Setting up Oracle 11g Data Guard for SAP customers

Database Version: 11g Release 2
Preface ................................................................................................ 1
Requirements ...................................................................................... 6
Aim of the document ........................................................................... 6
Preparatory work ................................................................................. 7
Structure of the standby database .................................................... 14
Creating standby redo log files .......................................................... 17
Configuring Data Guard Broker ......................................................... 18
Deploying the transfer of logs and function check ......................... 24
Implementing SAP BRARCHIVE with Data Guard ......................... 25
Disaster handling / role transition ...................................................... 27
Reconnect SAP instance to database ............................................... 46
Performance ...................................................................................... 50
Annex ................................................................................................ 52
Preface

The document explains the structure of a physical standby database with Oracle Data Guard in an SAP environment. It indicates all the steps needed to successfully install and configure an Oracle Data Guard system with a physical standby database and the logical order in which they must be carried out.

To enable you to operate the standby database (Oracle Data Guard), a description of how to configure the Data Guard Broker is also provided. In just a few steps this service allows you to swap the database roles. This means that in the event of a disaster, what is known as a switchover or failover is undertaken almost automatically. The database administrator can initiate the process with just one command.

Overview of Oracle Data Guard

Oracle Data Guard is the management, monitoring, and automation software infrastructure that creates, maintains, and monitors one or more standby databases to protect enterprise data from failures, disasters, errors, and data corruptions.

Data Guard maintains standby databases as consistent copies of the production database as far as transactions are concerned. These standby databases can be located at remote disaster recovery sites thousands of miles away from the production data center, or they may be located in the same city, same campus, or even in the same building. If the production database becomes unavailable because of a planned or an unplanned outage, Data Guard can switch any standby database to the production role, thus minimizing the downtime associated with the outage, and preventing any data loss.
Data Guard functionality

A Data Guard configuration consists of a production database, also known as the primary database, and up to nine standby database(s), which are consistent copies of the primary database as far as transactions are concerned. Data Guard maintains this transactional consistency using redo data. As transactions occur in the primary database, redo data is generated and written to the local redo log files. With Data Guard, this redo data is also transferred to the standby sites and applied to the standby databases, keeping them synchronized with the primary database. Data Guard allows the administrator to choose whether this redo data is sent synchronously or asynchronously to a standby site.

The underlying technologies for standby databases are Data Guard Redo Apply (physical standby database), and Data Guard SQL Apply (logical standby database). A physical standby database has on-disk database structures that are identical to the primary database on a block-for-block basis, and is updated using Oracle media recovery. A logical standby database is an independent database that contains the same data as the primary database. It is updated using SQL statements, and has the advantage that it can be used for recovery and for other tasks such as reporting and queries at the same time.

Data Guard enables role transitions between the primary database and a chosen standby database, reducing overall downtime during planned outages and unplanned failures.
The primary and standby databases, as well as their various interactions, may be managed using SQL*Plus. Data Guard also offers a distributed management framework called the Data Guard Broker, which automates and centralizes the creation, maintenance, and monitoring of a Data Guard configuration. For easier manageability, administrators may use either Oracle Enterprise Manager or the Broker’s own specialized command-line interface (DGMGRL) to take advantage of the Broker’s management capabilities.

The following diagram shows the Data Guard components.

**Benefits of Data Guard**

- **Disaster recovery and high availability** - Data Guard provides an efficient and comprehensive disaster recovery and high availability solution. Automatic failover and easy-to-manage switchover capabilities allow quick role transitions between primary and standby databases, minimizing the downtime of the primary database for planned and unplanned outages.

- **Complete data protection** - A standby database also provides an effective safeguard against data corruptions and user errors. Storage level physical corruptions on the primary database do not spread to the standby database. Similarly, logical corruptions or user errors that cause the primary database to be permanently damaged can be resolved. Finally, the redo data is validated at the time it is received at the standby database and also when applied to the standby database.
• **Efficient utilization of system resources** - A physical standby database can be used for backups and read-only reporting, thereby reducing the primary database workload and saving valuable CPU and I/O cycles. In Oracle Database 10g Release 2, a physical standby database can also be easily converted back and forth between being a physical standby database and an open read/write database. A logical standby database allows its tables to be simultaneously available for read-only access while they are updated from the primary database. A logical standby database also allows users to perform data manipulation operations on tables that are not updated from the primary database. Finally, additional indexes and materialized views can be created in the logical standby database for better reporting performance.

• **Flexibility in data protection to balance availability against performance requirements** - Data Guard offers the Maximum Protection, Maximum Availability and Maximum Performance modes to help enterprises balance data protection against system performance requirements.

• **Protection from communication failures** - If network connectivity is lost between the primary and one or more standby databases, redo data cannot be sent from the primary database to those standby databases affected. Once connectivity is reestablished, the missing redo data is automatically detected by Data Guard and the necessary archive logs are automatically transmitted to the standby databases. The standby databases are resynchronized with the primary database, with no manual intervention by the administrator.

• **Centralized and simple management** - Data Guard Broker automates management and monitoring tasks across the multiple databases in a Data Guard configuration. Administrators may use either Oracle Enterprise Manager or the Broker's own specialized command-line interface (DGMGRL) to take advantage of this integrated management framework.

• **Integrated with Oracle database** - Data Guard is available as an integrated feature of the Oracle Database (Enterprise Edition) at no extra cost.
Data Guard process architecture

As shown in the following figure, Data Guard uses several processes of the Oracle database instance to achieve the automation necessary for disaster recovery and high availability.

On the primary database, Data Guard uses the Log Writer (LGWR) process or multiple Archiver (ARCH) processes to collect transaction redo data. In order to ensure isolation from network disruptions, the Log Writer process uses specialized background processes, called LogWriter Network Server (LNS) process, to synchronously or asynchronously transmit the redo data to the standby database. The Archiver processes transmit the redo data to the standby database directly. The primary database also has the Fetch Archive Log (FAL) process to provide a client-server mechanism for transmitting archived logs to the standby database following a loss of communication between the primary and standby database(s), for automatic gap resolution and resynchronization.

On the standby database, Data Guard uses one or more Remote File Server (RFS) processes to receive redo data from the primary database, the Managed Recovery Process (MRP) to apply redo data to the physical standby database, and the Logical Standby Process (LSP) to apply SQL-translated redo data to the logical standby database.

If the Data Guard Broker is enabled, Data Guard also uses the Data Guard Broker Monitor (DMON) process to manage and monitor the primary and standby databases as a unified configuration.
Requirements

A second database host, configured in exactly the same way as the first, is needed to operate Oracle Data Guard. This means:

- Same operating system, e.g. AIX 5.3 ML6 on both hosts
- Same parameter settings on database and operating system (e.g. nfiles)
- Identical Oracle version, currently the 11.2.0.1 database patch set with individual patches as recommended in SAP note.
- Identical file system structure, especially for SAP data and Oracle home.
- The databases must be operated in ARCHIVELOG mode
- Use of server parameter file (SPFILE)

Please read SAP note 105047 on approving Oracle Data Guard for an SAP environment.

Aim of the document

The document provides step-by-step instructions for how to set up an Oracle Data Guard solution with a physical standby database. This should enable the user to manage the Data Guard system created using the Data Guard Broker.

- Learn how to setup a Data Guard configuration and the needed environment. There are prepared examples for setup all the configuration files and sql scripts at the annex.
- Read about the possibilities of that solution in case of a desire or necessary outages for maintances. Instructions for managing with Data Guard Broker, but also complete manual steps are given.
- Learn how to setup a full automated failover solution, which will let SAP application reconnect to database without doing anything manual, using Data Guard feature “Fast-Start failover” in conjunction with “Client connection timeout” and database role event.

The topics contain cross-references to the appropriate chapter in the Annex. The Annex is a collection of all the scripts and configuration commands you will have to write. Texts in [] are optional.
Preparatory work

The work that must be carried out in advance is described below. This includes the database parameters, SQLNet configuration and testing the SQLNet connection from the primary side to the standby host and vice versa.

Naming conventions

Shown here taking the example of the O11 system:

As a prerequisite for using Data Guard Broker all service_names, global_db_names and db_unique_name defined in listener.ora, tnsnames.ora and database parameter file has to be set up in lower case.

Not following this prerequisite ends with configuration errors later on, as Data Guard broker convert all input values like a mixed-case service_name (O11_oracle-lnx) to lower case (o11_oracle-lnx). But if the service_name is also defined in mixed_case at parameter file (O11_oracle-lnx), it will not be found by Data Guard broker, as it acts case sensitive.

For example the domain is set to WORLD in the document, but it can be set by individual needs.

<table>
<thead>
<tr>
<th>IDENTIFIER</th>
<th>PRIMARY DATABASE</th>
<th>PHYSICAL STANDBY DATABASE</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB-SID</td>
<td>O11</td>
<td>O11</td>
<td>Listener.ora</td>
</tr>
<tr>
<td>Host name</td>
<td>oracle-lnx</td>
<td>oracle-vm2</td>
<td>Listener / tnsnames.ora</td>
</tr>
<tr>
<td>DG config.</td>
<td>o11</td>
<td>o11</td>
<td>Configuration name for Broker</td>
</tr>
<tr>
<td>Service name</td>
<td>o11_oracle-lnx[.WORLD]</td>
<td>o11_oracle-vm2[.WORLD]</td>
<td>Tnsnames.ora</td>
</tr>
<tr>
<td>Listener name</td>
<td>Listener_o11</td>
<td>Listener_o11</td>
<td></td>
</tr>
<tr>
<td>Listener port</td>
<td>1527</td>
<td>1527</td>
<td>Listener.ora</td>
</tr>
<tr>
<td>Global_Dbname</td>
<td>o11_oracle-lnx_DGMGR [.WORLD]</td>
<td>o11_oracle-vm2_DGMGR [.WORLD]</td>
<td>Listener.ora [11g] DGMGR configured in dg broker as property StaticConnectIdentifier DGB and XPT will be auto-configured o demand</td>
</tr>
<tr>
<td></td>
<td>o11_oracle-lnx_DGB [.WORLD]</td>
<td>o11_oracle-vm2_DGB [.WORLD]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>o11_oracle-lnx_XPT [.WORLD]</td>
<td>o11_oracle-vm2_XPT [.WORLD]</td>
<td></td>
</tr>
<tr>
<td>DB_Unique_name</td>
<td>o11_north</td>
<td>o11_south</td>
<td>Spfile&lt;sid&gt;.ora Important: define in lower case!!</td>
</tr>
<tr>
<td>Oracle base</td>
<td>/oracle</td>
<td>/oracle</td>
<td></td>
</tr>
<tr>
<td>Data files</td>
<td>sapdata*</td>
<td>sapdata*</td>
<td></td>
</tr>
</tbody>
</table>
Database parameters

Parameters in addition to those normally specified in SAP note "Parameter recommendations for Oracle 11g" are needed to set up a Data Guard configuration between two databases. Below you will find a list of these additional parameters for both the primary and the standby database.

**Note:** We would recommend configuring the Data Guard Broker (chapter Configuring Data Guard Broker), then the parameters are set to the correct values depending on the properties configured once the Broker has been started. This takes precedence over all values set manually up until that point for the parameters in question.

If a Data Guard Broker is not being used, all parameters must be set manually in the parameter file. You will find an example of an application in the next section.

Definition of relevant parameters for Oracle Data Guard (see Oracle online documentation Data Guard Concepts and Administration)

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>log_archive_dest_1</td>
<td>Location where the archive logs are stored (locally)</td>
<td>Location=&quot;&lt;local directory&gt;&quot; [parameter] [valid_for = ( expression )]</td>
</tr>
<tr>
<td>log_archive_dest_2</td>
<td>1. Destination for transferring logs to the standby database. (remote)</td>
<td>SERVICE=&quot;&lt;service definition&gt;&quot; [parameter] [valid_for = ( expression )]</td>
</tr>
<tr>
<td></td>
<td>Other destinations are possible if necessary and/or if needed for other</td>
<td></td>
</tr>
<tr>
<td></td>
<td>standby databases (1 – n)</td>
<td></td>
</tr>
<tr>
<td>log_archive_dest_state_n</td>
<td>Status of destinations, possible values: ENABLE = active DEFER = inactive</td>
<td>ENABLE</td>
</tr>
<tr>
<td>local_listener</td>
<td>Communicates the database's local listener, needed for Data Guard</td>
<td>Name and definition in Tnsnames.ora or as net service string in the form (ADDRESS= ( PROTOCOL= protocol) (HOST= hostname) (PORT= port# ) )</td>
</tr>
<tr>
<td>log_file_name_convert</td>
<td>Enables file name components to be converted. In order to operate with</td>
<td>‘String’, ‘String’</td>
</tr>
<tr>
<td></td>
<td>standby log files, this parameter must be set to prevent unwanted log file clearing.</td>
<td></td>
</tr>
<tr>
<td>standby_file_management</td>
<td>Activates the automatic creation of new data files which are produced on the primary side during a tablespace</td>
<td>AUTO</td>
</tr>
</tbody>
</table>
### Oracle White Paper—Setting up Oracle 11g Data Guard for SAP customers

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>expansion.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OFF deactivates the feature</td>
<td></td>
<td></td>
</tr>
<tr>
<td>db_unique_name</td>
<td>Unique database name, important for identification because both databases are usually addressed with the same SID.</td>
<td>e.g. &lt;dbsid&gt;_&lt;location&gt;</td>
</tr>
</tbody>
</table>

**Additional database parameters when operating with Data Guard Broker**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>dg_broker_start</td>
<td>Starts or ends the DMON Data Guard Broker process</td>
<td>Manually: TRUE</td>
</tr>
<tr>
<td>TRUE = starts DMON</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FALSE = stops DMON</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

When configuring the Data Guard Broker, the following parameters are automatically written to the parameter file once the DMON process has been started:

<table>
<thead>
<tr>
<th>log_archive_dest_2</th>
<th>1. Destination for transferring logs to the standby database. (remote)</th>
<th>SERVICE=&quot;&lt;service definition&gt;&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>[parameter] [valid_for=&quot;# expression&quot;]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Is generated in full from the Broker configuration</td>
</tr>
<tr>
<td>log_archive_dest_state_n</td>
<td>Status of destinations, possible values: ENABLE = active</td>
<td>ENABLE</td>
</tr>
<tr>
<td></td>
<td>DEFER = inactive</td>
<td></td>
</tr>
<tr>
<td>log_archive_config</td>
<td>Configuration name of Data Guard configuration on the partner host.</td>
<td>Is created dynamically by the Data Guard Broker</td>
</tr>
</tbody>
</table>

Other parameters are generated dynamically by the Data Guard Broker based on the settings of configuration properties and placed in the SPFILE.

Defining the database unique name has an affect to the database diag directory. As of 11g the new diag feature replaces the old `background_dump_dest` and `user_dump_dest` parameters. Alert.log and tracefiles will be written in the `diagnostic_dest` subdirectories. Changing the database unique name in behalf of the Data Guard
configuration, a new subdirectory named with the db_uniquename will be created and all diagnostic output
will be redirected to this new folder.

Example of application for parameter settings:

Parameter Primary DB

*.db_unique_name='o11_north'
*.dg_broker_start=true
*.local_listener='(ADDRESS = (PROTOCOL = TCP)(HOST = oracle-lnx)(PORT = 1527))'
*.log_archive_dest_1='LOCATION="/oracle/O11/oraarch/O11arch", valid_for=(ONLINE_LOGFILE,ALL_ROLES)'
*.log_archive_dest_state_1='ENABLE'
*.log_archive_format='%t_%s_%r.dbf'
*.log_archive_max_processes=2
*.log_archive_min_succeed_dest=1
*.log_archive_trace=0
*.log_file_name_convert='O11','O11'
*.standby_file_management='AUTO'

Parameter Standby DB

*.db_unique_name='o11_south'
*.dg_broker_start=TRUE
*.local_listener='(ADDRESS = (PROTOCOL = TCP)(HOST = oracle-vm2)(PORT = 1527))'
*.log_archive_dest_1='location="/oracle/O11/oraarch/O11arch" mandatory valid_for=(all_logfiles,all_roles)'
*.log_archive_dest_state_1='ENABLE'
*.log_archive_format='%t_%s_%r.dbf'
*.log_archive_max_processes=2
*.log_archive_min_succeed_dest=1
*.log_archive_trace=0
*.log_file_name_convert='O11','O11'
*.standby_file_management='AUTO'

SQL*Net configuration

Data Guard needs a total of 3 different listener services per database.

<table>
<thead>
<tr>
<th>SERVICE IDENTIFICATION</th>
<th>DESCRIPTION</th>
<th>SERVICE NAME USED</th>
</tr>
</thead>
</table>
| *_XPT                   | Used to transfer the redo logs and for FAL requests | P: o11_north_XPT,[.WORLD]  
S: o11_south_XPT,[.WORLD] |
| *_DGB                   | Is used by the background processes for communication, e.g. for the heartbeat and RFS communication. | P: o11_north_DGB,[.WORLD]  
S: o11_south_DGB,[.WORLD] |
| *_DGMGRL                | Used for switchover and failover by the Data Guard Broker, configured as property StaticConnectIdentifier | P: o11_north_DGMGRL,[.WORLD]  
S: o11_south_DGMGRL,[.WORLD] |
The *XPT and *DGB services are usually initiated automatically by the Data Guard Broker and created dynamically, based on the database name given during create of configuration. This can be usually the same as the db_unique_name setting at database.

The *_DGMGRL service must however be defined in the Listener.ora in all cases.

Data Guard Broker uses this service at switchover to connect to the stopped instances and startup it again. If such a service would be dynamically (by instance) defined, it will not be available during instance shutdown. Thus definition has to be static with the listener.

This service is defined as "Global_DBName" in the listener.ora.

Services XPT and DGB should be created automatically by Data Guard Broker, if not it should be created manually in LISTENER.ORA. See the example listener.ora in annex.

It may be platform dependend if service will be created by Data Guard Broker or not.

Possible errors

- TNS-12514
  
  "TNS: could not resolve the connect identifier specified"

  *Cause:
  A connection to a database or other service was requested using a connect identifier, and the connect identifier specified could not be resolved into a connect descriptor using one of the naming methods configured. For example, if the type of connect identifier used was a net service name then the net service name could not be found in a naming method repository, or the repository could not be located or reached.

  *Action:
  - If you are using local naming (TNSNAMES.ORA file):
  - Make sure that "TNSNAMES" is listed as one of the values of the NAMES.DIRECTORY_PATH parameter in the Oracle Net profile (SQLNET.ORA)
  - Verify that a TNSNAMES.ORA file exists and is in the proper directory and is accessible.
  - Check that the net service name used as the connect identifier exists in the TNSNAMES.ORA file.
  - Make sure there are no syntax errors anywhere in the TNSNAMES.ORA file. Look for unmatched parentheses or stray characters. Errors in a TNSNAMES.ORA file may make it unusable.

In order to prevent error ORA-12514 during a switchover, the DGMGRL service must be created manually and more importantly without a domain even if the domain = WORLD is set. Should the ORA-12514 error still arise, the service should be defined once with domain and once without domain in order to satisfy both scenarios. For more details about the error, refer to the drc<sid>.log and sqlnet.log log files.
• TNS-12541
"TNS:no listener"

*Cause:
The connection request could not be completed because the listener is not running.

*Action:
Ensure that the supplied destination address matches one of the addresses used by the listener
- compare the TNSNAMES.ORA entry with the appropriate LISTENER.ORA file (or TNSNAV.ORA
if the connection is to go by way of an Interchange). Start the listener on the remote machine.

Check that the host name and port in the Listener.ora and tnsnames.ora configuration files match. Has the
listener been started?

The output:

    Lsnrctl services <listener_name>

Provides an overview of the services initiated. The services produced by the Data Guard Broker are then also
listed.
Example application for an SQL*Net configuration:

**Listener settings primary database**

```sql
ADMIN_RESTRICTIONS_LISTENER_O11 = on
LISTENER_O11 =
  (ADDRESS_LIST =
    (ADDRESS =
      (PROTOCOL = IPC)
      (KEY = O11.WORLD)
    )
    (ADDRESS =
      (PROTOCOL = IPC)
      (KEY = O11)
    )
    (ADDRESS =
      (COMMUNITY = SAP.WORLD)
      (PROTOCOL = TCP)
      (HOST = oracle-lnx)
      (PORT = 1527)
    )
  )
STARTUP_WAIT_TIME_LISTENER_O11 = 0
CONNECT_TIMEOUT_LISTENER_O11 = 10
SID_LIST_LISTENER_O11 =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = O11)
      (GLOBAL_DBNAME=o11_oracle-lnx)
      (ORACLE_HOME = /oracle/O11/11r2_32)
    )
    (SID_DESC =
      (SID_NAME = O11)
      (GLOBAL_DBNAME=o11_north_DGMGRL)
      (ORACLE_HOME = /oracle/O11/11r2_32)
    )
  )
```

**Listener Settings Standby DB**

```sql
ADMIN_RESTRICTIONS_LISTENER_O11 = on
LISTENER_O11 =
  (ADDRESS_LIST =
    (ADDRESS =
      (PROTOCOL = IPC)
      (KEY = O11.WORLD)
    )
    (ADDRESS =
      (PROTOCOL = IPC)
      (KEY = O11)
    )
    (ADDRESS =
      (COMMUNITY = SAP.WORLD)
      (PROTOCOL = TCP)
      (HOST = oracle-vm2)
      (PORT = 1527)
    )
  )
STARTUP_WAIT_TIME_LISTENER_O11 = 0
CONNECT_TIMEOUT_LISTENER_O11 = 10
SID_LIST_LISTENER_O11 =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = O11)
      (GLOBAL_DBNAME=o11_oracle-vm2)
      (ORACLE_HOME = /oracle/O11/11r2_32)
    )
    (SID_DESC =
      (SID_NAME = O11)
      (GLOBAL_DBNAME=o11_south_DGMGRL)
      (ORACLE_HOME = /oracle/O11/11r2_32)
    )
  )
```

**Tnsnames.ora primary database**

```sql
O11_oracle-lnx.WORLD=
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS =
        (COMMUNITY = SAP.WORLD)
        (PROTOCOL = TCP)
        (HOST = oracle-lnx)
        (PORT = 1527)
      )
    )
    (CONNECT_DATA =
      (SID = O11)
      (GLOBAL_NAME = O11)
      (SERVICE_NAME = o11_oracle-lnx)
    )
  )
O11_oracle-vm2.WORLD=
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS =
        (COMMUNITY = SAP.WORLD)
        (PROTOCOL = TCP)
        (HOST = oracle-vm2)
      )
    )
    (CONNECT_DATA =
      (SID = O11)
      (GLOBAL_NAME = O11)
      (SERVICE_NAME = o11_oracle-vm2)
    )
  )
```

**Tnsnames.ora Standby DB**

```sql
O11_oracle-vm2.WORLD=
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS =
        (COMMUNITY = SAP.WORLD)
        (PROTOCOL = TCP)
        (HOST = oracle-vm2)
        (PORT = 1527)
      )
    )
    (CONNECT_DATA =
      (SID = O11)
      (GLOBAL_NAME = O11)
      (SERVICE_NAME = o11_oracle-vm2)
    )
  )
O11_oracle-lnx.WORLD=
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS =
        (COMMUNITY = SAP.WORLD)
        (PROTOCOL = TCP)
        (HOST = oracle-lnx)
        (PORT = 1527)
      )
    )
    (CONNECT_DATA =
      (SID = O11)
      (GLOBAL_NAME = O11)
      (SERVICE_NAME = o11_oracle-lnx)
    )
  )
```
Checking the SQL*Net configuration

Once the configuration files have been created and implementation on the hosts is complete, you must check whether the databases can be accessed by the other partner host using SQL*Net. The TNSPING Oracle tool is used for this purpose.

    tnsping <DB-Name>
    lsnrctl services <listener_name>

Example:
DB-Name here corresponds to the identifier created in the tnsnames.ora for the database.

<table>
<thead>
<tr>
<th>PRIMARY HOST</th>
<th>STANDBY HOST</th>
</tr>
</thead>
<tbody>
<tr>
<td>tnsping o11_oracle-vm2[WORLD]</td>
<td>tnsping o11_oracle-lnx[WORLD]</td>
</tr>
<tr>
<td>lsnrctl services Listener_O11</td>
<td>lsnrctl services Listener_O11</td>
</tr>
</tbody>
</table>

A valid SQL*Net address string must be returned and must end with details of the runtime. If errors such as TNS-12541 "No listener" occur, this indicates incorrect configuration.

Then first check

- Whether the listener on the standby host has been started,
- Whether the host name is written correctly and can also be canceled
- Whether the port in Tnsnames.ora and the Listener.ora are the same.
- The service name used is identical.

Corrections can be undertaken on the basis of the error message.

Structure of the standby database

The database is usually produced from an OFFLINE or even an ONLINE backup of the production database. There are many ways of doing this but it can be easily done using e.g. the RMAN.

You will find the official Oracle description of how to set up a Data Guard environment in the Oracle documentation: Oracle Database High Availability Best Practices 10g Release 2 - Documentation.
Copying the production database (sample scenario)

In this scenario the datafile will be copied using ONLINE BACKUP feature with filesystem copies. Its also possible to do this if database is down, in that case “Begin Backup” and “End Backup” commands can be skipped.

Carry out on: oracle-lnx

SQL> alter database begin backup;

Copy data. This is done e.g. with the database user and SCP.

    scp -r /oracle/O11/sapdata* oracle@oracle-vm2:/oracle/O11/*

Carry out on: oracle-vm2oracle-lnx

SQL> alter database end backup;

As a rule when creating database files on the standby system you can use the same method as you would for a homogeneous system copy. Other possibilities are to:

- Restore a backup from tape
- Copy the database using RMAN

Copy database parameter file and password file

- Copy the database parameter file init<sid>.ora to the standby Oracle Home destination:
  $ORACLE_HOME/dbs
  Change all the parameters like given in the example found in annex.

- Copy the passwordfile: orapw<SID> to standby Oracle Home destination:
  $ORACLE_HOME/dbs
  The password can not be changed at standby side, as the database will be normally not opened.

Force logging

The most SAP databases are running with activated redo logging mode, as recommended, so all changes will be logged and transported to the Standby DB by logfiles.
For SAP BI systems there is an exception regarding the secondary indexes for the info cubes. All these indexes are created using the nologging option, therefore for these segments no redo will be written to logfiles. As an effect at the recovery at Standby DB, these blocks are marked as soft corrupt and will be skipped.
If standby database will become primary this will throw corruption errors during open.
An easy method to workaround this is by setting primary in permanent FORCE LOGGING mode, but it will have a negative effect for performance during the load phases of info cubes.
At the other hand, recreating all the secondary indexes in case of the role switch is needed, but will need time before the database will become productive. If a standby database is used for a SAP BI system, its on the customer to decide what way would be the best in his situation. For OLTP organized systems FORCE LOGGING is also an option, but not really needed, if there will be make no use from the nologging option.

To set a primary database in FORCE LOGGING mode use:

\`
ALTER DATABASE FORCE LOGGING;
\`

Creating a standby control file on the primary database

Carry out on: `oracle-lnx`

\`
SQL> startup mount
SQL> alter database create standby controlfile as '/oracle/O11/sapreorg/stdby_controf.dbf' reuse;
\`

You will find a suitable script in the Annex (`cre_stdbycontrolfile.sql`). But the command can also be run directly using SQLPLUS. The standby control file created now has to be copied to the standby host.

Copy data. This is done e.g. with the database user and SCP.

\`
scp -r /oracle/O11/sapreorg/stdby_controf.dbf oracle@oracle-vm2:/oracle/O11/sapreorg
\`

Distributing on the standby host

For example:

\`
cp /oracle/O11/sapreorg/stdby_controf.dbf ...
cp /oracle/O11/sapreorg/stdby_controf.dbf ...
\`

The file is copied into the directories which are stated in the database parameter file using the `CONTROLFILES=` parameter. The example in the Annex shows one possible process using shell script.
Mounting the standby database for the first time

This is done to test the function. It tests whether all database parameters match and whether the control files are in the right place and are readable. Also we can now take the chance to create a SPFILE from the copied init<SID>.ora.

```
SQL> connect / as sysdba
SQL> create spfile from pfile;
SQL> startup mount
```

The Data Guard Broker starts up but doesn't find a valid configuration, as a result of which error messages are logged in Alert.log and in Broker.log (drc<sid>.log).

For the remaining configuration period, it is very helpful to execute a tail -f on the alert log of both databases. This allows any errors to be detected and remedied immediately.

Creating standby redo log files

If the Data Guard logs are to be transferred with LGWR in ASYNC or SYNC mode, standby redo log files are needed. These receive the redo information "in time" from the primary database and buffer it. This ensures that if the online redo logs are lost on the primary side, all redo information is already present on the standby side.

If ARCH is used instead, standby redo log files do not have to be created, and if they have already been produced, they are not used.

The number of standby redo log files must always be 1 higher than the number of log groups defined. Should the amount of redo operations increase, this prevents the redo information from not being updated because there are no more free standby log files available. The size of the standby redo log files must match that of the normal redo log files. Therefore, please adjust the script in the Annex accordingly before using it to create the standby log files.

Standby log files should be produced on the standby side and on the primary side. The latter are needed in the event of a switchover.

Make sure that the directory exists on the file system.

Start SQLPLUS "connect / as sysdba"

Syntax:

```
SQL> ALTER DATABASE ADD STANDBY LOGFILE '<FILE DESTINATION>' SIZE <n>M REUSE;
```
Standby log files

Example:

```bash
[11g] replace
mkdir /oracle/O11/standbylog
SQL>
alter database add standby logfile '/oracle/O11/standbylog/srl1.dbf' size 100M reuse;
alter database add standby logfile '/oracle/O11/standbylog/srl2.dbf' size 100M reuse;
alter database add standby logfile '/oracle/O11/standbylog/srl3.dbf' size 100M reuse;
alter database add standby logfile '/oracle/O11/standbylog/srl4.dbf' size 100M reuse;
alter database add standby logfile '/oracle/O11/standbylog/srl5.dbf' size 100M reuse;
```

Configuring Data Guard Broker

Open primary database and mount standby database. The DMON process (Data Guard Monitor) on both database get active (dg_broker_start=TRUE).

Make sure that the listener on the standby database is started and that the database can be accessed with SQL*Net. (See Checking the SQL*Net configuration)

Any Error messages relating to the LOG_ARCHIVE_DEST_2 parameters can be prevented with

```sql
SQL> ALTER SYSTEM SET log_archive_dest_state_2='DEFER' SCOPE=MEMORY;
```

if log_archive_dest_2 has already been manually entered in the parameter file of the database.

To prevent any conflict with the automated broker settings, all manually set parameter for running a standby database should be removed from spfile before setup a broker configuration.

Using the Data Guard Broker the log_archive_dest_2 parameter is written to the SPFILE triggered by the Broker configuration.

Defining the protection mode

The transfer mode for the redo log files depends on the protection mode selected. Also refer to the documentation "Managing Data Protection mode".
First of all you should find out which protection mode Data Guard should be used in. Unless configured otherwise, the "Maximum Performance" default is used.

For the setup and testing phase “Maximal performance” mode is the best option. Later on, if the configuration works it can be changed for the configuration by particulary broker command. See Data Guard Broker Reference documentation for details.

<table>
<thead>
<tr>
<th>Protection Mode</th>
<th>Risk of Data Loss In the Event of a Disaster</th>
<th>Redo Transport Mechanism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Protection</td>
<td>Zero data loss</td>
<td>Synchronous (LGWR SYNC)</td>
</tr>
<tr>
<td>Maximum Availability</td>
<td>Zero data loss</td>
<td>Synchronous (LGWR SYNC)</td>
</tr>
<tr>
<td>Maximum Performance</td>
<td>Minimal data loss – usually few seconds</td>
<td>Asynchronous (LGWR ASYNC) or ARCH</td>
</tr>
</tbody>
</table>

The following modes are supported with SAP.

- **Maximum Performance**
  
  Maximum Performance mode is the default protection mode. It offers slightly less primary database data protection, but higher performance than Maximum Availability mode. In this mode, as the primary database processes transactions, redo data is asynchronously shipped to the standby database by the LGWR process. Alternatively, the Archiver process(es) (ARCH) on the primary database may also be configured to transmit the redo data in this mode. In any case, the commit operation of the primary database does not wait for the standby database to acknowledge receipt before completing the write process on the primary database. If any standby destination becomes unavailable, processing continues on the primary database and there is little or no effect on performance.

  In the case of failure of the primary database, redo data, which had not yet been sent to the standby database, is lost. However, if the network has sufficient throughput to keep up with peaks in redo traffic and the LGWR process is used to transmit redo to the standby server, the number of lost transactions will be very small or zero.

  The Maximum Performance mode should be used when availability and performance on the primary database are more important than the risk of losing a small amount of data. This mode is also suitable for Data Guard deployments over a WAN where the inherent latencies of the network may limit the suitability of synchronous redo transmission.
• **Maximum Availability**
  Maximum Availability mode offers the next highest level of data availability for the primary database. As with Maximum Protection mode, redo data is synchronously transmitted by LGWR from the primary database to the standby database, and the transaction is not committed on the primary database until it has been confirmed that the transaction data is available on disk on at least one standby server. However, in this mode, unlike the Maximum Protection mode, if the last participating standby database becomes unavailable e.g. because of network connectivity problems, processing continues on the primary database. The standby database may temporarily fall behind compared to the primary database, but when it is available again, the databases will automatically synchronize with no data loss, using accumulated archived logs on the primary database. Because of synchronous redo transmission, this protection mode can potentially impact on response time and throughput. Configuring a low latency network with sufficient bandwidth for peak transaction loads can minimize this impact.

  Maximum Availability mode is suitable for businesses that want the assurance of zero data loss protection, but do not want the production database to be impacted on by network/standby server failures. These businesses will accept the possibility of data loss should a second failure subsequently affect the production database before the initial network/standby failure is resolved.

• **Maximum Protection**
  Maximum Protection mode offers the highest level of data protection for the primary database, ensuring a comprehensive zero-data loss disaster recovery solution. When operating in Maximum Protection mode, redo records are synchronously transmitted by the LGWR process (through the LNS process) from the primary database to the standby database(s), and a transaction is not committed on the primary database until it has been confirmed that the transaction data is available on disk on at least one standby server. It is strongly recommended that this mode be configured with at least two standby databases. If the last participating standby database becomes unavailable, processing stops on the primary database. This ensures that no transactions are lost should the primary database fail after it loses contact with all of its standby databases.

  Because of the synchronous nature of redo transmission, this Maximum Protection mode can potentially impact on primary database response time. Configuring a low latency network with sufficient bandwidth for peak transaction load can minimize this impact. Stock exchanges, currency exchanges, and financial institutions are examples of businesses that may require this Maximum Protection mode.

"Maximum Performance" mode is recommended for operating a standby database with SAP when using LGWR. The Data Guard Broker configuration described here is therefore based on this recommendation. "Maximum Availability" mode is also available. This mode offers increased data security but decreases performance. The mode automatically switches back to "Maximum Performance" mode if synchronous communication cannot be maintained. “Max Availability Mode” is required if you want to setup Fast-Start Failover feature.
Delayed Redo apply or Real time apply

In the most of all cases at SAP, a delayed redo apply would be the best decision, as it will provide the best protection for human errors. So a delay of 3-4 hours protects the Standby DB for applying one or more destructive command to fast and give DBA a good chance to react by stopping the log apply and then extract unchanged data from the Standby database. This is one of the big advantages using a Standby database.

The amount of minutes to delay in past is defined by Data Guard Broker property DelayMins, for details see next chapter. How large the delay can be depends on

- the amount of redo created in a time window.
- the amount of redo can be applied in a time window.
- and the maximal time allowed for switch or failover which is defined by business needs.

Alternatively a Standby database can run in real time apply mode, that means, that all changes from the Primary database are immediately applied to Standby database, if the archive log would be recovered. At real time apply, this happens at least having the next logswitch at Primary.

Real time apply makes only sense in conjunction with configured and activated flashback database feature at the Standby database. Only then the database can be set back to the past again to retrieve data in case of human errors.

Real time apply is required for using Fast-Start failover.

For setup a Standby database with flashback database feature see Oracle White Paper:

"Setup Flashback Database on Data Guard Physical Standby Database for SAP Customers"

which you can find at [http://www.oracle.com/sap](http://www.oracle.com/sap) under ORACLE PRODUCTS FOR SAP ➔ Oracle Database ➔ Best Practice

Creating the configuration

The DGMGRL Broker interface should be started as the Oracle user:

```
oracle-lnx> DGMGRL
DGMGRL> connect sys
DGMGRL> password:
DGMGRL> create configuration <DG-Config-Name> as primary database is <DB_Unique_Name> connect identifier is <Service_name>;
```

Creating the Data Guard configuration

Example:

```
DGMGRL> create configuration o11 as primary database is o11_primary connect identifier is o11_oracle-lnx;
Configuration "o11" created with primary database "o11_primary"
```
Adding the physical standby database

The standby database must be mounted for this purpose.

DGMGRL> add database <DB_unique_Name> as connect identifier is <Service_name> maintained as physical;

Add physical standby database to DG config.

Example:
DGMGRL> add database o11_standby as connect identifier is o11_oracle-vm2 maintained as physical;
Database "o11_standby" added

Adapting the Data Guard Broker properties

The Data Guard Broker needs several settings relating to the databases, which deviate from the preset default settings. It is important that these properties match the corresponding database parameters, in case such parameters are already set.

A good way to make the configuration first time runnable is to use the defaults at first hand, and change the parameter needed later on using broker commands.

<table>
<thead>
<tr>
<th>PROPERTY NAME</th>
<th>CORRESP. DATABASE PARAMETER</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogXptMode</td>
<td>- within log_archive_dest_n</td>
<td>ASYNC should always be used when the LGWR is used to transfer logs in &quot;Maximum Performance&quot; mode. (This is the default)</td>
</tr>
<tr>
<td></td>
<td>Definition.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DelayMins</td>
<td>- within log_archive_dest_n</td>
<td>Length of delay on standby side in minutes Default = 0</td>
</tr>
<tr>
<td></td>
<td>Definition.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MaxFailure</td>
<td>- within log_archive_dest_n</td>
<td>Standard error tolerance for connection errors before an abort takes place. Default = 0</td>
</tr>
<tr>
<td></td>
<td>Definition.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>logFileNameConvert</td>
<td>Log_file_name_convert</td>
<td>Default = '&lt;SID\&gt;', '&quot;&lt;SID&gt;&quot; will be imported from database.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>hostname</td>
<td>-</td>
<td>use the physically hostname</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LocalListenerAddress</td>
<td>Local_listener</td>
<td>removed with 11.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Data Guard properties

Example:

Set the Delay for applying the redo to 3 hours:

DGMGRL> EDIT DATABASE oll_primary SET PROPERTY DelayMins='180';
DGMGRL> EDIT DATABASE oll_standby SET PROPERTY DelayMins='180';

Activating the configuration

The configuration must be activated before the Data Guard Broker can use it.

DGMGRL> ENABLE CONFIGURATION;

Errors may arise as a result of which the configuration cannot be activated. The Broker log file provides a check for this under:

/oracle/<SID>/saptrace/background/drc<sid>.log

Upon successful activation, the Data Guard Broker now undertakes the final parameterization for the databases on the basis of the properties and activates media recovery on the standby database in the background.

→ From this point on Data Guard is active.
Deploying the transfer of logs and function check

When using a Data Guard Broker configuration, the transfer of logs and the media recovery run without any user intervention.

To start the Data Guard and get it running, all that is required is:

- To set dg_broker_start=TRUE on both databases
- For the primary database to be open and
- For a standby database to have been mounted

The properties in the Broker can be changed at any time via the DGMGRL Broker interface. Please read the Oracle documentation Data Guard Broker Administration for more details.

Carry out a few manual log switches to perform the function check:

```
Alter system switch logfile;
```

Also check whether the log switch causes the redo to be transferred to the standby database. The Alert.log for both databases filled with corresponding entries.

On the primary database:

```
Wed Jan 20 22:09:07 2010
Thread 1 advanced to log sequence 151 (LGWR switch)
  Current log# 7 seq# 151 mem# 0: /oracle/O11/origlogA/log_g14m1.dbf
Wed Jan 20 22:09:43 2010
LNS: Standby redo logfile selected for thread 1 sequence 151 for destination LOG_ARCHIVE_DEST_2
```

On the standby database:

```
Wed Jan 20 22:14:45 2010
Media Recovery Waiting for thread 1 sequence 151 (in transit)
Thu Jan 21 05:37:52 2010
Archived Log entry 190 added for thread 1 sequence 151 ID 0x681c76d7 dest 3:
ARC0: Archive log thread 1 sequence 151 available in 5 minute(s)
Thu Jan 21 05:37:55 2010
Media Recovery Delayed for 5 minute(s) (thread 1 sequence 151)
Once the delay has passed, the media recovery should also be visible:
Thu Jan 21 05:42:56 2010
Media Recovery Log <logfile name>
```
Implementing SAP BRARCHIVE with Data Guard

As usual at SAP systems SAP tool BRARCHIVE is used for storing the archive logs to tape and clean up the archive log destination to prevent an archiver stuck. All archive logs which are written to log_archive_dest_1 will be handled by BRARCHIVE that way.
There is no change to BRARCHIVE configuration necessary. It can work like with any other standolane database.

Remember the way how redo information are processed since Oracle 10g Data Guard. As described, we now are using LGWR, what give some advantages compared using the ARCH process.
All redo will be immediately written to Standby Database log files using log_archive_dest_2 and therefore all that redo will be already available at Standby host. Further handling now is completely at the Standby Database, having a log switch at Primary will also switch the Standby logfile and then archive the former standby logfile to an archive file at Standby Database.
Such files will be applied by recover process (MRP) if the delay time is reached.

But what happens in the case Standby Database missed some standby logs and therefore even the standby archive log due to maintenance or temporary offline state of Standby Database?
The normal reaction of Standby Database is to request missing archive files, based on the log sequences from the Primary Database again. Unfortunately the files are already stored to tape and deleted by BRARCHIVE, so the ongoing FAL-Request will fail.
The missed files have to be restored back from tape again to log_archive_dest_1. This is time consuming and has to be done manually be DBA.

Using an additional Buffer destination

In case of a FAL-Request, Primary Database will check all its locally destinations for the requested files and if successfully send it again or returns an error if not found.
The idea is now, to define a secondary locally destination in log_archive_dest_3 to buffer all created archive files for the time of up to 24 hours.
That destination is not touched by BRARCHIVE. So the file should be still exists here, and FAL-Request can find and send the file from this secondary destination and no further manual action is needed.
Define this destination at Primary database like:

* .log_archive_dest_3='LOCATION="/oracle/PEC/oraarch2/PECarch", optional, valid_for=(ONLINE_LOGFILE,ALL_ROLES)'
For preventing any archiver stuck due to file system full at secondary destination, the parameter OPTIONAL has to be set. Starting the time of definition of the additionally locally destination primary ARCH process will write an archivelog file copy in parallel to this buffer destination too.

**Maintance the buffer destination**

The destination should be able to hold archive log files for the last 24 hours. How large the filesystem has to be depends on the size and amount of archive logs created for that timeframe. In some cases 24 hours would be too long and it will be needed too much disk space to hold all the files, therefore decide what is the best solution for your needs and money.

Calculate it like:

\[
\text{Destination size} = (\text{number of archives} \times \text{size}) / \text{hour} \times 24
\]

Create a independent filesystem which the given size. It can optional reside at any NFS share if necessary.

Implement a rolling delete routine, may be a cron job, for delete all files which are older then the given time (e.g. 24h) and already applied at Standby Database.

Using:

```sql
SQL> SELECT THREAD#, MAX(SEQUENCE#) AS "LAST_APPLIED_LOG"
     FROM V$LOG_HISORY GROUP BY THREAD#;
```

To find the last applied log from Standby Database to take care that the log can be deleted from destination.

**Taking backups from Standby Database using BRBACKUP**

From the Standby Database we can perform OFFLINE backups without having any impact to performance at the primary side. You can use BRBACKUP in a similar way as using it for the primary side. BRBACKUP only needs to know that is running at a Standby Database, so its logging and controlling tables are still written to primary side.

In configuration file `init<sid>.sap` at `$ORACLE_HOME/dbs` add the parameter

```plaintext
Backup_type = offline_standby
```

For more information see:

[http://help.sap.com/saphelp_40b/helpdata/de/0d/d311854a0c11d182b80000e829fbfe/content.htm](http://help.sap.com/saphelp_40b/helpdata/de/0d/d311854a0c11d182b80000e829fbfe/content.htm)
Disaster handling / role transition

Preparations

Also refer to "Data Guard Concepts and Administration " chapter 8.1.1 Preparing for a Role Transition (failover or switchover).
Verify that the initialization parameters for each database are configured correctly.

- The database, which takes on the role of the primary database, must run in ARCHIVELOG mode.
- Make sure that all files, which belong to a tablespace with temporary files, also exist on the standby side before the role transition takes place.

How long does the role transition take?

Data Guard provides the V$DATAGUARD_STATS view, which can estimate how long the process, will take.

Example:

```
SQL> COLUMN NAME FORMAT A18
SQL> COLUMN VALUE FORMAT A16
SQL> COLUMN TIME_COMPUTED FORMAT A24
SQL> SELECT * FROM V$DATAGUARD_STATS;
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>VALUE</th>
<th>UNIT</th>
<th>TIME_COMPUTED</th>
</tr>
</thead>
<tbody>
<tr>
<td>apply finish time</td>
<td></td>
<td>day(2) to second(1) interval</td>
<td>19-MAY-2008 09:29:07</td>
</tr>
<tr>
<td>apply lag</td>
<td>+00 00:15:51</td>
<td>day(2) to second(0) interval</td>
<td>19-MAY-2008 09:29:07</td>
</tr>
<tr>
<td>estimated startup time</td>
<td>10</td>
<td>second</td>
<td>19-MAY-2008 09:29:07</td>
</tr>
<tr>
<td>standby has been open</td>
<td>N</td>
<td></td>
<td>19-MAY-2008 09:29:07</td>
</tr>
<tr>
<td>transport lag</td>
<td>+00 00:00:00</td>
<td>day(2) to second(0) interval</td>
<td>19-MAY-2008 09:29:07</td>
</tr>
</tbody>
</table>

The example shows that there is no delay in the transfer of redo logs, but this is of importance only if another standby database is being cascaded. The active media recovery currently needs another 15:51 minutes.

The role transition will only take place once it is complete.
Switchover

During a switchover the roles of the databases are usually swapped. The previous standby database becomes the primary database and the previous primary one the standby one. This is done by parameterization and changing the control file.

Data Guard before a switchover

After a switchover

During the switchover phase both databases briefly assume the standby role.
Switchover with Data Guard Broker

The switchover is triggered by a simple command in the DGMGRL tool.

(“Data Guard Broker” chapter 6.8 scenario 7: Performing a Switchover Operation.)

It is important that the connect is made using a valid Listener Service which is statically defined in listener.ora and is defined in the local tnsnames.ora. For example “o11_oracle-lnx” service.

In case of a switchover connection can be made to the local DB service, here “o11_oracle-vm2” unique database name O11_SOUTH, or alternatively to the remote service “o11_oracle-lnx” unique database name O11_NORTH

```
DGMGRL> CONNECT SYS@o11_oracle-vm2
DGMGRL> Password: ***********
DGMGRL> show configuration
Configuration - o11
   Protection Mode: MaxPerformance
   Databases:
      o11_south - Primary database
      o11_north - Physical standby database
   Fast-Start Failover: DISABLED
Configuration Status:
   SUCCESS
DGMGRL> SWITCHOVER TO o11_north;
```

Switchover without Data Guard Broker

This approach is only needed if the Data Guard Broker has not been used. The steps have to be processed manually following a list. Very experienced database administrators should only undertake this approach, as it is very prone to error and can therefore result in problems when transitioning roles. This may even extend to scenarios where it is no longer possible to synchronize the standby database.

**Manual approach**

**Step 1**

Verify that it is possible to perform a switchover.

On the current primary database, query the SWITCHOVER_STATUS column of the V$DATABASE fixed view on the primary database to verify it is possible to perform a switchover. For example:

```
SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE;
SWITCHOVER_STATUS
---------------------
```

TO STANDBY
1 row selected

**Step 2**

Initiate the switchover on the primary database.

```sql
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PHYSICAL STANDBY
   WITH SESSION SHUTDOWN;
```

**Step 3**

Shut down and restart the former primary instance.

Shut down the former primary instance, restart and mount the database:

```sql
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP MOUNT;
```

At this point in the switchover process, both databases are configured as physical standby databases.

**Step 4**

Verify the switchover status in the V$DATABASE view.

```sql
SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE;
```

```
SWITCHOVER_STATUS
------------------
TO_PRIMARY
```
1 row selected

Any other value for the `switchover_status` will prevent a switchover. Follow the troubleshooting guide in “Oracle Data Guard Concepts and Administration” in this case.

**Step 5**

Switch the target physical standby database role to the primary role.

```sql
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY;
```
Finish the transition of the standby database to the primary role.
The task you perform depends on whether the physical standby database has ever been opened in read-only mode:

- If the physical standby database has not been opened in read-only mode since the last time it was started, issue the SQL ALTER DATABASE OPEN statement to open the new primary database:
  
  SQL> ALTER DATABASE OPEN;
  
  Then, go to step 7.

- If the physical standby database has been opened in read-only mode since the last time it was started, you must shut down the target standby database and restart it:
  
  SQL> SHUTDOWN IMMEDIATE;
  SQL> STARTUP;

**Step 7**

If necessary, restart log apply services on the standby databases. For the new physical standby database and for each other physical or logical standby database in the Data Guard configuration, if log apply services were not previously configured to continue operating through a switchover, use an appropriate command to restart log apply services.

**Step 8**

Begin sending redo data to the standby databases. Issue the following statement on the new primary database:

SQL> ALTER SYSTEM SWITCH LOGFILE;

**Failover to standby database**

**Limitation:**
This step must be taken when the primary system can no longer be accessed.
All redo transactions, which have not been written to the standby log files or are available, as an archive log cannot be recovered.

- If the database was operated in "Maximum Protection" or "Maximum Availability" mode, it is very likely that this will only affect the very last transaction which was not committed on the primary system and can therefore be considered as not having been written.

- When in "Maximum Performance" mode, this affects all transactions, which have not been transferred to the standby, redo logs since the last net buffer synchronization using the RFS processes running on the standby system.

- If the logs were transferred using ARCH, the last complete archive log and the online redo logs on the standby side will also be missing after the disaster.

- If log transfer was configured with LGWR, usually all REDO information produced before the disaster has been written to the standby redo log files.

- Check whether the online redo logs or archive logs on the primary database server are still readable and can be manually copied to the standby server.

Failover with Data Guard Broker

When using the DGMGRL tool, a failover can be carried out with just one command. Also refer to (“Data Guard Broker” chapter 6.9 Scenario 8: Performing a Manual Failover Operation)

DGMGRL> CONNECT SYS@o11_oracle-lnx
Failover without Data Guard Broker

This approach is only needed if the Data Guard Broker has not been used. The steps have to be processed manually following a list. Very experienced database administrators should only undertake this approach, as it is very prone to error and can therefore result in problems when transitioning roles. This may even extend to scenarios where it is no longer possible to synchronize the standby database.

**Manual approach**

**Step 1**

Identify and resolve any gaps in the archived redo log files. To determine if there are gaps in the archived redo log files on the target standby database, query the V$ARCHIVE_GAP view.

For example:

```
SQL> SELECT THREAD#, LOW_SEQUENCE#, HIGH_SEQUENCE# 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>90</td>
<td>92</td>
</tr>
</tbody>
</table>

Copy all missing log files (90,91,92) to log destination manually.

Register them:

```
SQL> ALTER DATABASE REGISTER PHYSICAL LOGFILE 'filespec1';
```

**Step 2**

Repeat Step 1 until all gaps are resolved.

**Step 3**

Copy any other missing archived redo log files.

```
SQL> SELECT UNIQUE THREAD# AS THREAD, MAX(SEQUENCE#) OVER (PARTITION BY thread#) AS LAST from V$ARCHIVED_LOG;
```

<table>
<thead>
<tr>
<th>THREAD</th>
<th>LAST</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
</tr>
</tbody>
</table>

**Step 4**
Initiate a failover on the target physical standby database. Issue the following statement to initiate the failover:

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

Note:
The `FORCE` keyword terminates active RFS processes on the target physical standby database.

Note:
Failover adds an end-of-redo marker to the header of the last log file being archived and sends the redo to all enabled destinations that are valid for the primary role (specified with the `VALID_FOR=(PRIMARY_ROLE, _LOGFILES)` or the `VALID_FOR=(ALL_ROLES, _LOGFILES)` attributes).

Note:
The `FINISH` keyword must follow all other keywords in the SQL statement except for `FORCE`, `WAIT`, or `NOWAIT`.

**Step 5**

Convert the physical standby database to the primary role.

After the recovery has finished, change the role of database to primary:

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY;
```

**Step 6**

Finish the transition of the standby database to the primary database role. The task you perform in this step depends on whether the physical standby database was ever opened in read-only mode:

- If the physical standby database has not been opened in read-only mode since the last time it was started, issue the `SQL ALTER DATABASE OPEN` statement to open the new primary database:

  ```
  SQL> ALTER DATABASE OPEN;
  Then, go to step 7.
  ```

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

  ```
  SQL> SHUTDOWN IMMEDIATE;
  SQL> STARTUP;
  ```

The target physical standby database has now undergone a transition to the primary database role.

**Step 7**
Back up the new primary database. If you choose online backup, set database in backup mode:

```
ALTER DATABASE BEGIN BACKUP;
```

Once the backup has finished, don’t forget to end backup mode with:

```
ALTER DATABASE END BACKUP;
```

### Restoring the standby database after a failover

If the primary database host can be used again, there are various ways of moving the database back onto the computer. You can use an offline backup from tape or a normal online backup via the file system. Oracle 10g also provides features for an automatic "Reinstate" command in the DGMGRL. However, this option is not available when using physical standby databases.

The procedure is the same as that used when creating a standby database.

- Set up using an online backup of the current production database
- Create a standby control file, copy and distribute to the new production host
- Renew the database property in the Data Guard Broker configuration:

```
DGMGRL> connect sys/<password>
DGMGRL> show configuration
DGMGRL> remove database <database name>
DGMGRL> add database <DB_unique_Name> as connect identifier is <Service_name>
maintained as physical;
DGMGRL> EDIT DATABASE <DB_unique_Name> set PROPERTY ...
```

Also see:

- Add physical standby database to Data Guard configuration.
- Data Guard properties
- The database service must also be recreated for Windows systems.

If the database has been recreated in this way, it is started up as a "physical standby database" and synchronized with the current production database with a normal delay until a scheduled downtime permits another switchover to return the database to its old role.
Fast-Start Failover

Definition

Fast-Start Failover enables an automated failover to standby database, in case the primary database goes down by incident or network loss. An Observer process is used to monitor the network connectivity and availability of the databases. The observer is a separate OCI client-side component that runs on a different computer from the primary and standby databases and monitors the availability of the primary database.

In case the connection to primary database got lost and does not come back in a defined timeout range, observer initiate a failover to standby database.

In difference to a manual failover, primary database will be reinstate by observer process automatically as a new standby database. No manual work is necessary by the DBA.

Later on, a gracefull planed switchover during a scheduled downtime will bring back the databases into their old roles.

See Oracle documentation Data Guard Broker - Fast-Start Failover chapter 5.5 for details.

- **before Fast-Start failover**
  Observer holds and monitor connections to primary and standby database. Primary is sending its redo to standby database.

- **Fast-Start Failover ensues**
  Observer has lost the connection to primary, even standby database too, so Fast-Start Failover is triggered. Standby database becomes the new primary.
• After Fast-Start Failover
Restoring the old primary database and bring it back into Mount state, enables observer to connect successfully again. Old primary will now reinstate as a standby database, triggered by observer process.

Events, which will trigger a Fast-Start failover and which are enabled by default are:

• Broken network connection between the observer and the primary database.
  If the connection is lost between the observer and the primary database, or there are network failures that cause the primary database to be isolated, the observer attempts a fast-start failover.

• Instance failures
  If a single-instance primary database (either RAC or nonRAC), or if all instances of a RAC primary database fail, the observer attempts a fast-start failover.

• Shutdown abort
  If a single-instance primary database (either RAC or nonRAC), or if all instances of a RAC primary database are shut down with the ABORT option, the observer attempts a fast-start failover. Fast-start failover will not be attempted for the other types of database shutdown (NORMAL, IMMEDIATE, TRANSACTIONAL).

• Offline datafiles
  If the observer determines that one or more datafiles in the primary database have been taken offline by the database because of I/O errors, the observer attempts a fast-start failover.

• Corrupted Dictionary
  Dictionary corruption of a critical database object. Currently, this state can be detected only when the database is open

• Corrupted Controlfile
  Controlfile is permanently damaged because of a disk failure.

Optional following events can be enabled too:

• Inaccessible Logfile
  LGWR is unable to write to any member of the log group because on an I/O error.

• Stuck archiver
  Archiver is unable to archive a redo log because the device is full or unavailable.

Except for the last condition (Offline datafiles), the observer attempts to reconnect to the primary database within the time specified by the FastStartFailoverThreshold configuration property before attempting a fast-start failover. When the primary database datafiles are offline, the observer initiates a fast-start failover.
immediately, without waiting for the amount of time specified by the FastStartFailoverThreshold property to expire.

Prerequisites

The following prerequisites must be met before the broker allows you to enable fast-start failover:

- Ensure the broker configuration is running in either maximum availability mode or maximum performance mode.
  (See “Data Guard Broker” Set the protection mode for configuration Section 4.6.1 for information about configuring the protection mode, standby redo logs, and the LogXptMode property.)
- running real time apply is no longer a prerequisite with 11g, but recommended to prevent warnings about fast_start failover is lagging.

Enable Flashback Database and set up a flash recovery area on both the primary database and the target standby database. This allows to jump back in time if necessary, so a delayed redo apply at standby database is not necessary and we can use real time apply .

See Oracle Whitepaper Setup Flashback Database on Data Guard Physical Standby Database for SAP Customers or Oracle documentation "Setting Up Flash Recovery Areas as Destinations" in Data Guard Concepts and Administration

- Install the DGMGRL command-line interface on the observer computer as described in “Data Guard Broker” - Oracle Data Guard Installation, Section 2.1.
- Configure the TNSNAMES.ORA file on the observer system so that the observer is able to connect to the primary database and the pre-selected target standby database.

Setup

See Oracle documentation “Data Guard Broker”- Enabling Fast-Start Failover, section 5.5.2.

Step 1: Configure and activate Flashback database

Follow the steps provided in Setup Flashback Database on Data Guard Physical Standby Database for SAP Customers. A Flashback area need to be set up at Primary and Standby database.

Step 2: Set the protection mode

Find out what protection mode is currently used:

```
SELECT PROTECTION_MODE, PROTECTION_LEVEL FROM V$DATABASE;
```

```
PROTECTION_MODE PROTECTION_LEVEL
----------------- ------------------
----------------- ------------------
```
Oracle 11g allows to implement Fast-Start Failover for the two modes “Maximum Availability” and also for “Maximum Performance”.

Using “Maximal Performance” mode

As this is the default mode, no further changes are necessary. At all take care that LogXPTMode is still set to 'ASYNC'.

If it was set to 'ARCH' before, set it back to 'ASYNC'

DGMGRL> EDIT DATABASE o11_north SET PROPERTY LogXptMode='ASYNC';
DGMGRL> EDIT DATABASE o11_south SET PROPERTY LogXptMode='ASYNC';

Set the property “FastStartFailoverLagLimit”

DGMGRL> EDIT CONFIGURATION SET PROPERTY FastStartFailoverLagLimit=10;

This configuration property establishes an acceptable limit, in seconds, that the standby is allowed to fall behind the primary in terms of redo applied, beyond which a fast-start failover will not be allowed. The lowest possible value is 10 seconds.

This property is used when fast-start failover is enabled and the configuration is operating in maximum performance mode.

Default value is set to 30 seconds.

Continue with Step 3.

Using “Maximal Availability” mode

First change the the matching Log transport mode (LogXptMode) to ‘SYNC’.

Configure redo transport services for this standby database using the LGWR, SYNC, and AFFIRM attributes of the LOG_ARCHIVE_DEST_n initialization parameter. This mode, along with standby redo log files, is required for the maximum protection or maximum availability protection modes. This redo transport service enables the highest grade of data protection to the primary database, but also incurs the highest performance impact.
DGMGRL> EDIT DATABASE PEC_hpcoesd1 SET PROPERTY LogXptMode='SYNC';
DGMGRL> EDIT DATABASE PEC_hpcoe8640r2 SET PROPERTY LogXptMode='SYNC';

Then change the protection mode of configuration to **Max Availability Protection**.

This protection mode provides the highest level of data protection that is possible without compromising the availability of the primary database. Like maximum protection mode, a transaction will not commit until the redo needed to recover that transaction is written to the local online redo log and to at least one remote standby redo log. Unlike maximum protection mode, the primary database does not shut down if a fault prevents it from writing its redo stream to a remote standby redo log. Instead, the primary database operates in maximum performance mode until the fault is corrected, and all gaps in redo log files are resolved. When all gaps are resolved, the primary database automatically resumes operating in maximum availability mode.

This mode ensures that no data loss will occur if the primary database fails, but only if a second fault does not prevent a complete set of redo data from being sent from the primary database to at least one standby database.

This protection mode is required if you enable Fast-Start failover.

DGMGRL> EDIT CONFIGURATION SET PROTECTION MODE AS MAXAVAILABILITY;
DGMGRL> show configuration

Configuration - o11
Protection Mode: MaxAvailability
Databases:
o11_north - Primary database
o11_south - Physical standby database
Fast-Start Failover: DISABLED

SQLPLUS:
SQL> SELECT PROTECTION_MODE, PROTECTION_LEVEL FROM V$DATABASE;

<table>
<thead>
<tr>
<th>PROTECTION_MODE</th>
<th>PROTECTION_LEVEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAXIMUM AVAILABILITY</td>
<td>MAXIMUM AVAILABILITY</td>
</tr>
</tbody>
</table>

It is not necessary to restart database after have changed the protection mode.

**Step 3: Configure REAL TIME APPLY**

Running real time apply requires to have standby redo logs in place. Setting up standby redo logs is already described in this whitepaper. To configure real time apply of redo stream, the broker property DelayMins is set to no value.

EDIT DATABASE o11_north SET PROPERTY DelayMins='';
EDIT DATABASE o11_south SET PROPERTY DelayMins='';
Step 4: Configure Fast-Start Failover

Specify the target standby database with the FastStartFailoverTarget property.

```
DGMGRL> EDIT DATABASE o11_north
SET PROPERTY FastStartFailoverTarget = 'o11_south';
DGMGRL> EDIT DATABASE o11_south
SET PROPERTY FastStartFailoverTarget = 'o11_north';
```

The target will be automatically set to the standby database if only 1 standby database is configured. In any other case it has to be set by the commands above.

Step 5: Set the FastStartFailoverThreshold property.

This property manages the time for failover. The default is set to 30 seconds, setting it to a different value overwrites the default and will give DBA a longer period to possibly stop the countdown.

```
DGMGRL> EDIT CONFIGURATION SET PROPERTY FastStartFailoverThreshold = 60;
```

This value depends on individual system level needs. It should be as short as possible but also give a realistic chance to prevent the failover if the primary database would be back to business in time, maybe due to a short network outage, so a failover is no longer necessary.

Step 6: Setup an Observer

An Observer can be created at any database home or particularly installed administrative client, which has a permanent network connection to the primary and the standby database.

For example, it can be setup at the consolidation system of a SAP landscape.

There is no need to install any different software if an available oracle home can be used. The observer will be provided by DGMGRL as a foreground process.

In the following example an observer is configured for a separate Unix user, which has permissions to read and execute on an existing oracle home from a different installation.

Example:

1.) create a user “observer” with group “dba”
2.) logon with user “observer”
3.) set environment to
   ```
   ORACLE_BASE=/oracle
   ORACLE_HOME=/oracle/<SID>/112_64
   LD_LIBRARY_PATH=$ORACLE_HOME/lib
   TNS_ADMIN=$HOME/config
   PATH=$ORACLE_HOME:$PATH
   ```
4.) create directory $HOME/config
5.) copy sqlnet.ora and tnsnames.ora from your primary database home to $HOME/config
6.) test with tnsping if the primary and standby database are reachable.

7.) Start “dmgmgrl” to test if the utility can be executed.

To start the observer, only the primary database must be running; it is not necessary for the target standby database to be running.

You can start the observer before or after you enable fast-start failover. However, it is recommended that you have the observer running whenever you have fast-start failover enabled. If Fast-Start failover is enabled, the observer immediately begins monitoring the status and connections to the primary and target standby databases.

To start the observer with DGMGRL, run the following command on the observer computer:

DGMGRL> connect sys@O11
Password: ********
DGMGRL> START OBSERVER;

This creates a foreground process, the prompt will never come back until a
DGMGRL> STOP OBSERVER

is executed from an second session. A scripted option which will log all output into an logfile and does not need to have a permaent window open for observer is described later.

After the observer was started the first time, Fast-Start Failover can now be enabled at Data Guard configuration.

Step 7: Enable Fast-Start Failover

To enable fast-start failover with DGMGRL, execute the ENABLE FAST_START FAILOVER command while connected to any database system in the broker configuration (but not to the observer). For example:

DGMGRL> ENABLE FAST_START FAILOVER;
Enabled.
DGMGRL> show configuration verbose
Configuration - o11

Protection Mode: MaxPerformance
Databases:
  o11_north - Primary database
  o11_south - (*) Physical standby database

(*) Fast-Start Failover target

Fast-Start Failover: ENABLED
Step 8 Verify the Fast_Start Failover environment

To verify the readiness of the fast-start failover configuration, issue the DGMGRL SHOW CONFIGURATION VERBOSE command or the SHOW FAST_START FAILOVER command on the primary database. For example:

DGMGRL> show fast_start failover

Fast-Start Failover: ENABLED

Threshold: 60 seconds
Target: o11_south
Observer: oracle-vm2
Lag Limit: 300 seconds
Shutdown Primary: TRUE
Auto-reinstate: TRUE

Configurable Failover Conditions
Health Conditions:
  Corrupted Controlfile YES
  Corrupted Dictionary YES
  Inaccessible Logfile NO
  Stuck Archiver NO
  Datafile Offline YES

Oracle Error Conditions:
(none)

Maintain the Observer

For generally information about working with an observer see Oracle documentation: Data Guard Broker Managing the observer.

Start the observer process:

Dgmgrl> connect sys@O11
Password:******
DGMGRL> START OBSERVER
Using a logfile

As observer is running as a foreground process, sometimes it would be better to run it in background and have all output logged to a file. Use the example below to execute in background:

```
oracle-vm2> dgmgrl -logfile observer.log sys/<password>@O11 "start observer" &
```

A logfile “observer.log” will created into the current directory.
For the connect we use a full qualified user account including password. As this may be not a secure option, because the SYS password is visible at the command line at particular operation systems, another way would be more secure to use

```
dgmgrl -logfile observer.log /@edu "start observer" &
```

but this need Oracle Walet Manager (Oracle Database Advanced Security Administrator’s Guide - section 9 Using Oracle Walet Manager) to be set up.

Test is `ps -ef | grep dgmgrl` give a password line back at you operating system:

```
observer  6416  6247  0 15:37 pts/2    00:00:00
dgmgrl -logfile observer.log             start observer
```

if not, using a full qualified login should not be a problem.

Stop the observer:
Open a second window with the observer unix user and type:

```
Dgmgrl> connect sys@O11
Password:********
DGMGRL> STOP OBSERVER
```

Or using a command line:
```
dgmgrl sys/manager@O11 "stop observer"
```

this can easy be scripted for a quicker use.

Monitoring:
You can check also at database if the Observer is present and the Fast-Start Failover status is still synchronized:

```
SQL> Select FS_FAILOVER_STATUS, FS_FAILOVER_OBSERVER_PRESENT from v$database;
FS_FAILOVER_STATUS | FS_FAILOVER_OBSERVER_PRESENT
---------------------|-----------------------------
SYNCHRONIZED         | YES    
```

View the Data Guard Broker “Fast-Start Failover Configuration Statistics and Status” from documentation for the value of column FS_FAILOVER_STATUS.
Disable Fast-Start Failover

There may be some reason to disable the feature. An overview can be find at section 5.5.4 in Oracle Data Guard Broker documentation. Disabling this let you still perform manual failover or switchover your Data Guard configuration.

The observer will not stopped by disabling Fast-start failover, it need stopped manually.

To disable Fast-Start failover in DGMGRL use:

```sql
DGMGRL> DISABLE FAST-START FAILOVER [FORCE];
```
Reconnect SAP instance to database

In principle, some different approaches are possible,

- using a **virtual host name** / IP address as is standard with a cluster – means: no changes in SAP profiles and SQL*Net config files in case of disaster

- using **Client connection timeout** with DATABASE_ROLE change event to trigger the database service start at the new primary side. - means: no changes in SAP profiles and SQL*Net config files in case of disaster

- using **prepared and tested profile files** with which the CI and application servers can be restarted should a disaster occur. - means: no changes at the SQL*Net profiles, but at SAP profiles in case of a disaster.

In all cases the SAP CI must again be able to access the database using the parameters set in the profile files. The CI must either be operated on a system independent of the database server, which would therefore still be available after the primary database server had experienced the disaster, or must then be restarted on a failover server. The failover server can also be identical to the secondary database server.

Consider to use one of the first both topics to have a gracefull reconnect of SAP without to have to restart any of that components. See the below SQL*Net configuration for details.

Reconnecting the CI

The SAP instance can either be reconnected by the newly started CI, which is on-hand with the previously preconfigured profile file DEFAULT.PFL and only actually starts to operate in the event of a disaster.

Set the following SAP profile parameters in the transaction RZ20 on the disaster CI or manually at file level at the default profile of SAP instance.

The file DEFAULT.PFL can be found in directory `/usr/SAP/PEC/SYS/profile`.

Set parameter SAPDBHOST to the current database hostname after the switch:

```
SAPDBHOST = <hostname of Standby DB>
```

There is no need to change the SID, as it will be still the same after a database role switch.

After that change start SAP CI to become parameter change active.

Each SAP instance has a start profile which possible includes an entry for setting the environment variable “dbs_ora_tnsname=” defining the database service for connect. Such a service will be provided by listsner and has to be configurat as a Net Service Name in tnsnames.ora.
Alternatively, if a virtual IP / host name is available or Client connection timeout setup is used, the disaster CI can run with the same parameter settings as the original CI. The modified host name is realized using the network layer. SAP CI is still bound to the previously defined VIP host name, even in case of a independent running SAP CI reconnection feature will take place. SAP CI will try reconnect to database until database service will become available again.

Reconnecting the application server

The profile parameter for the message server must be adapted to the application servers. However, if a virtual host name is used, no changes are required. The application servers must be restarted in order to log back onto the new CI (message server).

Enter the new message server on the application servers instance profiles and restart the instances.

```
rdisp/msname = <Host name of disaster CI>
```

SQL*Net configuration

Provided a virtual host name is not used, consider to use the described solution for the Client Connection timeout. Otherwise configuration in the TNSNAMES.ORA must be adapted to the disaster CI.

We would recommend entering and testing this in advance. If the disaster CI is on the same host as the standby database, the TNSNAMES.ORA of the database installation can be used if, for example, the environment variable TNS_ADMIN refers to the database home directory. TNS_ADMIN has been used again in combination with the instant client since 10g.

Client connection Timeout

This feature is available with 10.2 databases and above. It is based on load balancing entries and connect timeout settings in SQL*Net. A very good and detailed description for setting up this configuration can be found in the Oracle MAA Whitepaper “Client Failover Best Practices for Highly Available Oracle Databases: Oracle Database 10g Release 2”. Additional it needs to setup a database trigger for switching the database service at database role change event.

Example:

In TNSNAMES.ORA of client side, at SAP generally found at: /usr/sap/<SID>/SYS/profile/oracle

Create an Oracle Net service name that includes all primary and standby hosts in the ADDRESS_LIST, and allow load balancing.

```
O11_DVEBMGS00 =
```
In the client side SQLNET.ORA file, set the SQLNET.OUTBOUND_CONNECT_TIMEOUT parameter. This parameter enables clients to quickly traverse an address_list in the event of a failure. For example, if a client attempts to connect to a host that is unavailable, the connection attempt will be bounded to the time specified by the SQLNET.OUTBOUND_CONNECT_TIMEOUT parameter, after which the client attempts to connect to the next host in the address_list. This behavior continues for each host in the address_list until a connection is made. Setting the parameter to a value of 3 seconds will suffice in most environments.

SQLNET.OUTBOUND_CONNECT_TIMEOUT= t (time in seconds).

Switching database service at database role event

To make this working the database needs to provide a service “<SID>”, in our example it would be “O11”. Whereby we can possible connect to one of the above listed databases, as we will done using normal loadbalancing, we will find only 1 active primary database, in our Data Guard configuration which provide this service at one time.

May be the service is provided by the original primary instance or - after a desaster - by the overtaking standby database, which now owns the database role “PRIMARY”.

Switching this service is based an a database trigger, which will only fire at startup of database instance if the database role is “Primary”.

Such a service is set as default to SID for each the database.

All the created services can be listet using the query:

```
SQL> select service_id, name, enabled from dba_services;
```

```
SERVICE_ID NAME                                  ENABLED
---------- -------------------------- -------
 1 SYS$BACKGROUND                                         NO
 2 SYS$USERS                                              NO
 3 O11                                                    NO
```

This service is disabled by default, but can be enabled by a simple PL/SQL.

Another way would be, to define such a service in listener.ora. But that means, we have to modify or restart the listener service after a desaster at the standby host to provide this service. Having service enabled at both listeners at the same time, as we will get with a statically listener setup, will let fail connection entries, because
we cannot connect to a standby database with application users. Thus, this service should only be provided by
 database, which is open for use and that is the one with database role “PRIMARY”.

Using DBMS_SERVICE.START_SERVICE enable the DBA to switch the service to the active database
node using scripted solutions like an “AFTER STARTUP” database trigger.

We have to create a database trigger that fires on the database startup event to relocate the database service
'O11' to a Data Guard Standby database after it has transitioned to the primary role.

Example:

```sql
CREATE OR REPLACE TRIGGER manage_DBservice
  after startup on database
DECLARE
  role VARCHAR(30);
BEGIN
  SELECT DATABASE_ROLE INTO role FROM V$DATABASE;
  IF role = 'PRIMARY' THEN
    DBMS_SERVICE.START_SERVICE('O11');
  END IF;
END;
/
```

checking with :

```
[oracle@oracle-vm2 dbs]$ lsnrctl services listener_o11
LSNRCTL for Linux: Version 11.2.0.1.0 - Production on 21-JAN-2010 11:41:53
Copyright (c) 1991, 2009, Oracle. All rights reserved.
Connecting to (ADDRESS=(PROTOCOL=IPC)(KEY=O11.WORLD))
Services Summary... Service "O11.WORLD" has 1 instance(s).
  Instance "O11", status READY, has 1 handler(s) for this service...
  Handler(s):
    "DEDICATED" established:0 refused:0 state:ready LOCAL SERVER
```

This trigger will also be installed at standby side due to the redo transport. Next time standby side become
primary it will fire and start the service.
Performance

Various things may cause the log apply to not perform on the standby side and the database to continue to relapse in the recovery over time. This means that in the event of a disaster, the specified times (SLA) cannot be observed because the database cannot be recovered fast enough.

It is therefore well worth monitoring the performance of the standby database even during normal operation and intervening as early on as possible. Various $V$ views are available for this in mount status. Normal monitoring, e.g. using AWR views, is not possible because the database has to be open for this.

Performance monitoring

The most important performance views can be queried using SQLPLUS. One very common approach is provided by the Oracle Maximum Availability Architecture (MAA) paper "DataGuard Redo Apply and Media Recovery best Practices", which can be found at:


Parallel recovery

Another way of achieving a good recovery performance is to use parallel recovery.

This approach is however only worthwhile on servers with at least 8 CPUs because the overhead for managing the parallel processes otherwise partially cancels out the gain in performance.

When using Data Guard Broker, parallel recovery can be controlled using the "ApplyParallel" property.

DGMGRL> EDIT DATABASE <name> SET PROPERTY ApplyParallel='AUTO'

Possible values are:

- AUTO: The number of processes for the "Redo Apply" is calculated automatically by Oracle on the basis of the CPUs available. This is the default setting.
- NO: Parallel recovery is deactivated.

The following parameters should be set on the standby database. After a role transition, they should also be set on the now active standby database:

PARALLEL_MAX_SERVERS = <CPU*2>
DB_BLOCK_CHECKING=FALSE
PARALLEL_EXECUTION_MESSAGE >= 16348 \(\text{max} = 65536\)
Use a test run to compare the performance of serial and parallel recovery to establish the actual gain in performance.

As the number of parallel processes increases, CPU utilization and the I/O bandwidth used also goes up. Parallel recovery and the recovery slave processes use more system resources because of the IPC messaging overhead and recovery communication. Excess overheads may mean that the gain in performance is canceled out.
Annex

Database parameters of relevance to Data Guard, primary database

* .db_unique_name='o11_north'
* .dg_broker_start=true
* .local_listener='(ADDRESS = (PROTOCOL = TCP)(HOST = oracle-lnx)(PORT = 1527))'
* .log_archive_dest_1='LOCATION="/oracle/O11/oraarch/O11arch", valid_for=(ONLINE_LOGFILE,ALL_ROLES)'
* .log_archive_dest_state_1='ENABLE'
* .log_archive_format='%t_%s_%r.dbf'
* .log_archive_max_processes=2
* .log_archive_min_succeed_dest=1
* .log_archive_trace=0
* .log_file_name_convert='O11','O11'
* .standby_file_management='AUTO'

Database parameters of relevance to Data Guard, standby database

* .db_unique_name='o11_south'
* .dg_broker_start=TRUE
* .local_listener='(ADDRESS = (PROTOCOL = TCP)(HOST = oracle-vm2)(PORT = 1527))'
* .log_archive_dest_1='location="/oracle/O11/oraarch/O11arch" mandatory valid_for=(online_logfile,all_roles)'
* .log_archive_dest_state_1='ENABLE'
* .log_archive_format='%t_%s_%r.dbf'
* .log_archive_max_processes=2
* .log_archive_min_succeed_dest=1
* .log_archive_trace=0
* .log_file_name_convert='O11','O11'
* .standby_file_management='AUTO'

Listener settings, primary database

ADMIN_RESTRICTIONS_LISTENER_O11 = on
LISTENER_O11 =
ADDRESS_LIST =
  (ADDRESS =
    (PROTOCOL = IPC)
    (KEY = O11.WORLD)
  )
  (ADDRESS =
    (PROTOCOL = IPC)
    (KEY = O11)
  )
  (ADDRESS =
    (COMMUNITY = SAP.WORLD)
    (PROTOCOL = TCP)
    (HOST = oracle-lnx)
    (PORT = 1527)
  )
)
STARTUP_WAIT_TIME_LISTENER_O11 = 0
CONNECT_TIMEOUT_LISTENER_O11 = 10
TRACE_LEVEL_LISTENER_O11 = OFF
SID_LIST_LISTENER_O11 =
   (SID_LIST =
      (SID_DESC =
         (SID_NAME = O11)
         (GLOBAL_DBNAME=o11_oracle-lnx)
         (ORACLE_HOME = /oracle/O11/1102_64)
      )
      (SID_DESC =
         (SID_NAME = O11)
         (GLOBAL_DBNAME=o11_oracle-lnx_DGMGRL)
         (ORACLE_HOME = /oracle/O11/1102_64)
      )
      # optional
      (SID_DESC =
         (SID_NAME = O11)
         (GLOBAL_DBNAME=o11_oracle-lnx_DGMGRL.WORLD)
         (ORACLE_HOME = /oracle/O11/1102_64)
      )
      (SID_DESC =
         (SID_NAME = O11)
         (GLOBAL_DBNAME=o11_oracle-lnx_XPT.WORLD)
         (ORACLE_HOME = /oracle/O11/1102_64)
      )
      (SID_DESC =
         (SID_NAME = O11)
         (GLOBAL_DBNAME=o11_oracle-lnx_DGB.WORLD)
         (ORACLE_HOME = /oracle/O11/1102_64)
      )
      # end optional
   )
Listener settings, standby database

ADMIN_RESTRICTIONS_LISTENER_O11 = on
LISTENER_O11 =
  (ADDRESS_LIST =
   (ADDRESS =
    (PROTOCOL = IPC)
    (KEY = O11.WORLD)
   )
   (ADDRESS=
    (PROTOCOL = IPC)
    (KEY = O11)
   )
   (ADDRESS =
    (COMMUNITY = SAP.WORLD)
    (PROTOCOL = TCP)
    (HOST = oracle-vm2)
    (PORT = 1527)
   )
  )
STARTUP_WAIT_TIME_LISTENER_O11 = 0
CONNECT_TIMEOUT_LISTENER_O11 = 10
TRACE_LEVEL_LISTENER_O11= OFF
SID_LIST_LISTENER_O11=
  (SID_LIST =
   (SID_DESC =
    (SID_NAME = O11)
    (GLOBAL_DBNAME=o11_oracle-vm2)
    (ORACLE_HOME = /oracle/O11/1102_64)
   )
   (SID_DESC =
    (SID_NAME = O11)
    (GLOBAL_DBNAME=o11_oracle-vm2_DGMGRL)
    (ORACLE_HOME = /oracle/O11/1102_64)
   )
   # optional
   (SID_DESC =
    (SID_NAME = O11)
    (GLOBAL_DBNAME=o11_oracle-vm2_DGMGRL.WORLD)
    (ORACLE_HOME = /oracle/O11/1102_64)
   )
   (SID_DESC =
    (SID_NAME = O11)
    (GLOBAL_DBNAME=o11_oracle-vm2_XPT.WORLD)
    (ORACLE_HOME = /oracle/O11/1102_64)
   )
   (SID_DESC =
    (SID_NAME = O11)
    (GLOBAL_DBNAME=o11_oracle-vm2_DGB.WORLD)
    (ORACLE_HOME = /oracle/O11/1102_64)
   )
  )
# end optional
Tnsnames.ora, primary database

O11_oracle-lnx.WORLD=
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS =
        (COMMUNITY = SAP.WORLD)
        (PROTOCOL = TCP)
        (HOST = oracle-lnx)
        (PORT = 1527)
      )
    )
  )
  (CONNECT_DATA =
    (SID = O11)
    (GLOBAL NAME = O11)
    (SERVICE_NAME = o11_oracle-lnx)
  )
)

O11_oracle-vm2.WORLD=
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS =
        (COMMUNITY = SAP.WORLD)
        (PROTOCOL = TCP)
        (HOST = oracle-vm2)
        (PORT = 1527)
      )
    )
  )
  (CONNECT_DATA =
    (SID = O11)
    (SERVICE_NAME = o11_oracle-vm2)
  )
)

Tnsnames.ora, standby database

O11_oracle-vm2.WORLD=
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS =
        (COMMUNITY = SAP.WORLD)
        (PROTOCOL = TCP)
        (HOST = oracle-vm2)
        (PORT = 1527)
      )
    )
  )
  (CONNECT_DATA =
    (SID = O11)
    (GLOBAL NAME = O11)
    (SERVICE_NAME = o11_oracle-vm2)
  )
)

O11_oracle-lnx.WORLD=
(DESCRIPTION =
 (ADDRESS_LIST =
 (ADDRESS =
 (COMMUNITY = SAP.WORLD)
 (PROTOCOL = TCP)
 (HOST = oracle-lnx)
 (PORT = 1527)
 )
 )
 (CONNECT_DATA =
 (SID = O11)
 (SERVICE_NAME = o11_oracle-lnx)
 )
 )

Script: cre_stdbycontrolfile.sql
alter database create standby controlfile as
'oracle/O11/sapreorg/stdby_controlf.dbf' reuse;
scp -r /oracle/O11/sapreorg/stdby_controlf.dbf oracle@oracle-vm2:/oracle/O11/sapreorg

Script: cpy_controlf.sh

Creating the Data Guard configuration
create configuration o11 as primary database is o11_north connect identifier is o11_oracle-lnx;
Add physical standby database to DG config.
add database o11_south as connect identifier is o11_oracle-vm2 maintained as physical;
Data Guard properties

EDIT DATABASE o11_north SET PROPERTY LogXptMode='ASYNC';
EDIT DATABASE o11_north SET PROPERTY DelayMins='5';
EDIT DATABASE o11_north SET PROPERTY MaxFailure='100';
EDIT DATABASE o11_north set PROPERTY logfilenameconvert ="'O11','O11'";
EDIT DATABASE o11_north set PROPERTY hostname = 'oracle-lnx';
EDIT DATABASE o11_north set PROPERTY
StandbyArchiveLocation='/oracle/O11/oraarch/O11arch'

EDIT DATABASE o11_south SET PROPERTY LogXptMode='ASYNC';
EDIT DATABASE o11_south SET PROPERTY DelayMins='5';
EDIT DATABASE o11_south SET PROPERTY MaxFailure='100';
EDIT DATABASE o11_south set PROPERTY hostname = 'oracle-vm2';
EDIT DATABASE o11_south set PROPERTY logfilenameconvert ="'O11','O11'";
EDIT DATABASE o11_south set PROPERTY
StandbyArchiveLocation='/oracle/O11/oraarch/O11arch'

Standby log files

mkdir /oracle/O11/standbylog

alter database add standby logfile '/oracle/O11/standbylog/srl1.dbf' size 100M reuse;
alter database add standby logfile '/oracle/O11/standbylog/srl2.dbf' size 100M reuse;
alter database add standby logfile '/oracle/O11/standbylog/srl3.dbf' size 100M reuse;
alter database add standby logfile '/oracle/O11/standbylog/srl4.dbf' size 100M reuse;
alter database add standby logfile '/oracle/O11/standbylog/srl5.dbf' size 100M reuse;

References

Data Guard Broker HTML  PDF
Data Guard Concepts and Administration HTML  PDF
Oracle Database High Availability Best Practices 10g Release 2 - HTML
Documentation

Setup Flashback Database on Data Guard Physical Standby Database for SAP Customers HTML