



Oracle8i Standby Database

Lawrence To

**Centers of Expertise
Oracle Support Services
Oracle Corporation**

June 7 1999

Final

TABLE OF CONTENTS

INTRODUCTION.....	3
AUDIENCE	3
DOCUMENT OVERVIEW	3
BACKGROUND.....	4
THE STANDBY DATABASE CONFIGURATION	4
STANDBY DATABASE RECOVERY AND RECOVERY BASICS.....	6
CHOICE, DESIGN AND PLANNING.....	8
STANDBY DATABASE -- THE CORRECT CHOICE?	8
STANDBY DATABASE -- ONLY A SUBSET OF AN AVAILABILITY SOLUTION.....	11
DESIGN AND PLANNING ISSUES.....	13
8I STANDBY DATABASE ENHANCEMENTS.....	15
PREPARATION.....	17
CREATION OF STANDBY DATABASE	17
INITIALIZATION PARAMETERS.....	18
DEPLOYMENT.....	20
REMOTE ARCHIVAL OF ONLINE REDO LOGS	20
MOUNTING AND SHUTTING DOWN A STANDBY DATABASE	23
INITIATING STANDBY RECOVERY.....	23
MAINTENANCE AND MONITORING	25
ACTIVATION.....	30
ACTIVATING STANDBY DATABASE FOR PRODUCTION	30
OPENING STANDBY DATABASE AS READ ONLY.....	31
BIBLIOGRAPHY.....	34

ATTENTION READERS INCLUDING ORACLE EMPLOYEES. No part of this document may be reproduced without prior permission from the publishers, Centers of Expertise, Oracle Support Services.

Copyright © Oracle Corporation 1999 All rights reserved. Printed in the U.S.A.

Author: Lawrence To
Previous Authors: Brian Quigley, Sajjad Masud, Jasmin Nadic, Lawrence To
Previous Paper Name: Oracle8 Standby Database
Reviewers: Rick Anderson, Mark W. Johnson, Erhan Odok, Stefan Pommerenk, Roderick Manalac, Carol Colrain

Oracle is a registered trademark of Oracle Corporation. Oracle7, Oracle8, Oracle8i and Net8 are trademarks of Oracle Corporation.

All other products or company names are used for identification purposes only, and may be trademarks of their respective owners.

INTRODUCTION

Audience

You are interested in the Oracle8i standby database feature. The primary audience for this paper is professionals capable of administering a release Oracle8i database. Reference material appears at the end of this paper.

Document Overview

Oracle initiated official support for standby databases in Oracle7 Release 7.3. With each subsequent release, Oracle has enhanced this functionality. This document will highlight and explain how to best exploit the new Oracle8i standby database features.

After reading this paper, you should understand the following:

- The environments where a standby database is suitable
- The differences between Oracle8 and Oracle8i standby databases
- How to configure and implement a standby database
- What to monitor to safeguard the integrity of your standby database
- How to leverage the new Oracle8i standby features

This paper is structured with the following components:

- **Background**

This major section describes terminology and concepts behind standby databases and Oracle recovery.

- **Choice, Design And Planning**

This major section guides the reader in deciding if a standby database is the right choice for their business and environment.

- **Oracle8i Enhancements**

This major section describes the standby database differences and enhancements between Oracle versions.

- **Preparation**

This major section reviews preparations before deploying the standby feature.

- **Deployment**

This major section describes how to deploy an Oracle8i standby database.

- **Maintenance and Monitoring**

This major section describes how to maintain the standby database environment. The section also describes how and what to check on both the primary and standby databases to preserve standby database integrity.

- **Activation**

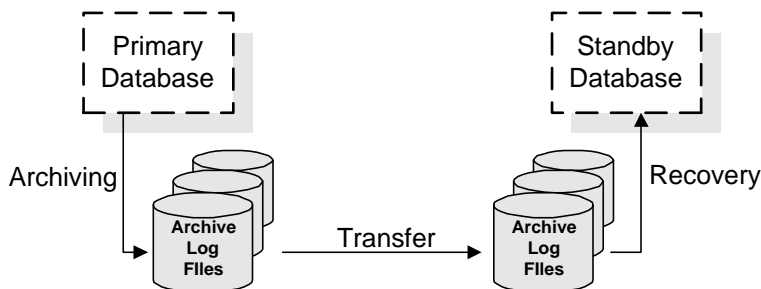
This major section describes how to activate the standby as a new production system and how to activate the standby database for READ ONLY purposes.

BACKGROUND

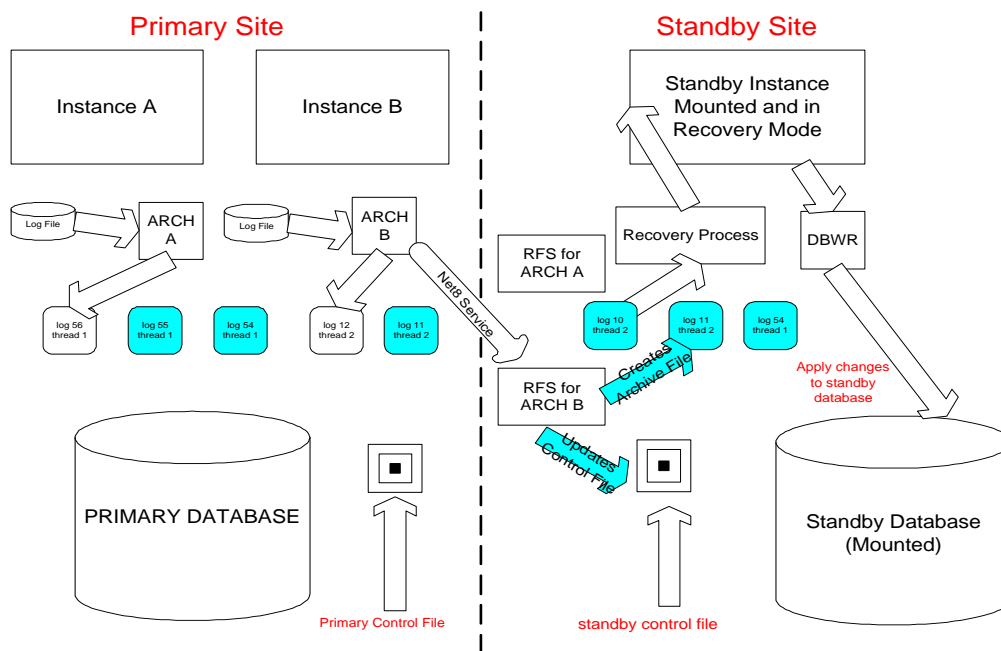
This major section provides a basic overview of standby databases and its recovery algorithms.

The Standby Database Configuration

A *primary database* is a database that contains production data. A *standby database* is a copy of the primary database leveraged essentially for disaster recovery. From the diagram below, the standby database is normally in a state of recovery, applying the primary database's archive logs. When the standby database is *activated*, the standby database becomes the production database and is no longer compatible with the primary database.



The next diagram illustrates an Oracle8i standby database configuration. The primary site can consist of one to many Oracle instances. Each instance will be associated with one thread of redo. All threads of redo need to be sent to the standby site. In versions 7.3 and higher, a standby control file exists. In Oracle8i, Oracle can be configured to send archives automatically to a local and standby archive destination. For each archiver (ARCH) process on the primary database, a Remote File Server (RFS) process will be spawned and persistent on the standby database instance. The RFS process consumes IO from the primary ARCH processes and creates the standby archive files. Subsequently, it updates the standby control file for each successfully archived file. Another enhancement of Oracle8i is "managed" recovery. Once mounted, the standby database can be placed in managed recovery mode. A recovery process will automatically check the standby control file and apply any relevant archive files.



Database Configuration and Implementation – Primary and Standby databases must reside on the same hardware and base operating system. The database versions should also be identical. At the very least, standby and primary databases should never cross major releases (e.g. 8.0 and 8.1 or 7.3 and 8.0). The `init.ora` parameter `compatible` must be identical, and configurations with different releases need to be tested and validated.

Creation of the standby can be briefly outlined as follows:

- Primary site
 - establish monitoring and file transfer scripts
 - backup initialization file(s) and database files
 - create standby controlfile
- Transfer files to the standby site
- Standby site
 - establish monitoring scripts
 - restore data files and standby control files
 - modify `init.ora` files
 - mount the standby database using the standby controlfile
 - initiate standby database in recovery mode

After standby database creation, the standby needs to be in recovery mode and continuously applying archives to ensure changes from the primary are *propagated* to the standby. Maintenance should be automatic so that propagation occurs quickly thus ensuring that the standby is very *current* with the primary. How closely the standby concurs with the primary depends on how quickly changes are propagated to the standby, your service level requirements (e.g. MTTR=30 minutes) and the type of outage protection (e.g. user error, loss of primary site). For example, if your MTTR is 10 minutes, you need to configure your total times to log switch, archive a log, propagate a log, and recover to be less than 10 minutes. Furthermore, if the standby database must protect from user errors such as a `drop table` command, the detection mechanism must alert and stop the standby database from applying the log that contains the change or corruption.

Maintenance can be briefly outlined as follows:

- Primary site
 - continue archiving to archive log files
 - monitor for any errors or NOLOGGING operations
- Transfer completed archive log files to the standby site
- Standby site
 - apply archives to standby database
 - monitor for standby database status and errors

Database recovery and monitoring are the keys to a successful standby database configuration. Database recovery updates the standby database with all the new changes. Monitoring is required to catch errors and special events such as adding a data file and NOLOGGING operations that disrupt recovery or upset the data similarities between the databases.

Important: The archive stream must not be interrupted since recovery cannot skip a lost or corrupted archive log.

Standby Database Recovery and Recovery Basics

The primary premise behind a standby database is to provide a recent or exact copy of the production database when the primary database fails. To stay current with all the primary database changes, Oracle leverages its robust recovery mechanisms and algorithms. The following section provides an overview of Oracle recovery with examples relevant to a standby database.

Oracle has four different recovery categories

1. Process or block recovery is required when a process dies while modifying a cache buffer.
2. Instance recovery is required when one instance, out of a multiple instance environment, fails.
3. Crash recovery is required when all instances of a database fail.
4. Media or standby recovery is required when rolling forward an older copy of a data file or database.

The first three recovery categories are detected automatically, and the current data sets are recovered using the online redo logs. The fourth category relies on restoring a consistent or inconsistent backup of one or more data files and initiating recovery manually. Standby database recovery falls into the fourth category. Media and standby database recovery depend on archive logs (copies of the online redo logs) to update a consistent (cold backup) or inconsistent (hot backup or RMAN online backup) copy of the datafiles.

All the recovery categories employ a number of algorithms and rules to ensure that data remains consistent and recoverable. The important ones are

- Page fix
- Log force at commit
- Write ahead logging
- Checkpoints
- Online log switching.

Page fix rule guarantees cache buffer consistency in case of process failure. Before changing a cache buffer, a user process must pin the buffer in exclusive mode, create redo and undo changes, and write redo and undo changes. The fix rule prevents any access to the buffer until the buffer is consistent and its “version” has been advanced. If the process fails, block recovery can restore the consistency of the block by reading the on disk copy of the block and applying the redo and undo changes. In case a primary database and all its processes crash, this rule helps to guarantee that all “current” changes required for block or database consistency is found in the online redo log. If the current redo log is available to the standby database, you can apply the last committed transactions and rollback any uncommitted transactions to the standby database.

Log-force-at-commit mechanism guarantees that all operations to redo and rollback a transaction are written to the redo log at commit time. The online redo logs store redo entries containing data and undo block changes. In case of a primary database failure, the standby database will contain all committed transactions if the online redo logs are applied.

Write ahead logging ensures that the log writer (LGWR) process is always ahead of database writer (DBWR) process. Before a process modifies a data block in the buffer cache, it must write redo records of the change and its undo changes to the log buffer. Similarly, DBWR writes a dirty buffer to disk only after LGWR confirms that redo and undo changes for that data block version are written to log files. Since we rely on the redo records within the redo logs for all recovery, the log files need to contain all data to restore database and transaction consistency for all cache-managed data blocks. This algorithm guarantees that Oracle can always advance (roll forward) a data block with subsequent changes from the redo.

Checkpoints demarcate points in the redo stream where all corresponding data blocks are written to the datafiles. All corresponding data block changes prior to the checkpoint System Commit Number (SCN) are guaranteed to be written to disk. Oracle uses five types of checkpoints: mini (e.g. drop table command), data file, tablespace, thread,

and database. In most cases, recovery will begin at the lowest, last successful checkpoint. At a minimum, the database checkpoint needs to advance before the corresponding online redo log may be overwritten. During standby database recovery, Oracle checkpoints datafiles and controlfiles after applying an archive log or after canceling recovery successfully (without incurring an error or process death). Checkpoints ensure that data is persistently stored in the datafiles and subsequent recovery actions need not apply old archives.

Before reusing an online redo log, **online log switching** verifies that all changes to an online redo log is

1. written to the data files and
2. archived to all **mandatory** archive destinations if the database is in ARCHIVELOG mode

The first condition allows Oracle to perform block level, crash or instance recovery automatically by applying the online redo logs. The second point safeguards all redo changes via the archive logs for media or standby database recovery. In Oracle8i, you can configure multiple mandatory archive destinations including a remote archive destination for the standby database. Before overwriting an online redo log, all mandatory archive destinations need to be written successfully.

Except for operating system errors, such as lost writes or media corruption, the Oracle recovery algorithms described above provide the necessary requirements for a complete standby database implementation.

CHOICE, DESIGN AND PLANNING

This major section gives some insight in determining if a standby database is a suitable solution for your business.

Standby Database -- The Correct Choice?

Oracle standby database can be leveraged for

- Disaster recovery if remote
- Failover
- Reporting database or query database
- Planned outage solution

The most common use of standby database is for disaster recovery; however, more customers are leveraging standby databases as a failover solution especially when hardware failover or OPS is not implemented. When configured properly, standby databases provide fast and sophisticated unplanned outage protection. The main disadvantages are

- Lack of scalability
- Cost
- UNRECOVERABLE or NOLOGGING operations are not propagated
- Data loss due to loss of online redo logs, if geo-mirroring is not leveraged

However, the question, "Is the standby database the right choice?" remains especially since other failover and disaster recovery solutions exist. Some alternative solutions include

- Oracle Parallel Server (OPS)
- Oracle Replication or Custom Replication
-
- Hardware Failover Solutions
- Third Party Disaster Recovery Solutions using Geo-Mirroring or Hardware Replication

Before implementing an Oracle standby database, you need to determine if a standby database satisfies your business needs. The following chart describes some of the requirements and alternatives to consider in making this determination.

Requirements	Examples or clarifications	Standby Database Characteristics	Alternatives
Disaster Recovery	Earthquakes, Fires, and Tornadoes that cripple the primary data center.	Provides an excellent disaster recovery solution if and only if the standby site is physically separate from the primary site and the disaster.	Replicated Site, Third Party mirroring solutions such as EMC SRDF.
Failover	Loss of primary node. Machine crash.	Can provide failover, but other solutions may be more suitable.	Oracle Parallel Server (OPS) and Hardware Failover

No Log Loss	Loss of the current online redo logs, current transactions, data corruption.	Can prevent data loss due to loss of current online redo logs by implementing a geo-mirroring solution. Data corruption is not prevented if it resides in the online redo logs and is applied on standby site.	Replicated Site, Third Party mirroring solutions such as EMC SRDF. They will have similar complications. More complex solutions (e.g. transaction split) have enormous cost and complexity.
Short Mean Time To Recovery – MTTR (15+ minutes)	<p>MTTR can be interpreted as either</p> <ul style="list-style-type: none"> • full elapsed time between a database crash and client reconnection or • elapsed time before the database becomes available again. <p>Customer must provide the clarification.</p>	Can provide low MTTR if recovery speed and shipment of archive logs are optimized.	For non-disaster scenarios such as node failure, OPS and Hardware Failover provide shorter MTTR. For disaster scenarios, replicated sites possibly provide shorter MTTR.
Higher Scalability	Provide resources for higher throughput and load.	Does NOT provide production scalability gains.	OPS and Replication can provide potential enhanced scalability.
Minimal performance impact on Primary Site.	Performance impact on the primary site and database.	Minimum to no performance impact on primary. Shipment of archives will require high network bandwidth and relatively low network latency.	Replication solutions incur higher performance degradation on primary system. Third party geo-mirroring solution requires higher network bandwidth.
Read Only Access or DSS Reporting	Provide read only access to data.	Can provide READ ONLY access. Recovery and READ ONLY modes are mutually exclusive. Oracle recommends creating a separate database for this purpose.	Replicated site provides both a potential Disaster Recovery solution and read only access.
Simplicity	How easy or difficult is it to implement? What is the maintenance cost?	Standby database is far easier to implement than replication and third party solutions.	If you do not have a good understanding of Oracle recovery, Third Party disaster recovery solutions may seem easier especially if it is part of a turnkey solution.
Scheduled Outages or Maintenance.	Can solution be leveraged for scheduled outages or upgrades?	Unsupported graceful switch over and switch back capabilities allow primary and standby to	Other failover and disaster recovery solutions have the same problem. Scheduled outages are

<p>Fallback capabilities</p>	<p>Can you fall back and switch over easily? Can you exchange roles between primary and standby?</p>	<p>switch roles in some scheduled outage scenarios. However, support for upgrade and migration maintenance is limited.</p> <p>Cannot fall back or switch over easily once standby database is activated using a RESETLOGS operation. A new standby needs to be built.</p> <p>Graceful switch over and switch back is feasible but not supported.</p>	<p>handled in a case by case basis.</p> <p>OPS, hardware failover and third party disaster recovery solutions may provide better fall back and failover solutions. However, they do not provide disaster recovery.</p>
<p>Low Cost</p>	<p>What is the cost of implementation?</p>	<p>High cost.</p> <p>Need to duplicate the primary site environment and enhance network topology.</p>	<p>Other disaster recovery solutions will cost the same or more.</p>

To summarize, a standby database can be the “right” choice if and only if

- business requirements justify the cost
- replication is not suitable for this environment
- critical data or database cannot be rebuilt fast enough by other recovery solutions

How do you know if the business justifies the cost? You need to determine the cost of downtime and clarify your goals. If your company loses business or market share over an extended downtime, creating a disaster recovery solution using a standby database is justified. Even if the probability of using a standby database is low, you need to calculate the cost savings when a standby database will be required. Does the investment provide sufficient savings to justify the cost? Most customers can quantify that an outage of 30 minutes may equate to 5 million dollars. Those same customers may easily justify spending one million dollars to build one or more standby sites. Furthermore, understanding your ultimate goals will dictate how you need to configure your standby environment. If the service level agreement with your end users dictate a short MTTR for disaster scenarios such as earthquakes, you can configure your primary and standby configurations to achieve a low MTTR. If you are concerned about user error or corruption that is propagated through redo, you need to guarantee that your detection mechanisms catch those events before applying the logs on the standby.

Scalability and reduction of response time have become critical issues in the world of e-commerce and other global businesses. A standby database does not scale an application nor reduce its response time. Replicated sites, on the other hand, may provide failover, disaster recovery, enhanced scalability and reduced response time. The reasons why Oracle replication and custom replication may not be a suitable disaster recovery solution are

- throughput limitations due to tracking and synching overhead, and
- complexity and difficulties with data conflicts.

Even with Oracle8i replication enhancements, large OLTP companies will find it difficult to implement replication as a disaster recovery solution due to its overhead and complexity. However, more customers will turn toward replication if possible.

Finally, if your allowable MTTR is a large interval (e.g. 2+ hours) and your critical data set is small, other recovery strategies may be equally satisfactory. If your business environment allows you to restore and recover the database via tapes or rebuild the older version of the database through an export file, a standby database solution may seem expensive and unnecessary. Furthermore, data marts that are constantly rebuilt can easily use their rebuild scripts or strategies to recover from a disaster.

Standby Database – Only a subset of an availability solution

To clarify, a standby database is not a panacea for all outages. A standby database is usually just one critical part of a full recovery plan. Although the standby database can be leveraged as a failover solution, OPS and hardware failover solutions are more appropriate. In many cases, OPS and Oracle Standby Database together provide a greater recovery plan. The following chart illustrates some system requirements to attain different availability service levels.

Table 3.3: System Availability Matrix

Required Availability	Unexpected Outage hours/year	Minimum Infrastructure Requirements (inclusive as required availability increases)
99.00%	87.6	<ul style="list-style-type: none"> ➤ Sound security practices ➤ Local system and hardware redundancy (redundant disks, multiple network connections, redundant controllers, etc.) ➤ Well-defined upgrade plan ➤ Inconsistent backups ➤ Sound backup/restore/recovery plan
99.25%	65.7	<ul style="list-style-type: none"> ➤ Fast backup and restore procedure (less than 4 hours) ➤ Monitoring facility (good detection) ➤ Multiple client hosts
99.50%	43.8	<ul style="list-style-type: none"> ➤ Fast backup and restore procedure (less than 2 hours) ➤ Automation of backup-restore-recovery operations ➤ Application failover ➤ Redundant copies onsite ➤ Archives online
99.75%	21.9	<ul style="list-style-type: none"> ➤ Oracle Parallel Server or some type of Failover system ➤ Object level recovery ➤ Disaster recovery site (less than 30 minutes; e.g. standby database)
99.90%	8.76	<ul style="list-style-type: none"> ➤ One or more disaster recovery sites (less than 10 minutes; e.g. standby database) ➤ Remote redundancy
99.99%	0.88	<ul style="list-style-type: none"> ➤ Full automation of detection and repair

99.999%	0.08	<ul style="list-style-type: none">➤ Three-tier with full replication of application➤ System and database, redundant pairs throughout the entire application workflow➤ Secure system where user errors are eliminated
---------	------	--

Design and Planning Issues

In this section, design and planning issues are briefly discussed.

Software Upgrades -- You must have Oracle8i installed to use the 8i standby features. If an upgrade is required, then it is necessary to consider the effect on your production database. You may need to upgrade your application software or your operating system. After such upgrades, system re-tuning is often required.

Hardware and Oracle -- The primary and standby sites *must* use the exact same release of Oracle8i for a specific operating system and platform. In some cases, the primary and standby databases may have different minor versions (not applicable for migrations) but the same *init.ora compatible* setting. Preferably, the primary and standby databases are physically distant and do not share resources such as the same power source and disks.

It is recommended that the hardware configurations at the primary and standby sites match. If the hardware is configured the same and the standby is activated, you are safe to assume that performance will be the same. Likewise, disk configurations and space usage should be the same at both sites to ensure that adding or resizing datafiles at the primary also succeed at the standby. Administration of both sites is also simplified. Although not recommended, it is possible to have different hardware configurations. In which case, you must ensure that the standby has sufficient disk space and its *init.ora* is adequately tuned to reflect available system resources.

Lost Transactions and MTTR -- To minimize or prevent transaction loss on a standby database due to missing archive logs or online redo logs, you need to guarantee that all available archive logs are shipped to the standby environment. Redo log sizing should reflect a reasonable log switch frequency during peak loads to ensure that they are available to the standby site. Furthermore, geo-mirroring of the online redo logs will guarantee that online redo logs are available to the standby in cases where the primary machine is not accessible when the standby database needs to be activated. Adequately sized redo logs, high network bandwidth, low network latency, and parallel recovery are examples of what you need to recover with no transaction loss given strict MTTR requirements.

Transfer Mechanism -- A fast and private interconnect between the primary and standby can reduce MTTR. Other network traffic must not impede the transfer of archive logs. They should be applied to the standby as soon as possible. However, you must ensure that only complete archive logs are transferred. Furthermore, your detection mechanism must detect user errors or corruption before applying the log that reflects that activity to the standby database.

Switching Users to the Standby -- You must choose a mechanism for users to switchover from the primary to the standby. In the most effective method, a software and/or hardware mechanism detects a failure and transparently connects users to the standby. This may require redesigning the user application.

Standby Automation -- The deployment and maintenance of the standby feature should be automated as much as possible. When a failure occurs, the standby database becomes the new production database. It will be desirable to switch back to the original primary site or use it as a standby site, which will require that a new standby database be deployed. In the case of maintaining the standby, the transfer and application of archives should be automatic. Lastly, error detection should be automated.

Error Detection -- You must implement mechanisms to automatically detect failures. First, a mechanism must initiate a switchover to the standby when the primary site fails. Second, a mechanism must detect primary site problems (particularly, in areas such as archiving and NOLOGGING operations) that affect the standby feature. Third, a mechanism must detect failures in the transfer mechanism. Fourth, a mechanism must detect standby site failures or problems.

Backups and Standby Rebuilds -- Although the standby feature is a robust disaster recovery mechanism, backups are still necessary. The standby site or sites are also susceptible to failure! Thus, you must prepare for the eventuality of performing media recovery at the primary using backups of the database files, including the archive logs. However, there is an additional benefit. The backups can be used to rebuild the standby during strategically convenient periods.

Training and Preventing Problems -- Application developers must become aware of issues that will affect the standby. For example, developers must be cautioned on the impact of NOLOGGING operations to the standby database.

8i Standby Database Enhancements

With the advent of Oracle8i, standby database has been enhanced considerably. The following chart compares features between Oracle releases.

Categories	Oracle V6 to Oracle7 Release 7.2.X	Oracle7 Release 7.3	Oracle8	Oracle8i
Supportability	Not Supported	Supported	Supported	Supported
Log all cache-managed blocks	Yes	Yes	Yes	Yes
OPS aware	Yes	Yes	Yes	Yes
Handles file status changes and datafile additions	No	Yes	Yes	Yes
Pre-clear online redo logs	No	Yes	Yes	Yes
Standby Controlfile	No	Yes	Yes	Yes
Oracle Managed Recovery	No	No	No	Yes
Remote Archive Destination	No	No	No	Yes
Oracle Remote Archival of Logs	No	No	No	Yes
Read Only Standby	No	No	No	Yes
Concurrent Read Only and Standby Recovery	No	No	No	No
Log unrecoverable or NOLOGGING changes	No	No	No	No

From Oracle version 7.3 to 8.0, few standby enhancements occurred. However, Oracle8i introduces important enhancements including

- Oracle managed recovery
- Remote archival destination and remote archival of online redo logs
- Read only standby database

The following table describes the above new features and notes some major advantages and disadvantages. Best practices on how to deploy your standby database using these features will be described later.

Oracle 8i New Feature	Description	Advantages	Disadvantages
Remote Archival of Redo Log Groups and Multiple Archive Destinations	Ability to automatically archive to the standby database. Utilizes Net8 service descriptor. Can have multiple mandatory archive destinations and processes.	Oracle automation. Checks and balances. Error checking. Multiple ARCH processes for scalability.	Files are transferred via Net8. Performance of shipping logs through Net8 needs to be assessed. Oracle Listener needs to be up. Standby database needs to be mounted. Network needs to be available with sufficient bandwidth and low latency.
Managed Database Recovery	Oracle standby database can be configured to automatically apply archives.	Oracle automation. Do not need to script this feature after setup. Can easily restart using manual recovery.	Oracle standby controlfile contains archive names updated by the RFS process. If archive files are shipped manually, Oracle managed recovery will not be aware of the new archives. Need to monitor for errors closely.
Read Only Standby Database	Oracle standby database can be open in READ ONLY mode.	Queries can be done. Sorts can be done if tempfiles and locally managed temporary tablespace are precreated. Recovery can be restarted.	No DML or DDL is allowed. Recovery and READ ONLY access are mutually exclusive.

PREPARATION

This major section reviews creating the standby database.

Creation of Standby Database

Ensure that the production database will run at the standby site. This is especially important if different configurations are used for the standby and primary. Also check that operating system environment variables are correctly set; in particular, ORACLE_HOME, ORACLE_SID, and any others specified in the installation guide.

If the two sites have different hardware configurations, then you must ensure that parameters are tuned for the standby configuration. If standby directory paths are different than the primary's, then the standby paths must be reflected in the initialization file.

Ensure that the database directory paths for the standby are valid and have sufficient disk space.

The steps for creating a standby database essentially remain the same from previous releases.

- restore datafiles
- restore standby controlfiles
- modify init.ora files
- setup primary database tnsnames.ora file and test connection (new 8i)
- setup listener on the standby side (new 8i)
- mount the standby database using the standby controlfile

The following table describes the steps to initially create the standby database.

Step	Primary Database	Standby Database
1. Backup of the primary database.	Consistent (cold) or inconsistent (hot) backup of primary database	
2. Create standby controlfile.	<code>ALTER DATABASE CREATE STANDBY CONTROLFILE AS '/dir/standby.ctl'</code>	
3. Restore data files, standby controlfile and configuration files to standby.		Copy operating system files to the standby site.
4. Modify Primary and Standby init.ora File	Refer to init.ora table.	Refer to init.ora table.
5. Modify Primary's tnsnames.ora file if using remote archiving.	Create Net8 alias to connect to standby database.	
6. Setup Oracle listener on the standby database.		Setup and startup Oracle listener. Modify listener.ora and sqlnet.ora files.
7. Ship previous archives over to the standby database	<code>ALTER SYSTEM ARCHIVE LOG CURRENT</code>	Copy archives to standby database.
8. Mount standby database		<code>STARTUP PFILE=initStandby.ora NOMOUNT</code> <code>ALTER DATABASE MOUNT STANDBY DATABASE</code>
9. Offline or rename relevant datafiles		Issue <code>ALTER DATABASE DATAFILE 'filename' OFFLINE</code> command for files that you do not want to recover. Issue <code>ALTER DATABASE RENAME FILE</code>

<p>10. Validate file statuses</p>		<p>'oldname' TO 'newname' command for files that have different directories and cannot be easily renamed using the db_file_name_convert init.ora parameters.</p> <p>All files that are part of the standby database need to have a status of online or system. Check v\$recover_file.</p>
<p>11. Start recovery</p>		<p>Refer to different recovery mode chart.</p>

Initialization Parameters

The standby and primary sites must have their own unique initialization files. The following table lists the important init.ora parameters for a primary and standby database configuration.

Initialization Parameter	Primary Site Value	Standby Site Value
COMPATIBLE	Must be 8.1.5 or higher.	Identical to primary value.
CONTROL_FILE	Primary controlfile path and name.	Standby controlfile path and name.
DB_BLOCK_SIZE	Same as at database creation time.	Identical to primary value.
DB_FILES		
DB_NAME		
LOCK_NAME_SPACE	N/A	Required if standby database is local.
		Local standby database is not recommended since it provides no protection for disaster recovery.
*** ARCHIVE PARAMETERS	LOG_ARCHIVE_START=TRUE	SAME AS PRIMARY
LOG_ARCHIVE_START		
LOG_ARCHIVE_FORMAT	Example: 'ARCH%T_%S.ARC'	SAME AS PRIMARY
LOG_ARCHIVE_MAX_PROCESSES	Recommended to set log_archive_max_processes=10	N/A
(1-10)	Parallelism will be required to archive multiple redo logs concurrently.	Set the same as Primary in case you switch roles.
LOG_ARCHIVE_DEST_N	LOG_ARCHIVE_DEST_1=	
Where N=1-5	"LOCATION= local directory MANDATORY"	
	LOG_ARCHIVE_DEST_2=	
	"SERVICE=Net8ServiceName MANDATORY REOPEN=30"	

Initialization Parameter	Primary Site Value	Standby Site Value
<p>LOG_ARCHIVE_DEST_STATE_N</p> <p style="text-align: center;">Where N=1-5</p> <p>For each LOG_ARCHIVE_DEST_N parameter, a corresponding LOG_ARCHIVE_DEST_STATE_N parameter must exist.</p> <p>LOG_ARCHIVE_DEST</p> <p>LOG_ARCHIVE_DUPLEX_DEST</p> <p>LOG_ARCHIVE_MIN_SUCCEED_DEST</p> <p>STANDBY_ARCHIVE_DEST</p> <p>DB_FILE_STANDBY_NAME_CONVERT</p> <p>LOG_FILE_STANDBY_NAME_CONVERT</p> <p>DB_BLOCK_BUFFERS</p> <p>RECOVERY_PARALLELISM</p> <p>LOG_ARCHIVE_TRACE (available in 8.1.6+)</p>	<p>LOG_ARCHIVE_DEST_STATE_1=ENABLE LOG_ARCHIVE_DEST_STATE_2=ENABLE</p> <p>Do not use.</p> <p>Override by LOG_ARCHIVE_DEST_N parameter.</p> <p>Not necessary if explicitly set mandatory option in the LOG_ARCHIVE_DEST_N parameter.</p> <p>Only relevant for local archive destinations.</p> <p>Not needed.</p> <p>Dynamic parameter. Used by Primary (ARCH) and Standby (RFS) to determine the level of tracking information to write to the trace file.</p>	<p>N/A</p> <p>Do not use.</p> <p>STANDBY_ARCHIVE_DEST= <standby archive destination>. The standby recovery process will look for standby archive files in this destination.</p> <p>Only use these parameters if the directory paths to datafiles or online redo logs are different at both sites.</p> <p>Values are used to convert primary path names to standby path names.</p> <p>Increase DB_BLOCK_BUFFERS and RECOVERY_PARALLELISM to speed up recovery.</p> <p>Dynamic parameter. Used by Primary (ARCH) and Standby (RFS) to determine the level of tracking information to write to the trace file.</p>

DEPLOYMENT

This major section describes how to deploy an Oracle8i standby database. It focuses on setting up remote archiving and evaluating the new recovery modes. The different standby database modes are described.

Remote archival of online redo logs

In Oracle8i, remote archival is achieved by the following steps:

- configuring `log_archive_dest_n` and `log_archive_dest_state_n` parameters in the primary's `init.ora` file
- configuring `tnsnames.ora` on the primary database
- configuring and starting up the Net8 listener for the standby database

Automatic remote archival is the preferred and easier configuration compared to shipping the archives manually; however, it requires a more in-depth performance assessment.

First, you need to setup the `log_archive_dest_n` and `log_archive_dest_state_n` `init.ora` parameters in your primary database. The following chart describes each parameter and its attributes.

Parameter Name	Attributes	Description
LOG_ARCHIVE_DEST_N N=1-5 (parsable parameter)	LOCATION	Describes local archive destination. Not relevant for remote archive destination.
RECOMMENDED: Multiple Archive Processes. Minimum one mandatory archive destination. One mandatory remote standby destination.	SERVICE (mutually exclusive with the LOCATION parameter)	Net8 service name used to communicate with a standby database. The listener and standby instance must be available for the service to work. Standby database must be mounted.
	MANDATORY / OPTIONAL	Mandatory implies that archiver needs to successfully write to the archive destination before LGWR overwrites the log file.
	REOPEN	Specifies the seconds to wait before retrying the connection after an error.
LOG_ARCHIVE_DEST_STATE_N N=1-5	ENABLE / DEFER	Specifies if the corresponding archive destination is available. If state is <i>defer</i> , the archive destination is ignored.

Next, you need to configure the primary database's tnsnames.ora with the corresponding service name. The following example shows a sample init.ora file and a tnsnames.ora entry.

Primary database init.ora file example:

```
log_archive_dest_1 = "location=/u03/standby/arch mandatory"
log_archive_dest_2 = "mandatory service=8istby reopen=30"
log_archive_dest_state_1=enable
log_archive_dest_state_2=enable
```

Primary database's tnsnames.ora file.

```
8istby.world =                               /* 8istby.world is my Net8 alias */
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS=(PROTOCOL=tcp)(HOST=poseidon)(PORT=6001)) /* poseidon standby db host */
      (ADDRESS=(PROTOCOL=tcp)(HOST=poseidon)(PORT=6002)) /* poseidon failover lsnr */
    )
    (CONNECT_DATA =
      (SID = crash)
      (GLOBAL_NAME = crash.world))           /* crash.world = standby database SID */
  )
```

Subsequently, on the standby server, you need to configure the Net8 listener.ora file. Below, you will find a sample listener entry.

```
LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP)(HOST = poseidon)(PORT = 6001))
      )
    )
  )
SID_LIST_LISTENER =
  (SID_DESC =
    (GLOBAL_DBNAME = crash.world)
    (ORACLE_HOME = /oas/oas/oas815/ohome)
    (SID_NAME = crash)
  )
LIST2 =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP)(HOST = poseidon)(PORT = 6002))
      )
    )
  )
SID_LIST_LIST2=
  (SID_DESC =
```

```
(GLOBAL_DBNAME = crash.world)
(ORACLE_HOME = /oas/oas/oas815/ohome)
(SID_NAME = crash)
)
```

Once you have started the listener by issuing a `lsnrctl start <listener name>`, you can validate if the Net8 service works by performing the following actions:

- `tnsping <service name>` from the primary database
- `connect username/password@ServiceName`

Before the archiver processes automatically ship to the standby destination, the Net8 listener must be up and the standby database must be mounted. Each primary ARCH process will create a Remote File Server (RFS) process on the standby database. Upon the first attempt to ship archives to the standby database, the RFS process will be created will become a persistent server process. If the RFS process dies or if the connect is lost, a new RFS process will be created on reconnection. The primary archiver process will communicate with the RFS through Net8. The RFS processes are responsible for the following tasks:

- consuming network I/O from the primary archiver processes
- creating standby archive files under the `standby_archive_dest`
- updating standby controlfiles with the archive information once the standby archive file completes successfully.

All archive files generated prior to setting up remote archiving needs to be shipped manually. Furthermore, if errors are encountered or if Oracle automatic archival is not fast enough, you can disable automatic archiving by issuing `ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_n = DEFER`. Alternatively, you can disable remote archiving by dynamically changing the `LOG_ARCHIVE_DEST_N` settings through an `ALTER SYSTEM` command.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_n = {null_string | {LOCATION=pathname | SERVICE=servicename}
[MANDATORY | OPTIONAL] [REOPEN[=retry_time_in_seconds]]
```

Mounting and Shutting Down a Standby Database

The following table describes the steps to start, mount, and shutdown the standby database.

Steps	Oracle Commands	Description
Starting up Oracle Instance	STARTUP PFILE=initStandby.ora NOMOUNT	Starts the Oracle instance. Points to a specific init.ora file.
Mount Oracle Standby Database	ALTER DATABASE MOUNT STANDBY DATABASE	Mounts standby database. Required by ARCH to create the RFS processes. Required for RFS processes to create the standby archive files. Required for recovery.
Shutdown Standby Database	SHUTDOWN NORMAL SHUTDOWN IMMEDIATE	May require canceling recovery first prior to shutdown. Can be issued from a different session but canceling recovery is recommended.

Initiating Standby Recovery

Once the archiving is established and the standby database is mounted, you can initiate either manual or automatic recovery. The following table describes the advantages and disadvantages of each.

Category	Oracle Command	Advantages	Disadvantages
1. Managed Recovery RECOMMENDED	RECOVER MANAGED STANDBY DATABASE	Automatic recovery. Error checking from the RFS process. Recovery process will check for new archives in the standby controlfile every 15 seconds. No user intervention to feed archive names to recovery process. Error messages are found in dynamic views or alert.log files.	Depends on RFS process to update standby control file with archive file names. Unaware of any archive logs that are not created via the RFS process. Requires monitoring. Still requires manual recovery if archives are shipped manually. Manual intervention or scripting required when errors are encountered.
2. Managed Recovery with timeout	RECOVER MANAGED STANDBY DATABASE TIMEOUT [INTERVAL]	Same as #1 but with explicit timeout interval set (minutes)	Same as #1
3. Stop Managed	RECOVER MANAGED STANDBY DATABASE	Issue in separate session.	CANCEL IMMEDIATE can leave database in an

Recovery	CANCEL [IMMEDIATE]		inconsistent state if recovery is in the middle of an archive log. Need to apply one more log or complete the log in that case.
4. Manual Recovery	RECOVER STANDBY DATABASE	More control. Can recover archives that were manually shipped or archives that are not contained in the standby controlfile.	Need to be scripted. Need to manually enter archive names or take default names.
5. Automatic recovery	SET AUTORECOVERY ON RECOVER [AUTOMATIC] [FROM LOCATION] STANDBY DATABASE	Automatically apply a batch of archives found in the standby archive destination.	Very little control. Will recover all archives until it cannot find a match.

During managed recovery, the recovery process will periodically read the standby controlfile and apply any relevant archive logs. Managed recovery depends on the RFS process to update the standby controlfile. If there are archives that exist in the standby archive destination but not in the controlfile, manual recovery (options 4 or 5) can be utilized to apply the rest of the archives.

You can tune the standby database to expedite recovery. Here are some tips:

- set recovery_parallelism = minimum (number of devices accessed, number of CPUs)
- increase db_block_buffers
- tune DBWR and use asynchronous IO
- tune IO and use disk striping when appropriate

In most cases, the primary and standby database file layout should be identical. Disk striping on the primary host implies that you will do the same on the standby host.

MAINTENANCE and MONITORING

This major section describes how to maintain and monitor the standby feature after deployment.

Maintenance of the standby database ensures that changes made to the primary are fully propagated to the standby. Whenever possible, these tasks should be automated through Oracle or customized scripts. Most changes to the primary are propagated using normal propagation, which is easy to automate. However, special attention is needed for changes that are not propagated or cannot be propagated.

The table below indicates whether a command propagates normally or requires extra administrative efforts to be fully propagated. It also describes best practices in monitoring these events.

Event	Primary Detection	Standby Detection	Recommendations																		
ARCHIVING ERRORS	<p>Error column in v\$archive_destination.</p> <p>Alert.log</p> <p>ARCHIVE LOG LIST</p> <p>Before manually shipping archives, verify that archiving the log has completed by checking ARCHIVED column in v\$log.</p> <p>Archive Trace Files</p>	RFS process trace file.	<p>Multiple Archive Processes.</p> <p>Minimum one mandatory archive destination.</p> <p>One mandatory remote standby destination.</p> <p>Create scripts to push or pull archives if errors occur or due to lack of performance.</p> <p>Monitor Primary Database Alert.log and v\$archive_destination.</p>																		
ARCHIVING TRACING	<p>New init.ora parameter LOG_ARCHIVE_TRACE is a dynamic parameter available in versions 8.1.6 and higher. Used by Primary (ARCH) and Standby (RFS) to determine the level of tracking information to write to the trace file. Assists in debugging and monitoring ARCH and RFS processes.</p> <table border="1"> <thead> <tr> <th>Setting</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Tracking disabled (the default)</td> </tr> <tr> <td>1</td> <td>Track archival of REDO log file</td> </tr> <tr> <td>2</td> <td>Track archival status per destination</td> </tr> <tr> <td>4</td> <td>Track archival operational phases</td> </tr> <tr> <td>8</td> <td>Track destination activity</td> </tr> <tr> <td>16</td> <td>Track detailed destination activity</td> </tr> <tr> <td>32</td> <td>Track archive log destination parameter settings</td> </tr> <tr> <td>64</td> <td>Track ARCH process state activity</td> </tr> </tbody> </table> <p>Individual tracking levels can be combined by setting LOG_ARCHIVE_TRACE to the sum of the desired levels (i.e., level 3 combines level 1 and level 2 trace output).</p>			Setting	Description	0	Tracking disabled (the default)	1	Track archival of REDO log file	2	Track archival status per destination	4	Track archival operational phases	8	Track destination activity	16	Track detailed destination activity	32	Track archive log destination parameter settings	64	Track ARCH process state activity
Setting	Description																				
0	Tracking disabled (the default)																				
1	Track archival of REDO log file																				
2	Track archival status per destination																				
4	Track archival operational phases																				
8	Track destination activity																				
16	Track detailed destination activity																				
32	Track archive log destination parameter settings																				
64	Track ARCH process state activity																				
THREAD EVENTS	<p>Alert.log</p> <p>v\$thread.</p>	Alert.log	<p>Automatic propagation through archive files.</p> <p>Threads are dropped or</p>																		

REDO LOG CHANGES	Alert.log, V\$log, v\$logfile Check stale or invalid statuses in the online redo logs. <code>SELECT STATUS FROM V\$LOGFILE WHERE STATUS IS NOT NULL;</code>	N/A	enabled automatically through redo records in the redo logs. Standby controlfile is updated. Redo log changes do not impact standby database unless a log file is cleared or lost. In those cases, the standby database needs to be rebuilt. Pre-clear the logs on the standby database with <code>ALTER DATABASE CLEAR LOGFILE</code> command.
CONTROL FILE CHANGES	Alert.log	N/A	Recreate standby controlfile. Recreate Standby database if RESETLOGS option is utilized.

Event	Primary Detection	Standby Detection	Recommendations
PRIMARY DATABASE RECOVERY	Alert.log	N/A	Recreate Standby database if point in time recovery or RESETLOGS option is utilized.
Tablespace Status Changes	Db_tablespaces, alert.log	Verify all data files are online. Check v\$recover_file.	Automatically propagated. Files remain online.
Offline Tablespace	Db_tablespaces, alert.log	Verify all data files are online. Check v\$recover_file.	Automatically propagated. Files remain online.
Online Tablespace	Db_tablespaces, alert.log	Verify all data files are online. Check v\$recover_file.	Automatically propagated. Files remain online.
Read Only Tablespace	Db_tablespaces, alert.log	Verify all datafiles are online. Check v\$recover_file.	Automatically propagated. Files remain online.
Read Write Tablespace	Db_tablespaces, alert.log	Verify all datafiles are online. Check v\$recover_file.	Automatically propagated. Files remain online.
Add Datafile	Db_data_files, alert.log	Message: ORA-283,1670,1157,1110 Standby Recovery stops.	Need to manually create datafile. <code>ALTER DATABASE CREATE DATAFILE 'filename';</code>

			Recommence recovery.
--	--	--	----------------------

Event	Primary Detection	Standby Detection	Recommendations
Create Tablespace	Db_data_files, alert.log	Message: ORA-283,1670,1157,1110 Standby Recovery stops.	Need to manually create datafile. ALTER DATABASE CREATE DATAFILE 'filename';
Drop tablespace	Db_data_files, alert.log	Standby alert.log Recovery dropped tablespace message in alert.log	Remove operating system file.
Offline Immediate	V\$recover_file, alert.log Require recovery when tablespace is attempted to be online.	Verify all datafiles are online. Check v\$recover_file.	Files remain online. Tablespace is fine after standby database activation.

Offline data file	V\$recover_file, alert.log Require recovery when tablespace or datafiles attempted to be online.	Verify all datafiles are online. Check v\$recover_file.	Files remain online. Datafile is fine after standby database activation.
Offline datafile drop	V\$recover_file, alert.log Require recovery when tablespace or datafile is attempted to be online.	Verify all data files are online. Check v\$recover_file.	Files remain online. Datafile is fine after standby database activation.
Rename Datafile	Alert.log	No impact.	No impact
NOLOGGING or UNRECOVERABLE operations	select unrecoverable_change#, unrecoverable_time from v\$datafile; Direct loader "invalidate block range" redo entry in online redo log.	File blocks are invalidated. File blocks are not touched if blocks are in the future of redo.	Monitor primary database v\$database view for this event. Data files need to be refreshed. Stop Standby Recovery Corresponding Data Files can be offlined. Recommence Standby Recovery Restore new backup files. Stop Standby Recovery Online Data Files. Recommence Standby Recovery

Recovery Progress	Alert.log	V\$recovery_log, alert.log	Monitor standby database recovery through the alert.log. Make sure the standby is not falling too far behind the primary database.
-------------------	-----------	----------------------------	--

In most cases, the standby database does not need to be refreshed when there are changes to the primary database. The entire standby database needs to be refreshed after one of the following events:

- resetlogs operation occurs on the primary database
- one of the redo logs or archive logs is lost or corrupted

Refreshing a standby database is the same as recreating or rebuilding a standby database.

A partial refresh such as restoring a single tablespace, datafile or creating a new standby controlfile is required whenever one of the following events occur

- NOLOGGING or UNRECOVERABLE changes are applied to a datafile
- a standby datafile or controlfile suffers media corruption
- primary controlfile is recreated with NORESETLOGS option

The following tables describe how to refresh a standby control file or refresh one standby datafile.

Refreshing Standby Control File

Primary Site	Commands
Create new standby controlfile	ALTER DATABASE CREATE STANDBY CONTROLFILE;
Archive up to current log	ALTER DATABASE ARCHIVE LOG CURRENT;
Transfer files to standby site	Transfer the new standby controlfile and archive files to the standby. Do not remove old standby controlfile instead rename it.
Standby Site	
Shutdown recovery	CANCEL RECOVERY SHUTDOWN IMMEDIATE
Start and mount standby database	STARTUP PFILE=initstandby.ora NOMOUNT ALTER DATABASE MOUNT STANDBY DATABASE
Check file statuses and file names	SELECT * FROM V\$RECOVER_FILE; SELECT * FROM V\$DATAFILE; (Recovery only focuses on online data files)
Rename file(s), offline unwanted files, and online essential files	ALTER DATABASE RENAME FILE 'OLDNAME' TO 'NEWNAME' ALTER DATABASE DATAFILE unwanted OFFLINE; ALTER DATABASE DATAFILE wanted ONLINE;
Restart recovery	RECOVER STANDBY DATABASE;

Refreshing One Data File in the Standby Database

Primary Site	Commands
<p>Primary Site</p> <p>Event invalidated a standby file</p> <p>Create another backup of this file</p>	<p>Examples include NOLOGGING or UNRECOVERABLE operations on a file. Media corruption or failure of a specific standby database file.</p> <p>Example:</p> <pre>ALTER TABLESPACE tablespace_name BEGIN BACKUP</pre> <p>Backup file through operating system commands.</p> <pre>ALTER TABLESPACE tablespace_name END BACKUP</pre>
<p>Standby Site</p> <p>Shutdown recovery</p> <p>Offline corresponding datafile</p> <p>Restart recovery</p> <p>Restore new backup of datafile</p> <p>Shutdown recovery</p> <p>Online corresponding datafile</p> <p>STEP B:</p> <p>Check file statuses and rename files if necessary</p> <p>Restart recovery</p>	<pre>CANCEL RECOVERY</pre> <p>If the file is small and transfer time is minimal, you can transfer the file first and then skip to step B.</p> <pre>ALTER DATABASE DATAFILE 'FILENAME' OFFLINE</pre> <pre>RECOVER STANDBY DATABASE;</pre> <p>Transfer file from primary database to standby database.</p> <pre>CANCEL RECOVERY</pre> <pre>ALTER DATABASE DATAFILE 'FILENAME' ONLINE</pre> <pre>ALTER DATABASE RENAME FILE 'OLDNAME' TO 'NEWNAME'</pre> <pre>SELECT * FROM V\$RECOVER_FILE;</pre> <pre>SELECT * FROM V\$DATAFILE;</pre> <pre>RECOVER STANDBY DATABASE;</pre> <p>Recovery process may ask to reapply an older archive file.</p>

ACTIVATION

This major section describes how to activate the standby as a new production system and how to open the standby database for READ ONLY purposes.

Activating standby database for Production

The standby database should only be activated in case of an emergency. When the primary database is inaccessible or corrupt, the standby database can be activated and accessed by your end users. Prior to activation, all archives and possibly the current online redo logs are applied. To reduce the activation interval, the standby database online redo logs should have been pre-cleared. By pre-clearing the logs, the ACTIVATE command will skip the step of reformatting the standby database's online redo logs. The init.ora parameters may require adjustments prior to starting as production database. To activate the standby database, issue the command, `ALTER DATABASE ACTIVATE STANDBY DATABASE.`

The following table shows the steps involved in activating the standby feature.

Steps	Description
Transfer files to standby site	If possible, transfer any remaining archives to be applied from the primary. If possible, <code>ALTER SYSTEM ARCHIVE LOG CURRENT</code> the last logs and apply them on the standby database. This will allow the standby to be recovered up to the last transaction committed at the primary. Copy the online redo logs to the standby database only as the last resort. Make sure you only copy the current online redo logs while the database instance(s) are shutdown. Alternatively, the online redo logs can be geo-mirrored on the standby database.
Standby Site	
Start and mount standby database	<code>Startup pfile=\$ORACLE_HOME/dbs/initstby.ora nomount</code>
Check file statuses	All relevant files should be online. <code>Select * from v\$recover_file;</code>
Recover remaining archives and possibly online redo logs	<code>RECOVER STANDBY DATABASE;</code> You need to explicitly provide the online redo log names if you did not successfully archived them.
Activate standby database	<code>ALTER DATABASE ACTIVATE STANDBY DATABASE;</code>
Restart Standby Database	<code>Shutdown immediate</code> <code>Startup pfile=initProduction.ora</code>
Switch users over	Contact users directly, or, initiate network IP switching software.
Investigate new standby	Your new primary database must be safeguarded. Instigate backups, and when possible, build a new standby on the primary or third site.

The `ALTER DATABASE ACTIVATE STANDBY DATABASE` command initiates a RESETLOGS operation. The primary and standby databases are no longer compatible. Previous redo is invalidated and cannot be applied. A new standby database needs to be created. You need to follow the steps to create a new standby database.

An undocumented and unsupported method exists for bringing up the standby database as the production database without executing an ACTIVATE or RESETLOGS. The technique is called Graceful Switchover and Switchback of a standby database. To implement graceful switchover, please refer to the Graceful Switchover and Switchback of Oracle Standby Database paper in the MetaLink Server Technical Library.

Opening standby database as READ ONLY

Starting with Oracle8i, a database can be opened in READ ONLY mode without invalidating the standby database. Queries can be executed on the READ ONLY database; however, DDL and other DML operations are prohibited. Temporary tables cannot be created. Recovery can recommence easily after restarting the standby database. Unfortunately, recovery and READ ONLY access are mutually exclusive operations on the same database. It is recommended to create a separate READ ONLY database if such access is required for long periods.

The following table describes how to open a standby database in READ ONLY mode for queries while allowing disk sorts. The main prerequisite is to create a LOCALLY MANAGED temporary tablespace made of tempfiles. A LOCALLY MANAGED temporary tablespace is a requirement for disk sorts in a READ ONLY database. Normal (or dictionary managed) temporary tablespaces trigger DML operations on data dictionary tables (uet\$ and fet\$) to allocate space for disk sorts. However, LOCALLY MANAGED temporary tablespaces use bitmaps within the tablespace to track space utilization and tempfiles for disk sorts. Tempfiles are not stored in the data dictionary, and information in tempfiles is not written to the redo log file. Therefore, the database may write sort data to these files even when in READ ONLY mode. You will be required to create one or more tempfiles manually after opening the standby database in READ ONLY mode.

Steps	Description
Primary Database	
Create LOCALLY MANAGED Temporary Tablespace	<pre>CREATE TEMPORARY TABLESPACE tbspaceName TEMPFILE'filename' SIZE integer [M/K]EXTENT MANAGEMENT LOCAL UNIFORM SIZE integer [M/K];</pre>
Alter all users	<pre>ALTER USER username TEMPORARY TABLESPACE tbspaceName;</pre>
Standby Site	
Start and mount standby database	<pre>STARTUP PFILE= initstandby.ora NOMOUNT</pre>
Mount standby database	<pre>ALTER DATABASE MOUNT STANDBY DATABASE</pre>
Check file statuses	All files should be online. <pre>select * from v\$recover_file;</pre>
Recover standby database	<pre>RECOVER MANAGED STANDBY DATABASE</pre>
Cancel recovery	<pre>RECOVER MANAGED STANDBY DATABASE CANCEL</pre>
Open in read only mode.	<pre>ALTER DATABASE OPEN READ ONLY;</pre>
Create temporary files	<pre>ALTER TABLESPACE tbspaceName ADD TEMPFILE 'filename' SIZE [M/K]</pre> Since the ts\$ table contains the tbspaceName entry, you can add a tempfile to the tablespace. Creating tempfiles does not update the data dictionary or any other object inside the database
Log on to user	<pre>CONNECT username/password</pre> Need to connect to a user whose temporary tablespace points to tbspaceName. You cannot alter this dynamically in a READ ONLY database.
Run queries requiring disk sorts	Run reports and queries. Check database statuses. Check data content.

The following table briefly describes how to restart standby database recovery.

Steps	Description
Standby Site	STANDBY DATABASE is open in READ ONLY mode
Shutdown standby	SHUTDOWN IMMEDIATE
Start standby database	STARTUP PFILE= initstandby.ora NOMOUNT
Mount standby database	ALTER DATABASE MOUNT STANDBY DATABASE
Check file statuses	All files should be online. select * from v\$recover_file;
Recover standby database	RECOVER MANAGED STANDBY DATABASE

Bibliography

General background:

- *Oracle8i Backup and Recovery Guide*
- *Oracle8i Server Concepts Manual*
- *Oracle8i Server Administrator's Guide*
- *Oracle8i Server SQL Reference*
- *Oracle8i Server Reference Manual*

Further reading material available online through Metalink:

- Note 67488.1, *Oracle8i Standby Database New Features and Read Only Databases* by Jbarlow.uk, 1999
- *Standby Database Prior to 7.3*, by Lawrence To, in the *Technical Reports Compendium Volume I, 1995*
- *Standby Database* by Brian Quigley and Lawrence To
- *Graceful Switchover and Switchback using Standby Databases* by Lawrence To

The compendiums listed above are published by the Center of Expertise.

Oracle (confidential) documents available only to Oracle employees:

- *Standby Database, Release 7.3, Functional Specification* , by Lynne C. Thieme
- *Standby Database 7.3 Internals*, by Lawrence To
- *Oracle 7.2 Recovery Outline* , by Andrea Borr
- *Remote Archival Redo Log Brown Bag* from Terry Hart
- *DSI 303 Internals and Module 2 Database Recovery Internals*
- *Oracle8i Standby Database*, technical seminar by Lawrence To, 1999