced Database Failover	Database failover changes one of the standby databases into the role of primary database. A forced failover may result in lost application data even when standby redo logs are configured on the standby database. You will need to reinstantiate the original primary database and all other standby databases.

Role transition requires all instances whose roles have changed to be shut down and restarted before the new role become effective.

Figure 5–1 is a road map of the operations that are discussed in the following sections of this chapter.

Figure 5–1 Standby Database Road Map



This section contains the following topics:

- Database Switchover
- Database Switchback

- Graceful Database Failover
- Forced Database Failover

5.2.1 Database Switchover

You can switch a database role from primary to standby, as well as from standby to primary, without resetting the online redo logs of the new primary database. This is known as a **database switchover** operation, instead of a failover operation, because there is no loss of application data, and there is no need to reinstantiate the standby databases, including the other standby databases not involved in the switchover operation.

On the primary database, the SQL `ALTER DATABASE COMMIT TO SWITCHOVER TO STANDBY` statement is used to prepare the current primary database for switchover to the standby role. On one of the standby databases, the SQL `ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY` statement is used to prepare the standby database for switchover to the primary role.

5.2.2 Database Switchback

This is a database switchover, performed in reverse, which results in the original primary database becoming a new primary database. There is no actual **database switchback** operation. Use the switchover procedure instead.

5.2.3 Graceful Database Failover

Database failover changes one of the standby databases into the role of primary database. You should perform a standby database failover only in the event of a software or system failure that results in the loss of the primary database.

One of the consequences of a **graceful database failover** is that the original primary database must be reinstantiated. However, the other standby databases in the configuration do not need to be reinstantiated.

Depending on the log transport services attributes, it may be possible to automatically recover some or all of the primary database modifications. The SQL `ALTER DATABASE RECOVER MANAGED STANDBY DATABASE FINISH` statement is used to finalize the recovery of the archived logs on the standby database.

To issue the SQL `ALTER DATABASE RECOVER MANAGED STANDBY DATABASE FINISH` statement, you must set compatibility to 9.0.1.0.0 in the initialization parameter file. Furthermore, when you apply the standby redo logs, compatibility

must remain at 9.0.1.0.0 from this point forward. Add the following line to the initialization parameter file:

```
compatible=9.0.1.0.0
```

If the SQL `ALTER DATABASE RECOVER MANAGED STANDBY DATABASE FINISH` statement is canceled before the standby redo logs have been applied, you can return to an 8.0 or 8.1 release.

5.2.4 Forced Database Failover

Database failover changes one of the standby databases into the role of primary database. You should perform a standby database failover only in the event of a software or system failure that results in the loss of the primary database.

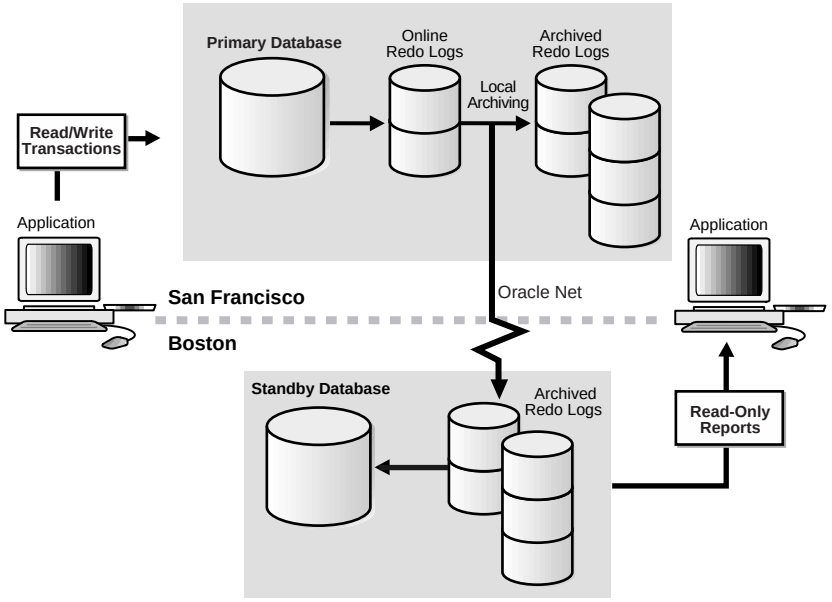
One of the consequences of a **forced database failover** is that the original primary database and all other standby databases must be reinstantiated. Another consequence is that there may be lost application data unless the standby and primary databases have been configured to run in guaranteed protection mode.

The SQL `ALTER DATABASE ACTIVATE STANDBY DATABASE` statement changes the state of a standby database to an active database and prepares it to become the primary database.

5.3 Switching Over Your Database

You can switch a database role from primary to standby, as well as from standby to primary. Figure 5–2 shows a typical two-site Data Guard configuration before the roles of the databases are switched.

Figure 5–2 Typical Data Guard Configuration Before Switchover



The switchover operation is performed in two phases: first, the primary database role is switched to the standby role; then, a standby database is selected to assume the primary database role.

Switchover is always initiated from the primary database and is always completed on a standby database.

When you perform a switchover operation, the existing control file type is converted in place. That is, the primary database current control file is converted into a standby control file, and a standby control file is converted into a primary control file.

This section contains the following topics:

- Switchover Precautions
- Preparation for Successful Switchover
- Switching the Primary Database Role to Standby
- Switching the Standby Database Role to Primary
- Multiple Standby Databases in Switchover

- Standby Databases Not Involved in Switchover
- Identifying Active Instances
- Identifying Active SQL Sessions
- Validating the Switchover Transition

5.3.1 Switchover Precautions

The act of switching database roles should be a well-planned activity. The primary and standby databases involved in the switchover operation should have as small a transactional lag as possible.

Oracle Corporation highly recommends that you consider performing a full, consistent backup of the primary database prior to starting the switchover procedure.

You cannot use a switchover operation to perform a **rolling upgrade** of Oracle software. However, it may be possible to use a switchover operation to perform a hardware-based rolling upgrade. It may be possible to perform an application software rolling upgrade if necessary schema modifications and data updates have been made to the primary database prior to the switchover operation.

5.3.2 Preparation for Successful Switchover

This section describes how to set up log transport services on the standby database in preparation for switchover. Before starting a switchover operation, verify that:

- You have proper initialization parameter files. See Section 5.3.2.1.
- You have network connectivity between the primary and standby databases. See Section 5.3.2.2.
- All Real Application Cluster instances have been shut down. See Section 5.3.2.3.
- All sessions have been disconnected from the instance. See Section 5.3.2.4.
- The standby database is mounted and managed recovery is active. See Section 5.3.2.5.

5.3.2.1 Initialization Parameter Files

Oracle Corporation suggests that you maintain two database initialization parameter files at both the primary and standby databases. This will allow you to easily change the databases from the primary role to the standby role or from the standby role to the primary role.

Most initialization parameters in the initialization parameter files of the primary and standby databases should be identical, although some initialization parameters such as CONTROL_FILES, LOCK_NAME_SPACE, and DB_FILE_NAME_CONVERT may differ.

See Also: Section 3.5.1

5.3.2.2 Network Connectivity

To run a standby database in a Data Guard environment, you must configure an Oracle Net connection between the primary and standby databases so that you can archive the redo logs to the standby site, and so that the FAL client can fetch missing log files from the FAL server. Furthermore, all primary and standby databases in the configuration should have network connectivity to all other databases in the configuration.

For each primary and standby database in the configuration, you should have entries in the `tnsnames.ora` file to identify the other databases in the configuration. You must also have corresponding entries in the `listener.ora` file for each database.

See Also: *Oracle Net Services Administrator's Guide* for information about configuring and administering the `listener.ora` and `tnsnames.ora` initialization files

5.3.2.3 Exclusive Instance Access

When your database is using Real Application Clusters, only one instance is allowed to perform the switchover operation. Shut down all other instances prior to performing the switchover operation.

See Also: Section 5.3.7

5.3.2.4 Exclusive SQL Session Access

The switchover operation requires that only one session be active on the instance performing the switchover. Manually terminate all other sessions prior to the switchover operation. Active sessions may include processes other than SQL sessions.

See Also: Section 5.3.8

5.3.2.5 Standby Database Accessible

The standby database must be mounted and ideally should be in managed recovery mode before you start a primary database switchover operation.

See Also: Section 4.3

5.3.3 Switching the Primary Database Role to Standby

You can switch a database from the primary role to the standby role when the instance is either mounted or open.

At least one standby database must be active in the Data Guard configuration. The standby database must be mounted and have proper network connectivity. Ideally, the standby database that has been targeted to become the new primary database will be mounted and actively recovering when the database roles are switched. If the standby database is open for read-only access, the standby switchover operation will take longer.

The database that will become the new primary database should be in ARCHIVELOG mode.

For a database using Real Application Clusters, only one primary instance is allowed to perform the switchover operation. Shut down all other instances prior to the switchover operation.

The switchover operation requires that only one SQL session be active on the primary instance. Terminate all other SQL sessions prior to switchover.

To verify whether or not it is possible to perform a switchover operation, refer to the SWITCHOVER_STATUS column of the V$DATABASE fixed view on the primary database. For example:

```
SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE;
 SWITCHOVER_STATUS
 -----------------
 TO STANDBY
 1 row selected
```

The TO STANDBY value of the SWITCHOVER_STATUS column indicates you are allowed to switch the database from the primary role to the standby role.

> **See Also:** V$DATABASE in Chapter 10, "Fixed Views" for possible values of the SWITCHOVER_STATUS column and Section 5.3.8 for information on identifying active SQL sessions

To perform the primary-to-standby role switchover operation, issue the following SQL statement:

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO STANDBY;
```

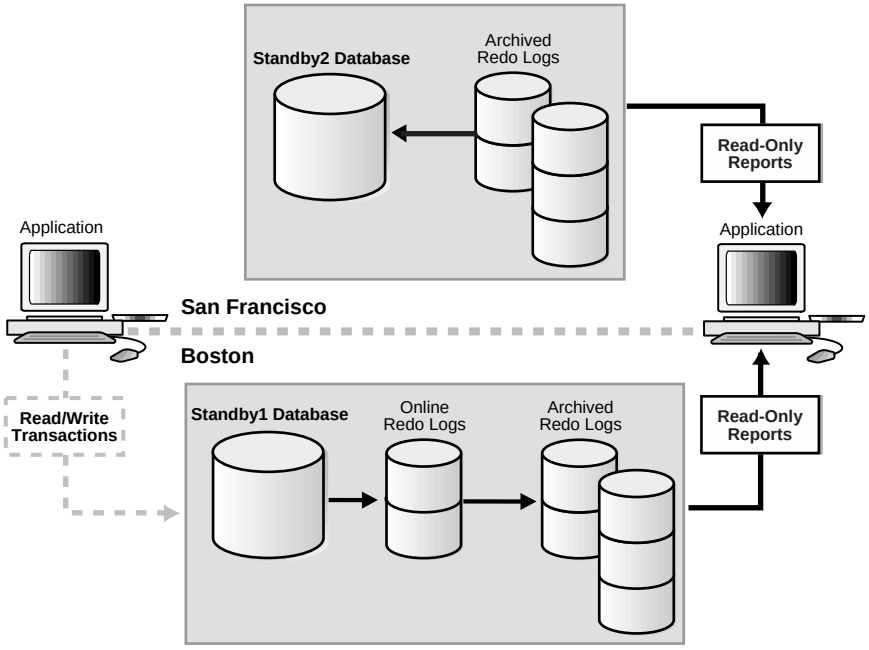
The primary database role will now be converted into a standby database. The current control file will be backed up to the current SQL session trace file prior to the switchover operation; this makes it possible to reconstruct a current control file, if necessary.

Note: Should you wish to convert the original primary database back, you must use the standby switchback procedure. See Section 5.3.4.

Following completion of the switchover operation, you must shut down the instance and restart it as a standby database.

Figure 5–3 shows the Data Guard environment after the primary database has been switched over to a standby database and before the original standby database has become the new primary database. Therefore, you temporarily have two standby databases.

Figure 5–3 Standby Databases Before Switchover to the New Primary Database



5.3.4 Switching the Standby Database Role to Primary

You can switch a database from the standby role to the primary role when the instance is either mounted or open for read-only access. You can also use this procedure to revert a primary database to its original role.

For a database using Real Application Clusters, only one standby instance is allowed to perform the switchover operation. Shut down all other instances prior to the switchover operation.

The switchover operation requires that only one SQL session be active on the standby instance. Terminate all other SQL sessions prior to switchover.

The primary database must have previously switched roles to the standby database. Also, the standby database must have applied the switchover notification from the primary database.

The standby database must be in managed recovery mode prior to starting a switchover operation, so that the primary database switchover operation request

can be coordinated. If managed recovery is not active, or the primary database switchover notification cannot be processed, the switchover operation cannot proceed.

To verify whether it is possible to perform a switchover operation, query the SWITCHOVER_STATUS column of the V$DATABASE fixed view on the standby database. For example:

```
SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE;
SWITCHOVER_STATUS
-----------------
TO PRIMARY
1 row selected
```

The TO PRIMARY value of the SWITCHOVER_STATUS column indicates you are allowed to switch the database from the standby role to the primary role.

After you switch the primary database to the standby role and the switchover notification has been received by the standby database, you should then manually place the standby database in managed recovery mode. This allows the primary database switchover notification to be processed by the standby database. To verify whether the switchover notification has been processed by the standby database, query the SWITCHOVER_STATUS column of the V$DATABASE fixed view on the standby database. For example:

```
SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE;
SWITCHOVER_STATUS
-----------------
SWITCHOVER PENDING
 1 row selected
```

The SWITCHOVER PENDING value of the SWITCHOVER_STATUS column indicates the standby database is about to switch from the standby role to the primary role.

> **See Also:** V$DATABASE in Chapter 10, "Fixed Views" for possible values of the SWITCHOVER_STATUS column and Section 5.3.8 for information on identifying active SQL sessions

To perform the primary-to-standby-role operation, issue the following SQL statement:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE
```

To perform the standby-to-primary role switchover operation, issue the following SQL statement:

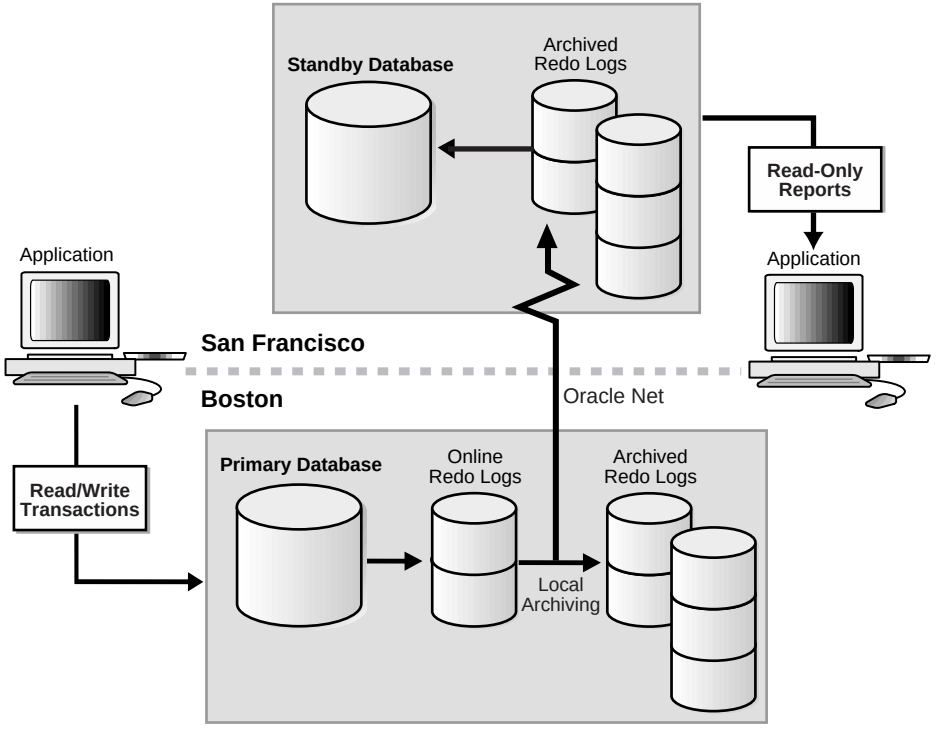
```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY;
```

> **Note:** The `COMMIT TO SWITCHOVER TO PRIMARY` clause automatically creates online redo logs. This may significantly increase the time required to complete the commit operation. Oracle Corporation recommends that you always manually add the online redo logs to the standby database prior to starting the switchover operation. Also, define the `LOG_FILE_NAME_CONVERT` initialization parameter to correlate the standby database path names to the online redo logs.

The selected standby database will now be converted into a primary database. The current standby control file will not be backed up to the current trace file. You can always reconstruct the standby control file from the original primary database which is a new standby database that has a copy of the control file.

Following completion of this operation, you must shut down the instance and restart it as a primary database. Figure 5–4 shows a Data Guard environment after a switchover has taken place.

Figure 5–4 Data Guard Environment After Switchover



5.3.5 Multiple Standby Databases in Switchover

Normally, you choose only one standby database to become the new primary database. You switch the primary database to the standby role and choose one standby database to be switched to the primary role.

In some cases, however, you may choose to switch multiple databases from the standby role to the primary role. If you switch over multiple standby databases, each will become a separate and distinct primary database. Any remaining standby databases can only be connected to a single primary database.

5.3.6 Standby Databases Not Involved in Switchover

There is no need to reinstantiate any other standby databases not involved in the switchover operation. These standby databases will continue to function normally.

5.3.7 Identifying Active Instances

When your database is using Real Application Clusters, active instances prevent a switchover from being performed. When other instances are active, an attempt to switch over fails with the following error message:

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO STANDBY;
ALTER DATABASE COMMIT TO SWITCHOVER TO STANDBY *
ORA-01105: mount is incompatible with mounts by other instances
```

Should this error occur, query the GV$INSTANCE view as follows to determine which instances are causing the problem:

```
SQL> SELECT INSTANCE_NAME, HOST_NAME FROM GV$INSTANCE
  2> WHERE INST_ID <> (SELECT INST_ID FROM V$INSTANCE);
INSTANCE_NAME HOST_NAME
------------- ---------
INST2         standby2
```

In the previous example, the identified instance must be manually shut down before the switchover can proceed. You can connect to the identified instance from your instance and issue the SHUTDOWN statement remotely, for example:

```
SQL> CONNECT sys/change_on_install@standby2 AS SYSDBA
SQL> SHUTDOWN;
SQL> EXIT
```

5.3.8 Identifying Active SQL Sessions

Active SQL sessions prevent a switchover from being processed. Active SQL sessions may include other Oracle processes.

When sessions are active, an attempt to switch over fails with the following error message:

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO STANDBY;
ALTER DATABASE COMMIT TO SWITCHOVER TO STANDBY *
ORA-01093: ALTER DATABASE CLOSE only permitted with no sessions connected
```

Should this occur, query the V$SESSION view to determine which processes are causing the error. For example:

```
SQL> SELECT SID, PROCESS, PROGRAM FROM V$SESSION
  2> WHERE TYPE = 'USER'
  3>  AND SID <> (SELECT DISTINCT SID FROM V$MYSTAT);
SID        PROCESS   PROGRAM
```

```
---------  --------  ------------------------------------------------
         7      3537  oracle@nhclone2 (CJQ0)
        10
        14
        16
        19
        21
 6 rows selected.
```

In the previous example, the JOB_QUEUE_PROCESSES parameter corresponds to the CJQ0 process entry. Because the job queue process is a user process, it is counted as a SQL session that prevents switchover from taking place. The entries with no process or program information are threads started by the job queue controller.

Verify that the JOB_QUEUE_PROCESSES parameter is set using the following SQL statement:

```
SQL> SHOW PARAMETER JOB_QUEUE_PROCESSES;
NAME                                 TYPE      VALUE
------------------------------------ -------   --------------------
job_queue_processes                  integer   5
```

Then set the parameter to 0. For example:

```
SQL> ALTER SYSTEM SET JOB_QUEUE_PROCESSES=0;
Statement processed.
```

Because JOB_QUEUE_PROCESSES is a dynamic parameter, you can change the value and have the change take effect immediately without having to restart the instance. You can now retry the switchover procedure.

You should not modify the parameter in your initialization parameter file. After you shut down the instance and restart it after switchover has completed, the parameter will be reset to the original value. This applies to both primary and standby databases.

Table 5–2 summarizes the common process that prevent switchover and what corrective action you need to take.

Table 5–2 Common Processes That Prevent Switchover

Type of Process	Process Description	Corrective Action
CJQ0	The Job Queue Scheduler Process	Change the JOB_QUEUE_PROCESSES dynamic parameter to the value 0. The change will take effect immediately without having to restart the instance.
QMN0	The Advanced Queue Time Manager	Change the AQ_TM_PROCESSES dynamic parameter to the value 0. The change will take effect immediately without having to restart the instance.
DBSNMP	The Oracle Enterprise Manager Intelligent Agent	Issue the agentctl stop command from the operating system prompt.

5.3.9 Validating the Switchover Transition

After the SQL ALTER DATABASE COMMIT TO SWITCHOVER TO STANDBY statement has been issued on the primary database, the resulting standby database is automatically placed in managed recovery mode. This is necessary to ensure that all unapplied redo logs have been applied to the new standby database. Upon completion of the recovery operation, the standby database instance is dismounted.

Examine the new standby alert log to ensure that recovery has finished on the new standby database. Subsequent standby database switchover operations cannot take place until this happens and the original standby database acknowledges that it has received all available archived logs.

The following is an example of a successful managed recovery operation as shown in the standby alert log:

```
MRP0: Media Recovery Complete: End-Of-REDO
Resetting standby activation ID 3830496333 (0xe450bc4d)
MRP0: Background Media Recovery process is now terminated
```

If the last primary database online redo log is not archived to the original standby database, or if the SQL ALTER DATABASE COMMIT TO SWITCHOVER TO STANDBY statement is issued before the archived log has been applied to the original standby database, you will have two standby databases and no primary database.

If the primary database finished archiving the last online redo log, you can query the SEQUENCE# column in the V$ARCHIVED_LOG view to see if the last archived log has been applied on the standby database. If not, you can manually copy the

archived log from the primary database to the standby database and register it with the SQL `ALTER DATABASE REGISTER LOGFILE filespec` statement. If you then start up the managed recovery process, the archived log will be automatically applied. Query the `SWITCHOVER_STATUS` column in the `V$DATABASE` view to verify that switchover to the primary role is now possible.

```
SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE;
SWITCHOVER_STATUS
-----------------
TO PRIMARY
1 row selected
```

In some situations where an error has occurred, it may still be possible to revert the new standby database back to the original primary database using the following steps:

1. When the switchover procedure to change the role from primary to standby was initiated, a trace file was written in the log directory that contained the SQL statements to re-create the original primary control file. Locate the trace file and extract those SQL statements into a temporary file. Execute the temporary file from SQL*Plus. This should convert the new standby database back to the original primary role.
2. Create a new standby control file. This is necessary to resynchronize the primary and standby databases. Copy the standby control file to the standby sites.
3. Shut down the original standby instance and restart it.

If this procedure is successful and archive gap management is enabled, the FAL processes will start and rearchive any missing archived logs to the standby database. Force a log switch on the primary database and examine the alert logs on both the primary and standby databases to ensure that the archived log sequence numbers are correct.

> **See Also:** Section 4.5 for information about archive gap management and Section 4.9.5 for information about locating the trace files

5.4 Database Failover

A **failover** operation is typically used when a catastrophic failure occurs on the primary database, and there is no possibility of recovering the primary database in

a timely manner. The primary database is discarded and the standby database assumes the primary database role.

You should not fail over to a standby database except in an emergency. After failover, the standby database becomes an ordinary primary database and loses its previous standby database capability. That is, you cannot return the database to a standby mode that is compatible with the original primary database.

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

- Switchover
  - No data is lost even when standby redo logs are not available on the standby database.
  - No database reinstantiation is necessary.
- Failover
  - Data loss is possible even when standby redo logs are available on the standby database, unless the standby database is running in guaranteed protection mode.
  - The original primary database must be reinstantiated.
  - Other standby databases may need to be reinstantiated.

Note: It is not always necessary to fail over to the standby database. In some cases, recovery of the primary database may be faster. See Section 5.4.6.4.

This section contains the following topics:

- Graceful Failover
- Forced Failover

5.4.1 Data Divergence and Data Loss

Data divergence is the temporary state of the primary database that occurs when a standby database becomes inaccessible. While in the data divergence state, the primary database is able to commit transactions whose data modifications are not immediately available on the standby database. Data divergence also occurs when you use the LGWR or ARC*n* process for asynchronous standby database archival operations.

Note: Use of the LGWR process in conjunction with synchronous standby database archival operations will prevent data divergence as long as continuous network connectivity is available.

Data loss occurs when you fail over to a standby database whose corresponding primary database is in the data divergence state. To prevent primary database data divergence if network connectivity fails, use the guaranteed protection mode.

Note: When you have multiple standby databases, the primary database may have diverged relative to one standby database, but not necessarily with other standby databases. Also, the degree of divergence may differ with each standby database.

5.4.2 Primary Database Protection Modes Overview

The primary database protection modes are failure policies that dictate how to manage the primary database if a standby database becomes inaccessible. There are four primary database protection modes:

- Guaranteed protection
- Instant protection
- Rapid protection
- Delayed protection

Guaranteed protection mode dictates that the primary database modifications must always be available on at least one standby database; data divergence is prohibited

by terminating the primary instance if it loses network connectivity to the last standby database.

Instant protection mode dictates that the primary database modifications always be available on at least one standby database. Unlike guaranteed protection mode, data divergence is not prohibited by the instant protection mode. Data divergence exists for the duration that standby database connectivity is unavailable, and can be resolved when network connectivity to the primary database is reestablished.

Rapid protection mode indicates that primary database modifications are available to the standby database as soon as possible with minimal effect on primary database performance.

Delayed protection mode indicates that primary database modifications will ultimately be available on the standby site, as long as the network is active. Both the rapid and delayed protection modes allow the primary database to diverge from all standby databases, even when network connectivity is available. The degree of data divergence is equivalent to the data contained in the unarchived primary database online redo logs.

5.4.3 Standby Database No-Data-Loss Failover

On a standby database, no-data-loss failover is possible only when the corresponding primary database is operating in a data protection mode that provides no-data-divergence semantics, such as guaranteed and instant protection mode. This means that all primary database archived redo logs necessary for no-data-loss failover are available. However, it is possible that the archived redo logs have not yet been applied to the standby database. No-data-loss failover requires that all archived redo logs are first applied. If the archived redo logs are not applied, failover to a standby database will result in data loss relative to the primary database.

5.4.4 Standby Database Data-Loss Failover

Standby database data-loss failover occurs when primary database modifications have not all been applied, or when the corresponding primary database is operating in a data protection mode that allows data-divergence semantics, such as rapid and delayed protection modes. This results in standby database data loss relative to the primary database. The amount of data loss can be controlled by primary database archived log destination attributes and the availability of standby redo logs at the standby database.

5.4.5 Planning for Database Failover

It is possible for the standby database to have lost transactions committed on the original primary database. This occurs because the current log of the primary database contains database modifications that may not be available to the standby database. Therefore, if the primary database current log is unavailable, data loss may be possible on the standby database.

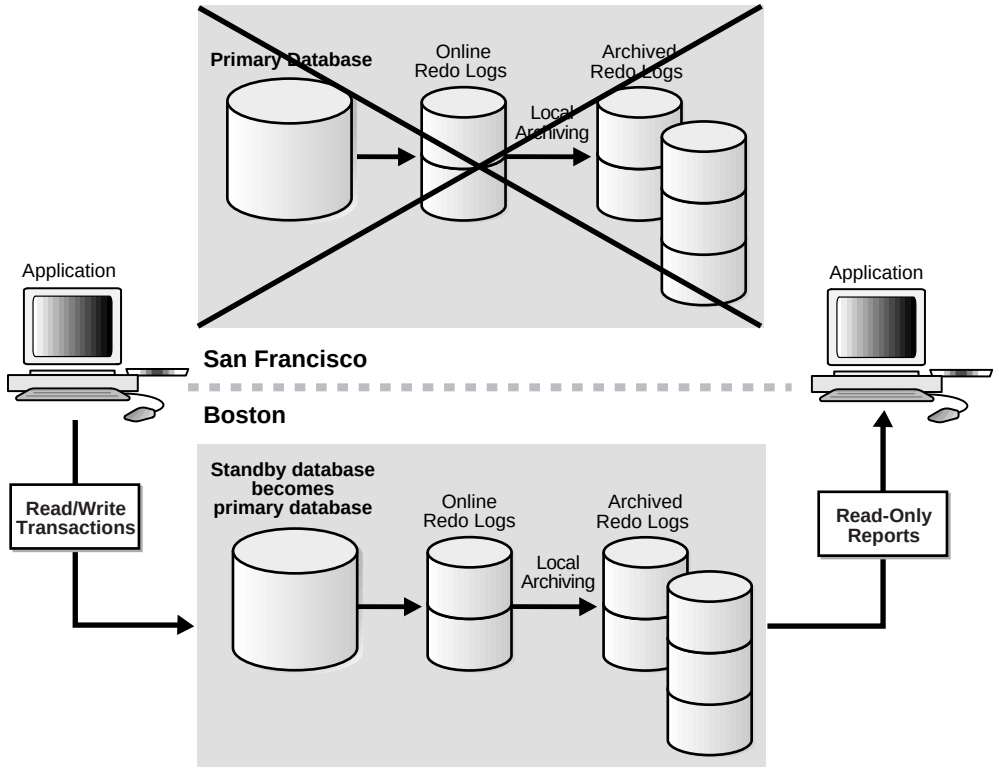
To avoid or minimize data loss during a database failover operation, always use standby redo logs. See Section 3.6.3.4 for information on creating standby redo logs. If the primary database is lost, the use of standby redo logs allows primary database modifications to be automatically recovered on the standby database. Depending on the log transport services options used, this may provide no-data-loss or minimal-data-loss failover.

If the standby database does not use standby redo logs, database failover will almost always result in data loss. However, if the primary database is not completely lost, it may be possible to manually recover the primary database online redo logs and apply them on the standby database.

In both cases, the original primary database is incompatible with the resulting new primary database.

Figure 5–5 shows the result of a failover operation from a primary database in San Francisco to a standby database in Boston.

Figure 5–5 Failover to a Standby Database



The presence of standby redo logs on the standby database determines whether a graceful failover is possible. Depending on the log transport services destination attributes, a graceful failover may provide no data loss or minimal data loss. A forced failover will result in intentional data loss. These two types of failover are discussed in Section 5.4.6 and Section 5.4.7.

5.4.6 Graceful Failover

Database failover changes one of the standby databases into the role of primary database. You should perform a standby database failover only in the event of a software or system failure that results in the loss of the primary database.

This section contains the following topics:

- No-Data-Loss Failover

- Minimal-Data-Loss Failover
- Switching Database Roles After Graceful Failover
- Primary Database No-Data-Loss Recovery

5.4.6.1 No-Data-Loss Failover

No-data-loss failover to the standby database can be achieved when the primary database log transport services has been previously set up to provide this capability. The primary database must be using the guaranteed or instant protection mode. These modes require that the primary database have the following archived log destination attributes:

- `LGWR` - archived by the LGWR process
- `SYNC` - synchronous network transmission
- `SERVICE` - standby database
- `AFFIRM` - synchronous archived log disk I/O (implied if primary database is in `PROTECTED` mode)
- `REGISTER` - archived log registration on the standby database
- `MANDATORY` - online redo log must be archived to standby

Furthermore, the archived log destinations cannot have the following attribute:

- `DEPENDENCY` - archived log destination dependency

In addition, standby databases participating in the no-data-loss environment must have standby redo logs available.

See Also: Section 3.6.3.4 for an overview of standby redo logs

If a failure occurs at the primary site, there will be incomplete standby redo logs that can be salvaged with the `RECOVER ... FINISH` statement. For example:

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

When you start managed recovery, specify the `FINISH` control option to define the mode of the recovery operation. You can also specify the `FINISH` control option to alter the mode of an ongoing managed recovery operation. If the `FINISH` control option is being used to alter the mode of an ongoing managed recovery operation, specifying the `NOWAIT` option allows control to be returned to the foreground process before the recovery operation completes.

When the NOWAIT option is used, the V$MANAGED_STANDBY fixed view may be checked to verify when the managed recovery operation is finished. There should be no rows selected by the following statement after recovery has completed:

```
SQL> SELECT PROCESS FROM V$MANAGED_STANDBY;
no rows selected
```

5.4.6.2 Minimal-Data-Loss Failover

Minimal-data-loss failover to the standby database can be achieved when the primary database log transport services have been previously set up to provide this capability. Standby databases participating in the minimal-data-loss environment must have the following archived log destination attributes:

- LGWR - archived by the LGWR process
- ASYNC - asynchronous network transmission
- SERVICE - standby database
- REGISTER - archived log registration on the standby database

Furthermore, the archived log destinations cannot have the following attribute:

- DEPENDENCY - archived log destination dependency

In addition, standby databases participating in the minimal-data-loss environment must have standby redo logs available.

> **See Also:** Section 3.6.1.3 for an overview on minimal-data-loss failover

Using a lower block count for the archived log destination ASYNC attribute minimizes the amount of potential data loss but possibly decreases primary database throughput. The archived log destination AFFIRM attribute can also be used to further reduce potential data loss.

If a failure occurs at the primary site, there will be incomplete standby redo logs on the standby database. The standby redo log contents can be salvaged with the RECOVER...FINISH statement, for example:

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

Note: This is the same statement used for no-data-loss recovery. This statement finishes recovery of the standby redo logs, applying as much data to the standby database as possible.

5.4.6.3 Switching Database Roles After Graceful Failover

After a graceful failover, subsequent redo logs from the original primary database cannot be applied to the standby database. The standby database is "on hold." The standby redo logs have been archived and should be copied to, registered, and recovered on other standby databases derived from the original primary database.

You must now change a standby database into the new primary database. Change one of the available standby databases, which may include the original primary database, to the primary role by issuing the following statement:

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY;
```

No reinstantiation of the other standby databases is required. Once the archived standby redo logs have been registered and recovered on all destinations, the other standby databases are ready to receive redo logs from the new primary database. To register the copied archived redo logs, issue the following statement on each standby database:

```
SQL> ALTER DATABASE REGISTER LOGFILE '/standby/arch_dest/arch_1_101.arc';
SQL> ALTER DATABASE REGISTER LOGFILE '/standby/arch_dest/arch_1_102.arc';
```

The original primary database must be reinstantiated from a backup of the new primary database.

5.4.6.4 Primary Database No-Data-Loss Recovery

It may be faster to recover the primary database, even you are when using a no-data-loss environment.

When the primary database is operating in no-data-loss mode and a system failure occurs, you can restart the primary database without incurring data loss on the corresponding standby databases. The following failure scenarios are automatically recovered by log transport services:

- Instance recovery, which is only possible in a Real Application Clusters configuration, occurs in an open primary database when one instance discovers that another instance has failed. A surviving instance automatically uses the online redo log to recover the committed data in the database buffers that was lost when the instance failed. Further, the Oracle database server undoes any

transactions that were in progress on the instance when it failed and then clears any locks held by the failed instance after recovery is complete.

- Failure recovery occurs when either a single-instance database fails or all instances of a multi-instance primary database fail. In failure recovery, an instance must first open the database and then execute recovery operations. In general, the first instance to open the database after a failure or `SHUTDOWN ABORT` automatically performs failure recovery.

In both of these scenarios, the standby databases participating in the no-data-loss environment will be automatically resynchronized with the recovered primary database. However, for the resynchronization to occur, the primary database archived log destinations must be properly established to identify the standby databases, and network connectivity must exist between the primary database and the standby databases.

> **Note:** When the primary database is running in a mode other than no-data-loss, the archiver (ARC*n)* process is responsible for performing resynchronization activities to the primary database.

5.4.7 Forced Failover

Most failures can be resolved at a primary site within a reasonable amount of time. However, a database that needs complete restoration and recovery (or a database that is partially restored and recovered) can be resolved quickly by failing over to the standby database. Failures such as the following are good reasons to switch to the standby database:

- Loss of the current online redo log
- Loss of system or rollback tablespace
- Loss of hardware and the database
- Inadvertent database operations that cannot be recovered quickly such as dropping a tablespace

There are two methods to perform a failover operation. Which to use depends on whether the standby database contains standby redo logs. If standby redo logs are available, you must either perform a graceful failover operation or disregard the standby redo logs intentionally and perform a forced failover operation. If standby redo logs are not available, your only method is to **activate** the standby database, which is a forced failover operation.

You should not perform a forced failover to the standby database except in an emergency.

> **Note:** You should not fail over to 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, you should attempt to archive the **current online redo log** on the primary database manually, and then copy and apply all available archived redo logs to the standby database that you plan on changing to the primary database role.

Before you can execute a forced failover, you must stop the primary database archival operations to the standby database. See Section 3.3.1.2 for information about disabling archived log destinations using the DEFER attribute of the LOG_ARCHIVE_DEST_STATE_*n* initialization parameter.

This section contains the following topics:

- Recovering Primary Database Modifications
- Standby Database Failover
- Intentional Data Loss

5.4.7.1 Recovering Primary Database Modifications

Prior to performing the standby database failover operation, you should always attempt to transfer as much of the missing primary database contents to the standby database. The more database modifications that can be transferred to the standby site, the closer the new primary database will match to the original primary database.

Four steps are necessary to move modifications from the primary database to the standby database:

1. If possible, archive the current log group of the primary database locally, for example:

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

   However, in a true database failover situation, this is seldom possible. If you are able to sufficiently recover the primary database to permit online redo log archival, then you can probably recover the entire primary database.

2. By default, you cannot failover to a standby database whose control file was created from a primary database operating in protected mode, even if the primary database is no longer available. To force a failover, you must take the standby database out of protected mode by issuing the following statement:

```
SQL> ALTER DATABASE SET STANDBY DATABASE UNPROTECTED;
```

> **Note:** This statement must be issued on the standby database.

3. If possible, manually copy all primary database archived log groups to the standby database.

   Use an appropriate operating system utility for transferring binary data, for example:

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

4. Register the copied archived logs with the standby database using the SQL `ALTER DATABASE REGISTER LOGFILE` statement. Each archived log must be registered individually, for example:

```
SQL> ALTER DATABASE REGISTER LOGFILE '/standby/arch_dest/arch_1_101.arc';
SQL> ALTER DATABASE REGISTER LOGFILE '/standby/arch_dest/arch_1_102.arc';
```

5. Apply all available archived logs to the standby database, for example:

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

> **Note:** Do not use the `RECOVER ... FINISH` statement when applying manually copied archived logs to the standby database.

This procedure rolls the standby database forward as close to the time of the primary database failure as possible.

5.4.7.2 Standby Database Failover

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='cluster_database';
NAME                 VALUE
-------------------- -----------------------------------
```

```
cluster_database            FALSE
1 row selected.
```

If the value is `true`, then the database is *not* mounted exclusively and you must shut down, set the parameter to `false`, and restart the instance. If the value is `false`, then the database is mounted exclusively.

2. Fail over to the standby database:

   Note: This statement resets the online redo logs.

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

   Note: The `ACTIVATE STANDBY DATABASE` clause automatically creates online redo logs. This may significantly increase the time required to complete the activate operation. Oracle Corporation recommends that you always manually add the online redo logs to the standby database prior to starting the failover operation. Also, define the `LOG_FILE_NAME_CONVERT` initialization parameter to correlate the standby database path names to the online redo logs.

3. Shut down the standby instance:

   ```
   SQL> SHUTDOWN IMMEDIATE
   ```

4. At this point, the former standby database is now your primary database. Back up your new primary database. This task, while not required, is a recommended safety measure because you cannot recover changes made after failover without a backup.

5. Start the new primary instance in read / write mode:

   ```
   SQL> STARTUP pfile=initPRIMARY.ora
   ```

   Now that you have switched the database role, use the proper initialization parameter file.

See Also: Section 3.5.3 for information about setting up initialization parameter files for the standby and primary roles

5.4.7.3 Intentional Data Loss

If the standby database has standby redo logs, you should always attempt a graceful failover operation. However, this may result in undesirable modifications being applied to the standby database. The following shows the error produced if you attempt to activate a standby database without applying all received archived logs:

```
SQL> ALTER DATABASE ACTIVATE STANDBY DATABASE;
ERROR at line 1:
ORA-16140: standby online logs have not been recovered
```

You can force the failover to the standby database, which will result in lost data. For example:

```
SQL> ALTER DATABASE ACTIVATE STANDBY DATABASE
  2> SKIP STANDBY LOGFILE;
```

Note: Oracle Corporation recommends that you always apply all data to the standby database prior to the failover operation.

5.5 Role Transition Summary

Table 5–3 summarizes the database role transition options and requirements.

Table 5–3 Summary of Role Transitions

Summary of Role Transitions	Database Switchover	Graceful Database Failover - No Data Loss	Graceful Database Failover - Minimal Data Loss	Forced Database Failover
Data Loss Potential	No data loss	No data loss	Minimal data loss	Probable data loss
Data Divergence Potential	Not applicable	No divergence if protected Possible divergence if unprotected	Possible divergence	Probable divergence
Database Reinstantiation Requirements	None	Primary database	Primary database	Primary and all other standby databases

Table 5–3 (Cont.) Summary of Role Transitions

Summary of Role Transitions	Database Switchover	Graceful Database Failover - No Data Loss	Graceful Database Failover - Minimal Data Loss	Forced Database Failover
Required Archived Log Destination Attributes	SERVICE REGISTER	SERVICE LGWR SYNC AFFIRM REGISTER MANDATORY	SERVICE LGWR ASYNC REGISTER	SERVICE
Standby Redo Logs	Optional	Required	Required	Optional
SQL Statement Issued on the Primary Database	ALTER DATABASE COMMIT TO SWITCHOVER TO STANDBY	Not applicable	Not applicable	Not applicable
SQL Statement Issued on the Standby Database	ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY	ALTER DATABASE RECOVER MANAGED STANDBY DATABASE FINISH ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY	ALTER DATABASE RECOVER MANAGED STANDBY DATABASE FINISH ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY	ALTER DATABASE ACTIVATE STANDBY DATABASE

5.6 Backing Up the Primary Database Using the Standby Database

You can use the Recovery Manager utility (RMAN) to back up the primary database at the standby site. This allows the standby database to offload the task of database backup from the primary database. Using RMAN at the standby site, you can back up the datafiles and the archived redo logs while the standby database is in managed recovery mode. These backups can be later restored to the primary database with RMAN.

The primary control file, however, still must be backed up at the primary site. This is not a problem, because the control file is significantly smaller than the datafiles.

See Also: *Oracle9i Recovery Manager User's Guide* for more details about RMAN backup and recovery of a primary database using a standby database and Section 6.12 for a relevant scenario

5.7 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:

- Dynamic Performance Views (Fixed Views)
- Monitoring the Primary and Standby Databases
- Determining Which Logs Have Been Applied to the Standby Database
- Determining Which Logs Have Not Been Received by the Standby Site

5.7.1 Dynamic Performance Views (Fixed Views)

The Oracle database server contains a set of underlying views that are maintained by the server. These views are often called **dynamic performance views** because they are continuously updated while a database is open and in use, and their contents relate primarily to performance. These views are also called **fixed views,** because they cannot be altered or removed by the database administrator.

These view names are prefixed with either V$ or GV$, for example, `V$ARCHIVE_DEST` or `GV$ARCHIVE_DEST`.

Standard dynamic performance views (V$ fixed views) store information on the local instance. In contrast, global dynamic performance views (GV$ fixed views), store information on all open instances. Each V$ fixed view has a corresponding GV$ fixed view.

The following fixed views contain useful information for monitoring the Data Guard environment:

- `V$ARCHIVE_DEST`

  Describes the archived redo log destinations associated with the current instance on the primary site. You can use this view to find out the current settings of your archived redo log destinations.

- V$ARCHIVE_DEST_STATUS

  This view, an extension of the V$ARCHIVE_DEST view, displays runtime and configuration information for the archived redo log destinations. You can use this view to determine the progress of archiving to each destination. It also shows the current status of the archive destination.

- V$ARCHIVE_GAP

  Provides information about archive gaps. You can use this view to find out the current archive gap that is blocking recovery.

- V$ARCHIVED_LOG

  Displays archived redo log information from the control file, including archived log names. This view gives you information on which log has been archived to where on your primary database. On the primary database, this view describes the logs archived to both the local and remote destinations. On a standby database, this view provides information about the logs archived to this standby database. You can use this fixed view to help you to track archiving progress to the standby site.

- V$DATABASE

  Contains database information from the control file. You can use this view to quickly find out if your database is a primary or a standby database, as well as its switchover status.

- V$DATAFILE

  Contains datafile information from the control file. You can query this fixed view to verify that the standby datafiles are correctly renamed after your standby database is instantiated.

  See Also: Table 6–1 and Section 6.3.2 for information on renaming standby datafiles when instantiating a standby database

- V$LOG

  Contains log file information from the online redo logs. You can use the information about the current online log on the primary database from this view as a reference point to determine how far behind your standby database is in receiving and applying logs.

- V$LOGFILE

  Contains information about the online redo logs and standby redo logs.

- V$LOG_HISTORY

  Contains log history information from the control file, including a record of the latest archived log that was applied.

- V$MANAGED_STANDBY

  Displays current and status information for some Oracle database server processes related to Data Guard. This view can show both foreground and background processes. You can use this view to monitor the various Data Guard recovery and archiving processes.

- V$STANDBY_LOG

  Provides information about the standby redo logs. Standby redo logs are similar to online redo logs, but they are only used on a standby database receiving logs from the primary database using the log writer process.

  See Also: Chapter 10, "Fixed Views" and the *Oracle9i Database Reference* for additional information on view columns

5.7.2 Monitoring the Primary and Standby Databases

Table 5–4 indicates whether a primary database event is automatically administered by log transport services and log apply services or requires additional intervention by the database administrator (DBA) to be propagated to the standby database. It also describes how to respond to these events.

Table 5–4 Troubleshooting Primary Database Events

Primary Database Event	Primary Site Problem Report Location	Standby Site Problem Report Location	Recommended Response
Archiving errors	■ ERROR column of V$ARCHIVE_DEST view ■ Alert log ■ ARCHIVED column of V$LOG view ■ Archiving trace files	Remote file server (RFS) process trace file	Fix the problem reported in the alert log file or trace file and resume archiving to the destination. If the problem persists or archiving performance is degraded, you can create scripts to send the archived redo logs.
Thread events	■ Alert log ■ V$THREAD view	Alert log	Thread events are automatically propagated through archived logs, so no extra action is necessary.
Redo log changes	Alert log V$LOG view and STATUS column of V$LOGFILE view	Alert log	Redo log changes do not affect the standby database unless a redo log is cleared or lost. In these cases, you must rebuild the standby database. See Section 2.3. Pre-clear the logs on the standby database with the ALTER DATABASE CLEAR LOGFILE statement. See Section 5.8.9.
Issue a CREATE CONTROLFILE statement	Alert log	Database functions normally until it encounters redo depending on any parameter changes.	Re-create the standby control file. See Section 5.8.8. Re-create the standby database if the primary database is opened with the RESETLOGS option. See Section 2.3.
Managed recovery performed	Alert log	Alert log	Re-create the standby database if the RESETLOGS option is utilized. See Section 2.3.
Tablespace status changes (made read / write or read-only, placed online or offline)	■ DBA_TABLESPACES view ■ Alert log	■ Verify that all datafiles are online. ■ V$RECOVER_FILE view	Status changes are automatically propagated, so no response is necessary. Datafiles remain online.

Table 5–4 (Cont.) Troubleshooting Primary Database Events

Primary Database Event	Primary Site Problem Report Location	Standby Site Problem Report Location	Recommended Response
Add datafile or create tablespace	■ DBA_DATA_FILES view ■ Alert log	■ ORA-283, ORA-1670, ORA-1157, ORA-1110 ■ Standby recovery stops.	If you have not set the STANDBY_FILE_MANAGEMENT initialization parameter to auto, you must re-create the control file on the standby database. See Section 5.8.8.
Drop tablespace	■ DBA_DATA_FILES view ■ Alert log	Alert log	If you have not set the STANDBY_FILE_MANAGEMENT initialization parameter to auto, you must refresh the control file on the standby database. See Section 6.3.6.
Tablespace or datafile taken offline, or datafile is deleted offline	■ V$RECOVER_FILE view ■ Alert log The tablespace or datafile requires recovery when you attempt to bring it online.	■ Verify that all datafiles are online. ■ V$RECOVER_FILE view	Datafiles remain online. The tablespace or datafile is fine after standby database activation.
Rename datafile	Alert log	Alert log	See Section 5.8.2.
Unlogged or unrecoverable operations	■ Direct loader invalidates block range redo entry in online redo log. Check V$DATAFILE view. ■ V$DATABASE view	Alert log. File blocks are invalidated unless they are in the future redo, in which case they are not touched.	Unlogged changes are not propagated to the standby database. See Section 5.8.7 on how to apply these changes.
Recovery progress	■ V$ARCHIVE_DEST_STATUS view ■ Alert log	■ V$ARCHIVED_LOG view ■ V$LOG_HISTORY view ■ V$MANAGED_STANDBY view ■ Alert log	Check the views on the primary site and the standby site for recovery progress. See Section 4.9.

Table 5–4 (Cont.) Troubleshooting Primary Database Events

Primary Database Event	Primary Site Problem Report Location	Standby Site Problem Report Location	Recommended Response
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 Section 2.3.
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. See Chapter 7, "Initialization Parameters".

5.7.3 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.

You can also use the `APPLIED` column in the `V$ARCHIVED_LOG` fixed view on the standby database to find out which log is applied on the standby database. The column displays `YES` for the log that has been applied. For example:

```
SQL> SELECT THREAD#, SEQUENCE#, APPLIED FROM V$ARCHIVED_LOG;

   THREAD#  SEQUENCE# APP
---------- ---------- ---
         1          2 YES
```

```
         1          3 YES
         1          4 YES
         1          5 YES
         1          6 YES
         1          7 YES
         1          8 YES
         1          9 YES
         1         10 YES
         1         11 NO

10 rows selected.
```

5.7.4 Determining Which Logs Have Not Been Received by the Standby Site

Each archive destination has a destination ID assigned to it. You can query the DEST_ID column in the V$ARCHIVE_DEST fixed view to find out your destination ID. You can then use this destination ID in a query on the primary database to discover logs that have not been sent to a particular standby site.

For example, assume the current local archive destination ID on your primary database is 1, and the destination ID of one of your remote standby databases is 2. To find out which logs have not been received by this standby destination, issue the following query on the primary database:

```
SQL> SELECT local.thread#, local.sequence# from
  2> (select thread#, sequence# from v$archived_log where dest_id=1) local
  3>  where
  4> local.sequence# not in
  5> (select sequence# from v$archived_log where dest_id=2 and
  6> thread# = local.thread#);

   THREAD#  SEQUENCE#
---------- ----------
  1        12
  1        13
  1        14
```

The preceding example shows the logs that have not yet been received by standby destination 2.

5.8 Responding to Events That Affect the Standby 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 offline or bringing it online
- Changing tablespace status from read/write to read-only and the reverse
- Renaming a datafile
- Dropping a tablespace

However, you might need to perform maintenance on the standby database when you add a datafile to the primary database or create a tablespace on the primary database.

In addition, you must re-create the standby database entirely when you restore a **backup control file** on the primary database or open the primary database with the RESETLOGS option.

This section contains the following topics:

5.8.1 Adding or Dropping Tablespaces and Adding or Deleting Datafiles in the Primary Database

Adding or dropping a tablespace and/or adding or deleting a datafile in the primary database generates redo logs that, when applied to the standby database, automatically add or delete the datafile name or add or drop the tablespace in 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.

Data Guard automatically updates tablespaces and datafiles on the standby site if you use the STANDBY_FILE_MANAGEMENT parameter with the auto option. For example:

```
STANDBY_FILE_MANAGEMENT=auto
```

> **See Also:** Section 6.3.1 for more information on adding datafiles and tablespaces and Section 6.3.3 for more information on deleting datafiles and dropping tablespaces

5.8.2 Renaming Datafiles on the Primary Database

If you rename a datafile on your primary database, the new name does 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, you must update the standby database control file.

> **See Also:** Section 6.3.2 for a scenario that explains renaming a datafiles on the primary database

5.8.3 Adding or Deleting 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 original primary database.

See Also: Section 5.8.8 for instructions on refreshing the standby database control file

5.8.4 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 Also: Section 2.3 and Section 6.9 for information on re-creating the standby database

5.8.5 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 Section 5.8.8.

5.8.6 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, to skip recovery of datafiles that were not copied to the standby database, take the missing datafiles offline using the following statement on the standby database:

```
SQL> ALTER DATABASE DATAFILE 'MISSING00004' OFFLINE DROP;
```

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

5.8.7 Detecting Unlogged or Unrecoverable Operations

> **Caution:** Unlogged or unrecoverable operations invalidate the standby database and may require substantial DBA administrative activities.

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

5.8.7.1 Propagating Unrecoverable Operations Manually

Primary database operations using the `UNRECOVERABLE` option are not propagated to the standby database because these operations 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, the Oracle database server writes error messages in the standby database alert log file. The following error message is displayed:

```
Errors in file /oracle/rdbms/log/rcv12_ora_150.trc:
ORA-01578: ORACLE data block corrupted (file # 1, block # 36031)
ORA-01110: data file 1: '/oracle/dbs/s1_t_db1.dbf'
ORA-26040: Data block was loaded using the NOLOGGING option
```

To correct your standby database following an unrecoverable operation, you will need to perform one or more of the following tasks:

- Take the affected datafiles offline in the standby database and drop the tablespace after failover. See Section 5.8.6.
- Re-create the standby database from a new database backup. See Section 2.3.
- Back up the affected tablespace and archive the current logs in the primary database, copy 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: Section 5.8.6 and Section 6.4

5.8.7.2 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 the Oracle database server 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: V$DATAFILE in Chapter 10, "Fixed Views" and the *Oracle9i Database Reference* for more information about the V$DATAFILE view

5.8.8 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 deleting files.

Caution: Do this only if all archived logs have been applied to the standby database. Otherwise, the archived logs are lost when you create the new standby control file.

To refresh the standby database control file:

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

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

2. Shut down the standby database:

   ```
   SQL> SHUTDOWN IMMEDIATE;
   ```

3. Start a SQL session on the primary database and create the control file for the standby database:

   ```
   SQL> ALTER DATABASE CREATE STANDBY CONTROLFILE AS 'stbycf.dbf';
   ```

4. Copy 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 database 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> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE;
   ```

5.8.9 Clearing Online Redo Logs

After creating the standby database, you can clear online redo logs on the standby site to optimize performance by issuing the following statement, where 2 is the number of the log group:

```
SQL> ALTER DATABASE CLEAR LOGFILE GROUP 2;
```

This statement optimizes failover to the standby database because it is no longer necessary for the Oracle database server to clear the logs before failover. Clearing 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.

If you clear the logs manually, the Oracle database server realizes at activation that the logs already have zeros and skips the clearing 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, the Oracle database server clears the online logs automatically during failover.

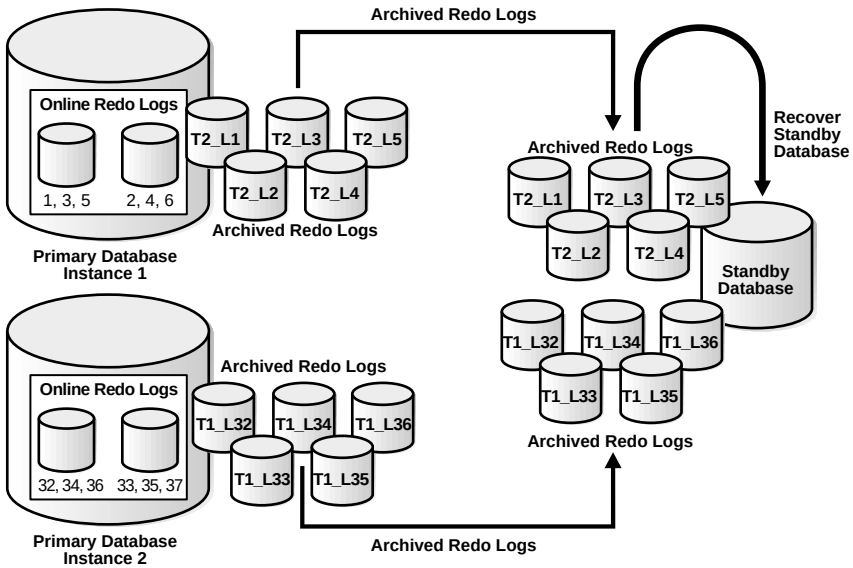
5.9 Standby Database with an Oracle Real Application Clusters Configuration

You can use a standby database to protect a primary database using Real Application Clusters. The following table describes the possible combinations of instances in the primary and standby databases:

Instance Combinations	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 instance of the primary database archives its own online redo logs to the standby database. For example, Figure 5–6 illustrates a Real Application Clusters database with two primary database instances (a multi-instance primary database) archiving redo logs to a single-instance standby database.

Figure 5–6 Archiving Redo Logs from a Multi-Instance Primary Database



In this case, Instance 1 of the primary database transmits logs 1, 2, 3, 4, 5 while Instance 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 a Real Application Clusters configuration, and the standby database is in managed recovery mode, then a single instance of the standby database applies all sets of logs transmitted by the primary instances. In this case, the standby instances that are *not* applying redo cannot be in read-only mode while managed recovery is in progress; in most cases, the non-recovery instances should be shut down, although they can also be mounted.

> **See Also:** *Oracle9i Real Application Clusters Installation and Configuration* for information about configuring a database for Real Application Clusters

This section contains the following topic:

- Setting Up a Cross-Instance Archival Database Environment

5.9.1 Setting Up a Cross-Instance Archival Database Environment

It is possible to set up a cross-instance archival database environment. Within a Real Application Cluster configuration, each instance directs its archived redo logs to a single instance of the cluster. This instance is called the **recovery instance** and is typically the instance where managed recovery is performed. This instance typically has a tape drive available for RMAN backup and restore support. Example 5–1 shows how to set up the LOG_ARCHIVE_DEST_*n* initialization parameters for archiving redo logs across instances. Execute this example on all instances except the recovery instance.

Example 5–1 Setting Destinations for Cross-Instance Archival

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_1 = 'LOCATION=archivelog MANDATORY REOPEN=120';
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_1 = enable;
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_2 = 'SERVICE=prmy1 MANDATORY REOPEN=300';
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2 = enable;
```

Destination 1 is the repository containing the local archived redo logs required for instance recovery. This is a mandatory destination. Because the expected cause of failure is lack of adequate disk space, the retry interval is 2 minutes. This should be adequate to allow the DBA to purge unnecessary archived redo logs. Notification of

destination failure is accomplished by manually searching the primary database alert log.

Destination 2 is the primary database on the instance where RMAN is used to back up the archived redo logs from local disk storage to tape. This is a mandatory destination, with a reconnect threshold of 5 minutes. This is the time needed to fix any network-related failures. Notification of destination failure is accomplished by manually searching the primary or standby database alert log.

Cross-instance archival is available using the ARC*n* process only. Using the LGWR process for cross-instance archival results in the RFS process failing and the archive log destination being placed in the Error state.

> **See Also:** Appendix D, "Standby Database Real Application Cluster Support"

6

Data Guard Scenarios

This chapter describes the following standby database scenarios:

6.1 Scenario 1: Creating a Standby Database on the Same Site

This scenario describes the creation of a standby database `standby1` on the same site as the primary database `primary1`. The site is a UNIX system 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-site standby database in managed recovery mode, you can keep it continuously up-to-date.

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

6.1.1 Step 1: Plan the Standby Database.

Because the site 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 system 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`

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.

6.1.2 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 is shut 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.dbf
/fs1/dbs/tbs_02.dbf
/fs1/dbs/tbs_11.dbf
/fs1/dbs/tbs_12.dbf
/fs1/dbs/tbs_21.dbf
/fs1/dbs/tbs_22.dbf
/fs1/dbs/tbs_13.dbf
/fs1/dbs/tbs_23.dbf
/fs1/dbs/tbs_24.dbf
/fs1/dbs/tbs_31.dbf
/fs1/dbs/tbs_32.dbf
/fs1/dbs/tbs_41.dbf
/fs1/dbs2/tbs_42.dbf
/fs1/dbs2/tbs_51.dbf
/fs1/dbs2/tbs_52.dbf
/fs1/dbs2/tbs_03.dbf
/fs1/dbs3/tbs_14.dbf
/fs1/dbs3/tbs_25.dbf
/fs1/dbs3/tbs_33.dbf
/fs1/dbs3/tbs_43.dbf
/fs1/dbs3/tbs_53.dbf
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=initPRIMARY1.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.ctl';
   ```

4. Copy 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 transfer of datafiles can take a long time, first copy the control file, copy 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.ctl /fs2/dbs/cf1.ctl
   % cp /fs1/temp/tbs* /fs2/dbs
   ```

6.1.3 Step 3: Configure Oracle Net.

To run a standby database in a Data Guard environment, you must configure an Oracle Net 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 site. Because you are using the local naming method, you must create new entries in the `tnsnames.ora` file. You must also add corresponding entries in the `listener.ora` file.

> **Note:** If, in the future, you choose to manage this standby database using the Data Guard broker, you must use the TCP/IP protocol instead of the IPC protocol. See Section 6.2.4 for an example using the TCP/IP protocol.

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 `primary1` database:

```
primary1 = (DESCRIPTION=
           (ADDRESS=(PROTOCOL=tcp) (PORT=1521) (HOST=dlsun183))
           (CONNECT_DATA=(SID=primary1))
)
```

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=1521) (HOST=dlsun183))
)

SID_LIST_LISTENER = (SID_LIST=
   (SID_DESC=(SID_NAME=primary1) (ORACLE_HOME=/fs3/oracle))
)
```

Currently, the listener is listening on port `1521` of the host `dlsun183` for database `primary1`.

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=/fs3/oracle)))
```

Now that you have edited the `listener.ora` file, you must start the listener:

```
% lsnrctl

LSNRCTL for Solaris: Version 9.0.0.0.0 - Development on 09-MAR-2001 14:13:40

Copyright (c) 1991, 2001, 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 Oracle Net Configuration Assistant graphical user interface to configure the network files.

See Also: *Oracle Net Services Administrator's Guide*

6.1.4 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 initialization parameter file looks as follows:

```
db_name=primary1
```

```
control_files=(/fs1/dbs/cf1.ctl,/fs1/dbs/cf2.ctl)
compatible=9.0.1.0.0
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 the Oracle database server can archive to the destination.

Parameter/Option	Meaning
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 the Oracle database server 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 the Oracle database server 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/initPRIMARY1.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.

6.1.5 Step 5: Configure the Standby Database Parameter File.

You know that the initialization parameters shown in Table 6–1 play a key role in the standby database recovery process, and decide to edit them.

Table 6–1 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 9.0.1.0.0 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	You must rename your standby datafile filenames to direct the standby database to correctly access its datafiles. A convenient way to do so is to use this parameter. If you have more than one set of files mapping from your primary datafiles to your standby datafiles, you need to specify multiple mapping pairs as values to this parameter. For this example, you decide to map primary datafiles from /fs1/dbs to /fs2/dbs, from /fs1/dbs2 to /fs2/dbs, and from /fs1/dbs3 to /fs2/dbs.
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 primary database parameter file. Consequently, you leave this parameter alone.
FAL_SERVER	This parameter specifies the net service name that the standby database should use to connect to the FAL server. You decide to set the name to primary1.
FAL_CLIENT	This parameter specifies the net service name that the FAL server should use to connect to the standby database. You decide to set the name to standby1.
LOCK_NAME_SPACE	This parameter 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 site. You decide to set the name to standby1.

Table 6–1 (Cont.) Configuring Standby Database Initialization Parameters

Parameter	Setting
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	You must rename your online log filenames on the standby database to allow correct access of its online log file. Even though the online log files are not used until you fail over to the standby database, it is good practice to rename your online log files. Much like the DB_FILE_NAME_CONVERT initialization parameter, you can specify more than one mapping pair as values to this parameter. For this example, you decide to map the primary online log file from /fs1/dbs to the standby online log file location /fs2/dbs.
STANDBY_ARCHIVE_DEST	The Oracle database server uses this value to create the name of the logs received from the primary site. You decide to set it to /fs2/arc_dest/.

Edit the standby database parameter file as follows:

```
db_name = primary1                #The same as PRMYinit.ora
control_files = (/fs2/dbs/cf1.ctl)
compatible = 9.0.1.0.0
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/arc_dest/
db_file_name_convert = ('/fs1/dbs','/fs2/dbs',
                        '/fs1/dbs2','/fs2/dbs',
                        '/fs1/dbs3','/fs2/dbs')
log_file_name_convert = ('/fs1/dbs','/fs2/dbs')
lock_name_space = standby1
fal_server=primary1
fal_client=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
```

6.1.6 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 AS sysdba
   SQL> STARTUP NOMOUNT PFILE=/fs3/oracle/dbs/initSTANDBY1.ora;
   SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
   ```

3. 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@primary1 AS sysdba
   SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_2 = 'SERVICE=standby1
     2> OPTIONAL REOPEN=180';
   SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2 = ENABLE;
   ```

6.1.7 Step 7: 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 that log apply services will wait until log transport services 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;
```

The standby database is now in managed recovery. When you enable managed recovery, log apply services automatically identify and resolve any archive gaps that may exist. As the primary database archives redo logs to the standby site, the standby database automatically applies them.

6.2 Scenario 2: Creating a Standby Database on a Remote Site

This scenario describes the creation of a standby database `standby1` on a remote site. 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 Data Guard environment.
- The primary database site name is `prmyhost`.
- The remote site 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.

6.2.1 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. 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
   ----------------------------------------------------------------------
   /oracle/dbs/dbf_1.dbf
   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):

```
% cp /oracle/dbs/dbf_1.dbf /backup
```

4. Reopen the primary database as follows:

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

6.2.2 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 ARCHIVELOG 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.ctl'
```

   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.

6.2.3 Step 3: Transfer the Datafiles and Control File to the Standby Site.

Transfer 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
```

6.2.4 Step 4: Configure Oracle Net.

This scenario assumes that the TCP/IP network protocol is used to connect to the primary and the standby databases. To achieve remote archiving of redo log files, Oracle Net must be set up correctly from the primary database to the standby database. To allow the standby database to fetch archive gaps from the primary database, Oracle Net must again be set up correctly from the standby database to the primary database. This step involves editing the following files:

- `tnsnames.ora` file on the primary and standby site
- `listener.ora` file on the primary and standby site

1. Configure the `tnsnames.ora` file on the primary site.

   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=1521) (HOST=stbyhost))
      (CONNECT_DATA=(SID=stdby1)))
   ```

2. Configure the `listener.ora` file on the standby site.

   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=1521) (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. You can either create a new listener or add a new address to an existing listener.

3. Configure the `tnsnames.ora` file on the standby site.

You need to add a new TNS name entry at the standby site to allow the standby database to connect to the primary database. The following entry is added:

```
primary1 =
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(PORT=1601)(HOST=prmyhost))
                     (CONNECT_DATA=(SID=prmy)))
```

4. Configure the `listener.ora` file on the primary site.

The following lines are added to the default `LISTENER` entry:

```
LISTENER = (
               ADDRESS_LIST=(
               .
               .
               .
               (ADDRESS=(PROTOCOL=tcp) (PORT=1601) (HOST=prmyhost)))
               .
               .
               .
          )
```

```
SID_LIST_LISTENER = (
                        SID_LIST= (
                        .
                        .
                        .
                        (SID_DESC=(SID_NAME=prmy) (ORACLE_HOME=/oracle)))
                        .
                        .
                        .
                    )
```

Make sure the SID_NAME in the `listener.ora` file matches the SID in the `tnsnames.ora` file in step 3. Also make sure the PORT and HOST values are the same in the two files.

> **See Also:** *Oracle Net Services Administrator's Guide* for detailed directions on using the Oracle Net Manager

6.2.5 Step 5: Start the Listener on the Primary and Standby Site.

Start the standby listener on the standby site. For example:

```
% lsnrctl start stdby1_listener
```

Normally, your default listener is started on your primary site. Restart the default listener on the primary database to pick up the new definitions. For example:

```
% lsnrctl stop
% lsnrctl start
```

6.2.6 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 primary site, issue a command similar to the following:

   ```
   % rcp /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.ctl

Parameter	Value
STANDBY_ARCHIVE_DEST	/fs2/oracle/stdby/
LOG_ARCHIVE_DEST_1	/fs2/oracle/stdby/
LOG_ARCHIVE_FORMAT	stdby_%t_%s
DB_FILE_NAME_CONVERT	('/oracle/dbs','/fs2/oracle/stdby')
LOG_FILE_NAME_CONVERT	('/oracle/dbs','/fs2/oracle/stdby')
LOG_ARCHIVE_START	true
FAL_SERVER	standby1
FAL_CLIENT	primary1

The STBYinit.ora file looks as follows:

```
#
#parameter file STBYinit.ora
#

db_name=primary1                    # The same as PRMYinit.ora

# The following parameter has changed from PRMYinit.ora
control_files=/fs2/oracle/stdby/stbycf.ctl

# 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=/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=('/oracle/dbs','/fs2/oracle/stdby')
log_file_name_convert=('/oracle/dbs','/fs2/oracle/stdby')
log_archive_start=true
log_archive_trace=127
fal_server=standby1
fal_client=primary1
```

6.2.7 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 Section 6.2.4 for information on how to configure the `tnsnames.ora` file.

6.2.8 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;
   ```

6.2.9 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
     2> MANDATORY REOPEN=60';
   SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;
   ```

6.2.10 Step 10: 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;
```

6.3 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 Dropping a Tablespace in the Primary Database
- Adding or Dropping Online Redo Logs
- Altering Control Files
- Refreshing the Standby Database Control File
- Physical Changes That Require You to Rebuild the Standby Database

6.3.1 Adding a Datafile to the Primary Database

To maintain consistency when you add a datafile to the primary database, you must also 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. The steps you perform depend on how the datafile was created on the primary database and how the initialization parameter STANDBY_FILE_MANAGEMENT is defined on the standby database.

If	Then
You create a new datafile on the primary database with the STANDBY_FILE_MANAGEMENT initialization parameter set to auto on the standby database	The new datafile is automatically created on the standby database.
If you create a new datafile on the primary database with the STANDBY_FILE_MANAGEMENT initialization parameter undefined or set to manual on the standby database	You must manually copy the new datafile to the standby database and re-create the standby control file.
You copy an existing datafile from another database to primary database	You must also copy the new datafile to the standby database and re-create the standby control file.

Section 6.3.1.1 illustrates the steps to enable the automatic creation of datafiles on the standby database. Section 6.3.1.2 illustrates the manual steps necessary if standby file management is not automatic.

6.3.1.1 Enabling the Automatic Creation of Datafiles on the Standby Database

When you create a new datafile on the primary database, the datafile is automatically created on the standby database if the initialization parameter STANDBY_FILE_MANAGEMENT is set to auto in the standby database initialization file.

The following steps describe how to set up your standby database to automate the creation of datafiles.

1. Add the following line to your standby database initialization file.

```
STANDBY_FILE_MANAGEMENT=auto
```

2. Shut down the standby database and restart it to pick up the changes in the initialization file and then restart managed standby.

```
SQL> SHUTDOWN;
SQL> STARTUP NOMOUNT pfile=initSTANDBY.ora
SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT;
```

   Alternatively, you can dynamically set the STANDBY_FILE_MANAGEMENT initialization parameter with an ALTER SYSTEM statement to avoid shutting down and restarting the instance. For example:

```
SQL> ALTER SYSTEM SET STANDBY_FILE_MANAGEMENT=auto;
```

3. Add a new tablespace to the primary database.

```
SQL> CREATE TABLESPACE new_ts datafile 't_db2.dbf'
  2> SIZE 1m AUTOEXTEND ON MAXSIZE UNLIMITED;
```

4. Archive the current redo log so it will get copied to the standby database.

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

5. Verify that the new datafile was added to the primary database.

```
SQL> SELECT name FROM v$datafile;
NAME
----------------------------------------------------------------------
/oracle/dbs/t_db1.dbf
/oracle/dbs/t_db2.dbf
```

6. Verify that the new datafile was added to the standby database.

```
SQL> SELECT name FROM v$datafile;
NAME
----------------------------------------------------------------------
/oracle/dbs/s2t_db1.dbf
/oracle/dbs/s2t_db2.dbf
```

6.3.1.2 Manually Creating New Datafiles on the Standby Database

The following steps illustrate how to manually create datafiles on the standby database when standby file management is manual.

1. Make sure managed recovery is caught up by comparing the log sequence numbers in the V$LOG fixed view on the primary database with the V$MANAGED_STANDBY fixed view on the standby database.

```
SQL> -- on the primary
SQL> SELECT SEQUENCE#, ARCHIVED, STATUS
  2> FROM v$log
  3> WHERE STATUS = 'CURRENT';

SEQUENCE# ARC STATUS
---------- --- ----------------
       123 NO  CURRENT

SQL> -- on the standby
SQL> SELECT PROCESS, STATUS, SEQUENCE#
  2> FROM v$managed_standby;

PROCESS STATUS        SEQUENCE#
------- ------------ ----------
MRP0    WAIT_FOR_LOG        123
```

2. Shut down the standby database.

```
SQL> SHUTDOWN;
```

3. Add a new tablespace to the primary database.

```
SQL> CREATE TABLESPACE new_ts DATAFILE 't_db2.dbf'
  2> SIZE 1m AUTOEXTEND ON MAXSIZE UNLIMITED;
```

4. Verify that the new datafile has been added to the primary database.

```
SQL> SELECT name FROM v$datafile;
NAME

--------------------------------------------------------------------------------
/oracle/dbs/t_db1.dbf
/oracle/dbs/t_db2.dbf
```

5. Set the new tablespace offline, copy the new datafile to the standby database with an operating system command, and then set the new tablespace back online.

```
SQL> ALTER TABLESPACE new_ts OFFLINE;

% cp t_db2.dbf s2t_db2.dbf
```

```
SQL> ALTER TABLESPACE new_ts ONLINE;
```

6. Re-create the standby control file and copy it to the standby database.

```
SQL> ALTER DATABASE CREATE STANDBY CONTROLFILE AS 'scf2.ctl' REUSE;

% rcp scf2.ctl <standby location>
```

7. Start the standby database and start managed recovery.

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

8. Archive the current redo log on the primary database so it will get copied to the standby database.

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

9. Verify that the datafile has been added on the standby database.

```
SQL> SELECT name FROM v$datafile;
NAME

----------------------------------------------------------------------
/oracle/dbs/s2t_db1.dbf
/oracle/dbs/s2t_db2.dbf
```

6.3.2 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.

> **Note:** Unlike adding or deleting datafiles, the STANDBY_FILE_MANAGEMENT initialization parameter has no effect when renaming datafiles.

1. To rename the datafile at the primary site, you must take the tablespace offline:

```
SQL> ALTER TABLESPACE tbs_4 OFFLINE;
```

2. Exit SQL and execute the following command to rename the datafile on the system:

```
% mv tbs_4.dbf tbs_x.dbf
```

3. Go back into SQL and execute the following statements to rename the datafile in the database and to bring the tablespace back online:

```
SQL> ALTER TABLESPACE tbs_4 RENAME DATAFILE 'tbs_4.dbf' TO 'tbs_x.dbf';
SQL> ALTER TABLESPACE tbs_4 ONLINE;
```

4. At the primary site, create the standby database control file by issuing the following statement:

```
SQL> ALTER DATABASE CREATE STANDBY CONTROLFILE AS 'stbycf.ctl';
```

5. Ensure that the standby database has applied all of the online redo logs by issuing the following statement:

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

6. Shut down the standby database with the `IMMEDIATE` option:

```
SQL> SHUTDOWN IMMEDIATE;
```

7. 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.ctl stbyhost:/fs2/oracle/stdby/
```

8. Rename the datafile at the standby site:

```
% mv tbs_4.dbf tbs_x.dbf
```

9. Start the standby database instance without mounting the database:

```
SQL> STARTUP NOMOUNT PFILE=STBYinit.ora;
```

10. Mount the standby database:

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

11. 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: '/oracle/dbs/stdby/tbs_x.dbf'
```

6.3.3 Deleting a Datafile or Dropping a Tablespace in the Primary Database

When you delete one or more datafiles or drop one or more tablespaces in the primary database, you also need to delete the corresponding datafiles or drop the corresponding tablespaces in the standby database. You also need to refresh the standby database control file.

1. Drop the tablespace at the primary site:

   ```
   SQL> DROP TABLESPACE tbs_4;
   % rm tbs_4.dbf
   ```

2. Refresh the standby database control file. See Section 6.3.6 for the steps.

3. Delete the corresponding datafile at the standby site:

   ```
   % rm tbs_4.dbf
   ```

If the STANDBY_FILE_MANAGEMENT initialization parameter is set to auto, dropped tablespaces and deleted datafiles from the primary database are automatically dropped or deleted from the standby database. However, if the STANDBY_FILE_MANAGEMENT initialization parameter is set to manual, or if the parameter is not defined, you must manually drop tablespaces or delete datafiles from the standby database and re-create the standby control file.

6.3.4 Adding or Dropping Online Redo Logs

One method of tuning available to the database administrator (DBA) is changing the size and number of online redo logs. Consequently, when you add or drop an online redo log at the primary site, it is important that you refresh the standby database control file.

1. Add or drop 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 Section 6.3.6 for the steps.

6.3.5 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, datafiles, 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 primary1 NORESETLOGS
  2>    LOGFILE  'prmy1.log' SIZE 100K, 'prmy2.log' SIZE 100K
  3>    DATAFILE 'dbf_1.dbf' SIZE 10M MAXLOGFILES 52 ARCHIVELOG;
```

2. Refresh the standby database control file. See Section 6.3.6 for the steps.

Note: 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.

6.3.6 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
- Create a temporary tablespace
- Drop a tablespace
- Add one or more online redo logs
- Remove one or more online redo logs

- Alter control files

Whether you must refresh the control file after dropping a tablespace depends on the setting of the STANDBY_FILE_MANAGEMENT initialization parameter. If the parameter is not set, or it is set to manual, you must refresh the control file after dropping a tablespace. If the parameter is set to auto, you do not have to refresh the control file after dropping a tablespace, as this is done automatically.

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.ctl';
   ```

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.ctl 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. 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: '/oracle/dbs/stdby/tbs_1.dbf'
```

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 archive gap have not been successfully applied. If you receive these errors, manually resolve the gaps and repeat step 7.

See Also: Section B.3 for information on resolving an archive gap

6.3.7 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

See Also: Section 2.3

6.4 Scenario 4: Recovering After the NOLOGGING Clause Is 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 copied 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: '/oracle/dbs/stdby/tbs_1.dbf'
ORA-26040: Data block was loaded using the NOLOGGING option
```

To recover after the NOLOGGING clause is 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                                               UNRECOVERABLE
   -------------------------------------------------- -------------
   /oracle/dbs/tbs_1.dbf                                       5216
   /oracle/dbs/tbs_2.dbf                                          0
   /oracle/dbs/tbs_3.dbf                                          0
   /oracle/dbs/tbs_4.dbf                                          0
   4 rows selected.
   ```

   b. Issue the following query in the standby database:

   ```
   SQL> SELECT name, unrecoverable_change# FROM v$datafile;
   NAME                                               UNRECOVERABLE
   -------------------------------------------------- -------------
   /oracle/dbs/stdby/tbs_1.dbf                                 5186
   /oracle/dbs/stdby/tbs_2.dbf                                    0
   /oracle/dbs/stdby/tbs_3.dbf                                    0
   /oracle/dbs/stdby/tbs_4.dbf                                    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.dbf` datafile is greater, so you need to copy the `tbs_1.dbf` 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.dbf /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.ctl;
   ```

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. 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: '/oracle/dbs/stdby/tbs_1.dbf'
   ```

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 archive gap have not been successfully applied. If you receive these errors, manually resolve the gaps and repeat step 8.

See Also: Section B.3 for information on resolving an archive gap

6.5 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 logs are archived to the mandatory archive destination. After you apply the archived redo logs 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 primary database with minimum loss of data.	In some cases, (such as network failure), the archived redo logs cannot reach the mandatory archive destination, causing the archiving of the redo log to stop. In the worst case, if all online redo logs 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.
		See Section 6.8.

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.	An archive gap may cause data loss because archive logs that are required to be applied to the standby database are unavailable. This results in the managed recovery operation terminating before all primary data has been applied to the standby database.

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 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 Section 6.10 for a description of this type of configuration.
5	San Francisco	Optional	This standby site receives archived redo logs, and applies them after an 8-hour time lag. See Section 6.11 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 site 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. This ReadMe file is created and maintained by the DBA and 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
- Perform the standby database failover operation
- 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	Oracle9i	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 system) 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-------------
```

6.6 Scenario 6: Switching Over a Primary Database to a Standby Database

Reversing the roles of a primary and standby database is sometimes referred to as a switchover or a switchback operation. The following steps outline what commands must be issued to perform a switchover operation.

For this discussion, boston is initially the primary database and la is initially the standby database. During the following switchover scenario, la will become the primary database and boston will become the standby database.

This scenario assumes that the primary and standby databases have been previously created and initialized.

See Also: Section 5.3.3, Section 6.1, and Section 6.2

6.6.1 Step 1: End Read or Update Activity on the Primary and Standby Databases.

Exclusive database access is required by the DBA before beginning a switchover operation. Ask users to log off the primary and standby databases or query the V$SESSION view to identify users that are connected to the databases and close all open sessions except the SQL*Plus session from which you are going to issue the switchover command.

See Also: *Oracle9i Database Administrator's Guide* for more information on managing users

6.6.2 Step 2: Prepare the Primary Database for Switchover.

On the primary database, boston, execute the following statement:

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO STANDBY;
```

This statement does the following:

- Closes the primary database
- Archives any unarchived log files and applies them to the standby database, la
- Adds an end-of-redo marker to the header of the last log file being archived
- Creates a backup of the current control file
- Converts the current control file into a standby control file

6.6.3 Step 3: Shut Down and Start Up the Former Primary Instance Without Mounting the Database.

Execute the following statement on boston:

```
SQL> SHUTDOWN NORMAL;
```

```
SQL> STARTUP NOMOUNT;
```

6.6.4 Step 4: Mount the Former Primary Database in the Standby Database Role.

Execute the following statement on `boston`:

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

6.6.5 Step 5: Prepare the Former Standby Database to Switch to the Primary Database Role.

Execute the following statement on `la`:

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY;
```

This statement does the following:

- Makes sure the last log file has been received and applied through the `end-of-redo` marker.
- Closes the database if it has been opened for read-only transactions
- Converts the standby control file to the current control file

6.6.6 Step 6: Shut Down the Database.

Execute the following statement on `la`:

```
SQL> SHUTDOWN;
```

6.6.7 Step 7: Start Up the Database in the Primary Role.

Execute the following statement on `la`:

```
SQL> STARTUP;
```

6.6.8 Step 8: Put the Standby Database in Managed Recovery Mode.

Execute the following statement on the standby database, `boston`, to place it in managed recovery mode:

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

6.6.9 Step 9: Start Archiving Logs from the Primary Database to the Standby Database.

Execute the following statement on the primary database, `la`:

```
SQL> ALTER SYSTEM ARCHIVE LOG START;
SQL> ALTER SYSTEM SWITCH LOGFILE;
```

6.7 Scenario 7: Configuring Client Application Failover

When a standby database is activated, it becomes a primary database, and is no longer capable of serving as a standby database. Client applications must redirect their connections from the original primary database to the newly activated primary 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

6.7.1 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 Also:** *Oracle Net Services 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 primary 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 primary database. You must establish the connection to the `ProductDB` database again.

6.7.2 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 Also:** *Oracle Net Services 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 Section 6.7.1 also applies to the Oracle Names server configuration.

6.7.3 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 Also:** *Oracle Net Services Administrator's Guide* 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 primary 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.

6.7.4 Manual Network Configuration

Instead of setting the configurations so that the client can automatically fail over to the new primary 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 primary database to the newly activated primary 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 primary 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 primary database will be sent to the new primary 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 primary 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 primary 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 must be replaced with the new settings.

6.8 Scenario 8: 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 an archive gap for the time when the network was down. The archive gap is automatically detected and resolved when managed recovery mode is enabled (when FAL_SERVER and FAL_CLIENT are defined in the initialization parameter file).

See Also: Section 4.5 for information on archive gaps

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.

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.

This scenario describes how to recover after a network failure.

6.8.1 Step 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 for the archived log destination that experienced the network failure. For example:

```
SQL> SELECT DEST_ID, STATUS, ERROR FROM V$ARCHIVE_DEST WHERE DEST_ID = 2;

DEST_ID    STATUS
---------- ---------
ERROR
----------------------------------------------------------------
         2  ERROR
ORA-12224: TNS:no listener
```

The query results show there are errors archiving to the standby database, and the cause of 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.

6.8.2 Step 2: Prevent the Primary Database from Stalling.

If you cannot solve the network problem quickly, and if the standby database is specified as a mandatory destination, 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
    2> 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
  2> 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.

6.8.3 Step 3: Archive the Current Redo Log.

On the primary database, archive the current redo log:

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

When the network is back up again, log apply services can detect and resolve the archive gaps automatically if you place the standby database in managed recovery mode.

6.9 Scenario 9: 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 primary database. After a period of time, you decide you want to fail back to the original primary system and make the primary database the standby database again.

6.9.1 Step 1: Create a Standby Database at the Original Primary Site.

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
CONTROL_FILES	/fs2/oracle/stdby/cf1.ctl
LOCK_NAME_SPACE	fallback
LOG_ARCHIVE_FORMAT	r_%t_%s.arc
STANDBY_ARCHIVE_DEST	/oracle/dbs/
LOG_ARCHIVE_DEST_1	'LOCATION=/oracle/dbs/'
DB_FILE_NAME_CONVERT	('/fs2/oracle/stdby','/oracle/dbs')
LOG_FILE_NAME_CONVERT	('/fs2/oracle/stdby','/oracle/dbs')

The `fallback.ora` file looks as follows:

```
#
#parameter file fallback.ora
#

db_name=primary1                #The same as PRMYinit.ora

control_files=/fs2/oracle/stdby/cf1.ctl
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=/oracle/work/tkinit.ora

# specific parameters for standby database
log_archive_format = r_%t_%s.arc
standby_archive_dest=/oracle/dbs/
log_archive_dest_1='LOCATION=/oracle/dbs/'
log_archive_dest_state_1 = ENABLE
```

```
db_file_name_convert=('/fs2/oracle/stdby','/oracle/dbs')
log_file_name_convert=('/fs2/oracle/stdby','/oracle/dbs')
log_archive_start=true
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 initialization parameters.

   Note: Section 6.2.7 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 primary database datafiles.
5. On the original standby site, create the standby database control file.
6. Copy the primary database datafiles 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 primary 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 primary database datafiles were backed up from the original standby site to the original primary site, as follows:

   ```
   % rcp /fs2/oracle/stdby/stdby_1_102.arc prmyhost:/oracle/dbs/r_1_102.arc
   % rcp /fs2/oracle/stdby/stdby_1_103.arc prmyhost:/oracle/dbs/r_1_103.arc
   ```

6.9.2 Step 2: Fail Over to the Standby Database at the Original Primary Site.

1. 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;
   ```

2. Fail over to the standby database:

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

3. Start the primary database at the original primary site:

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

4. Configure the network settings to enable client applications to access the primary database.

6.9.3 Step 3: Create a New Standby Database at the Original Standby Site.

Create a new standby database at the original standby site.

See Also: Section 6.2 for the steps to create a standby database

6.10 Scenario 10: Standby Database with No Ongoing Recovery

This scenario describes what the DBA needs to do when archived online redo logs are automatically copied 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 site running a Web server. The workload for the site is very heavy, and you do not want to add workload to the site by automatically applying redo logs as they arrive from the primary site. You decide to let the site 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 switch over the standby database as a primary 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. Section 6.2 describes 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 Section 6.2. However, in Section 6.2, 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.

6.10.1 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:

- Copying a Datafile to the Primary Database
- Responding to a Change in the Primary Database Control File
- Responding When the NOLOGGING Clause Is Specified

6.10.1.1 Copying a Datafile to the Primary Database

When a datafile is copied 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: '/oracle/dbs/stdby/tbs_4.dbf'
```

  Create the correponding new datafile by issuing the following statement:

```
SQL> ALTER DATABASE CREATE DATAFILE '/oracle/dbs/stdby/tbs_4.dbf';
```

> **Note:** When you create a datafile in the primary database, you can set up your environment so that a corresponding datafile is also created in the standby database. See Section 6.3.1.1.

6.10.1.2 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 Also: Section 6.3.6

6.10.1.3 Responding When the NOLOGGING Clause Is Specified

To recover after the NOLOGGING clause is 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 Also: Section 6.4

If you can afford to lose the changes incurred by specifying NOLOGGING at the primary site, you may choose to do nothing.

6.10.2 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. Query the V$ARCHIVE_GAP view on the standby database as follows:

   ```
   SELECT THREAD#, LOW_SEQUENCE#, HIGH_SEQUENCE#
   FROM V$ARCHIVE_GAP;

   THREAD#    LOW_SEQUENCE#   HIGH_SEQUENCE#
   ---------- -------------   --------------
            1            90               92
   ```

   The archive gap is the LOW_SEQUENCE# to the HIGH_SEQUENCE#. In this example, the gap is 90, 91, and 92 for thread 1. If no gap is selected in this step, go to step 5.

2. Identify the archived redo logs that need to be copied from the primary site to the standby site.

   Obtain the filenames of the logs in the archive gap by performing a query on the V$ARCHIVED_LOG view on the primary database as follows:

   ```
   SELECT name FROM V$ARCHIVED_LOG
   ```

```
WHERE thread#=1 AND sequence#<=92 AND sequence#>=90;

NAME
----------------------------------------
/oracle/dbs/r_1_90.arc
/oracle/dbs/r_1_91.arc
/oracle/dbs/r_1_92.arc
```

3. Copy the logs in the archive gap from the primary database to the standby database as follows:

```
% rcp /oracle/dbs/r_1_90.arc stbyhost:/fs2/oracle/stdby/stdby_1_90.arc
% rcp /oracle/dbs/r_1_91.arc stbyhost:/fs2/oracle/stdby/stdby_1_91.arc
% rcp /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 archive gap:

```
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 primary instance.

   If necessary, build the parameter file for the new primary 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;
```

6.11 Scenario 11: Standby Database with a Time Lag

By default, 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 primary database.

To create a standby database with a time lag, use the `DELAY` attribute of the `LOG_ARCHIVE_DEST_`*n* initialization parameters in the primary database initialization parameter file. The archived redo log files are still automatically copied from the primary site to the standby site, but the log files are not immediately applied to the standby database. The log files are applied when the specified time interval has expired.

This scenario use a 4-hour time lag 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.

See Also: Section 6.2 for details of normal standby database setup

6.11.1 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. Copy 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` parameter.

   You must use the directory specified by the `STANDBY_ARCHIVE_DEST` parameter when performing managed recovery.

   For example, assume you have set the parameter to the following value:

   ```
   STANDBY_ARCHIVE_DEST=/fs2/oracle/stdby_log/
   ```

6. Mount the standby database.
7. Configure the primary initialization parameter file.

   Edit the `LOG_ARCHIVE_DEST_`*n* initialization parameter to include the `DELAY` keyword.

   For example, to specify a 4-hour delay, set the parameter as follows:

   ```
   LOG_ARCHIVE_DEST_2 DELAY=240
   ```

   The `DELAY` attribute indicates that the archived redo logs at the standby site are not available for recovery until the 4-hour time interval has expired. The time interval (expressed in minutes) starts when the archived redo logs are successfully transmitted to the standby site.

6.11.2 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:

- Define the `STANDBY_FILE_MANAGEMENT` initialization parameter in the primary database initialization parameter file to ensure that when a datafile is created on the primary site, it is also created on the standby site. Section 6.3.1.1 shows the step-by-step procedure for automatically creating a datafile in the standby database.

- When a datafile from another database is copied to the primary site, you need to copy the datafile to the standby site. Section 6.3.1.2 explains this procedure.
- Before you refresh the standby database control file, you must apply all of the archived redo logs that have been copied 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 must wait for the specified time lag (for example, 4 hours) to reinforce the lag.

- To recover after the `NOLOGGING` clause is 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 archive gap.

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.

6.11.3 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 primary database whose status is a specified time (for example, 4 hours) before the current primary 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 primary instance:

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

6.11.4 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. Apply all of the archived redo logs:

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

2. Activate the standby database by issuing the following statement:

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

3. Shut down the standby database instance:

```
SQL> SHUTDOWN IMMEDIATE;
```

4. Start the new primary instance:

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

6.12 Scenario 12: Using a Standby Database to Back Up the Primary Database

This scenario describes using a physical standby database to back up the primary database. This allows the standby site to offload the task of backup at the primary site. The backup at the standby site can be done while the standby database is in the managed recovery mode. When the primary database needs to be restored, you can use the backup created at the standby site.

6.12.1 Step 1: Back Up the Standby Database.

The standby database can be used to back up the datafiles and the archived redo logs. The primary control file must be backed up at the primary site because the standby database uses a different standby control file. Because the control file is usually much smaller than the datafiles, backing up the control file at the primary site does not significantly affect the performance of the primary database.

To back up a database at the standby site:

1. At the standby site, start the Recovery Manager utility (RMAN) with the `NOCATALOG` option to back up the standby datafiles and log files. Assume that `stby` is the connect string of the standby database. For example:

```
% rman target sys/change_on_install@stby nocatalog

connected to target database: ADE3 (DBID=1417165739)
using target database controlfile instead of recovery catalog
```

2. Back up the standby datafiles. You can perform the backup while the standby database is still in managed recovery mode.

```
RMAN> run {
2> allocate channel c1 type disk;
3> backup database;
4> }

allocated channel: c1
```

```
channel c1: sid=13 devtype=DISK

Starting backup at 07-NOV-00
channel c1: starting full datafile backupset
channel c1: specifying datafile(s) in backupset
input datafile fno=00001 name=/oracle/dbs/s3t_db1.dbf
channel c1: starting piece 1 at 07-NOV-00
channel c1: finished piece 1 at 07-NOV-00
piece handle=/oracle/dbs/04c9t2ki_1_1 comment=NONE
channel c1: backup set complete, elapsed time: 00:00:35
Finished backup at 07-NOV-00

Starting Control File Autobackup at 07-NOV-00
warning - controlfile is not current, controlfile autobackup skipped
Finished Control File Autobackup at 07-NOV-00
released channel: c1
```

3. Back up the archived redo logs.

After you back up a database, Oracle Corporation recommends that you record the current sequence number and thread number applied at the standby site. This is important because the archived logs, after this backup, may be needed to restore the database at the primary site if the primary database loses its local archived logs.

For example, suppose we have only one thread and the sequence number at the time of database backup is 15. After several new logs are archived to the standby site, you can back up these archived logs using the following commands:

```
RMAN> run {
2> allocate channel c1 type disk;
3> backup archivelog from logseq 15;
4> }

allocated channel: c1
channel c1: sid=9 devtype=DISK

Starting backup at 07-NOV-00
channel c1: starting archive log backupset
channel c1: specifying archive log(s) in backup set
input archive log thread=1 sequence=15 recid=15 stamp=413043150
input archive log thread=1 sequence=16 recid=16 stamp=413043168
input archive log thread=1 sequence=17 recid=17 stamp=413043433
input archive log thread=1 sequence=18 recid=18 stamp=413043442
```

```
input archive log thread=1 sequence=19 recid=19 stamp=413043450
input archive log thread=1 sequence=20 recid=20 stamp=413043454
channel c1: starting piece 1 at 07-NOV-00
channel c1: finished piece 1 at 07-NOV-00
piece handle=/oracle/dbs/05c9t3c9_1_1 comment=NONE
channel c1: backup set complete, elapsed time: 00:00:03
Finished backup at 07-NOV-00

Starting Control File Autobackup at 07-NOV-00
warning - controlfile is not current, controlfile autobackup skipped
Finished Control File Autobackup at 07-NOV-00
released channel: c1
```

4. View the **backup set** using the `list` command:

```
RMAN> list backup;

List of Backup Sets
===================

BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ---------------
4       Full    104M       DISK        00:00:29     07-NOV-00
        BP Key: 4   Status: AVAILABLE   Tag:
        Piece Name: /oracle/dbs/04c9t2ki_1_1
  List of Datafiles in backup set 4
  File LV Type Ckp SCN    Ckp Time  Name
  ---- -- ---- ---------- --------- ----
  1       Full 73158      07-NOV-00
/oracle/dbs/s3t_db1.dbf

BS Key  Device Type Elapsed Time Completion Time
------- ----------- ------------ ---------------
5       DISK        00:00:02     07-NOV-00
        BP Key: 5   Status: AVAILABLE   Tag:
        Piece Name: /oracle/dbs/05c9t3c9_1_1

  List of Archived Logs in backup set 5
  Thrd Seq     Low SCN    Low Time  Next SCN   Next Time
  ---- ------- ---------- --------- ---------- ---------
  1    15      73139      07-NOV-00 73156      07-NOV-00
  1    16      73156      07-NOV-00 73158      07-NOV-00
  1    17      73158      07-NOV-00 73163      07-NOV-00
  1    18      73163      07-NOV-00 73165      07-NOV-00
  1    19      73165      07-NOV-00 73166      07-NOV-00
```

```
  1    20      73166      07-NOV-00 73167      07-NOV-00
```

5. You can also use the RMAN `BACKUP TABLESPACE` and `BACKUP DATAFILE` commands to back up individual tablespaces and datafiles.

6.12.2 Step 2: Restore the Backup at the Primary Site.

To restore the backup at the primary site, take the following steps:

1. Set up a **recovery catalog database** on the primary site.

   See Also: *Oracle9i Recovery Manager User's Guide* for instructions on setting up a catalog database

2. Ensure that the primary database is in the mounted state.
3. Move the standby control file and the **backup pieces** to the primary site.

   Note: If the backup is on disk, the backup pieces must reside under the same directory at the primary site as they do at the standby site. This is a current restriction of RMAN. This is not an issue if the backup is on tape, which is the preferred method.

4. Resynchronize the catalog database with the standby control file.

   Assume that `prmy` is the connect string of the primary database, and `rcat` is the connect string of the recovery catalog database.

   ```
   % rman target sys/change_on_install@prmy catalog rman/rman@rcat

   connected to target database: ADE3 (DBID=1417165739)
   connected to recovery catalog database

   RMAN> resync catalog from controlfilecopy 'scf3.ctl';
   ```

5. List the backup set obtained by the catalog database after the resynchronization:

   ```
   RMAN> list backup;

   starting full resync of recovery catalog
   full resync complete

   List of Backup Sets
   ===================
   ```

```
BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ---------------
182     Full    104M       DISK        00:00:29     07-NOV-00
        BP Key: 184   Status: AVAILABLE   Tag:
        Piece Name: /oracle/dbs/04c9t2ki_1_1
  List of Datafiles in backup set 182
  File LV Type Ckp SCN    Ckp Time  Name
  ---- -- ---- ---------- --------- ----
  1       Full 73158      07-NOV-00 /oracle/dbs/t_db1.dbf

BS Key  Device Type Elapsed Time Completion Time
------- ----------- ------------ ---------------
183     DISK        00:00:02     07-NOV-00
        BP Key: 185   Status: AVAILABLE   Tag:
        Piece Name: /oracle/dbs/05c9t3c9_1_1

  List of Archived Logs in backup set 183
  Thrd Seq     Low SCN    Low Time  Next SCN   Next Time
  ---- ------- ---------- --------- ---------- ---------
  1    15      73139      07-NOV-00 73156      07-NOV-00
  1    16      73156      07-NOV-00 73158      07-NOV-00
  1    17      73158      07-NOV-00 73163      07-NOV-00
  1    18      73163      07-NOV-00 73165      07-NOV-00
  1    19      73165      07-NOV-00 73166      07-NOV-00
  1    20      73166      07-NOV-00 73167      07-NOV-00
```

6. Restore the primary database:

```
RMAN> run {
2> allocate channel c1 type disk;
3> restore database;
4> }

allocated channel: c1
channel c1: sid=11 devtype=DISK

Starting restore at 07-NOV-00

channel c1: starting datafile backupset restore
channel c1: specifying datafile(s) to restore from backup set
restoring datafile 00001 to /oracle/dbs/t_db1.dbf
channel c1: restored backup piece 1
piece handle=/oracle/dbs/04c9t2ki_1_1 tag=null
params=NULL
```

```
channel c1: restore complete
Finished restore at 07-NOV-00
released channel: c1
```

7. Restore the archived log.

After restoring the database, you can try to open the database:

```
SQL> ALTER DATABASE OPEN;
alter database open
*
ERROR at line 1:
ORA-01113: file 1 needs media recovery
ORA-01110: data file 1: '/oracle/dbs/t_db1.dbf'
```

The ORA-01113 error indicates that media recovery is required on the database before you can open it. Issue the manual recovery statement, RECOVER DATABASE, as follows:

```
SQL> RECOVER DATABASE;
ORA-00279: change 73158 generated at 11/07/2000 14:12:47 needed for
thread 1
ORA-00289: suggestion : /oracle/dbs/db11_17.dbf
ORA-00280: change 73158 for thread 1 is in sequence #17

Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
```

The resulting error messages indicate that the database needs log sequence #17 for recovery. You can provide logs in several ways:

- If the primary database did not lose its local archived logs, then you can use these log files.
- You can move the standby archived logs back to the primary database. In this case, some renaming may be necessary.
- If you already backed up the archived log at the standby site, you can move the backup piece to the primary site, and restore these archived logs, as shown in the following example:

```
RMAN> run {
2> allocate channel c1 type disk;
3> restore archivelog from logseq 15;
4> }

allocated channel: c1
channel c1: sid=11 devtype=DISK
```

```
Starting restore at 07-NOV-00

channel c1: starting archive log restore to default destination
channel c1: restoring archive log
archive log thread=1 sequence=15
channel c1: restoring archive log
archive log thread=1 sequence=16
channel c1: restoring archive log
archive log thread=1 sequence=17
channel c1: restoring archive log
archive log thread=1 sequence=18
channel c1: restoring archive log
archive log thread=1 sequence=19
channel c1: restoring archive log
archive log thread=1 sequence=20
channel c1: restored backup piece 1
piece handle=/oracle/dbs/05c9t3c9_1_1 tag=null
params=NULL
channel c1: restore complete
Finished restore at 07-NOV-00
released channel: c1
```

After restoring the archived logs, you can recover the database and then open it.

Part II

Reference

This part provides reference material to be used in conjunction with the Oracle Data Guard standby database features. For more complete reference material, refer to the Oracle9*i* documentation set.

This part contains the following chapters:

7

Initialization Parameters

This chapter describes Oracle initialization parameters for each Oracle instance, including the primary database and each standby database in the Data Guard environment.

The members of a standby configuration are transactionally consistent copies of the primary member. The remote archival functionality of the standby feature provides the mechanism by which the redo log files of the primary database are transported to the designated standby sites. Redo logs generated by the primary database are transported and applied to the standby databases.

All initialization parameters are contained in an initialization file, the name of which is a variation of `init.ora` depending on your operating system. As an alternative to specifying parameters in the initialization file, you can modify dynamic parameters at runtime using the `ALTER SYSTEM SET` or `ALTER SESSION SET` statements.

> **See Also:** *Oracle9i Database Reference* and your Oracle operating system-specific documentation for more information about setting initialization parameters

Table 7–1 shows all of the initialization parameters that you need to implement a Data Guard environment. It describes whether a parameter applies to the primary database role, the standby database role, or both.

Table 7–1 Initialization Parameters Specific to the Oracle9i Data Guard Environment

Parameter	Description	For More Information
ARCHIVE_LAG_TARGET	Applies to the primary database role. Limits the amount of data that can be lost and effectively increases the availability of the standby database by forcing a log switch after the amount of time you specify (in seconds) elapses. That way, the standby database will not miss redo logs generated from a time range longer than a value of the ARCHIVE_LAG_TARGET parameter.	See *Oracle9i Database Reference*
COMPATIBLE	Applies to the primary and standby database roles. Always set the same value at the primary and standby databases. If different, you may not be able to archive the redo logs from your primary database to the standby database.	See *Oracle9i Database Reference*
CONTROL_FILES	Applies to the primary and standby database roles. Uniquely names the database control file. Always set this parameter to a different value from the CONTROL_FILES parameter in the primary database. The names of the control files for the standby database must exist at the standby site.	See *Oracle9i Database Reference*
CONTROL_FILE_RECORD_KEEP_TIME	Applies to the standby database role. Prevents reusing archived redo logs for a specified period of days. Setting this parameter prevents the ARCHIVELOG mechanism from overwriting information in the archived redo logs.	See Section 3.4.1.3
DB_FILE_NAME_CONVERT	Applies to the standby database role. Set to distinguish standby datafile filenames from primary datafile filenames. This parameter must be set on all standby databases.	See Section 4.6
DB_FILES	Applies to the primary and standby database roles. Specifies the maximum number of database files that can be open for this database.	See *Oracle9i Database Reference*

Table 7–1 (Cont.) Initialization Parameters Specific to the Oracle9i Data Guard Environment

Parameter	Description	For More Information
DB_NAME	Applies to the primary and standby database roles. Specifies a database identifier of up to 8 characters. Set it to the same value in the standby and primary initialization files.	See *Oracle9i Database Reference*
FAL_CLIENT	Applies to the standby database role. Assigns the FAL (fetch archive log) client name used by the FAL server to refer to the FAL client. This is the Oracle Net service name used by the primary database to refer to the standby database.	See Section 4.5
FAL_SERVER	Applies to the standby database role. Assigns the FAL (fetch archive log) server for the standby database.	See Section 4.5
LOCK_NAME_SPACE	Applies to the standby database role. Specifies the name space that the distributed lock manager (DLM) uses to generate lock names. Set this value if the standby database has the same name on the same cluster as the primary database.	See *Oracle9i Database Reference*
LOG_ARCHIVE_DEST_1[1]	Applies to the standby database role. Specifies the location of the archived logs for a standby database in recovery mode.	See Section 3.6.3.4
LOG_ARCHIVE_DEST_*n*[1]	Applies to the primary and standby database roles. Defines a destination and attributes for the archived redo log file group.	See Section 3.3.1.1
LOG_ARCHIVE_DEST_STATE_*n*	Applies to the primary and standby database roles. Specifies the state of the destination specified by the LOG_ARCHIVE_DEST_*n* parameters.	See Section 3.3.1.2

Table 7–1 (Cont.) Initialization Parameters Specific to the Oracle9i Data Guard Environment

Parameter	Description	For More Information
LOG_ARCHIVE_FORMAT	Applies to the primary and standby database roles. Indicates the format for filenames of log files. LOG_ARCHIVE_FORMAT and STANDBY_ARCHIVE_DEST are used to construct the fully qualified filenames for the archived redo logs.	See Section 3.6.3.4
LOG_ARCHIVE_MAX PROCESSES	Applies to the primary and standby database roles. Specifies the number of archiver processes to be invoked. This value is evaluated at instance startup if the LOG_ARCHIVE_START parameter has the value true; otherwise, this parameter is evaluated when the archiver process is invoked.	See *Oracle9i Database Reference*
LOG_ARCHIVE_MIN_SUCCEED_DEST	Applies to the primary and standby database roles. Defines the minimum number of destinations that must succeed in order for the online log file to be available for reuse.	See Section 3.4.2.4
LOG_ARCHIVE_START	Applies to the primary and standby database roles. Enables automatic archiving of filled groups each time an instance is started when LOG_ARCHIVE_START is set to true. To disable the automatic archiving of filled online redo log groups each time a database instance is started, set the LOG_ARCHIVE_START initialization parameter of a database's initialization parameter file to false.	See *Oracle9i Database Administrator's Guide*
LOG_ARCHIVE_TRACE	Applies to the primary and standby database roles. Optionally, set this parameter to an integer value to see the progression of the archiving of redo logs to the standby site. The Oracle database server writes an audit trail of the archived logs received from the primary database into a trace file.	See Section 4.9.5

Table 7–1 (Cont.) Initialization Parameters Specific to the Oracle9i Data Guard Environment

Parameter	Description	For More Information
LOG_FILE_NAME_CONVERT	Applies to the standby database role. 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.	See Section 4.6
REMOTE_ARCHIVE_ENABLE	Applies to the primary and standby database roles. Always set this parameter to `true` at the primary and standby databases in the Data Guard environment. This allows the standby database to receive redo logs for archiving from the primary database.	See Section 3.4.2
SORT_AREA_SIZE	Applies to the primary and standby database roles. Set this parameter to execute the `SELECT * FROM V$PARAMETER` statement when the database is not mounted, is mounted, or is open.	See Section 4.8.3

Table 7–1 (Cont.) Initialization Parameters Specific to the Oracle9i Data Guard Environment

Parameter	Description	For More Information
STANDBY_ARCHIVE_DEST	Applies to the standby database role. Defines the standby database destination for the archived redo log file group. STANDBY_ARCHIVE_DEST and LOG_ARCHIVE_FORMAT are used to construct the fully qualified filenames for the archived redo logs.	See Section 3.6.3.4
STANDBY_FILE_MANAGEMENT	Applies to the primary and standby database roles. When set to auto, this parameter automates the creation and deletion of datafile filenames on the standby site using the same filenames as the primary site. Use this parameter with the DB_FILE_NAME_CONVERT initialization parameter to automate the process of creating files with identical filenames on the standby database and primary database.	See Section 4.6
USER_DUMP_DEST	Applies to the primary and standby database roles. Determines the location of trace files for a database.	See Section 4.9.5

[1] The standby database assumes that the archived log file group is in the location specified by either the LOG_ARCHIVE_DEST or the LOG_ARCHIVE_DEST_n parameters in the standby initialization parameter file; both parameters cannot be specified.

8

LOG_ARCHIVE_DEST_n Parameters Attributes

This chapter provides syntax, values, and information on validity for attributes of the LOG_ARCHIVE_DEST_*n* initialization parameters. These attributes include:

- AFFIRM and NOAFFIRM
- ALTERNATE and NOALTERNATE
- ARCH and LGWR
- DELAY and NODELAY
- DEPENDENCY and NODEPENDENCY
- LOCATION and SERVICE
- MANDATORY and OPTIONAL
- MAX_FAILURE and NOMAX_FAILURE
- QUOTA_SIZE and NOQUOTA_SIZE
- QUOTA_USED and NOQUOTA_USED
- REGISTER and NOREGISTER
- REGISTER=location_format
- REOPEN and NOREOPEN
- SYNC and ASYNC

In addition, this chapter includes the following topics:

- About LOG_ARCHIVE_DEST_n Parameters Attributes
- Attribute Compatibility for Archive Destinations

About LOG_ARCHIVE_DEST_n Parameters Attributes

Each LOG_ARCHIVE_DEST_*n* (where *n* is an integer from 1 to 10) initialization parameter allows you to specify redo log archive destinations, including one required local destination and up to nine additional local or remote destinations. These parameters also allow you to set a number of archival options for each destination. See Chapter 3 for additional information.

AFFIRM and NOAFFIRM

Category	AFFIRM	NOAFFIRM
Attribute type	Keyword	Keyword
Minimum value		
Maximum value		
Default value	NOAFFIRM	NOAFFIRM
Requires attributes ...		
Conflicts with attributes ...	NOAFFIRM	AFFIRM
Valid when ...		
System / Session scope	SYSTEM and SESSION	SYSTEM and SESSION
Corresponding V$ARCHIVE_DEST column	AFFIRM	AFFIRM
Related V$ARCHIVE_DEST column	ASYNC_BLOCKS	ASYNC_BLOCKS

Examples

```
AFFIRM
NOAFFIRM
```

Use this option to ensure that archived redo log contents have been successfully written to disk and are immediately available for database recovery. The AFFIRM attribute indicates that all archived redo log I/O operations are to be performed synchronously. The NOAFFIRM attribute indicates that all archived redo log disk I/O operations are to be performed asynchronously.

The AFFIRM attribute has the potential to affect primary database performance. When you use the LGWR and AFFIRM attributes to indicate that the log writer process will synchronously write the locally archived redo log contents to disk, control is not returned to the users until the disk I/O operation has completed. When you use the ARCH and AFFIRM attributes to indicate that the ARC*n* process will synchronously write the archived redo logs to disk, the archival operation may take longer, and online redo logs may not be reusable.

Using the AFFIRM attribute does not affect performance when using the ASYNC attribute.

The AFFIRM column of the V$ARCHIVE_DEST fixed view indicates whether or not the AFFIRM attribute is being used for the associated destination.

The AFFIRM attribute is best used in conjunction with the SYNC and ASYNC attributes to provide the highest degree of control over negative performance effects on the primary database. The following table identifies the various combinations of these attributes and their potential for affecting primary database performance and data availability.

Network I/O Attribute	Archived Redo Log Disk I/O Attribute	Potential Primary Database Performance	Standby Database Data Availability
SYNC	AFFIRM	Lowest	Highest
SYNC	NOAFFIRM	Low	High
ASYNC	AFFIRM	High	Low
ASYNC	NOAFFIRM	Highest	Lowest

The highest degree of data availability also has the potential for the lowest primary database performance.

Note: When the primary database is in PROTECTED mode, redo log archiving destinations using the log writer process are also automatically placed in AFFIRM mode.

See Also: SYNC and ASYNC on page 8-33

ALTERNATE and NOALTERNATE

Category	ALTERNATE=*destination*	NOALTERNATE
Attribute type	String value	Keyword
Minimum value		
Maximum value		
Default value	NOALTERNATE	NOALTERNATE
Requires attributes ...		
Conflicts with attributes ...	NOALTERNATE	ALTERNATE
Valid when ...	REOPEN=0 or MAX_FAILURE > 0 and failure count is exceeded	
System / Session scope	SYSTEM and SESSION	SYSTEM and SESSION
Corresponding V$ARCHIVE_DEST column	ALTERNATE	ALTERNATE
Related V$ARCHIVE_DEST column	STATUS	STATUS

Examples

```
ALTERNATE=LOG_ARCHIVE_DEST_1
NOALTERNATE
```

Use the ALTERNATE attribute of the LOG_ARCHIVE_DEST_*n* parameters to define an alternate archiving destination to be used if archiving to the original archiving destination fails.

An archiving destination can have a maximum of one alternate destination specified. An alternate destination will be used when the transmission of an online redo log from the primary site to the standby site fails. If archiving fails and the REOPEN attribute is specified with a value of zero (0), or NOREOPEN is specified, the Oracle database server will attempt to archive online redo logs to the alternate destination on the next archival operation.

Use the NOALTERNATE attribute of the LOG_ARCHIVE_DEST_*n* parameters to prevent the original destination from automatically changing to an alternate destination when the original destination fails.

An alternate destination can reference a local or remote archiving destination. An alternate destination cannot be self-referencing.

A destination can also be in the ALTERNATE state; this state is specified using the LOG_ARCHIVE_DEST_STATE_*n* initialization parameters. The ALTERNATE state defers processing of the destination until such time as another destination failure automatically enables this destination, if the alternate destination attributes are valid. See Section 3.3.1.2 for more details about the LOG_ARCHIVE_DEST_STATE_*n* parameters.

The ALTERNATE attribute cannot be modified at the session level.

In the sample initialization parameter file in Example 8–1, LOG_ARCHIVE_DEST_1 will automatically fail over to LOG_ARCHIVE_DEST_2 on the next archival operation if an error occurs or the device becomes full.

Example 8–1 Automatically Failing Over to an Alternate Destination

```
LOG_ARCHIVE_DEST_1=
'LOCATION=/disk1 MANDATORY NOREOPEN ALTERNATE=LOG_ARCHIVE_DEST_2'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
LOG_ARCHIVE_DEST_2='LOCATION=/disk2 MANDATORY'
LOG_ARCHIVE_DEST_STATE_2=ALTERNATE
```

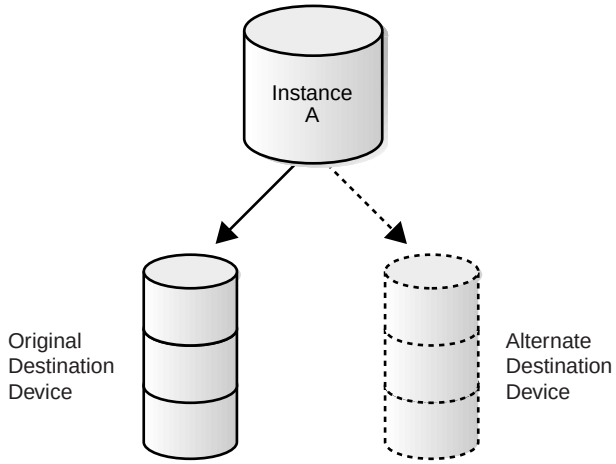
The sample initialization parameter file in Example 8–2 shows how to define an alternate Oracle Net service name to the same standby database.

Example 8–2 Defining an Alternate Oracle Net Service Name to the Same Standby Database

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1 MANDATORY'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
LOG_ARCHIVE_DEST_2='SERVICE=stby1_path1 NOREOPEN OPTIONAL ALTERNATE=LOG_ARCHIVE_
DEST_3'
LOG_ARCHIVE_DEST_STATE_2=ENABLE
LOG_ARCHIVE_DEST_3='SERVICE=stby1_path2 NOREOPEN OPTIONAL'
LOG_ARCHIVE_DEST_STATE_3=ALTERNATE
```

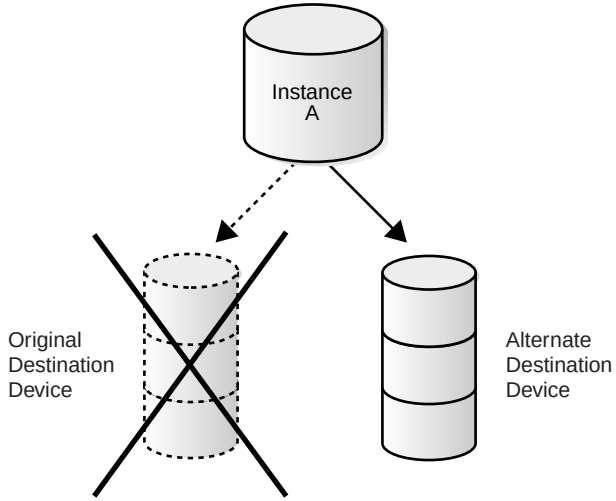
Figure 8–1 shows a scenario where online log files are archived to a local disk device.

Figure 8–1 Archiving Online Log Files to a Local Disk Device



If the original destination device becomes full or unavailable, the archival operation will be automatically redirected to the alternate destination device, as shown in Figure 8–2.

Figure 8–2 Redirecting the Archival Operation to an Alternate Destination Device



The REOPEN attribute takes precedence over the ALTERNATE attribute. The alternate destination will be used only if one of the following is true:

- The NOREOPEN attribute is specified.
- A value of zero (0) is specified for the REOPEN attribute.
- A non-zero REOPEN attribute and a non-zero MAX_FAILURE count have been exceeded.

The ALTERNATE attribute takes precedence over the MANDATORY attribute. This means that a destination will fail over to a valid alternate destination even if the current destination is mandatory.

The following is the archived redo log destination attribute precedence table:

Precedence	Attribute
Highest	MAX_FAILURE
	REOPEN
	ALTERNATE
Lowest	MANDATORY

The use of a standby database as the target of an alternate destination should be carefully handled. Ideally, a standby alternate destination should only be used to specify a different network route to the same standby database system.

If no enabled destination references the alternate destination, the alternate destination is implied to be deferred, because there is no automatic method of enabling the alternate destination.

An alternate destination can be manually enabled at runtime. Conversely, an alternate destination can also be manually deferred at runtime. See Section 3.3.2.2 for more information about changing initialization parameter settings using SQL at runtime.

There is no general pool of alternate archived redo log destinations. Ideally, for any enabled destination, the database administrator should choose an alternate destination that closely mirrors that of the referencing destination, although that is certainly not a requirement.

Each enabled destination can have its own alternate destination. Conversely, several enabled destinations can share the same alternate destination. This is known as an

overlapping set of destinations. Enabling the alternate destination determines the set to which the destination belongs.

Increasing the number of enabled destinations decreases the number of available alternate redo log archiving destinations.

> **Note:** An alternate destination is enabled for the next archival operation. There is no support for enabling the alternate destination in the middle of the archival operation, because that would require rereading already processed blocks, and so forth. This is identical to the REOPEN attribute behavior.

Any destination can be designated as an alternate given the following restrictions:

- At least one local mandatory destination is enabled.
- The number of enabled destinations must meet the defined LOG_ARCHIVE_MIN_SUCCEED_DEST parameter value.
- A destination cannot be its own alternate; however, this does not generate an error.

Session-defined destinations will not activate an alternate destination defined at the system level. Conversely, system-defined destinations will not activate an alternate destination defined at the session level.

If the NOALTERNATE attribute is specified, or if no alternate destination is specified, the destination does not automatically change to another destination upon failure.

If the REOPEN attribute is specified with a non-zero value, the ALTERNATE attribute is ignored. If the MAX_FAILURE attribute is also specified with a non-zero value, and the failure count exceeds the specified failure threshold, the ALTERNATE destination is enabled. Therefore, the ALTERNATE attribute does not conflict with a non-zero REOPEN attribute value.

ARCH and LGWR

Category	ARCH	LGWR
Attribute type	Keyword	Keyword
Minimum value		
Maximum value		
Default value	ARCH	ARCH
Requires attributes ...		
Conflicts with attributes ...	LGWR, ASYNC	ARCH
Valid when ...		
System / Session scope	SYSTEM and SESSION	SYSTEM only
Corresponding V$ARCHIVE_DEST column	ARCHIVER	ARCHIVER
Related V$ARCHIVE_DEST columns	PROCESS, SCHEDULE	PROCESS, SCHEDULE

Examples

ARCH
LGWR

The optional ARCH and LGWR attributes are used to specify the Oracle database server process that is responsible for transmitting redo log files to the corresponding destination.

The ARCH attribute indicates that redo logs are transmitted to the destination during an archival operation. The background archiver processes (ARC*n*) or a foreground archival operation serves as the redo log transport service. ARCH is the implicit default setting.

The LGWR attribute indicates that redo log files are transmitted to the destination concurrently as the online redo log is populated. The background log writer process (LGWR) serves as the redo log transport service. When transmitting redo logs to remote destinations, the LGWR process establishes a network connection to the destination instance. Because the redo log files are transmitted concurrently, they

are not retransmitted to the corresponding destination during the archival operation.

When you change the archiving process from the ARC*n* process to the LGWR process using the ARCH and LGWR attributes for an archive destination, the LGWR process will not start archiving until the next log switch operation. Conversely, when you change the archiving process from the LGWR process to the ARC*n* process, the LGWR process will continue to archive until the next log switch operation.

DELAY and NODELAY

Category	DELAY[=*minutes*]	NODELAY
Attribute type	Numeric	Keyword
Minimum value	0 minutes	
Maximum value	Unlimited	
Default value	NODELAY	NODELAY
Requires attributes ...	SERVICE	
Conflicts with attributes ...	LOCATION, NODELAY	DELAY
Valid when ...		
System / Session scope	SYSTEM and SESSION	SYSTEM and SESSION
Corresponding V$ARCHIVE_DEST column	DELAY_MINS	DELAY_MINS
Related V$ARCHIVE_DEST column	DESTINATION	DESTINATION

Examples

```
DELAY=60
NODELAY
```

When the standby database is in managed recovery mode, redo logs are automatically applied when they arrive from the primary database. However, to protect against the transfer of corrupted or erroneous data from the primary site to the standby site, you may want to create a time lag between archiving a redo log at the primary site and applying that archived redo log at the standby site.

Use the DELAY attribute of the LOG_ARCHIVE_DEST_*n* initialization parameters to specify a time lag for the application of redo logs at the standby site. The DELAY attribute does not affect the transmittal of redo logs to the standby site.

Note: Changes to the DELAY attribute take effect on the next archival operation. In-progress archival operations are not affected.

The DELAY attribute indicates that the archived redo logs at the standby site are not available for recovery until the specified time interval has expired. The time interval is expressed in minutes, and it starts when the redo log is successfully transmitted and archived at the standby site. If you specify the DELAY attribute without a time interval, the default time interval is 30 minutes.

You can use the DELAY attribute to set up a configuration where multiple standby databases are maintained in varying degrees of synchronization with the primary database. For example, assume primary database A supports standby databases B, C, and D. Standby database B is set up as the disaster recovery database and therefore has no time lag. Standby database C is set up to protect against logical or physical corruption, and is maintained with a 2-hour delay. Standby database D is maintained with a 4-hour delay and protects against further corruption.

You can override the specified delay interval at the standby site. To immediately apply an archived redo log to the standby database before the time interval has expired, use the NODELAY keyword of the RECOVER MANAGED STANDBY statement; for example:

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

See Also: Chapter 9, "SQL Statements"

DEPENDENCY and NODEPENDENCY

Category	DEPENDENCY=destination	NODEPENDENCY
Attribute type	String value	Keyword
Minimum value		
Maximum value		
Default value	NODEPENDENCY	NODEPENDENCY
Requires attributes ...	SERVICE, REGISTER	
Conflicts with attributes ...	NODEPENDENCY, LOCATION, NOREGISTER, QUOTA_SIZE, QUOTA_USED	DEPENDENCY
Valid when ...		
System / Session scope	SYSTEM only	SYSTEM only
Corresponding V$ARCHIVE_DEST column	DEPENDENCY	DEPENDENCY
Related V$ARCHIVE_DEST columns		

Examples

```
DEPENDENCY=LOG_ARCHIVE_DEST_1
NODEPENDENCY
```

A standby destination can be defined as being dependent upon the success or failure of an archival operation to another locally accessible destination. The dependent destination is known as the **child destination.** The destination on which the child depends is known as the **parent destination.**

Specifying a **destination dependency** can be useful in the following configurations:

- The standby database and the primary database are on the same node. Therefore, the archived redo logs are implicitly accessible to the standby database.
- Clustered file systems provide remote standby databases with access to the primary database archived redo logs.

- Operating system-specific network file systems provide remote standby databases with access to the primary database archived redo logs.
- Mirrored disk technology provides transparent networking support across geographically remote distances.
- Multiple standby databases are on same remote site, sharing access to common archived redo logs.

In these situations, although a physical archival operation does not occur for the dependent destination, the standby database needs to know the location of the archived redo logs. This allows the standby database to access the archived redo logs when they become available for managed recovery. The DBA must specify a destination as being dependent on the success or failure of a parent destination.

Consider the case of a two-node cluster where a primary node shares access to the destination with the standby node through a mirrored disk device. This configuration, where you maintain a local standby database, is useful for off-loading ad hoc queries and reporting functions.

The primary database archives a redo log locally and, upon successful completion, the archived redo log is immediately available to the standby database for managed recovery. This does not require a physical remote archival operation for the standby destination. In this case, two destinations are used: one for local archiving, and another for archiving at the standby site. The standby destination is not valid unless the primary destination succeeds. Therefore, the standby destination has a dependency upon the success or failure of the local destination.

The `DEPENDENCY` attribute has the following restrictions:

- Only standby destinations can have a dependency.
- The parent destination can be either a local or standby destination.
- The `DEPENDENCY` attribute cannot be modified at the session level.
- The `REGISTER` attribute is required.
- The `SERVICE` attribute is required.
- The alternate destination cannot be self-referencing.

When one or more destinations are dependent upon the same parent destination, all attributes of the dependent destinations still apply to that destination. It appears as if the archival operation were performed for each destination, when only one archival operation actually occurred.

Consider, for example, that two standby databases are dependent upon the archived redo log of a parent destination. You can specify different DELAY attributes for each destination, which allows you to maintain a staggered time lag between the primary database and each standby database.

Similarly, a dependent destination can specify an alternate destination, which itself may or may not be dependent upon the same parent destination.

Note: Dependent destinations do not participate in a standby no-data-loss environment.

LOCATION and SERVICE

Category	LOCATION=*location*	SERVICE=*service*
Attribute type	String value	String value
Minimum value		
Maximum value		
Default value		
Requires attributes ...		
Conflicts with attributes ...	SERVICE, DELAY, DEPENDENCY, REGISTER, NOREGISTER, ASYNC	LOCATION, QUOTA_USED, NOQUOTA_USED
Valid when ...		
System / Session scope	SYSTEM and SESSION	SYSTEM and SESSION
Corresponding V$ARCHIVE_DEST column	DESTINATION	DESTINATION
Related V$ARCHIVE_DEST column	TARGET	TARGET

Examples

```
LOCATION=/tmp/arc_dest
SERVICE=stby1
```

Each destination identifies either a local disk directory or a remotely accessed database. A remotely accessed database can be either a physical standby database or another instance of the current database.

If you use the LOCATION attribute, specify a valid path name for a disk directory on the system that hosts the primary database. Each destination that specifies the LOCATION attribute must identify a unique directory path name. This is the local destination for archiving redo logs.

Local destinations indicate that the archived redo logs are to reside within the file system that is accessible to the primary database. Locally archived redo logs remain physically within the primary database namespace. The destination parameter value specifies the local file system directory path where the archived redo logs will

be placed. When you configure log transport services, you must specify at least one local destination. This ensures that the locally archived redo logs are accessible should managed recovery of the primary database be necessary.

Locally archived redo logs are not used to maintain a transactionally consistent standby database.

If you specify SERVICE, specify a valid Oracle Net service name. The Oracle database server translates this net service name into a connection descriptor. The descriptor contains the information necessary for connecting to the remote database. The service name must have an associated database SID, so that the Oracle database server correctly updates the log history of the control file for the standby database.

Remote destinations indicate that the archived redo logs are to reside elsewhere and are physically outside of the primary database namespace.

Archiving redo logs to a remote destination requires a network connection and an Oracle database instance associated with the remote destination to receive the incoming archived redo logs.

The destination parameter value specifies the Oracle Net service name identifying the remote Oracle instance where the archived log files will be located.

Remotely archived redo logs are available to maintain a transactionally consistent copy of the primary database.

Remotely located archived redo log files are not currently accessible for managed recovery of the primary database.

> **See Also:** *Oracle Net Services Administrator's Guide* for more information on creating Oracle Net service names

The DESTINATION column of the V$ARCHIVE_DEST fixed view identifies the values that have been specified for a destination.

The TARGET column of the V$ARCHIVE_DEST fixed view identifies whether the destination is local or remote to the primary database.

MANDATORY and OPTIONAL

Category	MANDATORY	OPTIONAL
Attribute type	Keyword	Keyword
Minimum value		
Maximum value		
Default value	OPTIONAL	OPTIONAL
Requires attributes ...		
Conflicts with attributes ...	OPTIONAL	MANDATORY
Valid when ...		
System / Session scope	SYSTEM and SESSION	SYSTEM and SESSION
Corresponding V$ARCHIVE_DEST column	BINDING	BINDING
Related V$ARCHIVE_DEST columns		

Examples

```
MANDATORY
OPTIONAL
```

You can specify a policy for reuse of online redo logs using the attributes OPTIONAL or MANDATORY. The archival of an optional destination can fail, and the online redo logs will be overwritten. If the archival of mandatory destinations fail, online redo log files will not be overwritten.

By default, one destination is mandatory even if all destinations are designated to be optional. The following example shows how to set a mandatory archiving destination:

```
LOG_ARCHIVE_DEST_3 = 'LOCATION=/arc_dest MANDATORY'
```

The LOG_ARCHIVE_MIN_SUCCEED_DEST=*n* parameter (where *n* is an integer from 1 to 10) specifies the number of destinations that must archive successfully before the log writer process can overwrite the online redo logs. All mandatory destinations and non-standby optional destinations contribute to satisfying the

LOG_ARCHIVE_MIN_SUCCEED_DEST=*n* count. For example, you can set the parameter as follows:

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

When determining how to set your parameters, note that:

- Not specifying MANDATORY for a destination is the same as specifying OPTIONAL.
- You must have at least one local destination, which you can declare OPTIONAL or MANDATORY.

  At least one local destination will operationally be treated as mandatory, because the minimum value for the LOG_ARCHIVE_MIN_SUCCEED_DEST parameter is 1.
- The failure of any mandatory destination, including a mandatory standby destination, makes the LOG_ARCHIVE_MIN_SUCCEED_DEST parameter irrelevant.
- The LOG_ARCHIVE_MIN_SUCCEED_DEST value cannot be greater than the number of destinations, nor greater than the number of mandatory destinations plus the number of optional local destinations.
- If you defer a mandatory destination, and the online log is overwritten without transferring the redo log to the standby site, then you must transfer the redo log to the standby site manually.

The BINDING column of the V$ARCHIVE_DEST fixed view specifies how failure will affect the archival operation.

MAX_FAILURE and NOMAX_FAILURE

Category	MAX_FAILURE=*count*	NOMAX_FAILURE
Attribute type	Numeric	Keyword
Minimum value		
Maximum value		
Default value	NOMAX_FAILURE	NOMAX_FAILURE
Requires attributes ...	REOPEN	
Conflicts with attributes ...	NOMAX_FAILURE	MAX_FAILURE
Valid when ...	REOPEN=*seconds* when *seconds* > 0	
System / Session scope	SYSTEM only	SYSTEM only
Corresponding V$ARCHIVE_DEST column	MAX_FAILURE	
Related V$ARCHIVE_DEST columns	FAILURE_COUNT, REOPEN_SECS	

Examples

```
MAX_FAILURE=30
NOMAX_FAILURE
```

The MAX_FAILURE attribute specifies the maximum number of contiguous archival failures for the particular destination. Use this attribute for archiving destinations that you want to retry archival operations to after a failure, but not retry indefinitely.

You must use the MAX_FAILURE attribute in conjunction with a REOPEN attribute value greater than zero (0).

The MAX_FAILURE attribute corresponds to the MAX_FAILURE column of the V$ARCHIVE_DEST fixed view. The column FAILURE_COUNT identifies the number of contiguous failures that have occurred. The related column REOPEN_SECS identifies the REOPEN attribute value.

The MAX_FAILURE attribute identifies the maximum number of repeated archival operations that will be attempted before giving up.

Use of the NOMAX_FAILURE attribute, or use of a MAX_FAILURE attribute value of zero (0), indicates that an unlimited number of destination failures are allowed.

If the MAX_FAILURE attribute is not specified (or is specified as NOMAX_FAILURE), and the REOPEN attribute is specified with a non-zero value, the archival operation will be indefinitely attempted. If the destination has the MANDATORY attribute, the online redo log will not be reclaimable in the event of a repeated failure.

This attribute cannot be modified at the session level.

Each destination contains an internal operation failure count. The failure count is reset to zero (0) whenever the destination is modified by the user entering an ALTER SYSTEM SET statement. Runtime modifications made to the destination, such as changing the QUOTA_USED attribute (described in the QUOTA_USED and NOQUOTA_USED section), do not affect the failure count. The failure count can be displayed using the related FAILURE_COUNT column of the V$ARCHIVE_DEST fixed view.

If an archival operation fails for any reason, the failure count is incremented. If the failure count is greater than or equal to the MAX_FAILURE attribute value, the REOPEN attribute value is implicitly set to the value zero (0). The REOPEN_SECS column of the V$ARCHIVE_DEST fixed view displays the real value.

Dynamically changing the MAX_FAILURE attribute (or any other destination attribute) resets the destination failure count. This avoids the problem of setting the MAX_FAILURE attribute to a value less than the current failure count value.

Note: Once the destination failure count meets the specified MAX_FAILURE attribute value, the only way to reuse the destination is to modify the MAX_FAILURE attribute value or some other attribute.

QUOTA_SIZE and NOQUOTA_SIZE

Category	QUOTA_SIZE=*blocks*	NOQUOTA_SIZE
Attribute type	Numeric	Keyword
Minimum value	0 blocks	
Maximum value	Unlimited blocks	
Default value	NOQUOTA_SIZE	NOQUOTA_SIZE
Requires attributes ...	LOCATION	
Conflicts with attributes ...	NOQUOTA_SIZE, DEPENDENCY, SERVICE	QUOTA_SIZE
Valid when ...		
System / Session scope	SYSTEM only	SYSTEM only
Corresponding V$ARCHIVE_DEST column	QUOTA_SIZE	QUOTA_SIZE
Related V$ARCHIVE_DEST column	QUOTA_USED	QUOTA_USED

Examples

```
QUOTA_SIZE=100K
NOQUOTA_SIZE
```

The QUOTA_SIZE attribute indicates the maximum number of 512-byte blocks of physical storage on a disk device that may be consumed by a local destination. The value is specified in 512-byte blocks even if the physical device uses a different block size. The optional suffix values K, M, and G represent thousand, million, and billion, respectively (the value "1K" means 1,000 512-byte blocks).

Use of the NOQUOTA_SIZE attribute, or the QUOTA_SIZE attribute with a value of zero (0), indicates that there is unlimited use of the of the disk device by this destination; this is the default value.

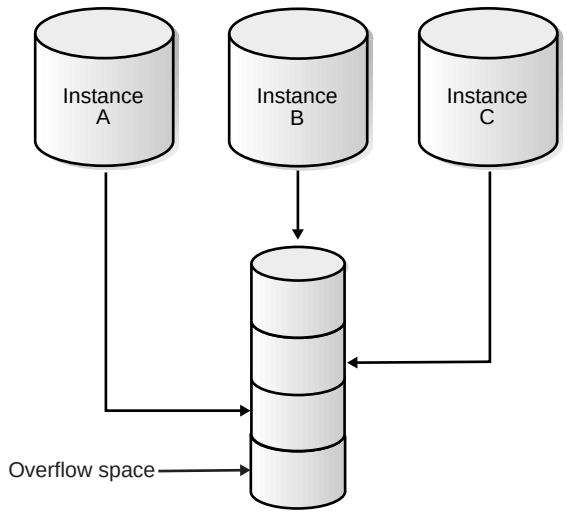
This attribute cannot be modified at the session level.

A local archiving destination can be designated as being able to occupy all or some portion of the physical disk. For example, in a Real Application Clusters environment, a physical archived redo log disk device may be shared by two or

more separate nodes (through a clustered file system, such as is available with Sun Clusters). As there is no cross-instance initialization parameter knowledge, none of the Real Application Clusters nodes is aware that the archived redo log physical disk device is shared with other instances. This can lead to significant problems when the destination disk device becomes full; the error is not detected until every instance tries to archive to the already full device. This seriously affects database availability.

For example, consider an 8-gigabyte (GB) disk device `/dev/arc_dest` that is further subdivided into node-specific directories `node_a`, `node_b`, and `node_c`. The DBA could designate that each of these instances is allowed to consume a maximum of 2 GB, which would allow an additional 2 GB for other purposes. This scenario is shown in Figure 8–3.

Figure 8–3 Specifying Disk Quota for a Destination



No instance will consume more than its allotted quota, so the conceptual picture displayed in Figure 8–3 holds true.

The quota is common to all users of the destination, including foreground archival operations, the archiver process, and even the log writer process.

Oracle Corporation highly recommends that the ALTERNATE attribute be used in conjunction with the QUOTA_SIZE attribute. However, this is not required.

See Also: ALTERNATE and NOALTERNATE on page 8-5

QUOTA_USED and NOQUOTA_USED

Category	QUOTA_USED=*blocks*	NOQUOTA_USED
Attribute type	Numeric	Keyword
Minimum value	0 blocks	
Maximum value	Unlimited blocks	
Default value	NOQUOTA_USED	NOQUOTA_USED
Requires attributes ...	LOCATION	
Conflicts with attributes ...	NOQUOTA_USED, DEPENDENCY, SERVICE	QUOTA_USED
Valid when	QUOTA_SIZE > 0	
System / Session scope	SYSTEM only	SYSTEM only
Corresponding V$ARCHIVE_DEST column	QUOTA_USED	QUOTA_USED
Related V$ARCHIVE_DEST column	QUOTA_SIZE	QUOTA_SIZE

Examples

```
QUOTA_USED=450K
NOQUOTA_USED
```

The QUOTA_USED attribute identifies the number of 512-byte blocks of data that have been archived on the specified local destination. Note that the value is specified in 512-byte blocks even if the physical device uses a different block size. The optional suffix values K, M, and G represent thousand, million, and billion, respectively (the value "1K" means 1,000 512-byte blocks).

This attribute cannot be modified at the session level.

If you specify a QUOTA_SIZE attribute value greater than zero (0) for a destination but do not specify a QUOTA_USED attribute value in the database initialization parameter file, the QUOTA_USED attribute value will be automatically determined when the database is initially mounted. The QUOTA_USED attribute value will default to the actual number of blocks residing on the local archiving destination device. If the calculated QUOTA_USED attribute value exceeds the QUOTA_SIZE

attribute value, the QUOTA_SIZE attribute value will be automatically adjusted to reflect the actual storage consumed.

This automatic calculation of the QUOTA_USED value applies only to local archiving destinations.

Note: The runtime value of the QUOTA_USED attribute changes automatically as online redo log archival operations are started. The QUOTA_USED attribute value is optimistically pre-incremented against the destination quota size. You need not change this attribute's value. In the event of archival failure, the QUOTA_USED value will be adjusted.

If, at runtime, you dynamically modify the QUOTA_SIZE attribute value, but not the QUOTA_USED attribute value, the QUOTA_USED attribute value will not be automatically recalculated.

For local destinations, the QUOTA_USED attribute value is incremented at the start of an archival operation. If the resulting value is greater than the QUOTA_USED attribute value, the destination status is changed to FULL and the destination is rejected before the archival operation begins.

The QUOTA_SIZE and QUOTA_USED attributes are very important because they can be used together to detect a lack of disk space before the archival operation begins.

Consider the case where the QUOTA_SIZE attribute value is "100K" and the QUOTA_USED attribute value is "100K" also. The destination status is VALID at this point. However, an attempt to archive 1 block will result in the QUOTA_USED attribute value being changed to "101K", which exceeds the QUOTA_SIZE attribute value. Therefore, the destination status is changed to FULL, and the destination is rejected before the archival operation begins.

The QUOTA_USED attribute has a default value of zero (0) for remote archiving destinations.

REGISTER and NOREGISTER

Category	REGISTER	NOREGISTER
Attribute type	Keyword	Keyword
Minimum value		
Maximum value		
Default value	REGISTER	REGISTER
Requires attributes ...	SERVICE, DEPENDENCY	SERVICE
Conflicts with attributes ...	NOREGISTER, LOCATION, NOALTERNATE	REGISTER, LOCATION, DEPENDENCY
Valid when ...		
System / Session scope	SYSTEM and SESSION	SYSTEM and SESSION
Corresponding V$ARCHIVE_DEST column	DESTINATION	DESTINATION
Related $ARCHIVE_DEST column	TARGET	TARGET

Examples

```
REGISTER
NOREGISTER
```

The REGISTER attribute indicates that the location of the archived redo log is to be recorded in the corresponding destination database control file. This is the implicit default setting.

For a standby destination database, this archived redo log registry serves as the manifest for the managed standby recovery operation.

By default, the location of the archived redo log, at a remote destination site, is derived from the destination instance initialization parameters: STANDBY_ARCHIVE_DEST and LOG_ARCHIVE_FORMAT.

The optional NOREGISTER attribute indicates that the location of the archived redo log is not to be recorded in the corresponding destination database control file. This setting pertains to remote destinations only. The location of each archived redo log is always recorded in the primary database control file.

The NOREGISTER attribute is incompatible with the LOCATION and DEPENDENCY attributes.

REGISTER=location_format

Category	REGISTER=*location_format*
Attribute type	String value
Minimum value	
Maximum value	
Default value	REGISTER
Requires attributes ...	DEPENDENCY
Conflicts with attributes ...	NOREGISTER, LOCATION
Valid when ...	
System / Session scope	SYSTEM and SESSION
Corresponding V$ARCHIVE_DEST column	DESTINATION
Related V$ARCHIVE_DEST column	TARGET

Examples

```
REGISTER=/dsk1/dir1/arc_%t_%s.arc
```

The optional REGISTER=*location_format* attribute is used to specify a filename format template for archived redo logs that is different from the default filename format template defined in the primary and standby database initialization parameter files. The default filename format template is a combination of the database initialization parameters STANDBY_ARCHIVE_DEST and LOG_ARCHIVE_FORMAT.

If the attribute value is not specified, then the setting is the same as REGISTER.

The REGISTER=*location_format* attribute is valid with remote destinations only. It requires the DEPENDENCY attribute and is incompatible with the LOCATION attribute.

REOPEN and NOREOPEN

Category	REOPEN [=*seconds*]	NOREOPEN
Attribute type	Numeric	Keyword
Minimum value	0 seconds	
Maximum value	Unlimited seconds	
Default value	300 seconds	NOREOPEN
Requires attributes ...		
Conflicts with attributes ...	NOREOPEN	REOPEN
Valid when ...		
System / Session scope	SYSTEM and SESSION	SYSTEM and SESSION
Corresponding V$ARCHIVE_DEST column	REOPEN_SECS	REOPEN_SECS
Related V$ARCHIVE_DEST column	MAX_FAILURE	MAX_FAILURE

Examples

```
REOPEN=30
NOREOPEN
```

The REOPEN attribute specifies the minimum number of seconds before the archiver process (ARC*n*, foreground, or log writer process) should try again to access a previously failed destination. You can turn off the option by specifying NOREOPEN.

REOPEN applies to all errors, not just connection failures. These errors include, but are not limited to, network failures, disk errors, and quota exceptions.

If you specify REOPEN without a value, the attribute has a default value of 300 seconds.

If you specify NOREOPEN, the failed destination remains disabled until:

- You manually reenable the destination
- You issue an ALTER SYSTEM SET or an ALTER SESSION SET statement with the REOPEN option

- The instance is restarted

SYNC and ASYNC

Category	SYNC	ASYNC=*blocks*
Attribute type	Keyword	Numeric
Minimum value		0 blocks
Maximum value		20,480 blocks
Default value	SYNC	SYNC
Requires attributes ...		LGWR
Conflicts with attributes ...	ASYNC	SYNC, LOCATION, ARCH
Valid when ...		
System / Session scope	SYSTEM and SESSION	SYSTEM only
Corresponding V$ARCHIVE_DEST column	TRANSMIT_MODE	TRANSMIT_MODE
Related V$ARCHIVE_DEST column		ASYNC_BLOCKS

Examples

```
SYNC
ASYNC=20480
```

When you use the log writer process (LGWR) to transmit redo logs to a standby database, you can specify the method by which the network I/O operations are performed.

The SYNC attribute indicates that all network I/O operations are to be performed synchronously, in conjunction with each write operation to the online redo log. Control will not be returned to the users until the redo information is received by the standby site, and the disk I/O operation on the standby site has completed. This attribute has the potential to affect primary database performance when you use slow network connections, but provides the highest degree of data availability.

The ASYNC attribute indicates that all network I/O operations are to be performed asynchronously, and control is returned to users immediately. The block count specified determines the size of the SGA network buffer to be used. In general, the slower the network connection you have, the larger the block count should be.

If you do not specify either SYNC or ASYNC, SYNC is the default. If you specify ASYNC without specifying a value, 2,048 is the default.

See Also: Appendix C, "Log Writer Asynchronous Network I/O"

With local archive destinations, there is no benefit to using asynchronous I/O operations. Therefore, the ASYNC attribute is incompatible with the LOCATION attribute.

Note: When the primary database is in protected mode, standby redo log archiving destinations using the log writer process are automatically placed in SYNC mode.

Attribute Compatibility for Archive Destinations

The LOG_ARCHIVE_DEST_*n* initialization parameters have many attributes. Some of these attributes conflict with each other. Some of the attributes require that other attributes are also defined. Table 8–1 lists the supported attributes and the requirements associated with each one.

Table 8–1 LOG_ARCHIVE_DEST_n Attribute Compatibility

Attribute	Requires...	Conflicts with...
AFFIRM		NOAFFIRM
NOAFFIRM		AFFIRM
ALTERNATE		NOALTERNATE
NOALTERNATE		ALTERNATE
ARCH		LGWR ASYNC
ASYNC	LGWR	SYNC LOCATION ARCH
DELAY	SERVICE	LOCATION NODELAY
NODELAY		DELAY
DEPENDENCY	SERVICE REGISTER	LOCATION NODEPENDENCY NOREGISTER QUOTA_SIZE QUOTA_USED
NODEPENDENCY		DEPENDENCY
LGWR		ARCH
LOCATION		SERVICE DEPENDENCY REGISTER NOREGISTER DELAY ASYNC
MANDATORY		OPTIONAL

Table 8–1 (Cont.) LOG_ARCHIVE_DEST_n Attribute Compatibility

Attribute	Requires...	Conflicts with...
MAX_FAILURE	REOPEN	NOMAX_FAILURE
NOMAX_FAILURE		MAX_FAILURE
OPTIONAL		MANDATORY
QUOTA_SIZE	LOCATION	DEPENDENCY SERVICE NOQUOTA_SIZE
NOQUOTA_SIZE		QUOTA_SIZE
QUOTA_USED	LOCATION	DEPENDENCY SERVICE NOQUOTA_USED
NOQUOTA_USED		QUOTA_USED
REGISTER	SERVICE DEPENDENCY	LOCATION NOREGISTER NOALTERNATE
NOREGISTER	SERVICE	LOCATION REGISTER DEPENDENCY
REOPEN		NOREOPEN
NOREOPEN		REOPEN
SERVICE		LOCATION QUOTA_USED NOQUOTA_USED
SYNC		ASYNC

9

SQL Statements

Several different SQL statements and SQL*Plus commands 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 9–1 describes relevant statements.

Table 9–1 Standby Database Statements

Statement	Description
`ALTER DATABASE ACTIVATE [PHYSICAL] STANDBY DATABASE [SKIP [STANDBY LOGFILE]]`	Changes the state of a standby database to an active database and prepares it to become the primary database. See Section 5.4.
`ALTER DATABASE ADD [STANDBY] LOGFILE [THREAD` *`integer`*`] [GROUP` *`integer`*`]` *`filespec`*	Adds one or more redo log file groups to the specified thread, making them available to the instance assigned the thread. The STANDBY keyword indicates that the redo log file is for use by standby databases only.
`ALTER DATABASE ADD [STANDBY] LOGFILE MEMBER '`*`filename`*`' [REUSE] TO` *`logfile-descriptor`*	Adds new members to existing redo log file groups. The STANDBY keyword is not required. If `GROUP` *`integer`* (of the *`logfile-descriptor`* clause) was added for standby use, all of its members will be used only for standby databases as well.
`ALTER DATABASE CREATE STANDBY CONTROLFILE AS '`*`filename`*`'`	Creates a standby control file. Issue this statement at the primary database. See Section 2.3.3.
`ALTER DATABASE DROP [STANDBY] LOGFILE` *`logfile_descriptor`*	Use the `DROP STANDBY LOGFILE` clause to drop all members of a standby redo log group. See *Oracle9i SQL Reference*.

Table 9–1 (Cont.) Standby Database Statements

Statement	Description
`ALTER DATABASE DROP [STANDBY] LOGFILE MEMBER ' filename '`	Use the `DROP STANDBY LOGFILE MEMBER` clause to drop one or more standby redo log members. Each '*filename*' must fully specify a member using the conventions for filenames on your operating system. See *Oracle9i SQL Reference*.
`ALTER DATABASE MOUNT STANDBY DATABASE`	Mounts a standby database.
`ALTER DATABASE OPEN READ ONLY`	Opens the standby database in read-only mode. See Section 4.7.2.
`ALTER DATABASE COMMIT TO SWITCHOVER TO [PHYSICAL] {PRIMARY | STANDBY} [[NO]WAIT]`	Prepares the current primary database for switchover to standby status or prepares the current standby database for switchover to primary status.
`ALTER DATABASE REGISTER [PHYSICAL] LOGFILE filespec`	Allows the registration of manually archived logs.
`ALTER DATABASE SET STANDBY DATABASE {PROTECTED | UNPROTECTED}`	Lets you specify whether your database environment is in no-data-loss mode. In this mode, the primary database is protected against data loss and divergence. See Section 3.6.3.5.
`ALTER DATABASE RECOVER [FROM '/dir'] STANDBY DATABASE`	Initiates manual recovery of the standby database. You can specify a nondefault directory for the archived redo logs. See Section B.2.1.
`ALTER DATABASE RECOVER MANAGED STANDBY DATABASE`	Initiates managed recovery of the standby database. See Section 4.3.2.
`ALTER DATABASE RECOVER MANAGED STANDBY DATABASE [TIMEOUT integer]`	The `TIMEOUT` option allows *integer* minutes of inactivity while the recovery waits for the next archived redo log to arrive from the primary database. If *integer* minutes go by without receiving another log to process, managed recovery automatically terminates. The `TIMEOUT` option can only be used when starting a foreground recovery. See Section 4.4.9.

Table 9–1 (Cont.) Standby Database Statements

Statement	Description
`ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL [NOWAIT]`	Causes log apply services to wait for managed recovery to finish writing the current log before canceling recovery. `NOWAIT` indicates that log apply services should not wait for the redo apply operation to complete before returning control to the user. See Section 4.4.1.
`ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL [IMMEDIATE] [NOWAIT]`	The `CANCEL` option terminates managed recovery. By default, the recovery process will finish processing the log it is currently working on before terminating. If `IMMEDIATE` is used, the recovery process will terminate after it finishes processing the redo block it is currently working on. When `NOWAIT` is used, control is returned to the process that issued the `CANCEL` statement without waiting for the recovery process to terminate. See Section 4.4.1.
`ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DELAY` *`integer;`*	The `DELAY` option specifies an absolute apply delay interval to the managed recovery operation. The managed recovery operation waits the specified number of minutes before applying the archived redo logs. See Section 4.4.2.
`ALTER DATABASE RECOVER MANAGED STANDBY DATABASE EXPIRE` *`integer;`*	The `EXPIRE` option specifies the number of minutes after which the managed recovery operation automatically terminates, relative to the current time. The managed recovery operation terminates at the end of the current archived redo log that is being processed. See Section 4.4.4.

Table 9–1 (Cont.) Standby Database Statements

Statement	Description
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT [FROM SESSION] [NODELAY] [PARALLEL *integer*]	The DISCONNECT option starts managed recovery in background mode. A managed recovery process (MRP) is created to perform the recovery in the background while the foreground process that issued the RECOVER statement can continue performing other tasks. See Section 4.4.3. The NODELAY option bypasses the specified delay interval and causes the archived redo logs to be applied to the standby database without delay. See Section 4.4.7. The PARALLEL option starts a parallel recovery using *'integer'* child processes that run in the background. A parallel recovery uses the child processes to spread the workload of the recovery across several processes. See Section 4.4.8.
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE FINISH [NOWAIT]	The FINISH option is used to complete managed recovery in preparation for a failover from the primary to the standby database. A managed recovery in FINISH mode first applies all available archived redo logs and then continues to recover available standby redo logs, while a recovery that is not in FINISH mode only applies the archived redo logs. Specifying the FINISH option causes the standby database to be up-to-date with the last committed transaction on the primary database. The FINISH option can be used when starting a managed recovery or when altering the mode of an ongoing managed recovery. If the FINISH option is used to alter the mode of an ongoing managed recovery, the NOWAIT option allows control to be returned to the foreground process before the recovery completes. See Section 4.4.5.

Table 9–1 (Cont.) Standby Database Statements

Statement	Description
`ALTER DATABASE RECOVER MANAGED STANDBY DATABASE NEXT integer;`	The NEXT option directs the managed recovery operation to apply a specified number of archived redo logs as soon as possible after the log transport services have archived them. See Section 4.4.6.
`ALTER DATABASE RECOVER MANAGED STANDBY DATABASE [NODELAY] [PARALLEL integer]`	The NODELAY option bypasses the specified delay interval and causes the archived redo logs to be applied to the standby database without delay. See Section 4.4.7. The PARALLEL option starts a parallel recovery using *'integer'* child processes that run in the background. A parallel recovery uses the child processes to spread the workload of the recovery across several processes. See Section 4.4.8.
`STARTUP NOMOUNT pfile='initSID.ora'`	Starts the standby instance without mounting the control file. You must execute this statement before mounting the standby database. See Section 4.3.1 and Section 4.7.2.

See Also: *Oracle9i SQL Reference* and *SQL*Plus User's Guide and Reference* for detailed information on these SQL statements

10

Fixed Views

This chapter describes the fixed views that are used in a Data Guard environment. This is a subset of the views that are available for use in a database. This chapter contains the following sections:

About Fixed Views

The Oracle database contains a set of underlying views that are maintained by the server and accessible to the database administrator. These **fixed views** are also called **dynamic performance views** because they are continuously updated while a database is open and in use, and their contents relate primarily to performance.

Although these views appear to be regular database tables, they are not. These views provide data on internal disk structures and memory structures. You can select from these views, but you can never update or alter them.

These view names are prefixed with either V$ or GV$, for example, `V$ARCHIVE_DEST` or `GV$ARCHIVE_DEST`.

Standard dynamic performance views (V$ fixed views) store information on the local instance. In contrast, global dynamic performance views (GV$ fixed views) store information on all open instances. Each V$ fixed view has a corresponding GV$ fixed view.

In most cases, the information available in fixed views persists across instance shutdowns. However, certain fixed view information is reset when the instance is shut down; these views are specifically identified in this chapter.

For additional information about views, see the *Oracle9i Database Reference.*

V$ARCHIVE_DEST

The V$ARCHIVE_DEST view describes, for the current instance, all the archived redo log destinations, their current value, mode, and status.

> **Note:** The information in this view does not persist across an instance shutdown.

This view contains the following columns:

Column	Description
DEST_ID	Identifies the log archive destination parameter
STATUS	Identifies the current status of the destination. Possible values are: ■ VALID - initialized and available ■ INACTIVE - no destination information ■ DEFERRED - manually disabled by the user ■ ERROR - error during open or copy ■ DISABLED - disabled after error ■ BAD PARAM - parameter has errors ■ ALTERNATE - destination is in an alternate state ■ FULL - exceeded quota size for the destination
BINDING	Specifies how failure will affect the archival operation. Possible values are: ■ OPTIONAL - successful archival is not required ■ MANDATORY - successful archival is required
NAME_SPACE	Identifies the scope of parameter setting. Possible values are: ■ SYSTEM - system definition ■ SESSION - session definition

Column	Description
TARGET	Specifies whether the archive destination is local or remote to the primary database. Possible values are: ■ PRIMARY - local ■ STANDBY - remote
ARCHIVER	Identifies the archiver process relative to the database where the query is issued. Possible values are: ■ ARC*n* ■ FOREGROUND ■ LGWR ■ RFS
SCHEDULE	Indicates whether the archival of this destination is INACTIVE, PENDING, ACTIVE, or LATENT
DESTINATION	Specifies the location where the archived redo logs are to be archived
LOG_SEQUENCE	Identifies the sequence number of the last archived redo log to be archived
REOPEN_SECS	Identifies the retry time, in seconds, after error
DELAY_MINS	Identifies the delay interval, in minutes, before the archived redo log is automatically applied to a standby database
PROCESS	Identifies the archiver process relative to the primary database, even if the query is issued on the standby database. Possible values are: ■ ARC*n* ■ FOREGROUND ■ LGWR
MANIFEST	Indicates whether the archived redo log is registered in the remote destination control file. If the archived redo log is registered, it is available to the managed recovery operation. Possible values are: ■ REGISTER ■ NOREGISTER
FAIL_DATE	Indicates the date and time of error
FAIL_SEQUENCE	Indicates the sequence number of the archived redo log being archived when the last error occurred

Column	Description
FAIL_BLOCK	Indicates the block number of the archived redo log being archived when the last error occurred
FAILURE_COUNT	Identifies the current number of contiguous archival operation failures that have occurred for the destination
MAX_FAILURE	Identifies the maximum number of contiguous archival operation failures allowed before the alternate destination is activated
ERROR	Displays the error text
ALTERNATE	Identifies the alternate destination, if any
DEPENDENCY	Identifies the dependent archive destination, if any
REGISTER	Specifies the template to be used to derive the location to be recorded
QUOTA_SIZE	Identifies the destination quotas, expressed in bytes
QUOTA_USED	Identifies the size of all the archived redo logs currently residing on the specified destination
MOUNTID	Identifies the instance mount identifier
AFFIRM	Specifies disk I/O mode
ASYNC_BLOCKS	Number of blocks specified for the ASYNC attribute
TRANSMIT_MODE	Specifies network transmission mode (SYNC or ASYNC)
TYPE	Indicates whether the archived log destination definition is PUBLIC or PRIVATE. Only PUBLIC destinations can be modified at runtime using the ALTER SYSTEM SET or ALTER SESSION SET statements. By default, all archived log destinations are PUBLIC.

V$ARCHIVE_DEST_STATUS

The V$ARCHIVE_DEST_STATUS view displays runtime and configuration information for the archived redo log destinations.

> **Note:** The information in this view does not persist across an instance shutdown.

The V$ARCHIVE_DEST_STATUS view contains the following columns:

Column	Description
DEST_ID	Identifies the log archive destination parameter
STATUS	Identifies the current status of the destination. Possible values are: ■ VALID - initialized and available ■ INACTIVE - no destination information ■ DEFERRED - manually disabled by the user ■ ERROR - error during open or copy ■ DISABLED - disabled after error ■ BAD PARAM - parameter has errors ■ ALTERNATE - destination is in an alternate state ■ FULL - exceeded quota size for the destination
TYPE	Identifies the type of archival destination database. Possible values are: ■ LOCAL - local to primary database ■ PHYSICAL - physical standby ■ CROSS-INSTANCE - an instance of the primary

Column	Description
DATABASE_MODE	Identifies the current mode of the archival destination database. Possible values are: ■ STARTED - instance started, not mounted ■ MOUNTED - mounted ■ MOUNTED-STANDBY - mounted standby ■ OPEN - open read / write ■ OPEN_READ-ONLY - open read-only
RECOVERY_MODE	Identifies the current mode of media recovery at the archival destination database. Possible values are: ■ IDLE - managed recovery is not active ■ MANUAL - manual media recovery active ■ MANAGED - managed recovery is active
DESTINATION	Specifies the location where the archived redo logs are to be archived
ARCHIVED_THREAD#	Identifies the thread number of the most recent archived redo log received at the destination
ARCHIVED_SEQ#	Identifies the log sequence number of the most recent archived redo log received at the destination
APPLIED_THREAD#	Identifies the thread number of the most recent applied redo log received at the destination
APPLIED_SEQ#	Identifies the log sequence number of the most recent applied redo log received at the destination
ERROR	Displays the error text
STANDBY_LOGFILE_ COUNT	Indicates the total number of standby redo logs created on the standby database
STANDBY_LOGFILE_ ACTIVE	Indicates the total number of standby redo logs on the standby database that are active and contain primary database online redo log information

V$ARCHIVE_GAP

The V$ARCHIVE_GAP view contains the following columns:

Column	Description
THREAD#	Specifies the thread number
LOW_SEQUENCE#	Specifies the low number of the log file
HIGH_SEQUENCE#	Specifies the high number of the log file

V$ARCHIVED_LOG

The V$ARCHIVED_LOG view displays archived redo log information from the control file, including archived log names. This view contains the following columns:

Column	Description
RECID	Archived log record ID
STAMP	Archived log record stamp
NAME	Archived log file name. If set to NULL, the log file was cleared before it was archived.
DEST_ID	The original destination from which the archived log was generated. Value is 0 if the destination identifier is not available.
THREAD#	Redo thread number
SEQUENCE#	Redo log sequence number
RESETLOGS_CHANGE#	Resetlogs change number of the database when this log was written
RESETLOGS_TIME	Resetlogs time of the database when this log was written
FIRST_CHANGE#	First change number in the archived log
FIRST_TIME	Time stamp of the first change
NEXT_CHANGE#	First change in the next log
NEXT_TIME	Time stamp of the next change
BLOCKS	Size of the archived log in blocks
BLOCK_SIZE	Redo log block size. This is the logical block size of the archived log, which is the same as the logical block size of the online log from which this archived log was copied. The online log logical block size is a platform-specific value that is not adjustable by the user.
CREATOR	Identifies the creator of the archived log
REGISTRAR	Identifies the registrar of the entry
STANDBY_DEST	Indicates if the entry is an archived log destination

Column	Description
ARCHIVED	Indicates that the online redo log was archived or that RMAN only inspected the log and created a record for future application of redo logs during recovery
APPLIED	Indicates whether the archived log has been applied to its corresponding standby database
DELETED	Specifies whether an RMAN DELETE command has physically deleted the archived log file from disk, as well as logically removing it from the control file of the **target database** and from the **recovery catalog**
STATUS	The status of this archived log. Possible values are: A - Available D - Deleted U - Unavailable X - Expired
COMPLETION_TIME	Time when the archiving completed
DICTIONARY_BEGIN	Indicates whether this log contains the start of a LogMiner dictionary
DICTIONARY_END	Indicates whether this log contains the end of a LogMiner dictionary
BACKUP_COUNT	Indicates the number of times this file has been backed up. Values range from 0 to 15. If the file has been backed up more than 15 times, the value remains 15.
END_OF_REDO	Indicates whether this archived redo log contains the end of all redo information from the primary database. Values are YES and NO.
ARCHIVAL_THREAD#	This is the redo thread number of the instance that performed the archival operation. This column differs from the THREAD# column only when a closed thread is archived by another instance.

V$DATABASE

The V$DATABASE view provides database information from the control file. This view contains the following columns:

Column	Description
DBID	The database identifier that is calculated when the database is created. This identifier is stored in all file headers.
NAME	Name of the database
CREATED	Creation date
RESETLOGS_CHANGE#	Change number at open resetlogs
RESETLOGS_TIME	Time stamp of open resetlogs
PRIOR_RESETLOGS_ CHANGE#	Change number at prior resetlogs
PRIOR_RESETLOGS_ TIME	Time stamp of prior resetlogs
LOG_MODE	Archive log mode
CHECKPOINT_CHANGE#	Last SCN checkpointed
ARCHIVE_CHANGE#	Last SCN archived
CONTROLFILE_TYPE	The type of control file. Possible values are: ■ STANDBY - indicates database is in standby mode ■ CLONE - indicates a clone database ■ BACKUP | CREATED - indicates database is being recovered using a backup or created control file ■ CURRENT - indicates the database is available for general use
CONTROLFILE_CREATED	Control file creation time stamp
CONTROLFILE_ SEQUENCE#	Control file sequence number incremented by control file transactions
CONTROLFILE_CHANGE#	Last change number in the backup control file. Set to NULL if the control file is not a backup.
CONTROLFILE_TIME	Last time stamp in the backup control file. Set to NULL if the control file is not a backup.

Column	Description
OPEN_RESETLOGS	Indicates whether the next database open allows or requires the resetlogs option
VERSION_TIME	The version time
OPEN_MODE	Open mode information
STANDBY_MODE	Indicates whether the database is protected
REMOTE_ARCHIVE	The value of the REMOTE_ARCHIVE_ENABLE initialization parameter. Possible values are: ■ TRUE ■ FALSE
ACTIVATION#	Number assigned to the database instantiation.
ARCHIVELOG_CHANGE#	Highest NEXT_CHANGE# (from the V$ARCHIVED_LOG view) for an archived log
DATABASE_ROLE	Current role of the database; either primary or standby
SWITCHOVER_STATUS	Specifies whether switchover is allowed. Possible values are: ■ NOT ALLOWED - Either this is a standby database and the primary database has not been switched first or this is a primary database and there are no standby databases. ■ SESSIONS ACTIVE - Indicates that there are active SQL sessions attached to the primary or standby database that need to be disconnected before the switchover operation is permitted. Query the V$SESSION view to identify the specific processes that need to be terminated. ■ SWITCHOVER PENDING - This is a standby database and the primary database switchover request has been received but not processed. ■ SWITCHOVER LATENT - The switchover was in pending mode, but did not complete and went back to the primary database. ■ TO PRIMARY - This is a standby database and is allowed to switch over to a primary database. ■ TO STANDBY - This is a primary database and is allowed to switch over to a standby database. ■ RECOVERY NEEDED - This is a standby database that has not received the switchover request.

V$DATAFILE

The V$DATAFILE view provides datafile information from the control file. This view contains the following columns:

Column	Description
FILE#	File identification number
CREATION_CHANGE#	Change number at which the datafile was created
CREATION_TIME	Time stamp of the datafile creation
TS#	Tablespace number
RFILE#	Tablespace relative datafile number
STATUS	Type of file (system or user) and its status. Possible values are: ■ OFFLINE - cannot be written to ■ ONLINE - can be written to ■ SYSTEM - system datafile ■ RECOVER - needs recovery ■ SYSOFF - offline system
ENABLED	Describes how accessible the file is from SQL. Possible values are: ■ DISABLED - no SQL access allowed ■ READ ONLY - no SQL updates allowed ■ READ WRITE - full access allowed
CHECKPOINT_CHANGE#	SCN at last checkpoint
CHECKPOINT_TIME	Time stamp of the last checkpoint
UNRECOVERABLE_ CHANGE#	Last unrecoverable change number made to this datafile. This column is always updated when an unrecoverable operation completes.
UNRECOVERABLE_TIME	Time stamp of the last unrecoverable change
LAST_CHANGE#	Last change number made to this datafile. Set to NULL if the datafile is being changed.
LAST_TIME	Time stamp of the last change

Column	Description
OFFLINE_CHANGE#	Offline change number of the last offline range. This column is updated only when the datafile is brought online.
ONLINE_CHANGE#	Online change number of the last offline range
ONLINE_TIME	Online time stamp of the last offline range
BYTES	Current datafile size in bytes; 0 if inaccessible
BLOCKS	Current datafile size in blocks; 0 if inaccessible
CREATE_BYTES	Size when created, in bytes
BLOCK_SIZE	Block size of the datafile
NAME	Datafile name
PLUGGED_IN	Describes whether the tablespace is plugged in. The value is 1 if the tablespace is plugged in and has not been made read / write; 0 if not.
BLOCK1_OFFSET	The offset from the beginning of the file to where the Oracle generic information begins. The exact length of the file can be computed as follows: BYTES + BLOCK1_OFFSET
AUX_NAME	The auxiliary name that has been set for this file

V$LOG

The V$LOG view contains log file information from the online redo logs. This view contains the following columns:

Column	Description
GROUP#	Log group number
THREAD#	Log thread number
SEQUENCE#	Log sequence number
BYTES	Size of the log in bytes
MEMBERS	Number of members in the log group
ARCHIVED	Archive status
STATUS	Indicates the log status. Possible values are: ■ UNUSED - The online redo log has never been written to. This is the state of a redo log that was just added, or just after a RESETLOGS when it is not the current redo log. ■ CURRENT - This is the current redo log. This implies that the redo log is active. The redo log could be open or closed. ■ ACTIVE - The log is active but is not the current log. It is needed for failure recovery. It may be in use for block recovery. It might or might not be archived. ■ CLEARING - The log is being re-created as an empty log after an ALTER DATABASE CLEAR LOGFILE statement. After the log is cleared, the status changes to UNUSED. ■ CLEARING_CURRENT - The current log is being cleared of a closed thread. The log can stay in this status if there is some failure in the switch, such as an I/O error writing the new log header. ■ INACTIVE - The log is no longer needed for instance recovery. It may be in use for managed recovery. It might or might not be archived. ■ INVALIDATED - Archived the current redo log without a log switch.
FIRST_CHANGE#	Lowest SCN in the log
FIRST_TIME	Time of first SCN in the log

V$LOGFILE

The V$LOGFILE view contains information about the online redo logs. This view contains the following columns:

Column	Description
GROUP#	Redo log group identifier number
STATUS	Status of this log member. Possible values are: ■ INVALID - file is inaccessible ■ STALE - contents are incomplete ■ DELETED - file is no longer used ■ blank (no value listed) - file is in use
MEMBER	Redo log member name
TYPE	Specifies whether this is a standby log or an online log. Possible values are: ■ STANDBY ■ ONLINE

V$LOG_HISTORY

The V$LOG_HISTORY view contains log history information from the control file. This view contains the following columns:

Column	Description
RECID	Control file record ID
STAMP	Control file record stamp
THREAD#	Thread number of the archived log
SEQUENCE#	Sequence number of the archived log
FIRST_CHANGE#	Lowest SCN in the log
FIRST_TIME	Time of first entry (lowest SCN) in the log
NEXT_CHANGE#	Highest SCN in the log

V$MANAGED_STANDBY

The V$MANAGED_STANDBY view displays current and status information information for some Oracle database server processes related to Data Guard.

Note: The information in this view does not persist across an instance shutdown.

The V$MANAGED_STANDBY view contains the following columns:

Column	Description
PROCESS	Type of process whose information is being reported. Possible values are: ■ RFS - remote file server ■ MRP0 - detached recovery server process ■ MR(fg) - foreground recovery session
PID	Operating system process identifier of process

Column	Description
STATUS	Current process status. Possible values are: ■ UNUSED - no active process ■ ALLOCATED - process is active but not currently connected to a primary database client ■ CONNECTED - network connection established to a primary database client ■ ATTACHED - process is actively attached and communicating to a primary database client ■ IDLE - process is not performing any activities ■ ERROR - process has failed ■ OPENING - process is opening the archived redo log ■ CLOSING - process has completed archival and is closing the archived redo log ■ WRITING - process is actively writing archived redo log data ■ RECEIVING - process is receiving network communication ■ ANNOUNCING - process is announcing the existence of a potential dependent archived redo log ■ REGISTERING - process is registering the existence of a completed dependent archived redo log ■ WAIT_FOR_LOG - process is waiting for the archived redo log to be completed ■ WAIT_FOR_GAP - process is waiting for the archive gap to be resolved ■ APPLYING_LOG - process is actively applying the archived redo log to the standby database
CLIENT_PROCESS	Identifies the corresponding primary database process. Possible values are: ■ ARCHIVAL - foreground (manual) archival process (SQL) ■ ARCH - background ARC*n* process ■ LGWR - background LGWR process
CLIENT_PID	Operating system process identifier of the client process
CLIENT_DBID	Database identifier of the primary database
GROUP#	Standby redo log group

Column	Description
THREAD#	Archived redo log thread number
SEQUENCE#	Archived redo log sequence number
BLOCK#	Last processed archived redo log block number
BLOCKS	Size of the archived redo log in 512-byte blocks
DELAY_MINS	Archived redo log delay interval in minutes
KNOWN_AGENTS	Total number of standby database agents processing an archived redo log
ACTIVE_AGENTS	Number of standby database agents actively processing an archived redo log

V$STANDBY_LOG

The V$STANDBY_LOG view contains the following columns:

Column	Description
GROUP#	Log group number
THREAD#	Log thread number
SEQUENCE#	Log sequence number
BYTES	Size of the log in bytes
USED	Number of bytes used in the log
ARCHIVED	Archive status
STATUS	Indicates the log status. Possible values are: ■ UNUSED - The online redo log has never been written to. This is the state of a redo log that was just added, or just after a RESETLOGS when it is not the current redo log. ■ CURRENT - This is the current redo log. This implies that the redo log is active. The redo log could be open or closed. ■ ACTIVE - The log is active but is not the current log. It is needed for failure recovery. It may be in use for block recovery. It might or might not be archived. ■ CLEARING - The log is being re-created as an empty log after an ALTER DATABASE CLEAR LOGFILE statement. After the log is cleared, the status changes to UNUSED. ■ CLEARING_CURRENT - The current log is being cleared of a closed thread. The log can stay in this status if there is some failure in the switch, such as an I/O error writing the new log header. ■ INACTIVE - The log is no longer needed for instance recovery. It may be in use for managed recovery. It might or might not be archived. ■ INVALIDATED - Archived the current redo log without a log switch.
FIRST_CHANGE#	Lowest SCN in the log
FIRST_TIME	Time of first SCN in the log

Column	Description
LAST_CHANGE#	Last change number made to this datafile. Set to NULL if the datafile is being changed.
LAST_TIME	Time stamp of the last change

Part III

Appendixes and Glossary

This part contains the following:

A

Troubleshooting the Standby Database

This appendix provides help troubleshooting a standby database. This appendix contains the following sections:

- Problems During Standby Database Preparation
- Problems Switching Over to a Standby Database

A.1 Problems During Standby Database Preparation

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

- The Standby Archive Destination Is Not Defined Properly
- The Standby Site Does Not Receive Logs Archived by the Primary Database
- You Cannot Mount the Standby Database

A.1.1 The Standby Archive Destination Is Not Defined Properly

If the STANDBY_ARCHIVE_DEST initialization parameter is not defined as a valid directory name on the standby site, the Oracle database server will not be able to determine the directory in which to store the archived redo logs. Check the DESTINATION and ERROR columns in the V$ARCHIVE_DEST view. For example, enter:

```
SQL> SELECT destination, error FROM v$archive_dest;
```

Make sure the destination is valid.

A.1.2 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).

A.1.3 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 or RMAN. You cannot use the following types of control file backups:
  - An operating system-created backup
  - A backup created using an `ALTER DATABASE` statement *without* the `STANDBY` option

A.2 Problems Switching Over to a Standby Database

If you encounter a problem switching over from a primary database to a standby database, it will probably be one of the following:

- Prepare to Switchover Fails
- Startup of Second Database Fails
- Archived Redo Logs Are Not Applied to the Standby Database After Switchover

A.2.1 Prepare to Switchover Fails

`ALTER DATABASE COMMIT TO SWITCHOVER TO PHYSICAL STANDBY` failed with ORA-01093 error "Alter database close only permitted with no sessions connected".

This error occurs because the `COMMIT TO SWITCHOVER` statement implicitly closed the database and, if there are any other user sessions connected to the database, the close fails.

Action: Make sure all user sessions are disconnected from the database. You can query the `V$SESSION` fixed view to see what sessions are still around. For example:

```
SQL> SELECT SID, PROCESS, PROGRAM FROM v$session;

       SID PROCESS   PROGRAM
---------- --------- -------------------------------------------------
         1 26900     oracle@dbuser-sun (PMON)
         2 26902     oracle@dbuser-sun (DBW0)
         3 26904     oracle@dbuser-sun (LGWR)
         4 26906     oracle@dbuser-sun (CKPT)
         5 26908     oracle@dbuser-sun (SMON)
         6 26910     oracle@dbuser-sun (RECO)
         7 26912     oracle@dbuser-sun (ARC0)
         8 26897     sqlplus@dbuser-sun (TNS V1-V3)
        11 26917     sqlplus@dbuser-sun (TNS V1-V3)

9 rows selected.
```

In the previous example, the first seven sessions are all server background processes. Among the two SQL*Plus sessions, one is the current SQL*Plus session issuing the query, and the other is an extra session that should be disconnected before the switchover operation.

A.2.2 Startup of Second Database Fails

Suppose the standby database and the primary database reside on the same site. After both the `ALTER DATABASE COMMIT TO SWITCHOVER TO PHYSICAL STANDBY` and the `ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY` statements are successfully executed, shut down and restart the standby database and the primary database. However, the startup of the second database fails with ORA-01102 error "cannot mount database in EXCLUSIVE mode."

This could happen during the switchover if you forget to set the `LOCK_NAME_SPACE` parameter in the initialization parameter file that is used by the standby database (that is, the original primary database). If the `LOCK_NAME_SPACE` parameter of the standby database is not set, the standby and the primary databases both use the same mount lock and cause the ORA-01102 error during the startup of the second database.

Action: Add `LOCK_NAME_SPACE=`*`unique_lock_name`* to the initialization parameter file used by the standby database and shut down and restart both the standby and the primary databases.

A.2.3 Archived Redo Logs Are Not Applied to the Standby Database After Switchover

The archived redo logs are not applied to the standby database after the switchover.

This may happen because some environment or initialization parameters have not been properly set after the switchover.

Action:

- Check the `tnsnames.ora` file at the primary site and the `listener.ora` file at the standby site. There should be entries for a listener at the standby site and a corresponding tnsname at the primary site.
- Start the listener at the standby site if it has not been started.
- Check if the `LOG_ARCHIVE_DEST_`*n* initialization parameters have been set to properly archive logs from the primary site to standby site. For example, query the `V$ARCHIVE_DEST` fixed view at the primary site as follows:

  ```
  SQL> SELECT DEST_ID, STATUS, DESTINATION FROM v$archive_dest;
  ```

  If you do not see an entry corresponding to the standby site, you need to set `LOG_ARCHIVE_DEST_`*n* and `LOG_ARCHIVE_DEST_STATE_`*n* initialization parameters.
- Set the `STANDBY_ARCHIVE_DEST` and `LOG_ARCHIVE_FORMAT` initialization parameters correctly at the standby site so that the archived redo logs are applied to the desired location.
- At the standby site, set the `DB_FILE_NAME_CONVERT` and `LOG_FILE_NAME_CONVERT` initialization parameters. Set the `STANDBY_FILE_MANAGEMENT` initialization parameter to `auto` if you want the standby site to automatically add new datafiles that are created at the primary site.

B

Manual Recovery

Although Oracle Corporation recommends that you use the managed recovery mode for your standby databases, you can also use **manual recovery mode.** You might choose manual recovery mode for any of the following reasons:

- Manual recovery mode allows you to have more than 10 standby databases, which is the limit with managed recovery.
- If you do not want to connect to the primary and standby databases over the Oracle Net network, then open the database in manual recovery mode.
- If you are using managed recovery mode and, for some reason, that mode no longer works, you can change to manual recovery mode.

This appendix explains how to work in manual recovery mode. It includes the following topics:

B.1 Preparing a Standby Database for Manual Recovery: Basic Tasks

Table B–1 summarizes 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 Oracle Net. If you do not wish to use Oracle Net to connect to the standby database, skip steps 4 and 5.

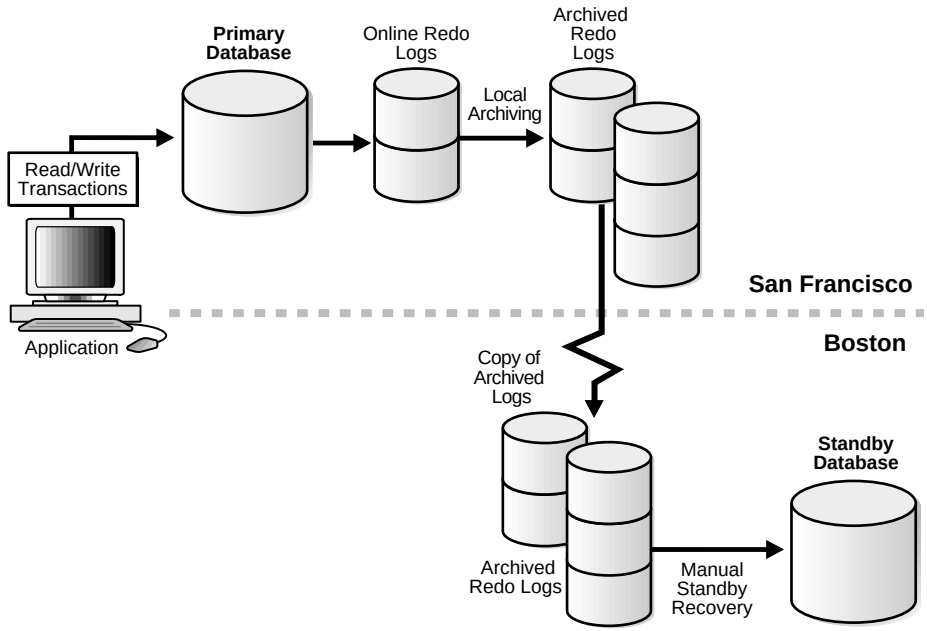
Table B–1 Task List: Preparing for Manual Recovery

Step	Task	Procedure
1	Either make a new backup of the primary database datafiles or access an old backup.	Section 2.3.2
2	Connect to the primary database and create the standby control file.	Section 2.3.3
3	Copy the backup datafiles and standby control file from the primary site to the standby site.	Section 2.3.4
4	If you want to create an Oracle Net connection to the standby database, create a service name.	Section 4.5 and LOCATION and SERVICE in Chapter 8
5	If you want to create an Oracle Net connection to the standby database, configure the listener on the standby site so that it can receive requests for connections to the standby instance.	Section 4.5
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.	Section 3.5
7	Start the standby instance and mount the standby database.	Section 4.3.1
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.	Section B.4

B.2 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. Figure B–1 shows a database in manual recovery mode.

Figure B–1 Standby Database in Manual Recovery Mode



This section contains the following topics:

- Initiating Manual Recovery Mode
- When Is Manual Recovery Required?

B.2.1 Initiating Manual Recovery Mode

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 Data Guard 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 10)
- Location specified by LOG_ARCHIVE_DEST_*n*

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 log transport services are not archiving logs automatically to the standby site, then manually copy the logs to the desired location on the standby site 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* parameters (where *n* is an integer from 1 to 10) or LOG_ARCHIVE_DEST_*n* 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 the Oracle database server generates archived redo logs, you must continually copy and apply them to the standby database to keep it current.

B.2.2 When Is Manual Recovery Required?

Manual recovery mode is required in a non-Data Guard environment. A non-Data Guard environment is one in which you manually:

- Transfer the archived redo logs from the primary site to the standby site
- Apply the archived redo logs to the standby database

Even if you implement a Data Guard environment, you may occasionally choose to perform manual recovery on the standby database. For example, you may choose to manually resolve an existing archive gap by using manual recovery mode.

B.3 Resolving Archive Gaps Manually

An **archive gap** is a range of archived redo logs created whenever you are unable to apply the next archived redo log generated by the primary database to the standby database. This section contains the following topics:

- What Causes Archive Gaps?
- Determining Whether an Archive Gap Exists
- Manually Transmitting the Logs in the Archive Gap to the Standby Site
- Manually Applying the Logs in the Archive Gap to the Standby Database

> **Note:** Typically, archive gaps are resolved automatically without the need for manual intervention. See Section 4.5 for more information about how log apply services automatically recover from gaps in the redo logs.

B.3.1 What Causes Archive Gaps?

An archive gap can occur whenever the primary database archives a log, but the log is not archived to the standby site. Because the standby database requires the sequential application of redo logs, media recovery stops at the first missing log encountered.

Archive gaps can occur in the following situations:

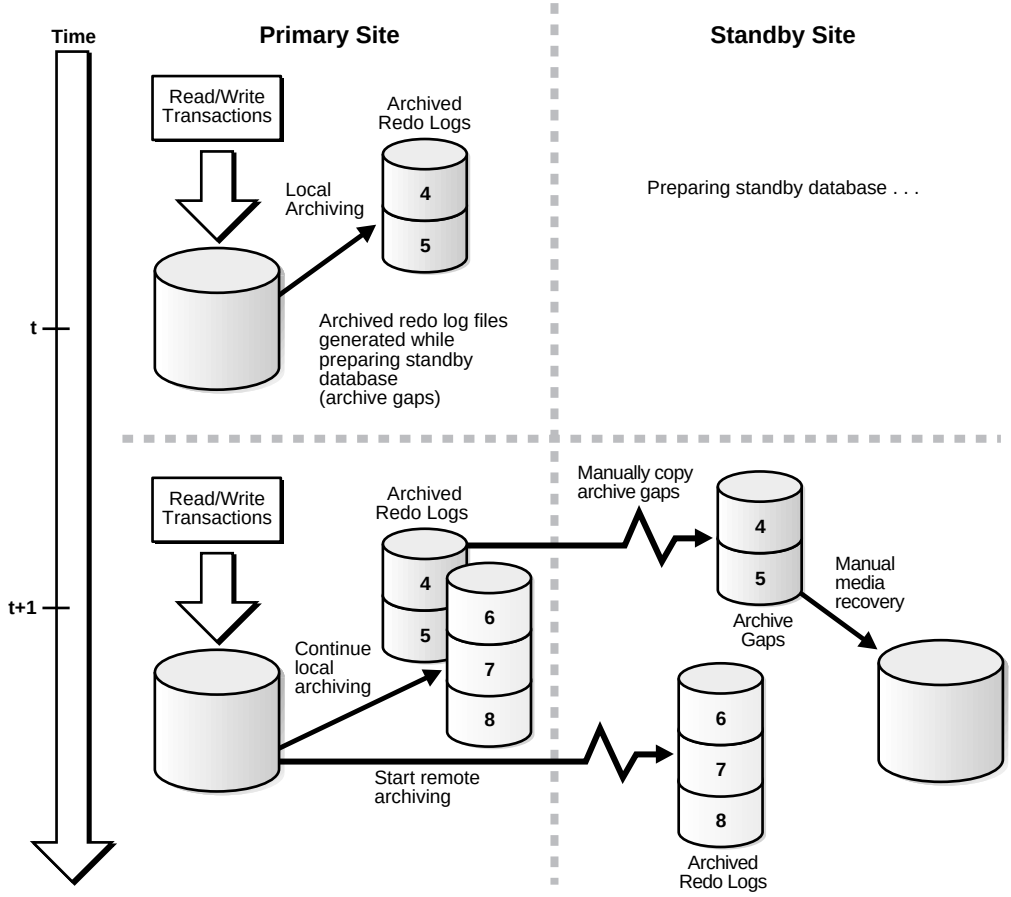
- Creation of the Standby Database
- Shutdown of the Standby Database When the Primary Database Is Open
- Network Failure Preventing the Archiving of Logs to the Standby Site

B.3.1.1 Creation of the Standby Database

One example of an archive gap 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 apply logs 101 to 150.

Another typical example of an archive gap occurs when you generate the standby database from a hot backup of an open database. For example, assume the scenario illustrated in Figure B–2.

Figure B–2 Manual Recovery of Archived Logs in an Archive Gap



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 log sequences 4 and 5.
3. At time *t + 1*, you start the standby instance.
4. `primary` archives log sequences 6, 7, and 8 to both the primary site and the standby site.

Archived log sequences 4 and 5 are now part of an archive gap, and these logs must be applied to the standby database.

B.3.1.2 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 when you change a control file parameter, such as MAXDATAFILE, in the primary database.

> **Note:** Performing a RESETLOGS operation on the primary database invalidates the standby database. If you reset the logs on the primary database, you must rebuild the standby database.

To avoid creating archive gaps, 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, the Oracle database server can create an archive gap.

> **Note:** If the standby site is specified as MANDATORY in one of the LOG_ARCHIVE_DEST_*n* parameters of the primary initialization parameter file, 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.

B.3.1.3 Network Failure Preventing the Archiving of Logs to the Standby Site

If you maintain a Data Guard 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:

- Section 3.4.2.3 for a detailed account of the significance of the OPTIONAL and MANDATORY attributes for standby archival
- Section 6.8 for a related scenario

B.3.2 Determining Whether an Archive Gap Exists

To determine whether there is an archive gap, query the V$ARCHIVE_GAP view. If an archive gap exists, the output of the query specifies the thread number and log sequence number of all logs in the archive gap. If there is *no* archive gap for a given thread, the query returns either no rows or an identical number in the LOW_SEQUENCE# and HIGH_SEQUENCE# columns.

To identify the logs in the archive gap:

1. Query the V$ARCHIVE_GAP view on the standby database as follows:

```
SQL> SELECT THREAD#, LOW_SEQUENCE#, HIGH_SEQUENCE#
  2> FROM V$ARCHIVE_GAP;
```

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

```
THREAD#    LOW_SEQUENCE#   HIGH_SEQUENCE#
---------- -------------   -------------
       1             460             463
       2             202             204
       3             100             100
```

Not every thread has an archive gap. As this example illustrates, the LOW_SEQUENCE# and HIGH_SEQUENCE# for thread 3 are identical, so no archive gap exists for this thread.

B.3.3 Manually Transmitting the Logs in the Archive Gap to the Standby Site

After you have obtained the log sequence numbers of the logs in the archive gap, you can obtain their filenames by querying the V$ARCHIVED_LOG view on the

primary site. 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 site as the primary database, or the standby database is on a remote site with a different directory structure than 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 copy logs in an archive gap to the standby site:

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

```
THREAD#    LOW_SEQUENCE#   HIGH_SEQUENCE#
---------- -------------   --------------
       1             460              463
       2             202              204
       3             100              100
```

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

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

```
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. On the standby site, 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. On the primary site, copy the archive gap 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/thread2_dest/arcr_2_202.arc /standby/arc_dest/log_2_202.arc
% cp /primary/thread2_dest/arcr_2_203.arc /standby/arc_dest/log_2_203.arc
% cp /primary/thread2_dest/arcr_2_204.arc /standby/arc_dest/log_2_204.arc
```

5. On the standby site, if the LOG_ARCHIVE_DEST and STANDBY_ARCHIVE_DEST parameter values are *not* the same, then copy the archive gap 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, the Oracle database server 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 archive gap logs to the standby database.

See Also: Section 4.9.3 and V$ARCHIVED_LOG in Chapter 10

B.3.4 Manually Applying the Logs in the Archive Gap to the Standby Database

After you have copied the logs in the archive gap to the standby site, you can apply them using the RECOVER AUTOMATIC statement.

To apply the archived redo logs in the archive gap:

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

   ```
   SQL> STARTUP NOMOUNT pfile=/oracle/admin/pfile/initSTBY.ora
   SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
   ```

2. Recover the database using the AUTOMATIC option:

   ```
   SQL> ALTER DATABASE RECOVER AUTOMATIC STANDBY DATABASE;
   ```

   The AUTOMATIC option automatically generates the name of the next archived redo log file needed to continue the recovery operation.

   After recovering the available logs, the Oracle database server 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 the Oracle database server has applied the available logs, by executing the following statement (or typing CTRL+C):

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

   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
   ```

Oracle Corporation recommends automatically applying the logs in the archive gap using the RECOVER MANAGED STANDBY DATABASE clause of the ALTER DATABASE statement.

See Also: Section 4.5 for additional information

B.4 Renaming Standby Files Manually

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.dbf	/standby/df1.dbf
/oracle/dbs/df2.dbf	/standby/df2.dbf
/data/df3.dbf	/standby/df3.dbf

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.dbf. 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.dbf' to '/standby/df3.dbf';
```

To rename a datafile manually:

1. Start up and mount the standby database (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.

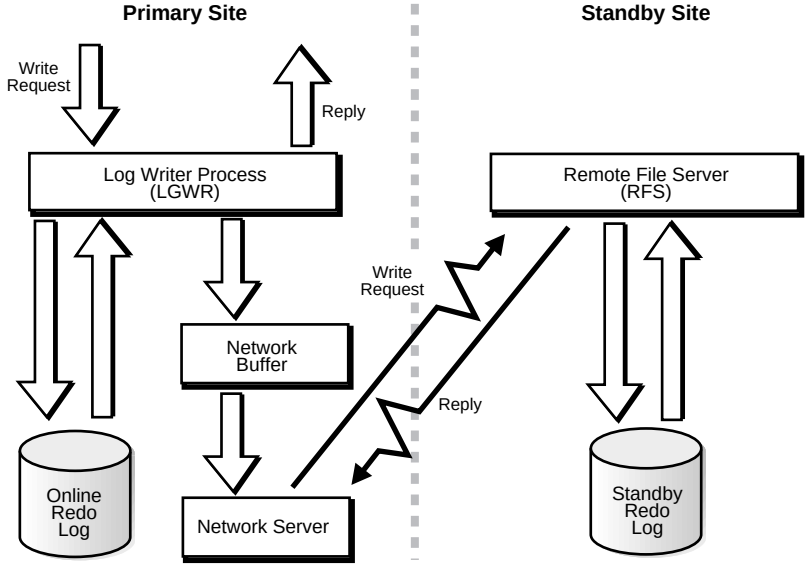
C

Log Writer Asynchronous Network I/O

This appendix describes the Data Guard algorithms for asynchronous network I/O when you use the log writer process (LGWR) to transmit primary database online redo log modifications to standby databases. This information is provided to help database administrators and support analysts know how asynchronous I/O is achieved, and how to diagnose and interpret asynchronous I/O-related network issues.

Figure C–1 shows the various processes involved in the asynchronous network I/O operation.

Figure C–1 Asynchronous Network I/O Processes



Data Guard achieves asynchronous network I/O using transparent network server processes. These network server processes are deployed by the LGWR process when the ASYNC attribute is specified for the LOG_ARCHIVE_DEST_*n* initialization parameters.

> **Note:** LGWR network server processes are different from LGWR I/O slave processes. However, both types of background processes are compatible with each other.

The network server processes are transparent because the remote file server (RFS) processes on the standby database do not know that network server processes are initiating the network I/O operations on the primary database. The network server processes mimic the same network I/O operations that occur when the LGWR process is set up for synchronous I/O operations. The network server process name is ora_lns*nn*_SID where *nn* is the arbitrary server number. For example, ora_lns5_prmy is background network server process number 5 for the database whose SID is prmy.

There is one network server process for each LGWR network connection to an asynchronous standby database. A network server process is created when the network connection to the standby database is established. A network server process remains active as long as the network connection to the standby database remains active. The abnormal termination of a network server process does not lead to instance shutdown. For example, if the system monitor (SMON) background process is unable to function, the Oracle instance ceases operation and fails. However, if one of the network server processes is terminated, the process monitor (PMON) background process wakes up and cleans up the resources that were being occupied by that network server process, and the Oracle instance continues to function. The PMON activity in cleaning up the resources is recorded in the corresponding alert log and trace files.

However, if a network server process terminates abnormally, the LGWR process discards the network connection for the remainder of the processing of the current online redo log. The network connection is reestablished, if possible, on the next log switch operation. Each network server process manages an internal network buffer, whose size is specified by the ASYNC attribute of the LOG_ARCHIVE_DEST_*n* initialization parameter. The network buffer resides in the private memory of the network server process. See Chapter 8 for information on the LOG_ARCHIVE_DEST_*n* initialization parameter.

The LGWR process sends the online redo modifications to the network server and the network server caches the online redo log modifications in the network buffer, if possible. If the online redo log modifications can be cached, the LGWR process continues operating as if the network I/O operation was successfully performed, when, in fact, it was not.

Because the network buffer resides on the primary database, the contents of this cache are susceptible to loss due to primary database system failure. To minimize this risk, the network server processes attempt to transmit the data to the standby database as long as the network does not become saturated.

Several events can also cause the network buffer to be transmitted to the standby database by the network server. These events are:

- If the LGWR request size exceeds the currently available buffer space, the existing network buffer is transmitted to the standby database. The LGWR process stalls until sufficient buffer space can be reclaimed, but this seldom occurs.
- A primary database online log switch forces the network buffer to be transmitted to the standby database.
- The primary database is shut down normally.

Note: An immediate shutdown of the primary database results in the network buffer contents being discarded because the network server process is shut down.

A standby database shutdown also causes the network buffer contents to be discarded.

- The primary database has no online redo log activity for a period of time. The duration of database inactivity cannot be controlled by the user.
- If the rate of redo generation exceeds the runtime network latency, the LGWR request is buffered if sufficient space is available. Otherwise, the existing buffer is transmitted to the standby database. The LGWR process stalls until sufficient buffer space can be reclaimed, but this seldom occurs.

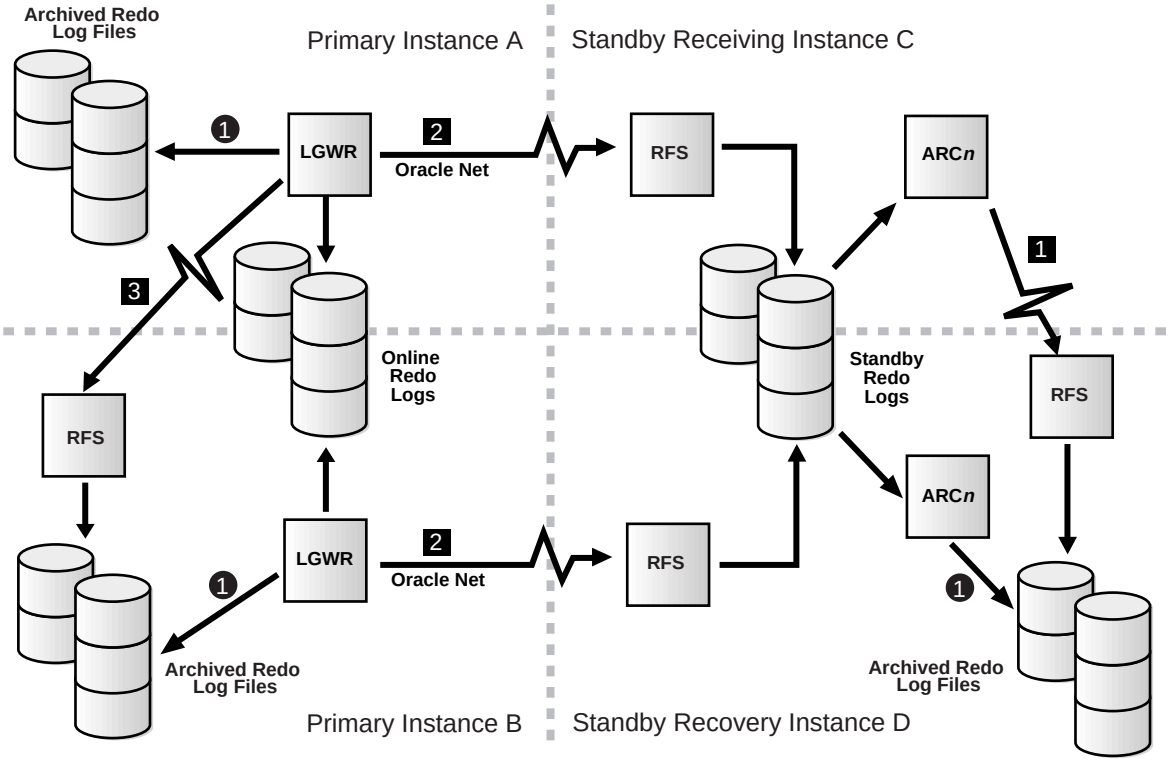
The network server process always performs synchronous network I/O; however, the LGWR process interprets this as asynchronous network I/O, because it is not blocked by the actual network I/O operation.

D

Standby Database Real Application Cluster Support

Oracle9*i* provides the ability to perform true database archival from a primary database to a standby database when both databases reside on a Real Application Cluster. This allows you to separate the log transport services processing from the log apply services processing on the standby database, thereby improving overall primary and standby database performance. Figure D–1 illustrates a standby database configuration on a Real Application Cluster.

Figure D–1 Real Application Cluster Standby Database



In Figure D–1, the numbers within circles indicate local connections and the numbers within boxes indicate remote connections.

When you use the standby database on a Real Application Cluster, any instance can receive archived logs from the primary database; this is the **receiving instance.** However, the archived logs must ultimately reside on disk devices accessible by the node on which the managed recovery operation is performed; this is the **recovery instance.** Transferring the standby database archived logs from the receiving instance to the recovery instance is achieved using the cross-instance archival operation, performed on the standby database.

See Also: Section 5.9.1

The standby database cross-instance archival operation requires use of standby redo logs as the temporary repository of primary database archived logs. Using the standby redo logs not only improves standby database performance and reliability, but also allows the cross-instance archival operation to be performed. However, because standby redo logs are required for the cross-instance archival operation, the primary database must use the log writer process (LGWR) to perform the primary database archival operation.

To set up a standby database in a Real Application Cluster environment:

Perform the following steps to set up log transport services on the standby database:

1. Create the standby redo logs. In a Real Application Cluster environment, the standby redo logs must reside on disk devices shared by all instances, such as raw devices.
2. On the recovery instance where the managed recovery process (MRP) is to operate, define the archived log destination to archive locally, because cross-instance archival is not necessary. This is accomplished using the `LOCATION` attribute of the `LOG_ARCHIVE_DEST_1` initialization parameter.

   Note: Unlike a primary database, a standby database is not required to have a local archived log destination.

3. On the receiving instance, define the archived log destination to archive to the node where the MRP is to operate. This is accomplished using the `SERVICE` attribute of the `LOG_ARCHIVE_DEST_1` initialization parameter.
4. Start the ARC*n* process on all standby database instances.
5. Start the MRP on the recovery instance.

To set up a primary database in a Real Application Cluster environment:

Perform the following steps to set up log transport services on the primary database:

1. On all instances, designate the LGWR process to perform the archival operation.

2. Designate the standby database as the receiving node. This is accomplished using the SERVICE attribute of the LOG_ARCHIVE_DEST_*n* initialization parameter.

Ideally, each primary database instance should archive to a corresponding standby database instance. However, this is not required.

Glossary

activate

To issue a SQL `ALTER DATABASE ACTIVATE STANDBY DATABASE` statement. This statement starts a forced database failover operation.

See also failover *and* forced database failover

ARCH

Setting this attribute on the `LOG_ARCHIVE_DEST_`*n* initialization parameter indicates that the archiver process (ARC*n*) will create archived redo logs on the primary database and also transmit redo logs for archival at specified destinations.

See also archiver process (ARCn)

archive gap

A range of archived redo logs created whenever you are unable to apply the next archived redo log generated by the primary database to the standby database.

archived redo log

A copy of one of the filled members of an online redo log group made when the database is in ARCHIVELOG mode. As the LGWR process fills each online redo log with redo records, log transport services copies the log to one or more offline archive log destinations. This copy is the archived redo log, also known as the *offline redo log*.

See also ARCHIVELOG mode, online redo log, *and* redo log

ARCHIVELOG mode

The mode of the database in which log transport services archives filled online redo logs to disk. Specify the mode at database creation or by using the SQL `ALTER DATABASE ARCHIVELOG` statement. You can enable automatic archiving either dynamically using the SQL `ALTER SYSTEM ARCHIVE LOG START` statement or by setting the initialization parameter `LOG_ARCHIVE_START` to `true`.

Running your database in ARCHIVELOG mode has several advantages over NOARCHIVELOG mode. You can:

- Back up your database while it is open and being accessed by users
- Recover your database to any desired point in time

To protect your database that is in ARCHIVELOG mode in case of failure, back up your archived logs.

See also archived redo log, NOARCHIVELOG mode, *and* redo log

archiver process (ARC*n*)

On the primary database site, the process (or a SQL session performing an archival operation) that creates a copy of the online redo logs, either locally or remotely, for standby databases. On the standby database site, the ARC*n* process archives the standby redo logs to be applied by the managed recovery process (MRP).

See also ARCH

archiving

The operation in which the ARC*n* background process copies filled online redo logs to offline destinations. You must run the primary database in ARCHIVELOG mode to archive redo logs.

ARC*n*

See archiver process (ARCn)

availability

The measure of the ability of a system or resource to provide the desired service when required. Measured in terms of the percentage of time the device is accessible out of the total time it is needed. Businesses that require uninterrupted computing services have an availability goal of 100%, or 24x365. The sum of availability and downtime equals 100%.

See also downtime *and* switchover

backup control file

A backup of the control file. Make the backup by:

- Using the Recovery Manager utility (RMAN) `backup` or `copy` command. Never create a backup control file by using operating system commands.
- Using the SQL statement `ALTER DATABASE BACKUP CONTROLFILE TO 'filename'`.

Typically, you restore backup control files when all copies of the current control file are damaged; sometimes you restore them before performing certain types of point-in-time recovery.

See also control file *and* current control file

backup piece

A physical file in a format specific to RMAN. The file belongs to only one backup set. A backup set usually contains only one backup piece. The only time RMAN creates more than one backup piece is when you limit the piece size using the `MAXPIECESIZE` option of the RMAN `ALLOCATE` or `CONFIGURE` command.

See also backup set *and* Recovery Manager (RMAN)

backup set

An RMAN-specific logical grouping of one or more physical files called *backup pieces*. The output of the RMAN `backup` command is a backup set. Extract the files in a backup set by using the RMAN `restore` command. You can multiplex files into a backup set, that is, intermingle blocks from input files into a single backup set.

There are two types of backup sets:

- Datafile backup sets, which are backups of any datafiles or a control file. This type of backup set is compressed, which means that it only contains datafile blocks that have been used; unused blocks are omitted.
- Archive log backup sets, which are backups of archived redo logs.

See also backup piece *and* Recovery Manager (RMAN)

broker

A distributed management framework that automates and simplifies most of the operations associated with the creation, control, and monitoring of a Data Guard configuration. The broker includes two user interfaces: Oracle Data Guard Manager (a graphical user interface) and the Data Guard command-line interface.

child destination

A log transport services archiving destination that is configured to receive redo logs from the primary database and depends on the successful completion of archival operations for the parent destination.

See also destinations, destination dependency, *and* parent destination

closed backup

A backup of one or more database files taken while the database is closed. Typically, a closed backup is also a whole database backup (a backup of the control file and all datafiles that belong to a database). If you closed the database cleanly, then all the files in the backup are consistent. If you shut down the database using a `SHUTDOWN ABORT` or the instance terminated abnormally, then the backups are inconsistent.

See also consistent backup

consistent backup

A whole database backup (a backup of the control file and all datafiles that belong to a database) that you can open with the `RESETLOGS` option without performing media recovery. In other words, you do not need to apply redo logs to datafiles in this backup for it to be consistent. All datafiles in a consistent backup must:

- Have the same checkpoint system change number (SCN) in their headers, unless they are datafiles in tablespaces that are read-only or offline normal (in which case they will have a clean SCN that is earlier than the checkpoint SCN)
- Contain no changes past the checkpoint SCN
- Match the datafile checkpoint information stored in the control file

You can only make consistent backups after you have made a clean shutdown of the database. The database must not be opened until the backup has completed.

See also closed backup

control file

A binary file associated with a database that maintains the physical structure and time stamps of all files in that database. The Oracle database server updates the control file continuously during database use and must have it available for writing whenever the database is mounted or open.

See also backup control file *and* current control file

current control file

The control file on disk for a primary database; it is the most recently modified control file for the current incarnation of the database. For a control file to be considered current during recovery, it must not have been restored from backup.

See also backup control file *and* control file

current online redo log

The online redo log file in which the LGWR background process is currently logging redo records. Those files to which LGWR is not writing are called inactive.

When LGWR gets to the end of the file, it performs a log switch and begins writing to a new log file. If you run the database in ARCHIVELOG mode, then the ARC*n* process or processes copy the redo data into an archived redo log.

See also online redo log *and* redo log

data divergence

Data divergence occurs when data modifications occur on the primary database and when connectivity to the standby database is not available. Data divergence also occurs when you use the LGWR or ARC*n* process for asynchronous standby database archival operations.

See also data loss *and* no data divergence

Data Guard

A distributed computing system that prevents or minimizes losses due to unplanned events (for example, human errors, environmental disasters, or data corruption) as well as to planned downtime (such as for routine maintenance tasks).

data loss

The loss of data. Data loss occurs when you fail over to a standby database whose corresponding primary database is in the data divergence state.

See also data divergence *and* graceful database failover

datafile

A physical operating system file on disk that was created by the Oracle database server and contains data structures such as tables and indexes. A datafile can only belong to one database.

See also tablespace

delayed protection

A log transport services data availability mode that offers the lowest level of data protection. In this mode, the archiver process writes redo log data asynchronously to a standby database. In a failover operation, it is possible to lose data from one or more logs that have not yet been transmitted.

destination dependency

Configuring log transport services so that archiving redo logs to a specified destination is dependent upon the success or failure of archiving redo logs to another destination.

See also child destination, destinations, *and* parent destination

destinations

Log transport services allow the primary database to be configured to archive redo logs to up to ten local and remote locations called destinations.

See also child destination, destination dependency, *and* parent destination

downtime

The measure of the inability of a system or resource to provide the desired service when required. Measured in terms of the percentage or amount of time the device is not accessible out of the total time it is needed. The sum of availability and downtime equals 100%.

A period of time for performing routine maintenance tasks, such as hardware and software upgrades and value-added services is *planned downtime*. The computing system is not available for productive operations during such scheduled maintenance tasks.

See also availability *and* switchover

failover

An unplanned event resulting from system or software failure.

If standby redo logs are present on the standby database, the failover is graceful and can occur with no data loss or with minimal data loss. If standby redo logs are not present on the standby database, the failover is forced and may result in the loss of application data.

See also activate, forced database failover, graceful database failover, role transition, *and* switchover

fetch archive log (FAL) client

A background Oracle database server process. The initialization parameter for the FAL client is set on the standby site. The FAL client pulls archived redo logs from the primary site and initiates and requests the transfer of archived redo logs automatically when it detects an archive gap on the standby database.

fetch archive log (FAL) server

A background Oracle database server process that runs on the primary or other standby databases and services the fetch archive log (FAL) requests coming from the FAL client. For example, servicing a FAL request might include queuing requests (to send archived redo logs to one or more standby databases) to an Oracle database server that runs the FAL server. Multiple FAL servers can run on the same primary database at one time. A separate FAL server is created for each incoming FAL request. The initialization parameter for the FAL server is set on the standby site.

forced database failover

Failover is an unplanned event resulting from system or software failure. If standby redo logs are not present, the failover is forced. A forced database failover changes one of the standby databases into the role of primary database, but may result in the loss of application data even when standby redo logs are configured on the standby database. You must reinstantiate the original primary database and all other standby databases after a forced database failover.

See also activate, failover, *and* graceful database failover

graceful database failover

Failover is an unplanned event resulting from system or software failure. If standby redo logs are present, the failover is graceful. A graceful database failover changes one of the standby databases into the role of primary database, and automatically recovers some or all of the original primary data. Therefore, it is necessary to reinstantiate only the primary database. The other standby databases in the configuration do not need to be reinstantiated. This is also known as *terminal recovery.* A graceful database failover can occur with no data loss or with minimal data loss.

See also data loss, failover, forced database failover, *and* no data loss

guaranteed protection

A log transport services data availability mode that can be set to ensure that redo logs are available at a standby database before primary database processing can

continue. Guaranteed protection protects against data divergence. This is also known as *protected mode*.

See also no data divergence *and* no data loss

hot backup

See open backup

instant protection

A log transport services data availability mode that can be set to ensure that redo logs are available at a standby database before the current database transaction is committed. Instant protection does not protect against data divergence.

See also data divergence, no data divergence, *and* no data loss

LGWR

See log writer process (LGWR)

listener

An application that receives requests by clients and redirects them to the appropriate server.

log apply services

The component of the Data Guard environment that is responsible for maintaining the standby database in either a managed recovery mode or an open read-only mode. The managed recovery mode automatically maintains and applies archived redo logs to maintain transactional synchronization with the primary database, and allows transactionally consistent read-only access to the data. In a Data Guard environment, the log apply services component coordinates its activities with the log transport services component.

See also log transport services *and* role management services

log switch

The point at which LGWR stops writing to the active redo log file and switches to the next available redo log file. LGWR switches when either the active log file is filled with redo records or you force a switch manually.

If you run your database in ARCHIVELOG mode, log transport services archive the redo data in inactive log files into archived redo logs. When a log switch occurs and LGWR begins overwriting the old redo data, you are protected against data loss because the archived redo log contains the old data. If you run in

NOARCHIVELOG mode, log transport services overwrite old redo data at a log switch without archiving it. Hence, you lose all old redo data.

See also redo log

log transport services

The component of the Data Guard environment that is responsible for the archival of the primary database online redo logs. Log transport services provide for the management of archived redo log permissions, destinations, transmission, reception, and failure resolution. In a Data Guard environment, the log transport services component coordinates its activities with the log apply services component.

See also log apply services, no data divergence, no data loss, *and* role management services

log writer process (LGWR)

The background process that collects transaction redo and updates the online redo logs. The log writer process can also create local archived redo logs and transmit online redo to standby databases.

managed recovery mode

An environment in which the primary database automatically archives redo log files to the standby site, initiated by entering the following SQL statement:

```
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE;
```

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

managed recovery process (MRP)

The managed recovery process applies archived redo log information to the standby database.

managed standby environment

See standby database environment

manual recovery mode

An environment in which the primary database does not automatically archive redo log files to the standby site. In this environment, you must manually transfer archived log files to the standby site and manually apply them by issuing the following SQL statement:

```
ALTER DATABASE RECOVER STANDBY DATABASE;
```

This mode allows you to recover a standby database manually.

MRP

See managed recovery process (MRP)

no data divergence

A log transport services option that can be configured to ensure that the primary and standby databases are always synchronized by prohibiting access to the primary database if connectivity to at least one standby database becomes unavailable.

See also guaranteed protection, instant protection, *and* log transport services

no data loss

A log transport services option that can be configured to ensure that data modifications made to the primary database are not acknowledged until those data modifications are also available on (but not necessarily applied to) the standby database.

See also graceful database failover, guaranteed protection, instant protection *and* log transport services

NOARCHIVELOG mode

The mode of the database in which log transport services do not require filled online redo logs to be archived to disk. Specify the mode at database creation or change it by using the SQL `ALTER DATABASE` statement. Oracle Corporation does not recommend running in NOARCHIVELOG mode because it severely limits the possibilities for recovery of lost data.

See also ARCHIVELOG mode

node

See site

non-managed recovery mode

See manual recovery mode

offline redo log

See archived redo log

online redo log

A set of two or more files that records all changes made to datafiles and control files. Whenever a change is made to the database, the Oracle database server generates a redo record in the redo buffer. The LGWR process flushes the contents of the redo buffer into the online redo log.

Every database must contain at least two online redo log files. If you are multiplexing your online redo log, LGWR concurrently writes the same redo data to multiple files. The individual files are called members of an online redo log group.

See also archived redo log, current online redo log, *and* redo log

open backup

A backup of one or more datafiles taken while a database is open. This is also known as *hot backup*.

parent destination

A log transport services archiving destination that has a child destination associated with it.

See also child destination, destination dependency, *and* destinations

physical standby database

A standby database that is physically identical to the primary database because recovery applies changes block-for-block using the physical row ID.

See also standby database

planned downtime

See availability, downtime, *and* switchover

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.

See also site

protected mode

See guaranteed protection *and* no data divergence

rapid protection

A log transport services data availability mode that can be set to ensure that redo logs are written as quickly as possible to the standby database with as little effect as possible on the performance of the primary database.

read-only database

A database opened with the SQL statement `ALTER DATABASE OPEN READ ONLY`. As their name suggests, read-only databases are for queries only and cannot be modified. A standby database can be run in read-only mode, which means that it can be queried while still serving as an up-to-date emergency replacement for the primary database.

See also read-only mode

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.

See also read-only database

receiving instance

When you use the standby database on a Real Application Cluster, any instance can receive archived logs from the primary database; this is the receiving instance.

See also recovery instance

recovery catalog

A set of tables and views used by Recovery Manager (RMAN) to store information about Oracle databases. RMAN uses this data to manage the backup, restore, and recovery of Oracle databases. If you choose not to use a recovery catalog, RMAN uses the target database control file. You should not store the recovery catalog in your target database.

See also recovery catalog database *and* Recovery Manager (RMAN)

recovery catalog database

An Oracle database that contains a recovery catalog schema.

See also recovery catalog

recovery instance

The node where managed recovery is performed. Within a Real Application Clusters configuration, each primary instance directs its archived redo logs to this node of the standby cluster.

See also receiving instance

Recovery Manager (RMAN)

A utility that backs up, restores, and recovers Oracle databases. You can use it with or without the central information repository called a *recovery catalog*. If you do not use a recovery catalog, RMAN uses the database's control file to store information necessary for backup and recovery operations. You can use RMAN in conjunction with a media manager, which allows you to back up files to tertiary storage.

See also backup piece, backup set, *and* recovery catalog

redo log

A file containing redo records. There are three types of redo logs: online redo logs, standby redo logs, and archived redo logs.

The online redo log is a set of two or more files that records all changes made to datafiles and control files. The LGWR process records the redo records in the log. The current online redo log is the one to which LGWR is currently writing.

The standby redo log is an optional location where the standby database can store the redo data received from the primary site. This redo data can be stored on the standby site using either standby redo logs or archived redo logs.

The archived redo log, also known as the *offline redo log*, is a copy of the online redo log that has been copied to an offline destination. If the database is in ARCHIVELOG mode, the ARC*n* process or processes copy each online redo log to one or more archive log destinations after it is filled.

See also archived redo log, ARCHIVELOG mode, current online redo log, log switch, online redo log, *and* standby redo log

reliability

The ability of a computing system or software to operate without failing.

remote file server (RFS)

The remote file server process on the standby site receives archived redo logs from the primary database.

RMAN

See Recovery Manager (RMAN)

role management services

The component of the Data Guard environment that is responsible for the changing of database roles. Database role transitions include switchover and switchback, as well as failover if the primary database is unavailable due to an unplanned shutdown.

See also log apply services *and* log transport services

role transition

A database can be in one of two mutually exclusive roles: primary or standby. You can change these roles dynamically as a planned transition (switchover) or you can change these roles as a result of an unplanned failure (failover).

See also failover *and* switchover

rolling upgrade

A software installation technique that allows a clustered system to continue to provide service while the software is being upgraded to the next release. This process is called a rolling upgrade because each site in the cluster is upgraded and rebooted in turn, until all sites have been upgraded. While each site is temporarily offline, the other sites take over the entire workload.

site

A managed unit of failover in a Data Guard configuration. A database is replicated across a set of sites, one replicant per site. Dependent applications are instantiated on a site. When a site holding a primary role fails, another site holding the standby role takes over the primary role responsibility and provides the desired service to users. Sites may be one of several types of *nodes*, which vary from one another in the degree of hardware complexity and software management.

See also primary site *and* standby site

standby database

An identical copy of a primary database that you can use for disaster protection. You can update your standby database with archived redo logs from the primary database to keep it current. Should a disaster destroy the primary database, you can fail over to the standby database and make it the new primary database. A standby database has its own initialization parameter file, control file, and datafiles.

See also physical standby database

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
- Number of host systems used by the databases
- Directory structures of the databases
- Network configuration

A configuration in which a primary database automatically archives redo logs to a standby site is a *managed standby environment*. 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.

standby redo log

The standby redo log is an optional set of log files where the standby database can store the redo data received from the primary site. (Redo data also can be stored on the standby site using archived redo logs.) Standby redo logs are created using the `ADD STANDBY LOGFILE` clause of the SQL `ALTER DATABASE` statement. Additional log group members can be added later to provide another level of reliability against disk failure on the standby site.

See also redo log

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.

See also site

switchback

A switchover performed in reverse that results in the original primary database becoming a new primary database. Once you have performed a database switchover operation, you can switch back to your original Data Guard configuration.

See also switchover

switchover

The process of intentionally switching a database role from primary to standby, as well as from standby to primary, without resetting the online redo logs of the new primary database. This is a switchover operation, instead of a failover operation, because there is no loss of application data, and there is no need to reinstantiate the standby databases, including other standby databases not involved in the switchover operation. You cannot use a switchover operation to perform a rolling upgrade of Oracle software. However, it may be possible to use a switchover operation to perform a hardware-based rolling upgrade.

See also availability, downtime, failover, role transition, *and* switchback

system change number (SCN)

A stamp that defines a committed version of a database at a point in time. The Oracle database server assigns every committed transaction a unique SCN.

tablespace

One or more logical storage units into which a database is divided. Each tablespace has one or more physical datafiles exclusively associated with it.

See also datafile

TAF

See transparent application failover (TAF)

target database

In RMAN, the database that you are backing up or restoring.

temporary tablespace

Tablespace of temporary tables created during the processing of an SQL statement. This allows you to add tempfile entries in read-only mode for the purpose of making queries. You can then perform on-disk sorting operations in a read-only database without affecting dictionary files or generating redo entries.

terminal recovery

See graceful database failover

transparent application failover (TAF)

The ability of client applications to automatically reconnect to a database and resume work after a failover occurs.

Index

A

B

C

D

E

F

G

I

J

L

M

N

O

P

Q

R

S

T

U

V