GENERAL CONCEPTS
Oracle Data Guard is the management, monitoring, and automation software that works with a production database and one or more standby databases to protect data against failures, errors, and corruption that might otherwise destroy your database.

DATA GUARD COMPONENTS
Oracle Data Guard 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.

**STANDBY DATABASE:**
A physical or logical standby database is a database replica created from a backup of a primary database. A Data Guard configuration consists of one production (or primary) database and up to nine standby databases. The databases in a Data Guard configuration are connected by Oracle Net and may be dispersed geographically. There are no restrictions on where the databases are located, provided that they can communicate with each other.

- A **physical standby** database is physically identical to the primary database on a block-for-block basis. It's maintained in managed recovery mode to remain current and can be set to read only; archive logs are copied and applied automatically.
- A **logical standby** database is logically identical to the primary database. It is updated using SQL statements. The SQL statements are "extracted" from the archive logs and then applied to the Logical Standby.

**LOG TRANSPORT SERVICES:**
Enables and controls the automated transfer of redo data within a Data Guard configuration from the primary site to each of its standby sites. Log transport services also control the level of data protection for your database. The DBA will configure log transport services to balance data protection and availability against database performance. Log transport services will also coordinate with log apply services and role management services for switchover and failover operations.

**NETWORK CONFIGURATION:**
The primary database is connected to one or more remote standby database via Oracle Net.

**LOG APPLY SERVICES:**
Log apply services apply the archived redo logs to the standby database.

**DATA GUARD BROKER:**
Data Guard Broker is the management and monitoring component with which you configure, control, and monitor a fault tolerant system consisting of a primary database protected by one or more standby database.
Data Guard can be used in combination with other Oracle High Availability (HA) solutions such as Real Application Clusters (RAC), Oracle Flashback, Oracle Recovery Manager (RMAN), and new database options for Oracle Database 11g that include Oracle Active Data Guard and Oracle Advanced Compression, to provide a high level of data protection.

Fig 1: DATA GUARD GENERAL VIEW
DATA GUARD ROLES
A database can operate in one of the two mutually exclusive roles: primary or standby database.

FAILOVER
A failover is done when the primary database (all instances of an Oracle RAC primary database) fails or has become unreachable and one of the standby databases is transitioned to take over the primary role. Failover should be performed when the primary database cannot be recovered in a timely manner. Failover may or may not result in data loss depending on the protection mode in effect at the time of the failover.

SWITCHOVER
A switchover is a role reversal between the primary database and one of its standby databases. A switchover guarantees no data loss and is typically done for planned maintenance of the primary system. During a switchover, the primary database transitions to a standby role, and the standby database transitions to the primary role.
**DATA GUARD INTERFACES**

Oracle provides three ways to manage a Data Guard environment:

**SQL*PLUS AND SQL STATEMENTS**
Using SQL*Plus and SQL commands to manage Data Guard environment. The following SQL statement initiates a switchover operation:

```sql
SQL> alter database commit to switchover to physical standby;
```

**DATA GUARD BROKER GUI INTERFACE (DATA GUARD MANAGER)**
Data Guard Manger is a GUI version of Data Guard broker interface that allows you to automate many of the tasks involved in configuring and monitoring a Data Guard environment.

**DATA GUARD BROKER COMMAND-LINE INTERFACE (CLI)**
It is an alternative interface to using the Data Guard Manger. It is useful if you want to use the broker from batch programs or scripts. You can perform most of the activities required to manage and monitor the Data Guard environment using the CLI. The following example lists the available commands:

```
$ dgmgrl
Welcome to DGMRL, type "help" for information.

DGMRL> help
The following commands are available:
  quit
  exit
  show             See "help show" for syntax
  enable           See "help enable" for syntax
  disable          See "help disable" for syntax
  help             [<command>]
  connect          <user>/</password> [@<connect>]
  alter            See "help alter" for syntax
  create           See "help create" for syntax
  remove           See "help remove" for syntax
  switchover       See "help switchover" for syntax
  failover         See "help failover" for syntax
  startup          See "help startup" for syntax
  shutdown         See "help shutdown" for syntax
```
PROCESS ARCHITECTURE

PHYSICAL STANDBY PROCESSES ARCHITECTURE (APPLY REDO LOGS)
A physical standby database is a byte for byte exact copy of the primary database. This also means that rowids stay the same in a physical standby database environment.

On the primary database site, the log writer process (LGWR) collects transactions from the log buffer and writes to the online redo logs. The archiver process (ARCH) creates a copy of the online redo logs, and writes to the local archive destination. Depending on the configuration, the archiver process or log writer process can also transmit redo logs to standby database. When using the log writer process, you can specify synchronous or asynchronous network transmission of redo logs to remote destinations. Data Guard achieves asynchronous network I/O using LGWR network server process (LNS). These network servers processes are deployed by LOG_ARCHIVE_DEST_n initialization parameter. Data Guard’s asynchronous log transport (i.e. the Maximum Performance mode) is recommended for a configuration in which the network distance is up to thousands of miles, providing continual maximum performance, while minimizing the risks of transaction loss in the event of a disaster.

On the standby database site, the remote file server process (RFS) receives archived redo logs from the primary database. The primary site launches the RFS process during the first log transfer. The redo logs information received by the RFS process can be stored as either standby redo logs or archived redo logs. Data Guard introduces the concept of standby redo logs (separate pool of log file groups). Standby redo logs must be archived by the ARCH process to the standby archived destination before the managed recovery process (MRP) applies redo log information to the standby database.

The fetch archive log (FAL) client is the MRP process. The fetch archive log (FAL) server is a foreground process that runs on the primary database and services the fetch archive log requests coming from the FAL client. A separate FAL server is created for each incoming FAL client.

FAL_SERVER specifies the FAL (fetch archive log) server for a standby database. The value is an Oracle Net service name, which is assumed to be configured properly on the standby database system to point to the desired FAL server.

FAL_CLIENT specifies the FAL (fetch archive log) client name that is used by the FAL service, configured through the FAL_SERVER parameter, to refer to the FAL client. The value is an Oracle Net service name, which is assumed to be configured
properly on the FAL server system to point to the FAL client (standby database). Thanks to the FAL_CLIENT and FAL_SERVER parameters, the managed-recovery process in the physical database will automatically check and resolve gaps at the time redo is applied. This helps in the sense that you don't need to perform the transfer of those gaps by yourself.

FAL_CLIENT and FAL_SERVER only need to be defined in the initialization parameter file for the standby database(s). It is possible; however, to define these two parameters in the initialization parameter for the primary database server to ease the amount of work that would need to be performed if the primary database were required to transition its role.

Prior to Oracle 11g, Redo Apply only worked with the standby database in the MOUNT state, preventing queries against the physical standby whilst media recovery was in progress. This has changed in Oracle 11g. When using Data Guard Broker (DG_BROKER_START = TRUE), the monitor agent process named **Data Guard Broker Monitor (DMON)** is running on every site (primary and standby) and maintain a two-way communication.

**LOGICAL STANDBY PROCESSES ARCHITECTURE (REDO LOGS CONVERTED TO SQL, CALLED SQL APPLY)**

The major difference between the logical and physical standby database architectures is in its log apply services. On Logical Standby, you can query it while simultaneously
applying transactions from the primary. This is ideal for business that requires a near real-time copy of your production DB for reporting. The key advantage for logical standby databases is that they're opened read/write, even while they're in applied mode. That is, they can be used to generate reports and the like. It is indeed a fully functional database. Also, additional indexes, materialized views and so on can be created.

Oracle (or more exactly the log apply services) uses the primary database's redo log, transforms them into SQL statements and replays them on the logical standby database. SQL Apply uses LOGMINER technology to reconstruct DML statements from the redo generated on the primary.

The **logical standby process (LSP)** is the coordinator process for two groups of **parallel execution process (PX)** that work concurrently to read, prepare, build, and apply completed SQL transactions from the archived redo logs sent from the primary database. The first group of PX processes read log files and extract the SQL statements by using LogMiner technology; the second group of PX processes apply these extracted SQL transactions to the logical standby database. The mining and applying process occurs in parallel. Logical standby database does not use standby online redo logs. Logical standby database does not have FAL capabilities in Oracle. All gaps are resolved by the proactive gap resolution mechanism running on the primary that polls the standby to see if they have a gap.
In addition to data protection and availability, Data Guard standby databases deliver high return on investment by supporting ad-hoc queries, reporting, backups, or test activity, while in standby role. Specifically:

- **The Active Data Guard option (Oracle Database 11g)** enables a physical standby database to be used for read-only applications while simultaneously receiving updates from the primary database. Queries executed on an active standby database return up-to-date results. An Active Data Guard standby database is unique compared to other physical replication methods in its ability to guarantee the same level of read consistency as the primary database while replication is active. More information [HERE](#).

- **Snapshot Standby** enables a physical standby database to be open read-write for testing or any activity that requires a read-write replica of production data. A Snapshot Standby continues to receive, but not apply, updates generated by the primary. When testing is complete, the Snapshot Standby is converted back into a synchronized physical standby database by first discarding the changes made while open read-write, and then applying the redo received from the primary database. Primary data is protected at all times. More information [HERE](#).

- A physical standby database, because it is an exact replica of the primary database, can also be used to offload the primary database of the overhead of performing backups. All recovery operations are interchangeable between primary and Data Guard physical standby databases.

---

**DATA PROTECTION MODES**

**Maximum Protection**: It offers the highest level of data availability for the primary database.

In order to provide this level of protection, the redo data needed to recover each transaction must be written to both the local (online) redo log and to a standby redo log on at least one standby database before the transaction can be committed. In order to guarantee no loss of data can occur, the primary database will shut down if a fault prevents it from writing its redo data to at least one remote standby redo log. Redo records are **synchronously** transmitted from the primary database to the standby database using LGWR process. 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 mode is usually configured with at least two standby databases. **Standby online redo logs** are required in this mode. Therefore, logical standby database cannot participate in a maximum protection configuration. The
log_archive_dest_n parameter needs to have the **LGWR SYNC AFFIRM option**, for example:

```bash
log_archive_dest_2='service=testdb_standby LGWR SYNC AFFIRM'
```

**Maximum Availability**: Provides the highest level of data protection that is possible without affecting the availability of the primary database. This protection mode is very similar to maximum protection where a transaction will not commit until the redo data needed to recover that transaction is written to both the local (online) redo log and to at least one remote standby redo log. Redo records are *synchronously* transmitted from the primary database to the standby database using LGWR process. Unlike maximum protection mode; however, the primary database will not shut down if a fault prevents it from writing its redo data to a remote standby redo log. Instead, the primary database will operate in maximum performance mode until the fault is corrected and all log gaps have been resolved. After all log gaps have been resolved, the primary database automatically resumes operating in maximum availability mode. This protection mode supports both physical and logical standby databases. **Standby online redo logs** are required in this mode. The log_archive_dest_n parameter needs to have the **LGWR SYNC AFFIRM option**, for example:

```bash
log_archive_dest_2='service=testdb_standby LGWR SYNC AFFIRM'
```

**Maximum Performance**: It is the *default* protection mode. It offers slightly less primary database protection than maximum availability mode but with higher performance. Redo logs are *asynchronously* shipped from the primary database to the standby database using either LGWR or ARCH process. When operating in this mode, the primary database continues its transaction processing without regard to data availability on any standby databases and there is little or no effect on performance. It supports both physical and logical standby databases. The log_archive_dest_n parameter needs to have the **LGWR ASYNC AFFIRM or NOAFFIRM option**, for example:

```bash
log_archive_dest_2='service=testdb_standby ARCH NOAFFIRM'
```

or

```bash
log_archive_dest_2='service=testdb_standby LGWR ASYNC NOAFFIRM'
```

<table>
<thead>
<tr>
<th>Mode</th>
<th>Log Writing Process</th>
<th>Network Trans Mode</th>
<th>Disk Write Option</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Maximum Protection</strong></td>
<td>LGWR</td>
<td>SYNC</td>
<td>AFFIRM</td>
</tr>
<tr>
<td>Zero data loss Double failure protection</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Setting the Level of Protection
This is achieved by using the following command syntax executed on the primary database.

```
ALTER DATABASE SET STANDBY DATABASE TO MAXIMIZE
{ PROTECTION | AVAILABILITY | PERFORMANCE };
```

The protection mode can be found by executing this query. PERFORMANCE is the default.

```
SELECT name, protection_mode, protection_level FROM v$database;
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>PROTECTION_MODE</th>
<th>PROTECTION_LEVEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>STBY</td>
<td>MAXIMUM PERFORMANCE</td>
<td>MAXIMUM PERFORMANCE</td>
</tr>
</tbody>
</table>

-No Data Loss Mode. The PROTECTION mode applies only to physical standby databases, using LGWR SYNC and will shutdown the primary database if no standby database can be written to.

-Minimal Data Loss. The AVAILABILITY mode prevents a transaction committing on the primary until all redo entries are written to at least one standby database. SYNC transport is required and this option is available to both logical and physical standby type databases. Unlike PROTECTION mode, which shuts down the primary database in the event of failure to pass redo entries to the standby, this mode simply lowers the protection mode to PERFORMANCE until the error is corrected.
- No Data Divergence. PERFORMANCE mode is the default setting and available for both physical and logical standby type databases. A transaction will commit to the primary before all redo entries are written to any standby database.

To ensure that minimal data loss will be encountered execute this command on the primary database. The database must be in mounted exclusive mode to execute this command.

*ALTER DATABASE SET STANDBY DATABASE TO MAXIMIZE AVAILABILITY;*

**Terms or Options to know**
These can be found in the Oracle documentation in Chapter 5 of the Data Guard Concepts and Administration Manual.

*AFFIRM* assures that archive logs are written to disk, primary or standby.

*MANDATORY* assures that redo logs are not overwritten until archive logs are successfully created. This should only apply to the primary database.

*REOPEN*=30 means that there will be a 30 second delay until LGWR process try again on a MANDATORY destination which failed.

*DELAY* is in minutes and does not stop the copy of an archive log file to a standby server but the application of redo on the standby after copying the archive log to the standby. This will not help primary database performance.

*Using ARCH instead of LGWR for the second standby database may help primary database performance but smaller sized log files would probably be required.*

SYNC=PARALLEL applies to LGWR only. Using ARCH waits for a switch on the primary, LGWR copies entries to a standby archive log, applied only at switch. ARCH will copy and apply at switch. LGWR is more efficient since it writes redo entries to all standby databases at once but a primary and two standby databases could possibly cause a performance issue for the primary database, possibly but unlikely!

Additionally multiple archiver processes can be created on the primary database. Increase the value of the LOG_ARCHIVE_MAX_PROCESSES parameter to start additional archiver processes. The default on my machine appears to be 2 and not 1 as stated in the manuals; probably because I have two standby databases.

*The ARCHIVE_LAG_TARGET parameter could be used to increase the frequency of log switches, thus sending less data to the standby databases more often. Specifies the maximum number of seconds between each log switch, so it will force a log switch when that number in seconds is reached. Used on Physical Implementation Only.*
Using Data Guard Redo Apply in a LAN the following is recommended:

- Use Maximum Protection or Maximum Availability modes for zero data loss; the performance impact was less than 3% in all synchronous tests. With a single remote archive destination, use the NOPARALLEL option (“lgwr sync=noparallel”).
- For very good performance and a minimal risk of transaction loss in the event of a disaster, use Maximum Performance mode, with LGWR ASYNC and a 10 MB async buffer (ASYNC=20480). LGWR ASYNC performance degraded no more than 1% as compared to using the ARCH transport. LGWR ASYNC also bounds the risk of potential transaction loss much better than the ARCH transport. The 10 MB async buffer outperformed smaller buffer sizes and reduced the chance of network timeout errors in a high latency / low bandwidth network.

Metropolitan and Wide Area Network (WAN)

Data Guard is used across a metropolitan area networks (MAN) or WANs to get complete disaster recovery protection. Typically a MAN covers a large metropolitan area and has network Round-Trip-Times (RTT) from 2-10 ms. For the MAN/WAN tests, different network RTT’s were simulated during testing to measure the impact of the RTT on the primary database performance. The tests were conducted for the following RTT’s: 2 ms (MAN), 10 ms, 50 ms, and 100 ms (WAN) Additionally, tests using Secure Shell (SSH) port forwarding with compression were also done for different RTT’s.

Best practices recommendations are:
- Use Maximum Protection and Maximum Availability modes over a MAN for zero data loss. For these modes, the network RTT overhead over a WAN can impact response time and throughput of the primary database. The performance impact was less than 6% with a 10 ms network RTT and a high transaction rate.
- For very good performance and a minimal risk of transaction loss in the event of a disaster, use Maximum Performance mode, with LGWR ASYNC and a 10 MB async buffer (ASYNC=20480). LGWR SYNC performance degraded no more than 2% as compared to remote archiving. The 10 MB async buffer outperformed smaller buffer sizes and reduced the chance of network timeout errors in a high latency / low bandwidth network.
- For optimal primary database performance throughput, use remote archiving (i.e. the ARCH process as the log transport). This configuration is best used when network bandwidth is limited and when your applications can risk some transaction loss in the event of a disaster.
• If you have sufficient memory, then set the TCP send and receive buffer sizes (these affect the advertised TCP window sizes) to the bandwidth delay product, the bandwidth times the network round trip time. This can improve transfer time to the standby by as much as 10 times, especially with the ARCH transport.

**Best Practices for Network Configuration and Highest Network Redo Rates**

• Set SDU=32768 (32K) for the Oracle Net connections between the primary and standby. Setting the Oracle network services session data unit (SDU) to its maximum setting of 32K resulted in a 5% throughput improvement over the default setting of 2048 (2K) for LGWR ASYNC transport services and a 10% improvement for the LGWR SYNC transport service. SDU designates the size of the Oracle Net buffer used to collect data before it is delivered to the TCP network layer for transmission across the network. Oracle internal testing of Oracle Data Guard has demonstrated that the maximum setting of 32767 performs best. The gain in performance is a result of the reduced number of system calls required to pass the data from Oracle Net buffers to the operating system TCP network layer. SDU can be set on a per connection basis with the SDU parameter in the local naming configuration file (tnsnames.ora) and the listener configuration file (listener.ora), or SDU can be set for all Oracle Net connections with the profile parameter DEFAULT_SDU_SIZE in the sqlnet.ora file. This is specially true for WAN environment.

• Use SSH port forwarding with compression for WAN’s with a large RTT when using maximum performance mode. Do not use SSH with compression for Maximum Protection and Maximum Availability modes since it adversely affected the primary throughput. Using SSH port forwarding with compression reduced the network traffic by 23-60% at a 3-6% increase in CPU usage. This also eliminated network timeout errors. With the ARCH transport, using SSH also reduced the log transfer time for RTT’s of 50 ms or greater. For RTT’s of 10ms or less, the ARCH transport log transfer time was increased when using SSH with compression.

• Ensure TCP.NODELAY is YES

To preempt delays in buffer flushing in the TCP protocol stack, disable the TCP Nagle algorithm by setting TCP.NODELAY to YES in the SQLNET.ORA file on both the primary and standby systems.

**PHYSICAL STANDBY IMPLEMENTATION**
NOTE = If you want to use the new way to setup Data Guard with RMAN, please check the following LINK

The following example shows how to set up Data Guard MANUALLY in this given environment:

1. The production database name is FGUARD
2. One primary database instance called FGUARD on host server_01; one physical standby database instance called FGUARD on host server_02.
3. Listener listener is on host server1, and pointed by TNS entry FGUARD
4. Listener listener is on host server2, and pointed by TNS entry FGUARD.
5. The purpose of TNS entry FGUARD and FGUARD are used for LGWR/ARCH process to ship redo logs to the standby site, and for FAL process to fetch redo logs from the primary site.
6. We will be implementing the configuration manually, but we will set dg_broker_start to true, so Data Guard broker can be used later.
7. The protection mode is set to best performance. Therefore, only local archive destination (log_archive_dest_1) is set to mandatory; the standby archive destination (log_archive_dest_2) is set to optional for LGWR process, with network transmission method of asynchronous and disk write option of no affirm.
8. The standby site is not using standby online redo logs. Therefore, the redo log reception option is archived logs.

Section 1: Site Information
Primary Site:
Database Name: FGUARD
Primary Server : server_01
Primary Instance Name: FGUARD
Primary Listener: LISTENER
Recovery Database: DR_FGUARD

Standby Site:
Database Name: FGUARD
Standby Server: server_02
Standby Instance name: FGUARD
Standby Listener: LISTENER
Production DB: PROD_FGUARD
Section 2: Oratab /etc/oratab entry:

<table>
<thead>
<tr>
<th>Primary Site:</th>
<th>Standby Site:</th>
</tr>
</thead>
<tbody>
<tr>
<td>FGUARD:/u01/app/oracle/product/11.2.0:Y</td>
<td>FGUARD:/u01/app/oracle/product/11.2.0:N</td>
</tr>
</tbody>
</table>

Section 3: Parameter file

Primary init.ora file:
archive_lag_target = 1800  # specifies how often in seconds we will force a switch log
db_name = FGUARD
db_unique_name = FGUARD
#fal_server = DR_FGUARD  # PROD DB used on tnsnames.ora
#fal_client = FGUARD  # this DB used on tnsnames.ora
log_archive_dest_1 = 'LOCATION=/u02/arch/PROD MANDATORY'  # Local Location of Archive Log Files
log_archive_dest_2 = 'SERVICE=DR_FGUARD reopen=60'  # Remote Service Name based on tnsnames.ora
log_archive_dest_state_1 = 'enable'
log_archive_dest_state_2 = 'enable'
log_archive_format = 'arch_t%t_s%s.dbf'
log_archive_start = true  # (not used on 10g)
standby_archive_dest = '/oracle/ARCH'
standby_file_management = 'AUTO'  # If auto, newly created tablespaces/datafiles must be created manually on the standby environment.
dg_broker_start = true

New Ones
service_names =FGUARD
instance_name =FGUARD
log_archive_config="dg_config=(FGUARD,DR_FGUARD)"
-->
>This parameter is required by the Data Guard Broker
log_archive_max_processes=5
log_archive_dest_1='location=D:\oracle\product\10.1.0\flash_recovery_area\FGUARD\ARCHIVELOG
valid_for=(ALL_LOGFILES,ALL_ROLES) db_unique_name=FGUARD'
log_archive_dest_2= 'service=standby LGWR SYNC AFFIRM
valid_for=(ONLINE_LOGFILES,PRIMARY_ROLE)
DB_UNIQUE_NAME=DR_FGUARD'
standby_archive_dest=D:\oracle\product\10.1.0\flash_recovery_area\PRIMARY\ARCHIVELOG

Optional parameters:
log_archive_dest_2='service=stby lgwr sync affirm mandatory reopen=180'
LOG_ARCHIVE_DEST_2 - Specifies the net service name of the standby database (check tnsnames.ora on primary database). You can either per destination use LGWR or ARCH or both, due to network traffic it is advised to use LGWR for at most one remote destination. Also the network transmission mode (SYNC or ASYNC) has to be specified in case primary database modifications are propagated by the LGWR. The NO DATA LOSS situation demands the SYNC mode, control is not returned to the executing application or user until the redo information is received by the standby site (this can have impact on the performance as mentioned).

Standby init.ora file:

```
db_name = FGUARD
db_unique_name = DR_FGUARD   --> MUST BE DIFFERENT FROM PRIMARY SITE
fal_server = PROD_FGUARD     #PROD DB used on tnsnames.ora
fal_client = FGUARD          #this DB used on tnsnames.ora
log_archive_dest_1 = 'LOCATION=/oracle/ARCH MANDATORY'   #This parameter should always coincide with the standby_archive_dest parameter
log_archive_dest_state_1 = 'enable'
#log_archive_dest_2 =
#'SERVICE=PROD_FGUARD reopen=60'
#log_archive_dest_state_2 = 'enable'
log_archive_format = 'arch_t%t_s%s.dbf'
log_archive_start = true    (not used on 10g)
standby_archive_dest = '/oracle/ARCH'          ##This parameter should always coincide with
                                             the log_archive_dest_1 parameter
standby_file_management = 'AUTO'
dg_broker_start = true
```
New Ones

service_names='DR_FGUARD'
instance_name=DR_FGUARD
control_files='D:\oradata\PRIMARY_STDBY.CTL'
log_archive_config="dg_config=(FGUARD,DR_FGUARD)"

> This parameter is required by the Data Guard Broker

log_archive_max_processes=5
log_archive_dest_1='location=D:\oracle\product\10.1.0\flash_recovery_area\PRIMARY\ARCHIVELOG
valid_for=(ALL_LOGFILES,ALL_ROLES)
db_unique_name=DR_FGUARD'
log_archive_dest_2= 'service=FGUARD LGWR SYN AFFIRM
valid_for=(ONLINE_LOGFILES,PRIMARY_ROLE)
DB_UNIQUE_NAME=FGUARD'
standby_archive_dest=D:\oracle\product\10.1.0\flash_recovery_area\PRIMARY\ARCHIVELOG

Optional Parameters:
db_file_name_convert=('disk1/oracle/oradata/payroll/','disk1/oracle/oradata/payroll/standby/')
or

db_file_name_convert=('PROD2/','PROD')

log_file_name_convert=('disk1/oracle/oradata/payroll/','disk1/oracle/oradata/payroll/standby/')
or

log_file_name_convert=('PROD2/','PROD')

DB_FILE_NAME_CONVERT - Specifies the location of datafiles on standby database. The two arguments that this parameter needs are: location of datafiles on primary database, location of datafiles on standby database. This parameter will convert the filename of the primary database datafiles to the filename of the standby datafile filenames. If the standby database is on the same system as the primary database or if the directory structure where the datafiles are located on the standby site is different from the primary site then this parameter is required. See Section 3.2.1 for the location of the datafiles on the primary database. Used on Physical Implementation ONLY.

LOG_FILE_NAME_CONVERT - Specifies the location of redo logfiles on standby database. The two arguments that this parameter needs are: location of redo logfiles on primary database, location of redo logfiles on standby database. This parameter will convert the filename of the primary database log to the filenames of the standby log. If
the standby database is on the same system as the primary database or if the directory structure where the logs are located on the standby site is different from the primary site then this parameter is required. Used on Physical Implementation ONLY.

Section 4: Listener.ora file

<table>
<thead>
<tr>
<th>Primary Site:</th>
<th>Standby Site:</th>
</tr>
</thead>
<tbody>
<tr>
<td>FGUARD = (DESCRIPTION =</td>
<td>FGUARD= (DESCRIPTION =</td>
</tr>
<tr>
<td>(ADDRESS_LIST =</td>
<td>(ADDRESS_LIST =</td>
</tr>
<tr>
<td>(ADDRESS = (PROTOCOL = TCP)(HOST = server1)(PORT = 1521)) ) )</td>
<td></td>
</tr>
<tr>
<td>(CONNECT_DATA =</td>
<td>(ADDRESS = (PROTOCOL = TCP)(HOST = server2)(PORT = 1521)) ) )</td>
</tr>
<tr>
<td>(SERVER = DEDICATED)</td>
<td>(CONNECT_DATA =</td>
</tr>
<tr>
<td>(SERVICE_NAME = FGUARD)</td>
<td>(SERVICE_NAME = FGUARD)</td>
</tr>
<tr>
<td>) )</td>
<td>) )</td>
</tr>
<tr>
<td>DR_FGUARD = (DESCRIPTION =</td>
<td>PROD_FGUARD = (DESCRIPTION =</td>
</tr>
<tr>
<td>(ADDRESS_LIST =</td>
<td>(ADDRESS = (ADDRESS = (PROTOCOL = TCP)(HOST = server1)(PORT = 1521)) ) )</td>
</tr>
<tr>
<td>(ADDRESS = (PROTOCOL = TCP)(HOST = server2)(PORT = 1521)) ) )</td>
<td></td>
</tr>
<tr>
<td>(CONNECT_DATA =</td>
<td>(CONNECT_DATA =</td>
</tr>
<tr>
<td>(SERVER = DEDICATED)</td>
<td>(SERVER = DEDICATED)</td>
</tr>
<tr>
<td>(SERVICE_NAME = FGUARD)</td>
<td>(SERVICE_NAME = FGUARD)</td>
</tr>
<tr>
<td>) )</td>
<td>) )</td>
</tr>
</tbody>
</table>

STEPS TO SET UP A PHYSICAL DATA GUARD ENVIRONMENT
The following steps show how to set up a Physical Data Guard environment:

STEP 1: THE PREPARATION

- Ensure primary database is in ARCHIVELOG mode
If Archive Log Mode is not enabled for your primary database, enable it using the following. First, you will need to define the following instance variables:
alter system set log_archive_dest_1 = 'LOCATION=/oracle/arch MANDATORY' scope=both;
alter system set log_archive_dest_state_1 = 'enable' scope=both;
alter system set log_archive_format = 'arch_t%t_s%s.dbf' scope=both;
shutdown immediate;
startup mount;
alter database archivelog;
alter database open;
archive log list;

- Enable Forced Logging. This option ensures that even in the event that a 'nologging' operation is done, force logging takes precedence and all operations are logged into the redo logs.
alter database force logging;

**Create standby redo log files (recommended):**
Standby redo logs are necessary for the higher protection levels such as Guaranteed, Instant, and Rapid. In these protection modes LGWR from the Primary host writes transactions directly to the standby redo logs. This enables no data loss solutions and reduces the amount of data loss in the event of failure. Standby redo logs are not necessary if you are using the delayed protection mode.
If you configure standby redo on the standby then you should also configure standby redo logs on the primary database. Even though the standby redo logs are not used when the database is running in the primary role, configuring the standby redo logs on the primary database is recommended in preparation for an eventual switchover operation

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 (performed by the LGWR process) is a requirement for selection of a standby redo log
You must have the same number of standby redo logs on the standby as you have online redo logs on production. They must also be exactly the same size.

```sql
select * from v$logfile;
GROUP# STATUS TYPE MEMBER IS_
---------- ------- ------ ---------------
1 ONLINE /export/home/oracle/temp/oracle/data/redo01.log NO
2 ONLINE /export/home/oracle/temp/oracle/data/redo02.log NO
3 ONLINE /export/home/oracle/temp/oracle/data/redo03.log NO

select bytes from v$log;
BYTES
----------
52428800
52428800
```
The following syntax is used to create standby redo logs:
SQL> alter database add standby logfile GROUP 4 size 50m;
SQL> alter database add standby logfile GROUP 5 size 50m;
SQL> alter database add standby logfile GROUP 6 size 50m;

select * from v$logfile;
GROUP# STATUS TYPE MEMBER
---------- ------ ------ ------------------

1 redo01.log ONLINE /export/home/oracle/temp/oracle/data/(NO)
2 redo02.log ONLINE /export/home/oracle/temp/oracle/data/(NO)
3 redo03.log ONLINE /export/home/oracle/temp/oracle/data/NO
4 standbyredo01.dbf STANDBY /export/home/oracle/temp/oracle/data/NO
5 standbyredo01.dbf STANDBY /export/home/oracle/temp/oracle/data/NO
6 standbyredo01.dbf STANDBY /export/home/oracle/temp/oracle/data/NO

select * from v$standby_log;
GROUP# DBID THREAD# SEQUENCE# BYTES USED ARC
STATUS FIRST_CHANGE# FIRST_TIME LAST_CHANGE# LAST_TIME
-------- ------- ------- -------- --------- -------
4 UNASSIGNED 0 0 52428800 512 YES

NOTE: In the above example db_file_name_convert and log_file_name_convert are not needed as the directory structure on the two hosts are the same. If the directory structure is not the same then setting of these parameters is recommended. Please reference ML notes 47325.1 and 47343.1 for further information.

Note here that the Primary init.ora on the Standby host to have log_archive_dest_2 use the alias that points to the Primary host. You must modify the Standby init.ora on
the standby host to have fal_server and fal_client use the aliases when standby is running on the Primary host.

- Setup the tnsnames.ora and listener.ora file for both primary and standby databases. (see section 4)

**STEP 2: BACKUP THE PRIMARY DATABASE DATAFILES**

- Shut down the primary database.
- Backup the primary database datafiles and online redo logs. A backup of the online redo logs is necessary to facilitate switchover.

```bash
$ cp /u02/oradata/prod/* /oracle/BCKUP
```

or

```sql
set pages 50000 lines 120 head off veri off flush off tie off
select 'cp ' || file_name || ' /oracle/BCKUP' from
dba_data_files
UNION
select 'cp ' || file_name || ' /oracle/BCKUP' from
dba_temp_files
UNION
select 'cp ' || member || ' /oracle/BCKUP' from v$logfile;
```

**STEP 3: CREATE THE PHYSICAL STANDBY DATABASE CONTROL FILE**

- Startup the Primary database and issue the following command to create the standby control file (it must be done AFTER the backup):

```sql
SQL> alter database create standby controlfile as '/oracle/BCKUP/standby.ctl';
```

**NOTE:** The controlfile must be created after the last time stamp for the backup datafiles.

**STEP 4: TRANSFER THE DATAFILES AND STANDBY CONTROL FILE TO THE STANDBY SITE**

- Transfer the backuped datafiles, redo log files and archived redo logs to the standby site.

```bash
$ rcp /oracle/BCKUP/* server2:/u02/oradata/FGUARD
```

- Transfer the standby control file to the standby site
$ rcp /oracle/BCKUP/standby.ctl
server2:/u01/app/oracle/admin/FGUARD/ctl/control01.ctl

- Also copy the orapwd file and init.ora files. A password file must be created on the Primary and copied over to the Standby site. The sys password must be identical on both sites. This is a key pre requisite in order to be able to ship and apply archived logs from Primary to Standby. If you need to generate a password file perform the following:
  
  cd $ORACLE_HOME/dbs
  orapwd file=orapwFGUARD password=oracle force=y ignorecase=y

  If needed perform the following on the other system:
  chmod 6751 orapwSID

- Put all of them on the correct places

**STEP 5: START THE LISTENERS ON BOTH PRIMARY AND STANDBY SITE**

- If the standby system is running on a Windows-based system, use the ORADIM utility to create a Windows Service and password file. For example:
  
  WINNT> oradim -NEW -SID databaseid -INTPWD password -STARTMODE manual

- Start the the listener on the primary and standby database
  
  $ lsnrctl start

**STEP 6: START THE STANDBY DATABASE (PRIMARY DATABASE ALREADY RUNNING)**

- Set the correct Oracle environment and copy all the files to each location
  
  If the standby is on a separate site with the same directory structure as the primary database then you 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.
  
  If the standby is on the same site as the primary database, or the standby database is on a separate site with a different directory structure the you must rename the primary datafiles in the standby control file after copying them to the standby site. This can be done using the *db_file_name_convert* and *log_file_name_convert* parameters or by manually using the alter database statements. If the directory structure is not the same then reference notes 47325.1 and 47343.1 for further information.
If you decided to rename them manually, you **MUST** use `ALTER DATABASE RENAME FILE` after the standby database is mounted to rename the database files and redo log files.

If needed, copy the Standby Controlfile that your created FROM the production DB to the appropriate location on the standby DB according your init.ora file

$ cd
$ cp standby.ctl /u03/app/oradata/FGUARD/control01.ctl
$ cp standby.ctl /u04/app/oradata/FGUARD/control02.ctl
$ cp standby.ctl /u05/app/oradata/FGUARD/control03.ctl

- Connect as sysdba.
  
  create spfile from pfile;

- Bring the database in nomount mode first.
  
  startup nomount;

- Mount the standby database.
  
  alter database mount standby database;

If you decided to rename the database files manually, you **MUST** use `ALTER DATABASE RENAME FILE <oldname> TO <newname>` after the standby database is mounted.

**STEP 7: PLACE THE STANDBY DATABASE IN MANAGED RECOVERY MODE**

- Issue the following command to bring the standby database in managed recover mode (start log apply services).
  
  `alter database recover managed standby database disconnect from session;`

  **NOTE:** The example includes the `DISCONNECT FROM SESSION` option so that log apply services run in a background session.

**STEP 8: MONITOR THE LOG TRANSPORT SERVICES AND LOG APPLY SERVICES**

- With the protection mode we are using, *Maximum Performance*, archiving of redo logs to the remote standby location do not occur until after a log switch. A log switch occurs, by default, when an online redo log becomes full. Issue a few log switches on the primary database.
SQL> alter system switch logfile;
or
SQL> alter system archive log current;

- Confirm the log files received on the standby archive destination.
- Check the standby alert log file to see if the new logs have applied to the standby database.

Media Recovery Log /u02/arch/FGUARD/prod_1482.arc
- Monitor the managed recovery.

```
select process, status, thread#, sequence#, block#, blocks
    from v$managed_standby;
```
or
```
select sequence#, first_time, next_time
    from v$archived_log order by sequence#;
```

- Verify that the new archived redo log was applied

From the standby database, query the V$ARCHIVED_LOG view to verify the archived redo log was applied.
```
select sequence#, archived, applied
    from v$archived_log order by sequence#;
```

<table>
<thead>
<tr>
<th>SEQUENCE#</th>
<th>ARCHIVED</th>
<th>APPLIED</th>
</tr>
</thead>
<tbody>
<tr>
<td>115</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>116</td>
<td>YES</td>
<td>YES</td>
</tr>
</tbody>
</table>

**Quick Steps for Creating the Physical Standby from Hot Backup**

**At PROD Site**
put proper values on the primary db using "alter system set...... "
create pfile from spfile;
alter system switch logfile;
alter system archive log current;
perform hot backup

vi Quick_Hot_Backup.sql
set serveroutput on
set heading off
set feedback off
Set verify off
accept destination prompt 'Enter destination (like /home/dpafumi/) : '
Set Termout off
spool hotbackups.sql
declare
fname varchar2(80);
tname varchar2(80);
tname1 varchar2(80);
aux varchar2(100);
cursor cur1 is
    select tablespace_name, file_name
    from v$datafile, sys.dba_data_files
    where enabled like '%WRITE%'
    and file# = file_id
    order by 1;
begin
    dbms_output.enable(32000);
    dbms_output.put_line('spool hotbackups.txt');
    if cur1%ISOPEN
    then
        close cur1;
    end if;
    open cur1;
    fetch cur1 into tname, fname;
    tname1 := tname;
    dbms_output.put_line('alter tablespace ||tname|| begin backup;');
    while cur1%FOUND loop
        if tname1 != tname then
            dbms_output.put_line('alter tablespace ||tname1|| end backup;');
            tname1 := tname;
            dbms_output.put_line('alter tablespace ||tname|| begin backup;');
        end if;
        dbms_output.put_line('!cp '||fname||' &destination');
        fetch cur1 into tname, fname;
    end loop;
    dbms_output.put_line('alter tablespace ||tname1|| end backup;');
    close cur1;
    dbms_output.put_line('alter system switch logfile;');
    dbms_output.put_line('!sleep 10');
dbms_output.put_line('!cp
/oracle/oracle7/app/oracle/admin/DIE/ARCHIVE/*.arc '||
'&&destination');
    dbms_output.put_line('alter database backup controlfile to
trace;');
    dbms_output.put_line('alter database backup controlfile to '||
CHR(39)|| '&&destination' || '/control.'||
to_char(sysdate,'DDMMYYYYHH24MISS')|| CHR(39) ||';');
    dbms_output.put_line('REM *** Copy Last file from udump ***');
    dbms_output.put_line('spool off');
end;
/
spool off
set heading on
set feedback on
set serveroutput off
-- Unremark/Uncomment the following line to run the script
-- or can be run from the sqlplus prompt.
-- @hotbackups
@Quick_Hot_Backup.sql
@hotbackups

--You can check your backup status with:
select  substr(b.status,1,10) status,
        substr(f.TABLESPACE_NAME,1,15) Ts_Name,
        substr(d.NAME,1,50) Datafile
from    v$backup b, dba_data_files f, v$datafile d
where   b.file# = d.file#
        and d.NAME = f.FILE_NAME;

- Compress files
- Copy compressed files to standby site
scp CC_* .Z oracle@10.10.10.10:/home/oracle/DBA_SCRIPTS/BCKP
........
- sqlplus "/ as sysdba"
    alter database create standby controlfile as '/home/dpafumi/standby.ctl';
    alter database backup controlfile to trace;
- transfer init.ora, passwd and ctrlfile.bin files to standby DB
scp standby.ctl oracle@10.10.10.10:/home/oracle/DBA_SCRIPTS/BCKP
-- transfer LATEST ARCH Log Files
alter system switch logfile;
alter system archive log current;
cd to arch location
cp ARCH*.dbf /home/dpafumi
scp ARCH*.dbf oracle@10.10.10.10:/home/oracle/DBA_SCRIPTS/BCKP

At standby site:
- Copy *.dbf, *log, standby.ctl and *.ora to proper locations
- If needed, copy the Standby Controlfile that your created FROM the production DB
to the appropriate location on the standby DB according your init.ora file
$ cd
$ cp standby.ctl /u03/app/oradata/FGUARD/control01.ctl
$ cp standby.ctl /u04/app/oradata/FGUARD/control02.ctl
$ cp standby.ctl /u05/app/oradata/FGUARD/control03.ctl

If needed perform the following on the other system:
chmod 6751 orapwSID

- Modify init.ora file containing correct information (like udump, ctrl file, etc)
Use if needed
db_file_name_convert=('/opt/oracle/product/10.1.0/db_1/oradata/FGUARD/','/data/oradata/FGUARD/)
        log_file_name_convert=('/opt/oracle/product/10.1.0/db_1/oradata/FGUARD/','/data/oradata/FGUARD/)

- sqlplus "/ as sysdba"
create spfile from pfile;
startup nomount;
alter database mount standby database;
recover standby database;
alter database recover managed standby database disconnect from session;

-- Test and Monitor Arch Log Files Transfer
-- On PROD
alter system switch logfile;

LOGICAL STANDBY
IMPLEMENTATION

**NOTE** = If you want to use the new way to setup Data Guard with RMAN, please check the following [LINK](#)

**Prior to creating the Logical Standby ensure the following:**
The physical organization in a logical standby database is different from that of the primary database, even though the logical standby database is created from a backup copy of the primary database. Thus, ROWIDs contained in the redo records generated by the primary database cannot be used to identify the corresponding row in the logical standby database.

Oracle uses primary-key or unique-key supplemental logging to logically identify a modified row in the logical standby database. When database-wide primary-key and unique-key supplemental logging is enabled, each UPDATE statement also writes the column values necessary in the redo log to uniquely identify the modified row in the logical standby database.

- If a table has a primary key defined, then the primary key is logged as part of the UPDATE statement to identify the modified row.
- In the absence of a primary key, the shortest non-null unique key is logged as part of the UPDATE statement to identify the modified row.
- In the absence of both a primary key and a non-null unique key, all columns of bounded size are logged as part of the UPDATE statement to identify the modified row. In other words, all columns except those with the following types are logged: LONG, LOB, LONG ROW, object type, and collections.

- Determine if the primary database contains tables and datatypes that were not supported by a logical stand by database. If the primary database contains tables that were unsupported, log apply services will exclude the tables from the logical standby database:

    ```sql
    SELECT DISTINCT OWNER, TABLE_NAME FROM DBA_LOGSTDBY_UNSUPPORTED ORDER BY OWNER, TABLE_NAME;
    ```

    To view the column names and data types which are not supported for one of the tables listed in the previous query, run the following select:

    ```sql
    col DATA_TYPE format a20
col COLUMN_NAME format a20
    SELECT COLUMN_NAME, DATA_TYPE FROM
    ```
DBA_LOGSTDBY_UNSUPPORTED WHERE OWNER='XXX' AND TABLE_NAME = 'YYY';

- Ensure that table rows in the primary database can be uniquely identified. Oracle recommends that you add a primary key or a non-null unique index to tables in the primary database, whenever possible, to ensure that SQL Apply can efficiently apply redo data updates to the logical standby database. Find tables without unique logical identifier in the primary database:

```
SELECT OWNER, TABLE_NAME FROM DBA_LOGSTDBY_NOT_UNIQUE WHERE (OWNER, TABLE_NAME) NOT IN (SELECT DISTINCT OWNER, TABLE_NAME FROM DBA_LOGSTDBY_UNSUPPORTED) AND BAD_COLUMN = 'Y';
```

If on the previous command you get a reference over the TSMSYS.SRS$ Table, you can safely ignore it. TSMSYS is created by rdbms/admin/cattsm.sql for transparent session migration. It’s expired & locked by default.

- Add a primary key to the tables that do not have to improve performance. If the table has a primary key or a unique index with a non-null column, the amount of information added to the redo log is minimal. If your application ensures the rows in a table are unique, you can create a disabled primary key RELY constraint on the table. This avoids the overhead of maintaining a primary key on the primary database.

```
ALTER TABLE TableA ADD PRIMARY KEY (id, name) RELY DISABLE;
```

When you specify the RELY constraint, the system will assume that rows are unique. Because you are telling the system to rely on the information, but are not validating it on every modification done to the table, you must be careful to select columns for the disabled RELY constraint that will uniquely identify each row in the table. If such uniqueness is not present, then SQL Apply will not correctly maintain the table.

To improve the performance of SQL Apply, add an index to the columns that uniquely identify the row on the logical standby database. Failure to do so results in full table scans during UPDATE or DELETE statements carried out on the table by SQL Apply.

- Ensure that Primary database is in archivelog mode.

```
SQL> archive log list;
```
- **Create a Physical Standby Database**
  For more information about how to create a physical standby database click [HERE](#).

Once the Physical Standby is created, you can move on.

**On STANDBY database**
1. Make sure that all the sequences generated on primary side has been applied on standby.
   
   ```sql
   SELECT ARCH.THREAD# "Thread", ARCH.SEQUENCE# "Last Sequence Received", APPL.SEQUENCE# "Last Sequence Applied"
   FROM
   (SELECT THREAD# ,SEQUENCE# FROM V$ARCHIVED_LOG WHERE (THREAD#,FIRST_TIME ) IN (SELECT THREAD#,MAX(FIRST_TIME) FROM V$ARCHIVED_LOG GROUP BY THREAD#)) ARCH,
   (SELECT THREAD# ,SEQUENCE# FROM V$LOG_HISTORY WHERE (THREAD#,FIRST_TIME ) IN (SELECT THREAD#,MAX(FIRST_TIME) FROM V$LOG_HISTORY GROUP BY THREAD#)) APPL
   WHERE ARCH.THREAD# = APPL.THREAD# ORDER BY 1;
   ```

2. Cancel the recovery on standby database
   ```sql
   ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;
   ```

**On PRIMARY database**
1. Review Destination Parameter
   ```sql
   show parameters log_archive_dest_1
   log_archive_dest_1 string LOCATION=F:\oracle\oradata\ARCH MANDATORY
   ```

2. On primary set the VALID_FOR parameter to make destination 1 as valid for online redo logs and role as all roles.
   This mean even if we change the role of primary server to standby server, online redo logs will get archived to archive destination
   ```sql
   alter system set log_archive_dest_1='location=F:\oracle\oradata\ARCH valid_for=(ONLINE_LOGFILES,ALL_ROLES)' scope=both;
   ```

3. Create another archive destination log_archive_dest_3 which will archive only standby redo logs and role should be set to standby_role.
   This means that in case if we make this primary as standby then standby redo logs will
be archived automatically at archive destination 3.

```
mkdir F:\oracle\oradata\ARCH\standby
alter system set
LOG_ARCHIVE_DEST_3='LOCATION=F:\oracle\oradata\ARCH\standby VALID_FOR=(STANDBY_LOGFILES,STANDBY_ROLE)' scope=both;
alter system set LOG_ARCHIVE_DEST_STATE_3='enable' scope=both;
```

4. Run DBMS_LOGSTDBY.BUILD package to create metadata for log miner to apply SQLs on logical standby site.
   A LogMiner dictionary must be built into the redo data so that the LogMiner component of SQL Apply can properly interpret changes it sees in the redo. Additionally, supplemental logging is set up to log primary key and unique-index columns. The supplemental logging information ensures each update contains enough information to logically identify each row that is modified by the statement.
   ```
   EXECUTE DBMS_LOGSTDBY.BUILD;
   ```

The DBMS_LOGSTDBY.BUILD procedure waits for all existing transactions to complete. Long-running transactions executed on the primary database will affect the timeliness of this command. The DBMS_LOGSTDBY.BUILD procedure uses Flashback Query to obtain a consistent snapshot of the data dictionary that is then logged in the redo stream. Oracle recommends setting the UNDO_RETENTION initialization parameter to 3600 on both the primary and logical standby databases.

After running this command, an entry will go in redo log file and when that redo log file gets shipped to standby, from that point onwards SQL apply will start on standby.

5. Supplemental logging should already be enabled on the primary database as a result of building the LogMiner Multiversioned Data Dictionary in the previous section.
   ```
   select supplemental_log_data_pk,supplemental_log_data_ui from v$database;
   SUP SUP
   --- ---
   YES YES
   ```
   If the supplemental logging is not enabled, execute the following
   ```
   SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (PRIMARY KEY,UNIQUE INDEX) COLUMNS;
   SQL> ALTER SYSTEM SWITCH LOGFILE;
   ```
On STANDBY database

1. The redo logs contain the information necessary to convert your physical standby database to a logical standby database.
To continue applying redo data to the physical standby database until it is ready to convert to a logical standby database, issue the following SQL statement:

```sql
ALTER DATABASE RECOVER TO LOGICAL STANDBY NEW_NAME;
```

On a Physical Standby, we used the same DB_Name as the primary.
But now we have to change the actual DB Name of the standby so it can become a logical standby database.
Data Guard will change the database name (DB_NAME) and set a new database identifier (DBID) for the logical standby.

For db_name, specify a database name that is different from the primary database to identify the new logical standby database.
If you are using a server parameter file (SPFILE) at the time you issue this statement, then the database will update the DB_NAME parameter with appropriate information about the new logical standby database. If you are using a PFILE, then the database issues a message reminding you to set the name of the DB_NAME parameter after shutting down the database.

The statement waits, applying redo data until the LogMiner dictionary is found in the log files.
This may take several minutes, depending on how long it takes redo generated to be transmitted to the standby database, and how much redo data need to be applied. If a dictionary build is not successfully performed on the primary database, this command will never complete.
You can cancel the SQL statement by issuing the ALTER DATABASE RECOVER MANAGED standby database CANCEL statement from another SQL session.

2. Create a New Password File
In 11g you do not need to re-create the password file when converting your physical standby to a logical standby. If you do, it will not work.
Because the conversion process changes the database name (that was originally set with the DB_NAME initialization parameter) for the logical standby database, you must re-create the password file.

```
cd $ORACLE_HOME/dbs
rm orapwfguard
orapwd file=orapwfguard password=MySysPassword
```
3. Modifying LOG_ARCHIVE_DEST_n parameters
You need to perform this modification because, unlike physical standby databases, logical standby databases are open databases that generate redo data and have multiple log files (online redo log files, archived redo log files, and standby redo log files).

```
alter system set
LOG_ARCHIVE_DEST_1='LOCATION=C:\oracle\product\10.2.0\oradata\FG
UARD_ARCH VALID_FOR=(ONLINE_LOGFILES,ALL_ROLES)' scope=both;
alter system set LOG_ARCHIVE_DEST_2='service=PROD_FGUARD LGWR
ASYNC VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)' scope=both;
alter system set
LOG_ARCHIVE_DEST_3='LOCATION=C:\oracle\product\10.2.0\oradata\FG
UARD_ARCH stdby_redo_log
VALID_FOR=(STANDBY_LOGFILES,STANDBY_ROLE)' scope=both;
alter system set LOG_ARCHIVE_DEST_STATE_1='enable' scope=both;
alter system set LOG_ARCHIVE_DEST_STATE_2='enable' scope=both;
alter system set LOG_ARCHIVE_DEST_STATE_3='enable' scope=both;
```

Log_archive_dest_1 -> Archived redo log files that store redo data generated by the logical standby database. It will also work if the role of database is Primary or Standby

Log_archive_dest_2 -> This is for shipping the online redo logs to primary server, in case this becomes primary server and primary becomes standby server. This is called auto role shifting. Till the time this database is standby it will not ship and redo logs to primary.

Log_archive_dest_3 -> Data received from the primary database (Arch Log Files generated on PROD will be shipped to this location).

This is for archiving all standby redo logs. This will work only when the role of database is Standby.

In case this becomes primary, it will not create any standby archive logs at this dest 3.

The following table show you various roles and redo log types that can be used in various situation.

<table>
<thead>
<tr>
<th>VALID_FOR Definition</th>
<th>Primary Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>ONLINE_LOGFILE, PRIMARY_ROLE</td>
<td>Active</td>
</tr>
<tr>
<td>ONLINE_LOGFILE, STANDBY_ROLE</td>
<td>Inactive</td>
</tr>
<tr>
<td>ONLINE_LOGFILE, ALL_ROLES</td>
<td>Active</td>
</tr>
<tr>
<td>STANDBY_LOGFILE, PRIMARY_ROLE</td>
<td>Error</td>
</tr>
<tr>
<td>STANDBY_LOGFILE, STANDBY_ROLE</td>
<td>Invalid</td>
</tr>
<tr>
<td>STANDBY_LOGFILE ALL_ROLES</td>
<td>Invalid</td>
</tr>
</tbody>
</table>
The following table describes the archival processing defined by the initialization parameters shown in this example for the new STANDBY logical standby database.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>When the STANDBY Database Is Running in the Primary Role</th>
<th>When the STANDBY Database Is Running in the Logical Standby Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOG_ARCHIVE_DEST_1</td>
<td>Directs archival of redo data generated by the primary database from the local online redo log files to the local archived redo log files in the Flash Recovery Area (USE_DB_RECOVERY_FILE_DEST).</td>
<td>Directs archival of redo data generated by the logical standby database from the local online redo log files to the local archived redo log files in the Flash Recovery Area (USE_DB_RECOVERY_FILE_DEST).</td>
</tr>
<tr>
<td>LOG_ARCHIVE_DEST_2</td>
<td>Directs transmission of redo data to the remote logical standby database FGUARD.</td>
<td>Is ignored; LOG_ARCHIVE_DEST_2 is valid only when FGUARD is running in the primary role.</td>
</tr>
<tr>
<td>LOG_ARCHIVE_DEST_3</td>
<td>Is ignored; LOG_ARCHIVE_DEST_3 is valid only when FGUARD is running in the standby role.</td>
<td>Directs archival of redo data received from the primary database to the local archived redo log files in C:\oracle\product\10.2.0\oradata\FGUARD_ARCH_stdby_redo_log.</td>
</tr>
</tbody>
</table>

**NOTE**
The DB_FILE_NAME_CONVERT initialization parameter is not honored once a physical standby database is converted to a logical standby database. This can be a problem, for example, when adding a non-OMF datafile to the primary database and the datafile paths are different between the primary and standby. If necessary, you should register a skip handler and provide SQL Apply with a replacement DDL string to execute by converting the path names of the primary database datafiles to the standby datafile path names. For more information check [HERE](#).

4. Mounting the Stanby Database
On the logical standby database, shutdown the instance and issue the STARTUP MOUNT statement to start and mount the database.

```
SHUTDOWN IMMEDIATE;
STARTUP MOUNT;
```

5. Open the Stanby Database
The new database is logically the same as your primary database, but it is transactionally inconsistent with the primary database, and thus incompatible for recovery operations. To open the new logical standby database, you must open it with the RESETLOGS option by issuing the following statement:

```
ALTER DATABASE OPEN RESETLOGS;
```
Because this is the first time the database is being opened, the database's global name is adjusted automatically to match the new DB_NAME initialization parameter.

```
select * from global_name;
```

```
select name , PROTECTION_LEVEL,DATABASE_ROLE from v$database;
NAME            PROTECTION_LEVEL     DATABASE_ROLE
---------          -------------------            ---------------
TESTDB            MAXIMUM PERFORMANCE       LOGICAL STANDBY
```

```
select instance_name , STATUS , INSTANCE_ROLE from v$instance ;
INSTANCE_NAME   STATUS       INSTANCE_ROLE
----------------   -----------   -----------------
winoradb        OPEN          PRIMARY_INSTANCE
```

As you can see here, the INSTANCE Name is different that the DB Name. The instance Name is used by the OS to identify the logical standby.

6. Start Applying Process
Issue the following statement to begin applying redo data to the logical standby database:

```
ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
```

If we get:

```
ERROR at line 1:
ORA-16239: IMMEDIATE option not available without standby redo logs
```

We got the above error, because we didn't create standby redo log files

Create standby redo logs as shown below.

```
alter database add standby logfile group 4
(''/slot/ems6826/oracle/oradata/sitst02/redo4.dbf'') size 100M;
alter database add standby logfile group 5
(''/slot/ems6826/oracle/oradata/sitst02/redo5.dbf'') size 100M;
alter database add standby logfile group 6
(''/slot/ems6826/oracle/oradata/sitst02/redo6.dbf'') size 100M;
ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
```

**Logical Standby Sanity Check**

Detect Database Role
```
select name, database_role from v$database;
```
Queries for PRIMARY Database
- Show Thread, Archived, Applied

```sql
select DEST_ID, THREAD#, SEQUENCE#, ARCHIVED, APPLIED, COMPLETION_TIME
from v$archived_log where DEST_ID = 2 order by SEQUENCE#;
```

<table>
<thead>
<tr>
<th>DEST_ID</th>
<th>THREAD#</th>
<th>SEQUENCE#</th>
<th>ARC</th>
<th>APP</th>
<th>COMPLETION_TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>1350</td>
<td>YES</td>
<td>YES</td>
<td>17/JAN/11 21:35:13</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1351</td>
<td>YES</td>
<td>YES</td>
<td>17/JAN/11 22:03:06</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1352</td>
<td>YES</td>
<td>YES</td>
<td>17/JAN/11 22:17:30</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1353</td>
<td>YES</td>
<td>NO</td>
<td>17/JAN/11 22:37:10</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1354</td>
<td>YES</td>
<td>NO</td>
<td>17/JAN/11 22:37:11</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1355</td>
<td>YES</td>
<td>NO</td>
<td>17/JAN/11 22:44:21</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1356</td>
<td>YES</td>
<td>NO</td>
<td>18/JAN/11 07:45:09</td>
</tr>
</tbody>
</table>

On STANDBY database

```sql
ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MON-YY HH24:MI:SS';
```

- Verify the archived redo log files were registered.

```sql
SELECT SEQUENCE#, FIRST_TIME, NEXT_TIME, DICT_BEGIN, DICT_END, TIMESTAM FROM DBA_LOGSTDBY_LOG ORDER BY SEQUENCE#;
```

<table>
<thead>
<tr>
<th>SEQUENCE#</th>
<th>FIRST_TIME</th>
<th>NEXT_TIME</th>
<th>DICT BEGIN</th>
<th>DICT END</th>
<th>TIMESTAMP</th>
</tr>
</thead>
</table>

- Show Threads Progress, Determine how much progress was made through the available logs

```sql
SELECT L.SEQUENCE#, L.FIRST_TIME,
    (CASE WHEN L.NEXT_CHANGE# < P.READ_SCN THEN 'YES' WHEN L.FIRST_CHANGE# < P.APPLIED_SCN THEN 'CURRENT' ELSE 'NO' END)
```

APPLIED FROM DBA_LOGSTDBY_LOG L, DBA_LOGSTDBY_PROGRESS P ORDER BY SEQUENCE#;

<table>
<thead>
<tr>
<th>SEQUENCE#</th>
<th>FIRST_TIME</th>
<th>APPLIED</th>
</tr>
</thead>
<tbody>
<tr>
<td>1353</td>
<td>17-JAN-11 22:17:14</td>
<td>CURRENT</td>
</tr>
<tr>
<td>1354</td>
<td>17-JAN-11 22:36:33</td>
<td>NO</td>
</tr>
<tr>
<td>1355</td>
<td>17-JAN-11 22:37:06</td>
<td>NO</td>
</tr>
<tr>
<td>1356</td>
<td>17-JAN-11 22:44:19</td>
<td>NO</td>
</tr>
</tbody>
</table>

- Verify Current SCN to see if all log file information was applied
SELECT APPLIED_SCN, NEWEST_SCN FROM DBA_LOGSTDBY_PROGRESS;

<table>
<thead>
<tr>
<th>APPLIED_SCN</th>
<th>NEWEST_SCN</th>
</tr>
</thead>
<tbody>
<tr>
<td>2998474027</td>
<td>2998474027</td>
</tr>
</tbody>
</table>

- Show File Location that I received and its status
SELECT FILE_NAME, SEQUENCE#, FIRST_CHANGE#, NEXT_CHANGE#, TIMESTAMP, DICT_BEGIN, DICT_END, THREAD# FROM DBA_LOGSTDBY_LOG ORDER BY SEQUENCE#;

<table>
<thead>
<tr>
<th>FILE_NAME</th>
<th>SEQUENCE#</th>
<th>FIRST_CHANGE#</th>
<th>NEXT_CHANGE#</th>
<th>TIMESTAMP</th>
<th>DIC</th>
<th>DIC</th>
<th>THREAD#</th>
</tr>
</thead>
<tbody>
<tr>
<td>C:\ORACLE\PRODUCT\10.2.0\ORADATA\WINORADB_ARCH_STDBY_REDO_LOG\ARCH_001_01353_697041245.LOG</td>
<td>1353</td>
<td>2998447810</td>
<td>2998451537</td>
<td>17-JAN-11 22:37:14</td>
<td>NO</td>
<td>NO</td>
<td>1</td>
</tr>
<tr>
<td>C:\ORACLE\PRODUCT\10.2.0\ORADATA\WINORADB_ARCH_STDBY_REDO_LOG\ARCH_001_01354_697041245.LOG</td>
<td>1354</td>
<td>2998451537</td>
<td>2998451552</td>
<td>17-JAN-11 22:37:14</td>
<td>NO</td>
<td>NO</td>
<td>1</td>
</tr>
<tr>
<td>C:\ORACLE\PRODUCT\10.2.0\ORADATA\WINORADB_ARCH_STDBY_REDO_LOG\ARCH_001_01355_697041245.LOG</td>
<td>1355</td>
<td>2998451552</td>
<td>2998451783</td>
<td>17-JAN-11 22:44:24</td>
<td>NO</td>
<td>NO</td>
<td>1</td>
</tr>
</tbody>
</table>
- Show what the Standby is doing, Inspect the process activity for SQL apply operations

column status format a50
column type format a12
select type, high_scn, status from v$logstdby;
SELECT substr(TYPE,1,10) Type, HIGH_SCN, substr(STATUS,1,55)
FROM V$LOGSTDBY;

<table>
<thead>
<tr>
<th>TYPE</th>
<th>HIGH_SCN</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>COORDINATOR</td>
<td>2998474028</td>
<td>ORA-16116: no work available</td>
</tr>
<tr>
<td>READER</td>
<td>2998474028</td>
<td>ORA-16240: Waiting for logfile (thread# 1, sequence# 1358)</td>
</tr>
<tr>
<td>BUILDER</td>
<td>2998474025</td>
<td>ORA-16116: no work available</td>
</tr>
<tr>
<td>PREPARER</td>
<td>2998474024</td>
<td>ORA-16116: no work available</td>
</tr>
<tr>
<td>ANALYZER</td>
<td>2998474025</td>
<td>ORA-16117: processing</td>
</tr>
<tr>
<td>APPLIER</td>
<td>2998474005</td>
<td>ORA-16116: no work available</td>
</tr>
<tr>
<td>APPLIER</td>
<td>2998474025</td>
<td>ORA-16116: no work available</td>
</tr>
<tr>
<td>APPLIER</td>
<td>2998473973</td>
<td>ORA-16116: no work available</td>
</tr>
<tr>
<td>APPLIER</td>
<td>2998473978</td>
<td>ORA-16116: no work available</td>
</tr>
</tbody>
</table>

- Show problematic Rows or Events

SELECT EVENT_TIME, STATUS, EVENT FROM DBA_LOGSTDBY_EVENTS ORDER BY EVENT_TIME, COMMIT_SCN;

- Check Coordinator Status

SELECT substr(NAME,1,20) Name, substr(VALUE,1,30) value
FROM V$LOGSTDBY_STATS
WHERE NAME LIKE 'coordinator%' or NAME LIKE 'transactions%';

<table>
<thead>
<tr>
<th>NAME</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>coordinator state</td>
<td>IDLE</td>
</tr>
<tr>
<td>transactions ready</td>
<td>1165</td>
</tr>
<tr>
<td>transactions applied</td>
<td>1165</td>
</tr>
<tr>
<td>coordinator uptime</td>
<td>285</td>
</tr>
</tbody>
</table>
Troubleshooting a Logical Standby

Setting up a Skip Handler for a DDL Statement
The DB_FILE_NAME_CONVERT initialization parameter is not honored once a physical standby database is converted to a logical standby database. This can be a problem, for example, when adding a non-OMF datafile to the primary database and the datafile paths are different between the primary and standby. This section describes the steps necessary to register a skip handler and provide SQL Apply with a replacement DDL string to execute by converting the path names of the primary database datafiles to the standby datafile path names.

This may or may not be a problem for everyone. For example, if you are using Oracle Managed Files (OMF), SQL Apply will successfully execute DDL statements generated from the primary to CREATE and ALTER tablespaces and their associated system generated path name on the logical standby.

On PRIMARY Standby using Oracle Managed Files (OMF)

show parameter db_create_file_dest

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>db_create_file_dest</td>
<td>string</td>
<td>/u02/oradata</td>
</tr>
</tbody>
</table>

create tablespace data2 datafile size 5m;

On LOGICAL Standby using Oracle Managed Files (OMF)

show parameter db_create_file_dest

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>db_create_file_dest</td>
<td>string</td>
<td>/u02/oradata</td>
</tr>
</tbody>
</table>

----------------------------- alert.log -----------------------------
Wed Jan 12 18:45:28 EST 2011
Completed: create tablespace data2 datafile size 5m

----------------------------------------
select tablespace_name, file_name
from dba_data_files
where tablespace_name = 'DATA2';

<table>
<thead>
<tr>
<th>TABLESPACE_NAME</th>
<th>FILE_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA2</td>
<td>/u02/oradata/FGUARD/datafile/o1mf_data2_6lwh8q9d_.dbf</td>
</tr>
</tbody>
</table>

On PRIMARY Standby using Oracle Managed Files (OMF)
alter tablespace data2 add datafile '/u05/oradata/MODESTO/data02.dbf' size 5m;

On LOGICAL Standby WITHOUT Oracle Managed Files (OMF)
If on the other hand, you attempt to specify a physical path name in the
CREATE/ALTER tablespace statement that does not exist on the logical standby,
SQL Apply will not succeed in processing the statement and will fail. Whenever SQL
Apply encounters an error while applying a SQL statement, it will stop and provide
the DBA with an opportunity to correct the statement and restart SQL Apply.

----------------------- alert.log -----------------------
Wed Jan 12 19:59:36 EST 2011
alter tablespace data2 add datafile '/u05/oradata/MODESTO/data02.dbf' size 5m
Wed Jan 12 19:59:36 EST 2011
ORA-1119 signalled during: alter tablespace data2 add datafile
'/u05/oradata/MODESTO/data02.dbf' size 5m...
LOGSTDBY status: ORA-01119: error in creating database file
'/u05/oradata/MODESTO/data02.dbf'
ORA-27040: file create error, unable to create file
Linux Error: 2: No such file or directory
LOGSTDBY Apply process P004 pid=31 OS id=28497 stopped
Wed Jan 12 19:59:36 EST 2011
Errors in file /u01/app/oracle/admin/turlock/bdump/turlock_lsp0_28465.trc:
ORA-12801: error signaled in parallel query server P004
ORA-01119: error in creating database file '/u05/oradata/MODESTO/data02.dbf'
ORA-27040: file create error, unable to create file
Linux Error: 2: No such file or directory
LOGSTDBY Apply process P004 pid=31 OS id=28497 stopped
LOGSTDBY Analyzer process P003 pid=30 OS id=28495 stopped
LOGSTDBY Apply process P005 pid=32 OS id=28499 stopped
LOGSTDBY Apply process P006 pid=33 OS id=28501 stopped
LOGSTDBY Apply process P007 pid=34 OS id=28503 stopped
LOGSTDBY Apply process P008 pid=35 OS id=28505 stopped
select event_timestamp, event, status from dba_logstdby_events;

<table>
<thead>
<tr>
<th>EVENT_TIMESTAMP</th>
<th>EVENT</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>12-JAN-11 07.59.36.134349 PM</td>
<td>alter tablespace data2 add datafile '/u05/oradata/MODESTO/data02.dbf' size 5m</td>
<td>ORA-01119: error in creating database file '/u05/oradata/MODESTO/data02.dbf' size 5m ESTO/data02.dbf'</td>
</tr>
</tbody>
</table>

Fixing it
1. Disable the database guard for this session so we can modify the logical standby.
   alter session disable guard;

2. Issue a compensating transaction or statement on the logical standby. For this example, issue the following command with the correct path:
   alter tablespace data2 add datafile '/u05/oradata/FGUARD/data02.dbf' size 5m;

3. Re-enable the database guard for this session.
   alter session enable guard;

4. Restart logical apply with a clause that will cause the failed transaction to be automatically skipped.
   alter database start logical standby apply immediate skip failed transaction;

5. Verify results.
   select tablespace_name, file_name from dba_data_files
   where tablespace_name = 'DATA2';
TABLESPACE_NAME      FILE_NAME
--------------------- ---------------------------------------------
DATA2                /u02/oradata/FGUARD/datafile/o1_mf_data2_61wh8g9d_.dbf
DATA2                /u05/oradata/FGUARD/data02.dbf

NOTE
It is possible to avoid errors of this nature on the logical standby database by registering a skip handler and provide SQL Apply with a replacement DDL string to execute by converting the path names of the primary database datafiles to the standby datafile path names. The steps to perform this are presented below. The actions below should be run on the logical standby database.

1. First, create the PL/SQL 'skip procedure' to handle tablespace DDL transactions.

```sql
CREATE OR REPLACE PROCEDURE sys.handle_tbs_ddl ( 
    old_stmt IN VARCHAR2,
    stmt_typ IN VARCHAR2,
    schema IN VARCHAR2,
    name IN VARCHAR2,
    xidusn IN NUMBER,
    xidslt IN NUMBER,
    xidsqn IN NUMBER,
    action OUT NUMBER,
    new_stmt OUT VARCHAR2
) AS
BEGIN
    -- All primary file specification that contain a directory
    -- '/u05/oradata/FGUARD' should be changed to the
    -- '/u01/another_location/oradata/FGUARD' directory
    -- specification.

    new_stmt := REPLACE(old_stmt, '/u05/oradata/FGUARD', '/u01/another_location/oradata/FGUARD');

    Action := DBMS_LOGSTDBY.skip_action_replace;

    EXCEPTION
        WHEN OTHERS THEN
            Action := DBMS_LOGSTDBY.skip_action_error;
            new_stmt := NULL;

END handle_tbs_ddl;
/
```
2. Stop SQL Apply.
alter database stop logical standby apply;

3. Register the skip procedure with SQL Apply.
execute dbms_logstdby.skip(stmt => 'TABLESPACE', proc_name => 'sys.handle_tbs_ddl');

4. Start SQL Apply.
alter database start logical standby apply immediate;

5. Perform a test.
On PRIMARY Standby
alter tablespace data2 add datafile
'/u05/oradata/FGUARD/data03.dbf' size 5m;

ON Logical Standby
-----------------------
alert.log -----------------------
Wed Jan 12 20:51:58 EST 2011
LOGSTDBY status: ORA-16110: user procedure processing of logical standby apply DDL
LOGSTDBY status: ORA-16202: Skip procedure requested to replace statement
Wed Jan 12 20:51:58 EST 2011
alter tablespace data2 add datafile
'/u01/another_location/oradata/FGUARD/data03.dbf' size 5m
Completed: alter tablespace data2 add datafile
'/u01/another_location/oradata/FGUARD/data03.dbf' size 5m

---------------------------------------------------------
select tablespace_name, file_name
   from dba_data_files
      where tablespace_name = 'DATA2';

TABLESPACE_NAME  FILE_NAME
------------------------
---------
DATA2  /u01/another_location/oradata/FGUARD/data03.dbf
**Logical Standby Database Activation (Role Transition)**
A database can operate in one of two mutually exclusive modes in an Oracle Data Guard configuration: primary or standby. Whenever the role is changed between the primary and standby, this is referred to as a role transition. Role transition plays an important part in Data Guard by providing an interface that allows DBA's to activate a standby database to take over as the primary production database or vice versa. There are two types of role transitions supported in Oracle: switchover and failover.

**Switchover**
1- Issue the following statement on the production database to enable receipt of redo from the current standby database:
   `ALTER DATABASE PREPARE TO SWITCHOVER TO LOGICAL STANDBY;`

2- On the current logical standby database, build the LogMiner dictionary and transmit this dictionary to the current production database:
   `ALTER DATABASE PREPARE TO SWITCHOVER TO PRIMARY;`

   Depending on the work to be done and the size of the database, the prepare statement may take some time to complete.

3- Verify the LogMiner Multiversioned Data Dictionary was received by the production database by querying the SWITCHOVER_STATUS column of the V$DATABASE fixed view on the production database. Initially, the SWITCHOVER_STATUS column shows PREPARING DICTIONARY while the LogMiner Multiversioned Data Dictionary is being recorded in theredo stream. Once this has completed successfully, the column shows PREPARING SWITCHOVER. When the query returns the TO LOGICAL STANDBY value, you can proceed to the next step.

4- When the SWITCHOVER_STATUS column of the V$DATABASE view returns TO LOGICAL STANDBY, convert the production database to a standby by issuing:
   `ALTER DATABASE COMMIT TO SWITCHOVER TO LOGICAL STANDBY with SESSION SHUTDOWN;`

5- Issue the following statement on the old standby database:
   `ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY;`

**Failover**
Copy and register any missing archived redo logs from PROD. Execute the following query on Standby:

```
COLUMN FILE_NAME FORMAT a55;
SELECT THREAD#, SEQUENCE#, FILE_NAME
FROM DBA_LOGSTDBY_LOG L
WHERE NEXT_CHANGE# NOT IN (SELECT FIRST_CHANGE#
                           FROM DBA_LOGSTDBY_LOG
                           WHERE L.THREAD# = THREAD#)
ORDER BY THREAD#, SEQUENCE#;
```

Register any log:

```
ALTER DATABASE REGISTER LOGICAL LOGFILE '/disk1/oracle/dbs/log-1292880008_7.arc';
```

Copy and register the online redo logs from the primary database (if possible). You can safely ignore the error ORA-01289: cannot add duplicate logfile

```
ALTER DATABASE REGISTER LOGICAL LOGFILE '/disk1/oracle/dbs/online_log1.log';
```

Turn off the apply delay interval.

```
ALTER DATABASE STOP LOGICAL STANDBY APPLY;
EXECUTE DBMS_LOGSTDBY.APPLY_UNSET('APPLY_DELAY');
ALTER DATABASE START LOGICAL STANDBY APPLY;
```

Initiate the failover by issuing the following on the target standby database:

```
ALTER DATABASE ACTIVATE LOGICAL STANDBY DATABASE FINISH APPLY;
```

This statement stops the RFS process, applies any remaining redo data, stops SQL Apply, and activates the logical standby database in the production role.

Note: To avoid waiting for the redo in the standby redo log file to be applied prior to performing the failover, omit the FINISH APPLY clause on the statement. Although omitting the FINISH APPLY clause will accelerate failover, omitting the clause will cause the loss of any unapplied redo data in the standby redo log. To gauge the amount of redo that will be lost, query the V$LOGSTDBY_PROGRESS view. The LATEST_SCN column value indicates the last SCN received from the production database, and the APPLIED_SCN column value indicates the last SCN applied to the standby database. All SCNs between these two values will be lost.
Monitoring Physical Data Guard (Detect Gap)

Perform a query of the V$ARCHIVE_GAP view from the physical standby database:
```
select * from v$archive_gap;
```

If you get results from this query, it means there is a gap. You can easily detect what is the problem by checking the alert.log file on your Primary DB.

Potential data loss window for a physical standby database (Archived logs not applied on a physical standby database)
-- On the standby, Get the sequence number of the last applied archive log.
```
select max(sequence#) Last_applied_arch_log from v$archived_log where applied='YES';
```

-- On the standby, Get the sequence number of the last complete archive log on the standby.
-- This is the last log the standby can apply without receiving additional archive logs from the primary.
```
SELECT min(sequence#) Last_archive_log_received FROM v$archived_log
WHERE (sequence#+1) NOT IN (SELECT sequence# FROM v$archived_log)
AND sequence# > &Last_applied_arch_log;
```

-- Connect to the primary database and obtain the sequence number of the current online log:
```
select sequence# from v$log where status='CURRENT';
```

-- The difference between 2nd query and 1st query should be 0
-- The difference between 3rd query and 1st query is the number of archive logs that the standby
-- database would not be able to recover if the primary host become unavailable

**Some GOOD queries to detect GAPS**
```
select process,status,client_process,sequence#,block#,active_agents,known_agents from v$managed_standby;
```
Run this at Primary
set pages 1000
set lines 120
column DEST_NAME format a20
column DESTINATION format a35
column ARCHIVER format a10
column TARGET format a15
column status format a10
column error format a15
select DEST_ID,DEST_NAME,DESTINATION,TARGET,STATUS,ERROR
from v$archive_dest
where DESTINATION is NOT NULL
/
select ads.dest_id,max(sequence#) "Current Sequence",
max(log_sequence) "Last Archived"
from v$archived_log al, v$archive_dest ad,
v$archive_dest_status ads
where ad.dest_id=al.dest_id and al.dest_id=ads.dest_id
group by ads.dest_id;

Run this at Standby

select max(al.sequence#) "Last Seq Recieved" from
v$archived_log al
/
select max(al.sequence#) "Last Seq Applied" from
v$archived_log al
where applied = 'YES'
/
select process, status, sequence# from v$managed_standby
/

select * from v$archive_gap;

<table>
<thead>
<tr>
<th>THREAD#</th>
<th>LOW_SEQUENCE#</th>
<th>HIGH_SEQUENCE#</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5134</td>
<td>5222</td>
</tr>
</tbody>
</table>

**Another Method**

Use the following SQL on the standby database (the database must be mounted).

```
SELECT high.thread#, "LowGap#", "HighGap#"
FROM
  ( SELECT thread#, MIN(sequence#)-1 "HighGap#"
  FROM
    ( SELECT a.thread#, a.sequence#
      FROM
      ( SELECT *
        FROM v$archived_log
      ) a,
        ( SELECT thread#, MAX(sequence#)gap1
          FROM v$log_history
          GROUP BY thread#
        ) b
      WHERE a.thread# = b.thread#
      AND a.sequence# > gap1
      )
    GROUP BY thread#
  ) high,
( SELECT thread#, MIN(sequence#) "LowGap#"
FROM
  ( SELECT thread#, sequence#
      FROM v$log_history, v$datafile
      WHERE checkpoint_change# <= next_change#
      AND checkpoint_change# >= first_change#
      )
    GROUP BY thread#
  ) low,
( SELECT thread#, MIN(sequence#) "HighGap#"
FROM
  ( SELECT thread#, MIN(sequence#)
      FROM v$log_history, v$datafile
      WHERE checkpoint_change# <= next_change#
      AND checkpoint_change# >= first_change#
      )
    GROUP BY thread#
) high,
( SELECT thread#, MIN(sequence#)
      FROM v$log_history, v$datafile
      WHERE checkpoint_change# <= next_change#
      AND checkpoint_change# >= first_change#
      ) low,
( SELECT thread#, MIN(sequence#)
      FROM v$log_history, v$datafile
      WHERE checkpoint_change# <= next_change#
      AND checkpoint_change# >= first_change#
      ) high,
( SELECT thread#, MIN(sequence#)
```
Suspend Physical Standby Recovery

To stop managed standby recovery:

-- Cancel protected mode on primary
CONNECT sys/password@primary1 AS SYSDBA
ALTER DATABASE SET STANDBY DATABASE UNPROTECTED;

-- Cancel recovery if necessary
CONNECT sys/password@standby1 AS SYSDBA
RECOVER MANAGED STANDBY DATABASE CANCEL;
ALTER DATABASE OPEN READ ONLY;

The database can subsequently be switched back to recovery mode as follows:

-- Startup managed recovery
CONNECT sys/password@standby1 AS SYSDBA
SHUTDOWN IMMEDIATE
STARTUP NOMOUNT;
ALTER DATABASE MOUNT STANDBY DATABASE;
RECOVER MANAGED STANDBY DATABASE DISCONNECT FROM SESSION;

-- Protect primary database
CONNECT sys/password@primary1 AS SYSDBA
ALTER DATABASE SET STANDBY DATABASE PROTECTED;

Activating A Physical Standby Database

Procedures to Open the DR Database in Query Mode
Once there, cancel recovery with (you could get an error message that you can ignore):
RECOVER MANAGED STANDBY DATABASE CANCEL;

Open the DB in Read Mode with:
ALTER DATABASE OPEN READ ONLY;
alter tablespace TEMP add tempfile
'/oracle/DBA_SCRIPTS/temp.dbf' size 1000m reuse;
Procedures to Put the Database back in DR Mode
Close the DB:
SHUTDOWN IMMEDIATE;

Mount the DB with:
startup nomount;
alter database mount standby database;

Open the DB in Recovery mode with:
recover standby database;
alter database recover managed standby database disconnect from session;

Procedures to Activate a DR Database as a PROD database
If the primary database is not available the standby database can be activated as a primary database using the following statements:

--Startup the DB in Mount Mode (if that is not already on place)
STARTUP NOMOUNT
ALTER DATABASE MOUNT STANDBY DATABASE;

--Try to receive and apply the latest Arch Logs from PROD
RECOVER MANAGED STANDBY DATABASE;
or
RECOVER STANDBY DATABASE;

-- Cancel recovery if necessary
RECOVER MANAGED STANDBY DATABASE CANCEL;
or
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE FINISH;

shutdown immediate;
startup nomount;
alter database mount standby database;

--You may need to rename log files like:
ALTER DATABASE RENAME FILE '/u01/app/oracle/oradata/CCOM/redo1.log' to
'/opt/app/oracle/oradata/CCOM/redo01.log';
ALTER DATABASE RENAME FILE '/u01/app/oracle/oradata/CCOM/redo2.log' to
'/opt/app/oracle/oradata/CCOM/redo02.log';

ALTER DATABASE ACTIVATE STANDBY DATABASE;
SHUTDOWN IMMEDIATE
STARTUP

Since the standby database is now the primary database it should be backed up immediately. The previous primary database can then be configured as a standby.

Create or Add a tempfile to the temporary tablespace: Ex:
create temporary tablespace TEMP2 TEMPFILE
'oracle/oradata/V901/temp2.dbf' size 100M;
or
alter tablespace TEMP add tempfile
'oracle/app/product/9.2/admin/oradata/temp01.dbf' size 100M;

**PHYSICAL STANDBY SWITCHOVER STEPS**

Unlike failover, a switchover operation is a planned operation. All the archive logs required bringing the standby to the primary’s point in time need to be available. The primary database’s online redo logs also must be available and intact. During switchover operation, primary and standby databases switch roles.

NOTE = Standby must be mounted before starting the switchover!!!!

**QUICK GUIDE**

_The Current PROD Site, that will become Standby_
select database_role, switchover_status from v$database; -- Here we would like to see "PRIMARY TO STANDBY"
alter database commit to switchover to physical standby with session shutdown;
shutdown immediate
startup nomount
alter database mount standby database;
alter system set log_archive_dest_state_2=defer;
ALTER SYSTEM SET fal_client='THIS_STANDBY' SCOPE=BOTH --This should be DR DB (Denver)
ALTER SYSTEM SET fal_server='PROD' SCOPE=BOTH; --This should be PROD (Falcon)

**** At this point we have 2 standby Databases ******

_The Current STDBY Site, that will become primary_
select database_role, switchover_status from v$database; -- Here we would like to
see "PHYSICAL STANDBY  TO PRIMARY"
alter database commit to switchover to primary;
shutdown immediate
startup
alter system set fal_client=NULL scope=both;
alter system set fal_server=NULL scope=both;
alter system set log_archive_dest_2='SERVICE=STANDBY reopen=60' scope=both;
alter system set log_archive_dest_state_2=enable scope=both;

OLD PRIMARY SITE
recover managed standby database disconnect

In this FULL Example, the standby database (STDBY) becomes the new primary, and the primary (PROD) becomes the new standby database.
The following are steps for switchover operation:

STEP 1: SWITCHOVER PREPARATION FOR FORMAL PRIMARY DATABASE
  •   End all activities on the primary and standby database
  •   Check primary database switchover status
SQL>
select database_role, switchover_status from v$database;

<table>
<thead>
<tr>
<th>DATABASE_ROLE</th>
<th>SWITTOVER_STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRIMARY</td>
<td>TO STANDBY</td>
</tr>
</tbody>
</table>

The SWITTOVER_STATUS column of v$database can have the following values:
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.
SWITTOVER PENDING - This is a standby database and the primary database switchover request has been received but not processed.
SWITTOVER 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, with no active sessions, that is allowed to switch over to a primary database.
TO STANDBY - This is a primary database, with no active sessions, that is allowed to switch over to a standby database.

RECOVERY NEEDED - This is a standby database that has not received the switchover request.

During normal operations it is acceptable to see the following values for SWITCHOVER_STATUS on the primary to be SESSIONS ACTIVE or TO STANDBY.
During normal operations on the standby it is acceptable to see the values of NOT ALLOWED or SESSIONS ACTIVE.

If SWITCHOVER_STATUS returns SESSIONS ACTIVE then you should either disconnect all sessions manually or you can use the following statement to close those sessions:
```
alter database commit to switchover to standby with session shutdown;
```

- Initiate the switchover operation on the primary database. Convert Primary DB to standby
```
SQL> alter system switch logfile;
SQL> alter database commit to switchover to physical standby;
```
or if you have current sessions connected, you can perform:
```
SQL> alter database commit to switchover to physical standby with session shutdown;
```

STEP 2: SHUTDOWN THE PRIMARY DATABASE AND BRING IT UP AS THE NEW STANDBY DATABASE

- Shutdown the primary database normally
```
SQL> shutdown immediate;
```

- Modify the former primary database’s initialization file
  - Add the following two parameters on the Primary init.ora File. These two parameters can also be set on the primary database ahead of time for future switchover operation.
  - fal_client = “PROD_01”
  - fal_server = “STDBY”
  or
  ```
  ALTER SYSTEM SET fal_client='PROD' SCOPE=BOTH;  #This DB used on tnsnames.ora
  ```
ALTER SYSTEM SET fal_server='STDBY' SCOPE=BOTH;  #The new PROD DB used on tnsnames.ora
- Remove the following parameters from the Primary init.ora File:
  log_archive_dest_2 and log_archive_dest_state_2. Or, just defer if you like
  ALTER SYSTEM SET log_archive_dest_2=''; SCOPE=BOTH;

  - Bring the former primary database in mount mode
    SQL> startup nomount;
    SQL> alter database mount standby database;

**STEP 3: SWITCHOVER PREPARATION FOR THE FORMER STANDBY DATABASE**

  - At this time, we have 2 standby Databases, now we will prepare the original standby and convert it to primary. Check standby database switchover status, if we see the "SESSIONS ACTIVE", we need to act as we did it before.
    SQL>
    select database_role, switchover_status from v$database;
    DATABASE_ROLE                      SWITCHOVER_STATUS
    -------------------------            ------------------
    PHYSICAL STANDBY TO PRIMARY

  - Initiate the switchover operation on the standby database
    SQL> alter database commit to switchover to primary;
    or
    SQL> alter database commit to switchover to primary with session shutdown;

**STEP 4: SHUTDOWN THE STANDBY DATABASE AND BRING IT UP AS THE NEW PRIMARY DATABASE**

  - Shutdown the standby database
    SQL> shutdown immediate;

  - Bring up the former standby database as the new primary database
    SQL> startup;

  - Modify the former standby database’s initialization file
    - fal_client = “STDBY”
    - fal_server = “PROD”
    - Add parameters log_archive_dest_2 and log_archive_dest_state_2
alter system set log_archive_dest_2='SERVICE=PROD reopen=60' scope=both;
alter system set log_archive_dest_state_2=enable scope=both;
ALTER SYSTEM SET fal_client='STBY' SCOPE=BOTH;  #This box when is in Standby
ALTER SYSTEM SET fal_server='PROD' SCOPE=BOTH;  #The "original" PROD box

STEP 5: ADD TEMP TABLESPACE
• Issue the following command to add TEMP tablespace
ALTER TABLESPACE TEMP ADD TEMPFILE 
'/u02/oradata/prod/temp01.dbf' SIZE 3072M reuse
AUTOEXTEND OFF;

STEP 6: PUT THE NEW STANDBY (OLD PROD DATABASE) IN MANAGED RECOVERY MODE
• Issue the following command on the new standby database.
SQL> alter database recover managed standby database disconnect from session;

STEP 7: SWITCH THE LOG FILES A COUPLE OF TIMES ON THE NEW PRIMARY DB
• Issue the following commands:
SQL> alter system switch logfile;
SQL> alter system switch logfile;

STEP 8: CHANGE TNS ENTRY FOR THE NEW PRIMARY DATABASE
• Change the TNS entry on all application hosts to point to the new primary Prod =
(description =
(address = (protocol = tcp) (host = server_02) (port = 1522)
(connect_data = (sid = stdby)))
)

Here is a nice Video about the steps:
http://www.youtube.com/watch?v=lcSz9PoqXu0&feature=player_embedded

PHYSICAL STANDBY FAILOVER STEPS
Failover is only performed as a result of an unplanned outage of the primary database. During a failover, the standby database (**prod_02**) becomes the new primary database. It is possible to have data loss.

The old primary (**prod_01**) has to be discarded and can not be used as the new standby database. You need to create a new standby database by backing up the new primary and restore it on host **server_01**. The time to create a new standby database exposes the risk of having no standby database for protection.

After failover operation, you need to modify TNS entry for ‘**prod**’ to point to the new instance and host name.

**Steps**

1- Initiate the failover by issuing the following on the target standby database:

```sql
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE FINISH FORCE;
```

Note: Include the **FORCE** keyword to ensure that the RFS processes on the standby database will fail over without waiting for the network connections to time out through normal TCP timeout processing before shutting down.

2- Convert the physical standby database to the production role:

```sql
ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY;
```

3- If the standby database was never opened read-only since the last time it was started, then open the new production database by issuing the following statement:

```sql
ALTER DATABASE OPEN;
```

If the physical standby database has been opened in read-only mode since the last time it was started, shut down the target standby database and restart it:

```sql
SHUTDOWN IMMEDIATE;
STARTUP;
```

Note: In rare circumstances, administrators may wish to avoid waiting for the standby database to complete applying redo in the current standby redo log file before performing the failover. (note: use of Data Guard real-time apply will avoid this delay by keeping apply up to date on the standby database).

If so desired, administrators may issue the **ALTER DATABASE ACTIVATE STANDBY DATABASE** statement to perform an immediate failover. This statement converts the standby database to the production database, creates a new resetlogs branch, and opens the database.

Create or Add a tempfile to the temporary tablespace: Ex:

```sql
create temporary tablespace TEMP2 TEMPFILE '/oracle/oradata/V901/temp2.dbf' size 100M;
```

or

```sql
alter tablespace TEMP add tempfile
```
IMPLEMENTATION TIPS

Here are several tips for implementing Data Guard:

TIP #1: PRIMARY ONLINE REDO LOGS
The number of redo groups and the size of redo logs are two key factors in configuring online redo logs. In general, you try to create the fewest groups possible without hampering the log writer process’s ability to write redo log information. In a Data Guard environment, LGWR process may take longer to write to the remote standby sites, you may need to add additional groups to guarantee that a recycled group is always available to the log writer process. Otherwise, you may receive incomplete logs on the standby sites. The size of redo log is determined by the amount of transaction needed to be applied to a standby database during database failover operation. A small size of redo will minimize the standby database lag time; however, it may cause more frequent log switches and require more redo groups for log switches to occur smoothly. On the other hand, large size redo logs may require few groups and less log switches, but it may increase standby database lag time and potential for more data loss. The best 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.

For example, the following message from the alert log may indicate a need for more log groups.

ORA-00394: online log reused while attempting to archive it

TIP #2: STANDBY ONLINE REDO LOGS VS. STANDBY ARCHIVED REDO LOGS
Online redo logs transferred from the primary database are stored as either standby redo logs or archived redo logs. Which redo log reception option should we choose? Here is the comparison chart:

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Standby Online Redo Logs</th>
<th>Standby Archived Redo Logs</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Pre-allocated files</td>
<td>- No extra ARCH process</td>
<td></td>
</tr>
<tr>
<td>- Can place on raw devices</td>
<td>- Reduce lag time</td>
<td></td>
</tr>
<tr>
<td>- Can be duplexed for more</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
TIP #3: ENFORCE LOGGING
It is recommended that you set the FORCE LOGGING clause to force redo log to be generated for individual database objects set to NOLOGGING. This is required for a no data loss strategy. Here is the SQL command to set FORCE LOGGING:

```
SQL> select force_logging from v$database;
FORCE_LOGGING
---------------
NO

SQL> alter database force logging;
```

TIP #4: RMAN BACKUP
A failover operation reset logs for the new primary. If you use RMAN to backup your database, you need to create a new incarnation of the target database. Otherwise, your RMAN backup will fail.

```
RMAN> reset database;
```

TIP #5: DISABLE LOG TRANSPORT SERVICES WHEN STANDBY DATABASE IS DOWN
When a standby database or host is down for maintenance, it is advisable to temporarily disable the log transport services for that site. Especially during a heavily transaction period, one behavior observed in Oracle is that when one of the standby database is down for maintenance, it can temporarily freeze the primary database even the data protection mode is set to rapid mode. To avoid such problem, you can issue this command on the primary database before bring down the standby database:

```
SQL> alter system set log_archive_dest_state_2 = defer;
```

When the standby database is up again, issue:
```
SQL> alter system set log_archive_dest_state_2 = enable;
```

TIP #6: STANDBY DATABASE UPGRADE
Steps to upgrade standby database to newer database version:
Step 1: Shutdown both primary and standby databases
Step 2: Install Oracle software on both primary and standby hosts
Step 3: Upgrade the primary database
Step 4: Rebuild standby database from the upgraded primary
**TIP #7: DATA GUARD BROKER**
Starting on Oracle9i R2 broker has made great improvements. The new configuration now support up to nine standby sites (including logical standby database). Both Data Guard Manager and CLI support switchover and failover operations.

**TIP #8: USING ‘DELAY’ OPTION TO PROTECT LOGICAL/PHYSICAL CORRUPTIONS**
You may utilize the delay option (if you have multiple standby sites) to prevent physical/logical corruption of your primary. For instance, your standby #1 may not have ‘Delay’ on to be your disaster recovery standby database. However, you may opt to implement a delay of minutes or hours on your standby #2 to allow recover from a possible physical or logical corruption on your primary database.

```
SQL> alter database recover managed standby database delay 5
disconnect;
```

**TIP #9: ALWAYS MONITOR LOG APPLY SERVICES AND CHECK ALERT. LOG FILE FOR ERRORS.**
If you are not using Data Guard broker, here is a script to help you to monitor your standby database recover process:

```
$ cat ckalertlog.sh
################################################################
####
## ckalertlog.sh
##
################################################################
####
#!/bin/ksh
export EDITOR=vi
export ORACLE_BASE=/u01/app/oracle
export ORACLE_HOME=$ORACLE_BASE/product/10.2.0
export ORACLE_HOME LD_LIBRARY_PATH=$ORACLE_HOME/lib
export TNS_ADMIN=/var/opt/oracle
export ORATAB=/var/opt/oracle/oratab
PATH=$PATH:$ORACLE_HOME:$ORACLE_HOME/bin:/usr/ccs/bin:/bin:/usr/sbin:
/sbin:/usr/openwin/bin:/opt/bin:.; export PATH
DBALIST="primary.dba@company.com,another.dba@company.com"; export

for SID in `cat $ORACLE_HOME/sidlist`
do
```
cd $ORACLE_BASE/admin/$SID/bdump
if [ -f alert_${SID}.log ]
then
    mv alert_${SID}.log alert_work.log
touch alert_${SID}.log
cat alert_work.log >> alert_${SID}.hist
grep ORA- alert_work.log > alert.err
fi
if [ `cat alert.err|wc -l` -gt 0 ]
then
    mailx -s "${SID} ORACLE ALERT ERRORS" $DBALIST < alert.err
fi
rm -f alert.err
rm -f alert_work.log
done

Place the script in a crontab:
#########################################################################
# Check Standby log file
#########################################################################
9,19,29,39,49,59 7-17 * * 1-5 /dba/scripts/ckalertlog.sh

Applying Patches with Standby
Beginning with Oracle Database 11.2, Oracle has introduced Standby-First Patch Apply to enable a physical standby to use Redo Apply to support different software patch levels between a primary database and its physical standby database for the purpose of applying and validating Oracle patches in rolling fashion. Patches eligible for Standby-First patching include:
• Patch Set Update (PSU)
• Critical Patch Update (CPU)
• Patch Set Exception (PSE)
• Oracle Database bundled patch
• Oracle Exadata Database Machine bundled patch
• Oracle Exadata Storage Server Software patch
Standby-First Patch Apply is supported for certified software patches for Oracle Database Enterprise Edition Release 2 (11.2) release 11.2.0.1 and later.
Refer to My Oracle Support Note 1265700.1 for more information and the README for each patch to determine if a target patch is certified as being a Standby-First Patch.

For other type of patches or older versions of Oracle
   - For Physical Standby use note ML Note 187242.1
- For Logical Standby use note ML Note 210989.1

Procedure to Apply a Patch Set with Physical Standby Database in Place (ML Note 187242.1)

NOTE: If you are using the Data Guard Broker, you should either disable the Data Guard Broker Configuration
DGMGRL> disable configuration;
or stop the Data Guard Broker (set db_broker_start=false) during the Upgrade.
You can enable the Data Guard Broker Configuration
DGMGRL> enable configuration;
or restart the Data Guard Broker (set db_broker_start=true) again once the Upgrade completed successfully.

1. Log in to the oracle account on both the PRIMARY and STANDBY hosts and make sure the environment is set to the correct ORACLE_HOME and ORACLE_SID.

2. On both the PRIMARY and STANDBY host uncompress and untar the downloaded patch set / interim patch file into a new directory.

3. Shut down the existing Oracle Server instance on the PRIMARY host with immediate priority.
Stop all listeners, agents and other processes running against the ORACLE_HOME. If using Real Application Clusters perform this step on all nodes.

   shutdown immediate
   % agentctl stop
   % lsnrctl stop


   recover managed standby database cancel;

5. Shutdown the STANDBY instance on the STANDBY host. Stop all listeners, agents and other processes running against the ORACLE_HOME.
If using Real Application Clusters perform this step on all nodes.

   shutdown immediate
   % agentctl stop
   % lsnrctl stop

6. Run the Installer and install the patchset on both PRIMARY and STANDBY host.
% ./runInstaller
If this is an interim patch, run opatch per the patch README.
If using Real Application Clusters, be sure the install has propagated to the other
nodes if using private ORACLE_HOMEs.
Please see the Patch readme for specific instructions.

7. Once the patchset/patch has been installed on on all hosts/nodes startup the
STANDBY listener on STANDBY host.
   % lsnrctl start

8. Startup nomount the STANDBY database.
   % sqlplus "/ as sysdba"
   startup nomount

9. Mount the STANDBY database.
   alter database mount standby database;

10. Place the STANDBY database in managed recovery mode.
    recover managed standby database nodelay disconnect;

11. Startup the PRIMARY instance on the primary host.
    % sqlplus "/ as sysdba"
    startup migrate

12. Ensure that remote archiving to the STANDBY database is functioning correctly
    by switching logfiles on the PRIMARY and verifying that v$archive_dest.status is
    valid.
    If you are not performing remote archiving make note of the current archive log
    sequence.
    alter system archive log current;
    select dest_id, status from v$archive_dest;

13. On the PRIMARY instance run the following script:
    @?/rdbms/admin/catpatch.sql
    For the interim patch, run any scripts as outlined in the README.

14. Once the catpatch.sql script / patch SQL scripts completes make note of the
    current log sequence and issue the following command:
    alter system archive log current;

15. Verify the STANDBY database has been recovered to the log sequence from step
    12.
    select max(sequence#) from v$log_history;
16. On the PRIMARY instance run the following command:
alter system disable restricted session;

17. Complete the remainder of the "Post Install Actions" from the Patch Set readme on the primary host.
Please note that it is not necessary to shudown the STANDBY in conjuction with the PRIMARY during the "Post Install Actions".

18. Once all "Post Install Actions" have been completed verify the STANDBY database has been recovered to the last archive log produced by the PRIMARY . On the PRIMARY :
select max(sequence#) from v$archived_log;

On the STANDBY :
select max(sequence#) from v$log_history;

---

**Resolving Problems**

After adding a datafile to primary database, recovery of the standby database fails with the following error messages:
ORA-01157: cannot identify/lock data file 16 - see DBWR trace file
ORA-01110: data file 16: '/oracle/oradata/FGUARD/undotbs02.dbf'
ORA-27037: unable to obtain file status

Problem Explanation:
The datafiles do not exist on the standby database.

Solution Description:
Create the datafile(s) on the standby database. When the files exist, recovery can continue.
The datafiles are not automatically created on the standby site. For example, the redo does not create a new datafile for you.
Then create datafile command from startup mount is:
alter database create datafile '/home/orahome/data/721/users02.dbf';

---

**Pausing/Starting from PROD**
alter system set log_archive_dest_state_2='defer';
alter system set log_archive_dest_state_2='enable';
Getting 'Restarting dead background process QMN0' on Alert Log File
If you get many of this messages, just perform the following:
```sql
alter system set aq_tm_processes=0 scope=both;
```

**Gap Detected**

If there is a gap on the arch log files, then you need to perform the following:

1. Copy the arch logs that doesn't exist on the DR box
2. Apply them by using the following command:
```sql
SQL> alter database recover managed standby database disconnect from session;
```

If you see errors on the Alert.log file like:
```
Fetching gap sequence for thread 1, gap sequence 5007-5060
Trying FAL server: PROD_FGUARD
Wed May 31 10:19:41 2006
Failed to request gap sequence. Thread #: 1, gap sequence: 5007-5060
All FAL server has been attempted.
Restarting dead background process QMN0
```

Then try with :
```sql
RECOVER AUTOMATIC STANDBY DATABASE;
```
If you get :
```sql
ORA-01153: an incompatible media recovery is active
```

Then stop/restart DR and try the last command again:
```sql
startup nomount;
alter database mount standby database;
RECOVER AUTOMATIC STANDBY DATABASE;
```

After recovery is done, then:
```sql
alter database recover managed standby database disconnect from session;
```

**Recovering After a Network Failure**

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:
```sql
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.

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:

```
SELECT DEST_ID, STATUS, ERROR FROM V$ARCHIVE_DEST WHERE DEST_ID = 2;
```

<table>
<thead>
<tr>
<th>DEST_ID</th>
<th>STATUS</th>
<th>ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>ERROR</td>
<td>ORA-12224: TNS:no listener</td>
</tr>
</tbody>
</table>

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

If you cannot solve the network problem quickly, and if the physical 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:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2 = DEFER;
```

When the network problem is resolved, you can enable the archive destination again:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2 = ENABLE;
```

# Change the archive destination from mandatory to optional:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_2 = 'SERVICE=standby1 OPTIONAL REOPEN=60';
```

When the network problem is resolved, you can change the archive destination from optional back to mandatory:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_2 = 'SERVICE=standby1 MANDATORY REOPEN=60';
```

Data Guard Scenarios
Synchronize a GAP on the STANDBY when the required archived logs are lost

Scenario:
All archived logs were removed from primary database. The standby had lagged far behind the primary, many required archived logs to close the gap were removed and no backup of them was available.
In order to close the gap you need to create an incremental backup that will contain all transactions since the last scn recorded on the standby database.

Implementation Steps

1- Cancel Recovery on Standby
SQL> alter database recover managed standby database cancel;

If you try to recover from the standby you will get:
SQL> recover standby database;
ORA-00279: change 4146871739 generated at 12/31/2008 11:39:03 needed for thread 1
ORA-00289: suggestion : Z:\ORACLE\ORADATA\SATI\ARCHIVE\1_205_674755717.ARC
ORA-00280: change 4146871739 for thread 1 is in sequence #205

2- Check Standby Database current scn
SQL> select current_scn from v$database;
CURRENT_SCN
-----------
4146871738

3- Create a Primary Database Incremental Backup FROM this SCN and a Control File for Standby
rman target sys/pass@PROD
backup incremental from scn 4146871738 database FORMAT 'Z:\BACKUP\FOR_STANDBY_%U' tag 'FORSTANDBY';
backup current controlfile for standby format 'Z:\BACKUP\FORSTDBYCTRL.bck';

4- Transfer The Incremental Backup Sets to the Standby Server

5- Restore controlfile on the Standby
rman target sys/pass@STANDBY
RESTORE STANDBY CONTROLFILE FROM 'Z:\BACKUP\FORSTDBYCTRL.BCK';

6- Catalog the Incremental Backups on The Standby Server
Note that for the catalog command to succeed you will need to move the backups to be within the Flash Recovery Area. When you execute the catalog command, RMAN will ask you if you want to catalog the new files, you will need to say YES.

catalog start with 'Z:\FRA\SATISTD\BACKUPSET';
searching for all files that match the pattern
Z:\FRA\SATISTD\BACKUPSET
List of Files Unknown to the Database
=======================================
File Name: Z:\FRA\SATISTD\BACKUPSET\FOR_STANDBY_A7K471DJ_1_1
File Name: Z:\FRA\SATISTD\BACKUPSET\FOR_STANDBY_A8K471DK_1_1
File Name: Z:\FRA\SATISTD\BACKUPSET\FOR_STANDBY_A9K471EF_1_1
File Name: Z:\FRA\SATISTD\BACKUPSET\FOR_STANDBY_AAK471GL_1_1
Do you really want to catalog the above files (enter YES or NO)?
yes
cataloging files...
cataloging done

7-Recover the Database and Cleanup Redologs
RMAN> recover database noredo;
... ...
SQL> alter database flashback off;
Database altered.
SQL> alter database flashback on;
Database altered.
SQL> alter database recover managed standby database disconnect from session;
Database altered.

If more archived logs were created on the primary since the finish of the SCN based incremental backup then you can copy them over and recover the standby database using the command : "recover standby database"

8- Enable the broker at both sites and check
When enabling the broker again it will take over the responsibility of managing the site and will resynchronize both sites
SQL> alter system set dg_broker_start=true scope=both;

Improvements in Oracle Data Guard in Oracle 10gr2
Automatic Deletion of applied archive logs: Once primary database Archived logs are
applied to a Logical Standby Database, they are deleted automatically without DBA intervention. This makes it easier to maintain both primary and logical standby databases. Physical standby databases have had this functionality since Oracle 10g Release 1, by using Flash Recovery Area option.

- **No downtime required to generate Logical Standby:** The primary database is no longer required to shutdown or be put in QUIESCING state, as we can create the logical standby database from a hotbackup of the primary database just like the physical standby database.

- **Online upgrades:** A lot of DBAs have dreamed about this for long time, the DBA no longer required to shutdown the primary database to upgrade from Oracle 10g release 2 with Data Guard option. First, upgrade the logical standby database to the next release, test and validate the upgrade, do a role reversal by switching over to the upgraded database, and then finally upgrade the old primary database.

- **New Datatypes Supported.** I always used to hesitate whenever I thought of logical standby databases, as some of my databases never meet the pre-requisite conditions. In 10g release2, Oracle supports most of the datatypes, such as NCLOB, LONG, LONGRAW, BINARY_FLOAT, BINARY_DOUBLE, IOTs.

- **Fast-Start Failover.** This capability allows Data Guard to automatically, and quickly fail over to a previously chosen, synchronized standby database in the event of loss of the primary database, without requiring any manual steps to invoke the failover, and without incurring any data loss. Following a fast-start failover, once the old primary database is repaired, Data Guard automatically reinstates it to be a standby database. This act restores high availability to the Data Guard configuration.

- **Easy conversion of a physical standby database to a reporting database.** A physical standby database can be activated as a primary database, opened read/write for reporting purposes, and then flashed back to a point in the past to be easily converted back to a physical standby database. At this point, Data Guard automatically synchronizes the standby database with the primary database. This allows the physical standby database to be utilized for read/write reporting and cloning activities.

- **Automatic deletion of applied archived redo log files in logical standby databases.** Archived logs, once they are applied on the logical standby database, are automatically deleted, reducing storage consumption on the logical standby and improving Data Guard manageability. Physical standby databases have already had this functionality since Oracle Database 10g Release 1, with Flash Recovery Area.
Using Flashback with Data Guard

If a Logical Mistakes happen, we can address them with the Flashback techniques, introduced in Oracle Database 10g already, even if in an Data Guard Environment. In case of “Flashback Table To Timestamp” or “Flashback Table To Before Drop”, there is nothing special to take into account regarding the Standby Database. It will simply replicate these actions accordingly.

If we do “Flashback Database” instead, that needs a special treatment of the Standby Database. This posting is designed to show you how to do that:

DGMGRL> show configuration
Configuration
  Name: mycf
  Enabled: YES
  Protection Mode: MaxAvailability
  Databases:
    prima - Primary database
    physt - Physical standby database
  Fast-Start Failover: DISABLED
  Current status for "mycf": SUCCESS

This is an 11g Database, but the shown technique should work the same with 10g also. Prima & Physt are both creating Flashback Logs:

SQL> connect sys/oracle@prima as sysdba
SQL> select database_role,flashback_on from v$database;
    DATABASE_ROLE FLASHBACK_ON
      ------------- ---------------
      PRIMARY       YES

SQL> connect sys/oracle@physt as sysdba
SQL> select database_role,flashback_on from v$database;
    DATABASE_ROLE FLASHBACK_ON
      --------------- ---------------
      PHYSICAL STANDBY YES

Now we perform the "error":

SQL> select * from scott.dept;
SQL> drop user scott cascade;

The Redo Protocol gets transmitted with SYNC to the Standby Database and is
applied there with Real-Time Apply. In other words:
The Logical Mistake has already reached the Standby Database. We could have
configured a Delay in the Apply there to address such scenarios. But that is somewhat
“old fashioned”; the modern way is to go with flashback. The background behind that
is, that in case of a Disaster, hitting the Primary Site, a Delay would cause a longer
Failover time.
I will now flashback the Primary to get back Scott:
SQL> shutdown immediate
SQL> startup mount
SQL> flashback database to timestamp systimestamp -
interval '15' minute;
SQL> alter database open resetlogs;
SQL> select * from scott.dept;

There he is again! Until now, that was not different from a Flashback Database
Operation without Data Guard. But now my Standby Database is no longer able to do
Redo Apply, because it is “in the future of the Primary Database”.
I need to put it into a time, shortly before the present time of the Primary, in order to
restart the Redo Apply successfully:
DGMGRL> show configuration
Configuration
    Name: mycf
    Enabled: YES
    Protection Mode: MaxAvailability
    Databases:
        prima - Primary database
        physt - Physical standby database
    Fast-Start Failover: DISABLED
    Current status for "mycf":
    Warning: ORA-16607: one or more databases have failed
DGMGRL> show database physt statusreport
STATUS REPORT
    INSTANCE_NAME SEVERITY ERROR_TEXT
 * ERROR ORA-16700: the standby database has
diverged
to the primary database
*ERROR ORA-16766: Redo Apply is stopped*

Please notice that the show statusreport clause is a new feature of 11g. In 10g, you need to look into the Broker Logfile to retrieve that problem.

```sql
SQL> connect sys/oracle@prima as sysdba
SQL> select resetlogs_change# from v$database;
RESETLOGS_CHANGE#
----------------
294223
```

```sql
SQL> connect sys/oracle@physt as sysdba
SQL> flashback database to scn 294221;
Flashback complete.
```

*Note = I subtracted 2 from the Resetlogs Change No. above to make sure that we get to a time before the error.*

```
DGMGRL> show configuration
Configuration
   Name: mycf
   Enabled: YES
   Protection Mode: MaxAvailability
   Databases:
      prima - Primary database
      physt - Physical standby database
Fast-Start Failover: DISABLED
Current status for "mycf":
Warning: ORA-16607: one or more databases have failed
```

```sql
DGMGRL> edit database physt set state=apply-on;
Succeeded.
```

```
DGMGRL> show configuration
Configuration
   Name: mycf
   Enabled: YES
   Protection Mode: MaxAvailability
   Databases:
      prima - Primary database
```
physt - Physical standby database
Fast-Start Failover: DISABLED
Current status for "mycf":
SUCCESS

Simple as that!!!

**Monitor Data Guard**

```sql
select 'Last applied : ' Logs, to_char(next_time,'DD-MON-YY:HH24:MI:SS') Time
    from v$archived_log
    where sequence# = (select max(sequence#) from v$archived_log where applied='YES')
union
select 'Last received : ' Logs, to_char(next_time,'DD-MON-YY:HH24:MI:SS') Time
    from v$archived_log
    where sequence# = (select max(sequence#) from v$archived_log);
```

<table>
<thead>
<tr>
<th>LOGS</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last applied</td>
<td>16-JUL-09:09:24:16</td>
</tr>
<tr>
<td>Last received</td>
<td>16-JUL-09:09:28:36</td>
</tr>
</tbody>
</table>

```sql
select NAME Name, VALUE Value, UNIT Unit
    from v$dataguard_stats
union
select null,null,' ' from dual
union
select null,null,'Time Computed: '||MIN(TIME_COMPUTED)
    from v$dataguard_stats;
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>VALUE</th>
<th>UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
apply finish time +00 00:02:07.2 day(2) to second(1) interval
apply lag +00 00:01:59 day(2) to second(0) interval
estimated startup time 16 second
standby has been open N
transport lag +00 00:00:00 day(2) to second(0) interval

Computed: 16-JUL-2009 09:33:16

select to_char(max(last_time),'DD-MON-YYYY HH24:MI:SS')
"Redo onsite"
   from v$standby_log

Redo onsite

--------------------
16-JUL-2009 09:42:44

More Information

Data Guard related Metalink Notes
734862.1 Step By Step Guide On How To Recreate Standby Control File When Datafiles Are On ASM And Using Oracle Managed Files
749947.1 Enabling Encryption for Data Guard Redo Transport using the Advanced Security Option
751600.1 Data Guard Physical Standby Switchover/Failover Detailed Best Practices, 10.2.0.1 - 10.2.0.4 - Data Guard Broker, EM, or SQL*Plus
729551.1 Redo Transport Compression in a Data Guard Environment, 11.1.0.6 - 11.1.0.7
751528.1 Oracle Data Guard and SSH
754065.1 Installing Database Vault in a Data Guard Environment, 10.2.0.3 - 11.1.0.7
737460.1 Changing Storage Definition in a Logical Standby Database
565535.1 Flashback Database Best Practices & Performance
559353.1 SQL Apply and Extended Datatype Support
756732.1 Potentially Longer Instance Recovery Timings in a MAA Configuration When Using Logical Standby
603361.1 Developer and DBA Tips for Pro-Actively Optimizing SQL Apply
454848.1 Installing and Using Standby Statspack in 11gR1
434164.1 Data Guard Redo Log Repository Example
416314.1 Reinstating a Logical Standby Using Backups Instead of Flashback Database
416310.1 Reinstating a Physical Standby Using Backups Instead of Flashback
413696.1 Data Guard Broker does not support Cold Failover Clusters
413484.1 Data Guard Support for Heterogeneous Primary and Standby Systems in Same Data Guard Configuration
414043.1 Role Transitions for Data Guard Configurations Using Mixed Oracle Binaries
409013.1 Cascaded Standby Database
395982.1 Data Guard Support for Mixed HP PA-RISC 64-bit and HP Itanium Environments
370434.1 How to make CRS aware of the role change in Data Guard environment?
368276.1 Steps to workaround issue described in Alert 308698.1
364290.1 Global Customer Service Escalation Process
331924.1 RMAN backups in Max Performance/Max Availability Data Guard Environment
312434.1 Oracle10g Data Guard SQL Apply Troubleshooting
305360.1 Data Guard and Oracle Standard Edition
304488.1 Using standby_file_management with Raw Devices
304061.1 Oracle Data Guard Readme for SQL Apply Release 10.1.0.4
304059.1 Oracle Data Guard Readme for SQL Apply Release 10.1.0.3
303421.1 Creating a Logical Standby with a Different Block Size Than Primary
300479.1 Rolling Upgrades with Logical Standby
290814.1 Rolling a Standby Forward using an RMAN Incremental Backup in 10g
278643.1 Applying Patchset with a 10g Logical Standby in Place
278641.1 Applying Patchset with a 10g Physical Standby in Place
278521.1 Upgrading to 10g with a Physical Standby in Place
278371.1 Creating a Logical Standby with Minimal Production Downtime
278108.1 Upgrading to 10g with a Logical Standby in Place
276636.1 Data Guard GUI ASM restrictions
273177.1 Removing Archived Redo Log Files No Longer Needed By SQL Apply
273015.1 Migrating to RAC using Data Guard
271463.1 Handling Problematic DDL with Data Guard SQL Apply
271455.1 Synchronizing tables in a Logical Standby Database
271448.1 Oracle Data Guard Switchover & Failover Best Practices
269954.1 Script to Collect Data Guard Logical Standby Table Information
257341.1 Example Usage of the DBMS_LOGSTDBY Package
243709.1 Monitoring Physical Standby Progress
241512.1 Script to Collect Data Guard Logical Standby Diagnostic Information
241438.1 Script to Collect Data Guard Physical Standby Diagnostic Information
241374.1 Script to Collect Data Guard Primary Site Diagnostic Information
239100.1 Data Guard Protection Modes Explained
234631.1 Creating a Logical Standby from a Hot Backup
233519.1 Known Issues with Logical Standby
233491.1 Data Guard Wait Events
233261.1 Tuning SQL Apply Operations for Logical Standby
232649.1 Data Guard Gap Detection and Resolution
232240.1 Performing Switchover in a Data Guard Configuration
180031.1 Creating a Data Guard Configuration

All Docs in ML for Data Guard