

Oracle Maximum
Availability Architecture

Disaster Recovery in the Oracle Cloud

Production and DR in the Cloud

ORACLE WHITE PAPER | JULY 2016



ORACLE®

Table of Contents

Introduction	1
Disaster Recovery in the Cloud with Data Guard and Active Data Guard	2
Enabling DR on the Oracle Cloud	2
Options to Instantiate Standby Database	3
Validating DR Readiness	4
Using Standby Database to reduce downtime during Planned Maintenance	4
Service Level Requirements	5
Security Requirements	5
Data Guard Runtime Monitoring	6
MAA Deployment and Ongoing Operational Procedures	8
MAA Case Study on Oracle Cloud Services	10
Operational Procedures	11
Site Outage Testing and Results	12
Conclusion	12
Appendix A: Case Study Details	13
Initial Site Setup	13
Procedure: Ready Initial Site For Data Guard	16
Procedure: Secondary Site Setup	21
Procedure: Site Test	34
Procedure: Site Test to Standby	34
Procedure: Switchover	34
Procedure: Failover	35
Procedure: Reinstate Standby	35
Appendix B – Data Guard Health Checks	36
Appendix C – Data Guard Run-Time Monitoring Queries	37
Appendix D – Health Check Queries	40
Appendix E – Configuring Client Failover	42
Appendix F – Creating and Dispersing Wallets	42
Appendix G – Standby Instantiation From Database Backup Cloud Service	44

Introduction

Oracle Maximum Availability Architecture (MAA) is Oracle's best practices blueprint based on proven Oracle high availability technologies, expert recommendations and customer experiences. The goal of MAA is to achieve optimal high availability for Oracle customers on private, public and hybrid clouds at the lowest cost and complexity. Data Guard and Active Data Guard provide disaster recovery (DR) for databases with recovery time objectives (RTOs) that cannot be met by restoring from backup. Customers use these solutions to deploy one or more synchronized replicas (standby databases) of a production database (the primary database) along with the application tier in physically separate locations to provide high availability, comprehensive data protection, and disaster recovery for mission critical data.

Customers may choose to deploy either a Data Guard or an Active Data Guard standby on the cloud depending upon their requirements. While there are some unique considerations to a cloud DR configuration, it follows the same Oracle MAA best practices as with any Data Guard deployment. This Oracle MAA blueprint details Oracle MAA Best Practices and provides a procedural overview for deploying DR on the Oracle Cloud using Java Cloud Service and Database Cloud Service through a case study. This paper is intended for a technical audience having knowledge of Oracle Weblogic Server, Oracle Database, Data Guard or Active Data Guard, and Oracle Database backup and recovery. This paper also assumes a basic understanding of services offered on the Oracle Cloud¹.

The procedural overview and case study provided in this paper address the manual deployment of a primary database, a standby database, and redundant application tiers deployed across two separate identity domains, each associated with a geographically separate cloud data center. This procedure does not apply to the automated deployment of Data Guard or Active Data Guard using new tooling recently introduced for Oracle Database Cloud Service (similar automation for Exadata Service is planned for future availability). Oracle MAA best practices for deployments that utilize automated tooling are being developed and will be published on the [Oracle Technology Network](#) when available.

The case study assumes that all persistent state is contained within database and so no other state must be replicated to the secondary site.

¹ <https://cloud.oracle.com/home>

Disaster Recovery in the Cloud with Data Guard and Active Data Guard

The [Oracle Cloud](#)² offers an extensive set of cloud services tailored to specific customer requirements: Platform as a Service (PaaS), Infrastructure as a Service (IaaS), and Software as a Service (SaaS). Disaster Recovery (DR) between data centers is deployed using the Oracle Database Cloud Service or Exadata Service (PaaS).

There are two options for DR in the cloud using Oracle Database Cloud Services:

- » Data Guard utilizing Enterprise Edition Service or High Performance Service.
- » Active Data Guard utilizing the Extreme Performance Service or Exadata Service.

Data Guard is included in Oracle Database Enterprise Edition and is supported by all Enterprise editions of [Oracle Database Cloud Services](#) (Enterprise, High Performance and Extreme Performance) and [Exadata Service](#).

Active Data Guard extends Data Guard capabilities by providing advanced features for data protection and availability as well as offloading read-only workload and backups from a production database. Active Data Guard is included in the [Extreme Performance Database Cloud Service](#) and [Exadata Service](#). Both the primary and standby services must be provisioned with the Extreme Performance Database Cloud Service or Exadata Service. Refer to Oracle [software license documentation](#)³ for more information on capabilities licensed with Active Data Guard.

Unless otherwise noted, the procedure explained in this paper applies equally to Data Guard and Active Data. For more information on Data Guard and Active Data Guard please refer to the [Data Guard home page on the Oracle Technology Network](#) and the [Active Data Guard white paper](#)⁴.

The procedural overview and case study provided in this paper address the manual deployment of a primary database, a standby database, and redundant application tiers deployed across two separate identity domains, each associated with a geographically separate cloud data center. This procedure does not apply to the automated deployment of Data Guard or Active Data Guard using new tooling recently introduced for Oracle Database Cloud Service. Oracle MAA best practices for deployments that utilize automated tooling are being developed and will be published on the [Oracle Technology Network](#) when available.

Enabling DR on the Oracle Cloud

Enabling DR on the cloud requires instantiation of a Data Guard standby database in the Oracle Database Cloud Service. Once instantiated, Data Guard maintains synchronization between the primary database and the standby database.

The Oracle Cloud provides all backend infrastructure and capabilities required for disaster recovery should the primary site become unavailable for any reason. This includes:

1. Ability to monitor the standby database and alert on major issues.
2. Ability to activate the standby to validate DR readiness and then convert back to a synchronized standby.

² <https://www.oracle.com/cloud/cloud-summary.html>

³ <https://docs.oracle.com/database/121/DBLIC/options.htm#DBLIC141>

⁴ <http://www.oracle.com/technetwork/database/availability/active-data-guard-wp-12c-1896127.pdf>

3. Utilization of the same Oracle MAA best practices as on-premises deployments. Use of additional Oracle MAA best practices specific to cloud deployments that are specified in this paper.
4. Ability to switchover (planned event) or failover (unplanned event) production to the standby database in the cloud during planned maintenance or unplanned outages. Once the failed primary database is repaired, the ability to automatically resynchronize it with the new production database in the cloud and then switch production back to the original database service.
5. Ability to failover both the application and database tiers to enable production applications to run in the Oracle Cloud when there is a complete site outage.
6. Flexibility for a standby database in the Oracle Cloud to support additional use cases beyond disaster recovery, including: offloading read-only production workloads to the cloud, development and test, source for thin-provisioned database clones, and offloading backups to the cloud, as shown in Figure 1.

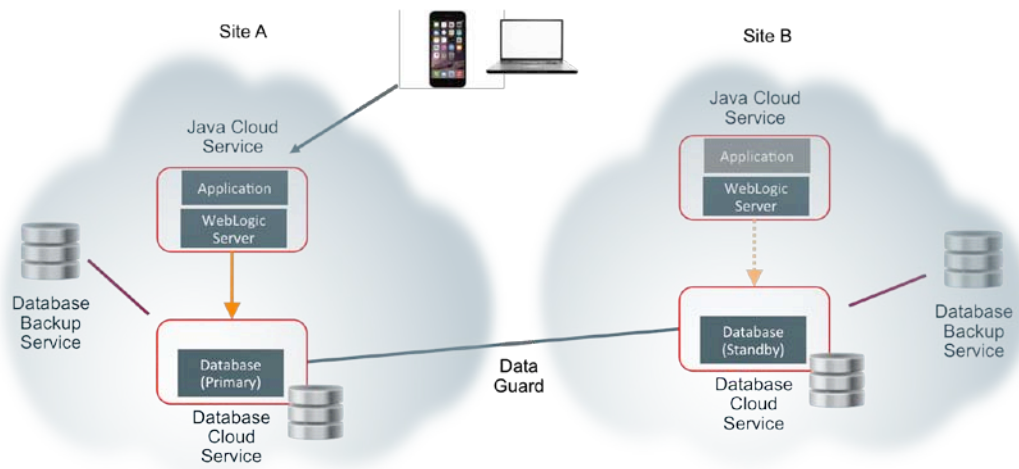


Figure 1: Disaster Recovery in the Oracle Public Cloud

Options to Instantiate Standby Database

Use one of the following options to instantiate a Data Guard standby in the cloud:

Duplicate from Primary Database

The primary production database is used to instantiate the standby database. Procedure: Secondary Site Setup documents the detailed steps for instantiating the standby from an active on-premises primary database.

Restore from Oracle Database Backup Cloud Service

Production database backups stored in Oracle Database Backup Cloud Service ([ODBCS](https://cloud.oracle.com/database_backup))⁵ are used to create the standby database using RMAN restore and recovery commands. Restore from the backup requires the same Password or TDE key.

⁵ https://cloud.oracle.com/database_backup

Validating DR Readiness

Best practice is to use Active Data Guard to offload read-only workload to the standby database to provide continuous, application-level validation that the standby is ready for production. This provides a level of assurance that is in addition to continuous Oracle block-level validation performed by Data Guard apply processes.

It is also best practice to periodically place the standby in read/write mode (using Data Guard Snapshot Standby) to validate its readiness to support read-write production workloads. A snapshot standby may also be used for a final level of pre-production functional and performance testing of patches and upgrades since the DR system is frequently sized similar to the production system. A Snapshot Standby continues to receive redo from the primary database where it is archived for later use, thus providing data protection at all times. Recovery time (RTO), however, will be extended by the amount of time required to convert the Snapshot Standby back to the standby database if a failover is required while testing is in progress. Note that additional storage is required for the fast recovery area when a standby is in snapshot mode (to hold archived redo received from the primary production database for later use and current redo and flashback logs generated by the snapshot standby). Please refer to Oracle documentation for additional details on [Data Guard Snapshot Standby](#)⁶.

Optionally you may perform an actual switchover or failover operation to the cloud for a complete end-to-end DR test.

Using Standby Database to reduce downtime during Planned Maintenance

There are several options for utilizing a standby database on the cloud for reducing planned downtime of the primary production database:

Standby-first Patch Apply

Many patches may be applied first to a physical standby for thorough validation. Customers who wish to minimize downtime will frequently patch the standby first, then switch production to the standby database, and then patch the original primary. Data Guard physical replication is supported between primary and standby running at mixed patch versions for patches that are standby-first eligible (documented in the patch read-me). The customer may also choose to run for a period of time with mixed patch versions between primary and standby to enable fast fallback to the unpatched version should there be any unanticipated problems with the patch. See My Oracle Support Note 1265700.1, [“Oracle Patch Assurance - Data Guard Standby-First Patch Apply”](#)⁷ for more details on patches eligible for the standby-first process.

Database Rolling Upgrade

Another beneficial use case for standby in the Oracle cloud is for database rolling upgrade to reduce downtime when upgrading to new database patch-sets and full Oracle releases. The transient logical process is used in Oracle 11g and Oracle 12c to temporarily convert a physical standby database to a logical standby, upgrade the logical standby to the new version, validate and when ready execute a Data Guard switchover. After the switchover completes, the original primary database is converted to a synchronized physical standby also operating at the new release. Refer to Oracle 11g [Database Rolling Upgrades Made Easy](#)⁸ or [Oracle 12c DBMS Rolling](#)⁹ for more information.

⁶ http://docs.oracle.com/database/121/SBYDB/manage_ps.htm#BACIEJJI

⁷ <https://support.oracle.com/epmos/faces/DocumentDisplay?id=1265700.1>

⁸ <http://www.oracle.com/technetwork/database/features/availability/maa-wp-11g-upgrades-made-easy-131972.pdf>

⁹ http://docs.oracle.com/database/121/SBYDB/dbms_rolling_upgrades.htm#CJACBBBC

Service Level Requirements

The administrator must determine service level expectations for availability, data protection, and performance that are practical for a given configuration and application. Service Levels must be established for each of three dimensions relevant to disaster recovery that are applicable to any Data Guard configuration:

- » **Availability:** Recovery Time Objective (RTO) describes the maximum acceptable downtime should an outage occur. This includes time required to detect the outage and to failover both the database and application connections so that service is resumed.
- » **Data Protection:** Recovery Point Objective (RPO) describes the maximum amount of data loss that can be tolerated. Achieving a desired RPO depends upon:
 - » Available bandwidth relative to network volume.
 - » The ability of the network to provide reliable, uninterrupted transmission.
 - » The Data Guard transport method used: either asynchronous for near-zero data loss protection, or synchronous for zero data loss protection.
- » **Performance:** Database response time may be different after failover if less capacity – compute, memory, I/O, etc., are provisioned at the standby system than in the primary system. This occurs when administrators purposefully under-configure standby resources to reduce cost; accepting reduced service level while in DR mode. MAA best practices recommend configuring symmetrical capacity at both primary and standby so there is no change in response time after failover. Rapid provisioning available with the cloud can enable a middle ground where there is less capacity deployed steady state, but the new primary is rapidly scaled-up should a failover be required.

Note: Independent of the service levels related to DR, all database instances created in the Oracle cloud conform to the service descriptions defined by the applicable Database Cloud Service¹⁰.

Security Requirements

Oracle MAA best practice recommends using Oracle Transparent Data Encryption (TDE) to encrypt primary and standby databases at rest. Conversion to TDE enables automatic encryption at rest for all DATA/INDEX tablespaces and encryption-in-flight of user data redo changes during replication to the cloud. Oracle Net encryption is also required for encryption-in-flight for other redo changes that are not encrypted by TDE (e.g. data from unencrypted tablespaces such as SYSTEM and SYSAUX).

Note: Data Guard and Active Data Guard use redo-based replication – a process that transmits redo generated by a primary database and applies those changes to a standby database using Oracle media recovery. This means that primary and standby databases are block for block identical copies of each other. Using TDE to encrypt a standby database also requires that the primary database be encrypted with TDE.

Using TDE to protect data is an important part of improving the security of the system. Users should however be aware of certain considerations when using any encryption solution, including:

¹⁰ <http://www.oracle.com/us/corporate/contracts/paas-iaas-public-cloud-2140609.pdf>

- » **Additional CPU overhead:** Encryption requires additional CPU cycles to calculate encrypted and decrypted values. TDE, however, is optimized to minimize the overhead by taking advantage of database caching capabilities and leveraging hardware acceleration for AES on Intel and SPARC CPUs. Most TDE users see little performance impact on their production systems after enabling TDE. If performance overhead is a concern, please refer to the [Oracle Database Advanced Security Guide](#)¹¹.
- » **Lower data compression:** Encrypted data compresses poorly because it must reveal no information about the original plain text data. Thus, any compression applied to TDE encrypted data will have low compression ratios. Hence, when TDE encryption is used, it is not recommended to use with redo transport compression. However, when TDE is used in conjunction with Oracle database compression technologies such as [Advanced Compression](#) or Hybrid Columnar Compression, compression is performed before the encryption occurs, and the benefits of compression and encryption are both achieved.
- » **Key management:** Encryption is only as strong as the key used to encrypt. Furthermore, the loss of the encryption key is tantamount to losing all data protected by that key. If encryption is enabled on a few databases, keeping track of the key and its lifecycle is relatively easy. As the number of encrypted databases grows, managing keys becomes an increasingly difficult problem. For users with large number of encrypted databases, it is recommended that [Oracle Key Vault](#)¹² on-premise be used to store and manage TDE master keys.

Data Guard Runtime Monitoring

There are two options for monitoring the run-time status of a Data Guard configuration: command line queries, or Enterprise Manager Cloud Control.

Option #1: Command Line Monitoring

Monitoring and alerting is required to ensure that the hybrid DR configuration is able to meet service levels established by the administrator for data protection (RPO) and availability (RTO).

See Appendix C – Data Guard Run-Time Monitoring Queries for a series of queries that monitor key aspects of the run time health of a Data Guard configuration:

- » Transport Lag
- » Apply Lag
- » Primary/Standby connection status
- » Data Guard Overall Status and Role Transition Readiness

The administrator will need to create scripts to automate the monitoring of the configuration such that alerts are sent when values exceed service level thresholds. This manual effort is not required if using Enterprise Manager 12c.


Option #2: Enterprise Manager 12c or 13

Oracle Enterprise Manager 12c or 13 streamlines and automates complex management tasks across the complete cloud lifecycle.

By deploying Management Agents onto the Oracle Cloud virtual hosts serving Oracle Cloud services, you are able to manage Oracle Cloud targets just as you would any other targets. The communication between Management Agents and on-premise Oracle management service instances is secure from external interference (requires Enterprise Manager Cloud Control 12c Release 1 (12.1.0.5) or higher). Support is provided for managing Oracle

¹¹ https://docs.oracle.com/database/121/ASOAG/asotrans_faq.htm#ASOAG10544

¹² <http://www.oracle.com/us/products/database/security/key-vault/overview/index.html>



Database and Fusion Middleware PaaS targets, as well as JVMD support for monitoring JVMs on Oracle Cloud virtual hosts.

Oracle Cloud Management includes the following features:

- » Automated agent deployment and configuration
- » Database and Java PaaS instances monitoring
- » Incident management including notifications and ticketing integration
- » Configuration management including Search and Inventory, comparison between on-premise and cloud instances, configuration history, and compliance
- » Cloning between on-premise and Oracle Cloud
- » One-off patching of Oracle Cloud database instances

Enterprise Manager provides a simple interface to monitor and manage a Data Guard environment, including:

- » Data Guard status
- » Transport Lag
- » Apply Lag
- » Estimated Database Role Transition Time
- » Primary or standby is or is not accessible

Alerts can be customized to notify of any event or when any of the above metrics exceed desired thresholds.

Refer to the [Enterprise Manager Cloud Control Administrator's Guide](#) for details

MAA Deployment and Ongoing Operational Procedures

In this document we assume as a starting point that the primary database and application are live and our goal is to transition to a dual site geographically distributed DR configuration.

In the following figure, we picture the states that a deployment goes through as it progresses from the initial single site implementation through the setup, testing and an eventual dual site MAA deployment. The systems will have a specific configuration in each state and there is a set of procedures that must be performed to move from one state to the other.

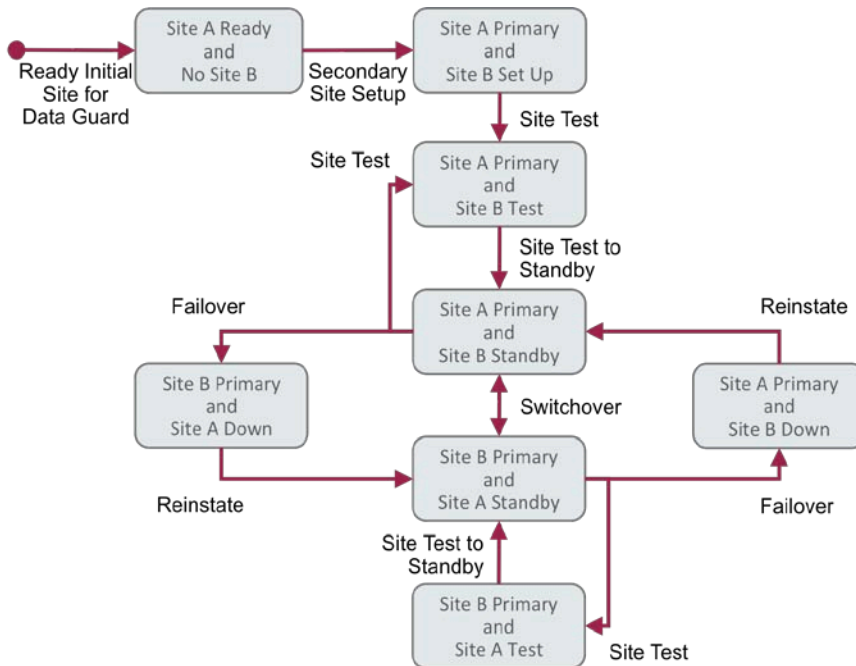


Figure 2: Site States and Operational Procedures

A summary of the operational procedures is provided in the following table:

TABLE 1 DESCRIPTION OF OPERATIONAL PROCEDURES

Operational Procedures (Each is linked to case study examples)	Description
Procedure: Ready Initial Site For Data Guard	Ready the primary site for Data Guard (assumes an existing deployment – see section Initial Site Setup for an example from the case study)
Procedure: Secondary Site Setup	Establish the secondary site database standby and application
Procedure: Site Test	Prepare the standby site for a site test.
Procedure: Site Test to Standby	Convert the site performing a site test back to standby mode.
Procedure: Switchover	Switch the roles so that the current standby site becomes the primary and the current primary site becomes the standby.
Procedure: Failover	Switch the current standby site to primary mode. The current primary site is assumed to be down or unavailable.
Procedure: Reinstate Standby	Reinstate the old primary site as a standby after failover.

The following table summarizes how the databases are configured in each state.

TABLE 2 DATABASE CONFIGURATIONS IN EACH SITE STATE

Site State	Database - Data Guard
Site A Ready and No Site B	Not configured
Site A Primary and Site B Set Up	Site A primary and Site B physical standby. Site B Snapshot standby during application setup.
Site A Primary and Site B Test	Site A primary and Site B snapshot standby.
Site A Primary and Site B Standby	Site A primary and Site B physical standby.
Site B Primary and Site A Down	Site B primary through failover, and Site A down.
Site B Primary and Site A Standby	Site B primary and Site A physical standby.
Site A Primary and Site B Down	Site A primary through failover and Site B down.
Site B Primary and Site A Test	Site B primary and Site A snapshot standby.

Application Access After Site Switchover or Failover

The public network IP address of the JCS instance is used to access the application, and this address is different on site A and site B. If an outage occurs on the primary site, and failover to the standby site is performed, it will be necessary for the application clients (users, client side applications) to use a different address to connect to the application when it is started on the secondary site. It may be possible to use a DNS push or intelligent client side application code to mask the address change from the end users, however, the details of how this can be done are very specific to a given customer environment and thus are out of the scope of this document.

MAA Case Study on Oracle Cloud Services

In this section we describe how the Maximum Availability Architecture described in the previous chapters was deployed on a system consisting of two geographically distributed sites each with a Database Cloud Service (DBCS) and Java Cloud Service (JCS) instance.

MedRec was used by the case study to represent a typical java based application. MedRec is an end-to-end sample Java EE application shipped with WebLogic Server that simulates an independent, centralized medical record management system. The MedRec application provides a framework for patients, doctors, and administrators to manage patient data using a variety of different clients. The application only stored data in the application database MEDREC PDB and so replication of all other storage artifacts was unnecessary. For this case study the single tenant version of MEDREC application was used.

The initial site (Site A) was located in Chicago, Illinois, USA and the secondary site (Site B) was located in Ashburn, Virginia, USA. The sites are approximately 600 miles apart.

The high level view of the configured system is pictured in Figure 3:

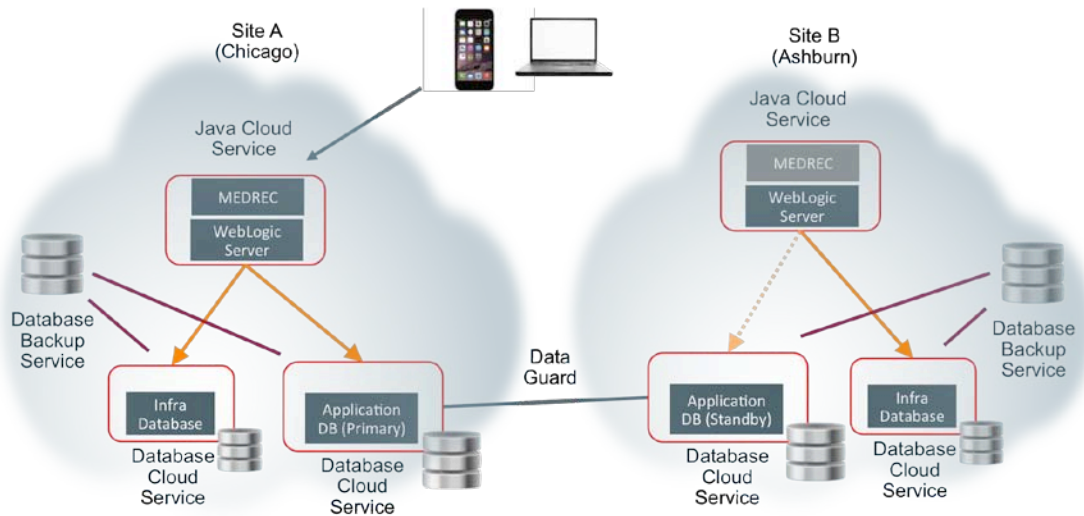


Figure 3: Example Disaster Recovery Configuration in the Oracle Public Cloud

The system parameters that were used for the case study are summarized in the following table:

TABLE 3 CASE STUDY SYSTEM PARAMETERS

System	Site A	Site B
Oracle Cloud Domain	domaina	domainb
Oracle Cloud User	joe@acme.com	joe@acme.com
Oracle Cloud Password	Acme1234#	Acme1234#
Database System Password (all databases)	Acme1234#	Acme1234#
JCS Instance	jcsa 2 OCPU	jcsb 2 OCPU
WLS Software Version	12.2.1.0	12.2.1.0
WebLogic User	weblogic	weblogic
WebLogic Password	Acme1234#	Acme1234#
JCS Backup Container	jcsaContainer	jcsbContainer
App DBCS Instance	appdba Extreme Performance 2 OCPU	appdbb Extreme Performance 2 OCPU
App DBCS Software Version	12.1.0.2	12.1.0.2
App Database Name	APPDB	APPDB
App Database Unique Name	APPDB	APPDBB
App Pluggable Database	MEDREC	MEDREC
App Instance Name	APPDB	APPDB
App Backup Container	appdbaContainer	appdbbContainer
JCS DBCS Instance	infraa Standard Edition 1 OCPU	infrab Standard Edition 1 OCPU
JCS Database Version	12.1.0.2	12.1.0.2
JCS Database Name	INFRAA	INFRAB
JCS Pluggable Database	INFRA	INFRA
JCS Instance Name	INFRAA	INFRAB
JCS Database Backup Container	infraaContainer	infrabContainer

Operational Procedures

The detailed steps that were followed to setup and test the complete system are documented in Appendix A: Case Study Details.

Site Outage Testing and Results

All servers on the primary site were stopped abruptly and site failover was performed. As the application was deployed at both sites (primary and standby) JCS instances as part of the DR configuration, there was no reconfiguration or redeployment of application during site switchover or failover. The standby site WLS managed server was just started to process any end-user traffic routed to Site-B

Observations on Site Outage

All users were lost on site outage and the site failover procedure was followed to restore service on the standby site. The failover procedure as documented in the section entitled Procedure: Failover was performed. and the timing for each step was recorded and is shown in Table 3 Recovery Times for Unplanned Site Outage:

TABLE 4 RECOVERY TIMES FOR UNPLANNED SITE OUTAGE

Recovery Step	Elapsed Time (seconds)	Cumulative Time (seconds)	Notes
Database Failover	37	37	Using Data Guard Broker
MEDREC Application Startup	63	100	
First User Login	5	105	

Each step in the process could have been executed by script with no delay between steps and in that case the total time to perform recovery, excluding the first user login, was estimated to be 100 seconds.

Conclusion

Disaster recovery in a cloud configuration consists of redundant applications tiers, a production database and a DR copy on the Oracle Cloud synchronized by Oracle Data Guard or Active Data Guard. Disaster Recovery on the Oracle Cloud eliminates the costs and complexity of owning and managing a remote facility as well as the capital expense of standby systems and software.

The use of Data Guard or Active Data Guard for disaster recovery eliminates the downtime and potential risk of relying upon a remote backup to restore service; production is quickly failed over to an already running, synchronized copy of your production database on the Oracle Cloud. The standby database on the cloud not only provides disaster recovery, it can also be used to seed clone databases for development and test as well as offloading read-only workloads from production databases.

Appendix A: Case Study Details

In this appendix we document the detail procedures that were followed during the case study setup and outage testing. The Initial Site Setup section is provided for reference. The setup will be different for each application.

Initial Site Setup

For reference, here are the steps that were performed to provision and configure Site A:

1. Provision Oracle Cloud Service Instances

1.1. Create Storage Containers

The following commands were executed to get the authentication token:

```
export SCSDOMAIN=domaina
export SCSUSER=joe@acme.com
export SCSPASSWORD='acme1234#'
curl -v -X GET \
  -H "X-Storage-User: Storage-${SCSDOMAIN}:${SCSUSER}" \
  -H "X-Storage-Pass: ${SCSPASSWORD}" \
  https://${SCSDOMAIN}.storage.oraclecloud.com/auth/v1.0
```

The following command set the authentication token:

```
export SCSAUTH=<token returned from previous step>
```

The following commands were executed to create the storage containers:

```
curl -v -X PUT \
  -H "X-Auth-Token: ${SCSAUTH}" \
  https://${SCSDOMAIN}.storage.oraclecloud.com/v1/Storage-${SCSDOMAIN}/jcsaContainer
curl -v -X PUT \
  -H "X-Auth-Token: ${SCSAUTH}" \
  https://${SCSDOMAIN}.storage.oraclecloud.com/v1/Storage-${SCSDOMAIN}/infraaContainer
curl -v -X PUT \
  -H "X-Auth-Token: ${SCSAUTH}" \
  https://${SCSDOMAIN}.storage.oraclecloud.com/v1/Storage-${SCSDOMAIN}/appdbaContainer
```

1.2. Provision JCS Infrastructure DBCS Instance

The following parameters were supplied when creating Site B's JCS Infrastructure DBCS instance from the Database Cloud Service – Service Console after clicking the “Create Service” button:

TABLE 5 CASE STUDY SITE 'B' JCS PARAMETERS

Page	Category	Option	Value
Subscription Type		Service Level	Oracle Database Cloud Service
		Billing Frequency	Monthly
Release		Software Release	12.1.0.2
Edition		Software Edition	Standard Edition
Service Details	Service Configuration	Service Name	infraa
		Shape	OC3 - 1 OCPU, 7.5 GB RAM
		Timezone	(UTC) Coordinated Universal Time(UTC)

	SSH Public Key	key.pub
Database Configuration	Usable Database Storage (GB)	25
	Administration Password	Acme1234#
	DB Name	INFRAA
	PDB Name	INFRA
	Character Set	AL32UTF8
	National Character Set	AL16UTF16
	Database Clustering with RAC	Unchecked
	Standby Database with Data Guard	Unchecked
	Enable Oracle Goldengate	Unchecked
	Include "Demos" PDB	Unchecked
Backup and Recovery Configuration	Backup Destination	Both Cloud Storage and Local Storage
	Cloud Storage Container	Storage-domaina/infraaContainer
	Cloud Storage User Name	joe@acme.com
	Cloud Storage Password	Acme1234#

1.3. Provision Application DBCS Instance

The following parameters were supplied when creating the DR DBCS instance from the Database Cloud Service – Service Console after clicking the “Create Service” button:

TABLE 6 CASE STUDY DATABASE CLOUD SERVICES (DBCS) PARAMETERS

Page	Category	Option	Value
Subscription Type		Service Level	Oracle Database Cloud Service
		Billing Frequency	Monthly
Release		Software Release	12.1.0.2
Edition		Software Edition	Enterprise Edition – Extreme Performance
Service Details	Service Configuration	Service Name	appdba
		Shape	OC4 - 2 OCPU, 15 GB RAM
		Timezone	(UTC) Coordinated Universal Time(UTC)
		SSH Public Key	key.pub
Database Configuration		Usable Database Storage (GB)	25
		Administration Password	*****
		DB Name	APPDB
		PDB Name	MEDREC
		Character Set	AL32UTF8
		National Character Set	AL16UTF16
		Database Clustering with RAC	Unchecked
		Standby Database with Data Guard	Unchecked
		Enable Oracle Goldengate	Unchecked
		Include "Demos" PDB	Unchecked
Backup and Recovery Configuration		Backup Destination	Both Cloud Storage and Local Storage
		Cloud Storage Container	Storage-domainb/appdbaContainer
		Cloud Storage User Name	joe@acme.com
		Cloud Storage Password	Acme1234#

1.4. Provision JCS Instance

The following parameters were supplied when creating Site A's JCS instance from the Java Cloud Service – Service Console after clicking the “Create Service” button:

TABLE 7 CASE STUDY SITE 'A' JCS PARAMETERS

Page	Category	Option	Value
Subscription Type		Service Level	Oracle Java Cloud Service
		Billing Frequency	Monthly
Software Release			12.2.1.0
Software Edition			Enterprise Edition
Service Details	Service Configuration	Service Name	jcsa
		Domain Partitions	0
		Shape	OC4 - 2 OCPU, 25 GB RAM
		SSH Public Key	key.pub
	WebLogic	Username	weblogic
		Password	Acme1234#
		Deploy Sample Application	Unchecked
	Database Configuration	Name	infra
		PDB Name	INFRA
		Administrator User Name	SYS
		Password	Acme1234#
	Backup and Recovery Configuration	Cloud Storage Container	Storage-domaina/jcsaContainer
		Cloud Storage User Name	joe@acme.com
		Cloud Storage Password	Acme1234#

2. Install and Configure Application

2.1. Create Encrypted Application Tablespace

The DATA tablespace used to store MEDREC application data was created in the MEDREC PDB with encryption on Site A' application database instance (appdba) by executing the following commands as sysdba:

```
ALTER SESSION SET CONTAINER = MEDREC;  
CREATE BIGFILE TABLESPACE "DATA" DATAFILE  
  '/u02/app/oracle/oradata/APPDB/MEDREC/MEDREC_data01.dbf' SIZE 3221225472  
  AUTOEXTEND ON NEXT 1073741824 MAXSIZE 33554431M  
  LOGGING ONLINE PERMANENT BLOCKSIZE 8192  
  EXTENT MANAGEMENT LOCAL AUTOALLOCATE ENCRYPTION using 'AES256' DEFAULT STORAGE(ENCRYPT)  
  SEGMENT SPACE MANAGEMENT AUTO;
```

2.2. Create Security Rule between JCS Instance and Application DBCS Instance at Site-A

A security rule was created between the JCS instance (jcsa) and APPDB DBCS DB Listener (security application)

It is recommended that the application to database traffic at each site is routed through the internal IP addresses using internal hostnames, instead of using the public IP addresses of DBCS and JCS instance. Refer Oracle Cloud Compute Documentation for more details for configuring security rules in Oracle Public Cloud

2.3. Install MEDREC Application

In the interest of brevity, the detailed steps for installing the MEDREC application are omitted.

If the applications to be deployed on the JCS instance have JMS and JTA requirements, it is recommended to externalize all those runtime artifacts into the database so that the data is protected from any disaster recovery event using Oracle Data Guard.

For more information about database persistent storage, see [Configuring WLS JMS with a Database Persistent Store](#) and [Configuring Database Stores to Persist Transaction Logs](#).

Procedure: Ready Initial Site For Data Guard

The purpose of this procedure is to prepare the initial site for DR. The steps performed were as follows:

1. Install Grid Infrastructure and Configure Oracle Restart

1.1. Install Grid Infrastructure Software

The Oracle Grid Infrastructure software was installed on the Site A's database host (appdba) and Oracle Restart was configured.

The following options were chosen during the installation:

- » Install option: Install Oracle Grid Infrastructure Software Only
- » Language: English
- » Oracle ASM Operator: dba
- » Oracle Base: /u01/app/oracle
- » Software Installation Location: /u01/app/12.1.0/grid

During the installation, for certain prerequisite check failures, Fix & Check Again was clicked to generate a fixup script (runfixup.sh). This fixup script was executed as the root user to complete the required pre-installation steps.

- » Zeroconf check was fixed by the fixup script

Also during the installation the following three warnings were ignored:

- » Insufficient swap space
- » NTP (not required for single instance)
- » Task resolv.conf Integrity

The output from the execution of the root.sh script documented an addition script that was executed to complete the Oracle Restart configuration.

The Oracle Grid Infrastructure (GI) has become an integral part of the application failover features for Oracle Data Guard. In the single instance case the GI provides Oracle Restart functionality.

The Grid Infrastructure is not installed with an Oracle Database Cloud Service in a standalone configuration and so it should be installed manually following the appropriate documentation for [12.1](#) or [11.2](#). The Oracle Grid infrastructure software is available for download from Oracle eDelivery. It is possible to download the software directly to the DBCS instance using the firefox browser.

Note: The steps followed in the case study assumed that the GI had been installed and configured. The Grid Infrastructure (GI) is not an absolute requirement for DR to function correctly; however, the steps in this document expect that it is installed. Alternate commands are available, but are not provided. GI is required for Fast Application Failover (FAN) functionality.

In order to run the Oracle installer on the OPC VM through the ssh connection, X11 forwarding must be enabled. Logged in as root via the opc user, set the following in /etc/ssh/sshd_config file:

```
X11Forwarding yes
X11DisplayOffset 10
X11UseLocalhost yes
```

As root, restart the ssh daemon:

```
service sshd stop
service sshd start
```

When connecting to the OPC VM used the following options to identify the connection as trusted:

```
ssh -o ServerAliveInterval=100 -Y oracle@<IPC VM IP address>
```

1.2. Configure the Database Listener in Oracle Restart

As user oracle on the Site A's database host (appdba):

```
lsnrctl stop listener
cp $ORACLE_HOME/network/admin/listener.ora \
  /u01/app/12.1.0/grid/network/admin/listener.ora
srvctl add listener -listener listener -oraclehome /u01/app/12.1.0/grid
srvctl start listener
```

The steps followed in the case study assume that the database listeners on all database servers are listening on port 1521.

1.3. Configure the Database in Oracle Restart

As user oracle on the Site A's database host (appdba):

```
srvctl add database -d APPDB -oraclehome $ORACLE_HOME -startoption open
```

Start the database and check its status using the srvctl command as below

```
srvctl status database -d APPDB
Database is not running.
```

```
srvctl start database -d APPDB
```

```
srvctl status database -d APPDB
Database is running.
```

2. Configure the Operating System and Database

2.1. Set TCP Socket Maximum Sizes

As documented in the Oracle Public Cloud documentation, you can gain root access to a provisioned cloud instance by connecting to the opc user and then using the sudo command:

```
sudo -s
```

The following commands were run as user root on Site A's database host (appdba):

```
sysctl -w net.core.rmem_max=10485760
sysctl -w net.core.wmem_max=10485760
```

The /etc/sysctl.conf file was also edited to reflect the changes so that they would be preserved during an instance reboot.

2.2. Size the Online Redo Logs

The redo logs created by default in the Database Cloud Service were 1GB in size and these were sufficient for our testing.

Online redo logs should be sized by this formula, but should not be less than 1GB in size:

$$\text{peak redo rate per minute} \times 20$$

Redo rates can be extracted from AWR reports during peak workload periods such as batch processing, quarter or year end processing. It is very important to use peak workload and not averages (averages can obscure peak redo rates and lead to provisioning redo logs that are too small).

2.3. Create Standby Redo Logs

The following commands were executed as user sys on Site A's application database (APPDB):

```
alter database add standby logfile thread 1 group 4
  '/u04/app/oracle/redo/stbyredo04.log' size 1073742336;
alter database add standby logfile thread 1 group 5
  '/u04/app/oracle/redo/stbyredo05.log' size 1073742336;
alter database add standby logfile thread 1 group 6
  '/u04/app/oracle/redo/stbyredo06.log' size 1073742336;
alter database add standby logfile thread 1 group 7
  '/u04/app/oracle/redo/stbyredo07.log' size 1073742336;
```

```
select * from v$logfile;
```

The MAA best practice for Standby Redo Logs (SRLs) is to create the same number as there are groups of Online Redo Logs (ORLs) **plus 1**. On RAC this must be done for each thread. Use the following query to discover how many redo log groups you have in each thread.

```
select thread#, count(group#) from v$log group by thread#;
```

SRLs should be created the same size as the largest of the ORLs. The group numbers are shared with the ORLs and so SRLs are created with higher group numbers. To gather the size of the largest ORLs and the current highest group number execute the following query:

```
select max(bytes), max(group#) from v$log;
```

The MAA best practice is that SRLs are not duplexed.

2.4. Encrypt the Database (Optional)

The DATA tablespace was created with encryption for the MEDREC PDB during the initial setup – see 2.1. Create Encrypted Application Tablespace.

The Data Guard primary and standby databases are physical replicas of each other. If a standby database in the cloud is to be encrypted at rest, then the primary must be encrypted first. Oracle MAA Best Practices provide guidance for converting to TDE with minimal downtime for [Oracle Database 11.2](http://www.oracle.com/technetwork/database/availability/tde-conversion-11g-2531187.pdf)¹³ or for [Oracle Database 12c](http://www.oracle.com/technetwork/database/availability/tde-conversion-12c-2537297.pdf)¹⁴. The following MAA whitepaper describes how conversion can be performed with minimal downtime - <http://www.oracle.com/technetwork/database/availability/tde-conversion-dg-3045460.pdf>.

2.5. Enable Force Logging, Flashback Database, and Set Recommended MAA Database Parameters

The following command was execute as user sys on Site A's application database (APPDB):

```
alter database force logging;
alter database flashback on;
alter system set DB_FLASHBACK_RETENTION_TARGET=120 scope=both sid='*';
alter system set remote_login_passwordfile='exclusive' scope=spfile sid='*';
alter system set DB_BLOCK_CHECKSUM=FULL;
alter system set DB_BLOCK_CHECKING=MEDIUM;
alter system set DB_LOST_WRITE_PROTECT=TYPICAL;
alter system set LOG_BUFFER=256M scope=spfile sid='*';
alter system set STANDBY_FILE_MANAGEMENT=AUTO scope=spfile sid='*';
```

NOTE: `DB_BLOCK_CHECKSUM`, `DB_BLOCK_CHECKING` and `DB_LOST_WRITE_PROTECT` are parameters that assist in preventing block corruption. The listed settings are recommended for the protection benefits though `DB_BLOCK_CHECKING` can have an impact on performance that should be considered when choosing a value. A value of `FALSE` turns `DB_BLOCK_CHECKING` off.

¹³ <http://www.oracle.com/technetwork/database/availability/tde-conversion-11g-2531187.pdf>

¹⁴ <http://www.oracle.com/technetwork/database/availability/tde-conversion-12c-2537297.pdf>

NOTE: Setting LOG_BUFFER to 256MB will aid asynchronous redo transport in reading redo from memory instead of having to do disk I/Os from the online redo logs.

2.6. Enable Archive Log Mode

As user sys on Site A's application database (APDDB):

```
alter system set log_archive_dest_1=
  'location=USE_DB_RECOVERY_FILE_DEST valid_for=(ALL_LOGFILES,ALL_ROLES)
  DB_UNIQUE_NAME=APPDB' scope=both sid='*';
shutdown immediate
startup mount
alter database archivelog;
alter database open;
alter system archive log current;
```

Changing a database's archive log mode requires a database restart.

Procedure: Secondary Site Setup

The step involves the deployment of the secondary site. It assumes that the steps performed in the section 'Procedure: Ready Initial Site For Data Guard' have been completed.

1. Establish the Oracle Cloud Services on the Secondary Site

1.1 Subscribe to Oracle Database Cloud Service at DR Location

The decision was made to locate the secondary site in Ashburn, and a subscription to the [Oracle Database Cloud Service](#)¹⁵ in that location was secured.

Refer to the [DBaaS cloud documentation](#) for subscription, activation and service creation details¹⁶.

1.2 Create Storage Containers for Database and JCS Backups

The following commands were executed to get the authentication token:

```
export SCSDOMAIN=domainb
export SCSUSER=joe@acme.com
export SCSPASSWORD=acme1234
curl -v -X GET \
  -H "X-Storage-User: Storage-${SCSDOMAIN}:${SCSUSER}" \
  -H "X-Storage-Pass: ${SCSPASSWORD}" \
  https://${SCSDOMAIN}.storage.oraclecloud.com/auth/v1.0
```

The following command set the authentication token:

```
export SCSAUTH=<token returned from previous step>
```

The following commands were executed to create the storage containers:

```
curl -v -X PUT \
  -H "X-Auth-Token: ${SCSAUTH}" \
  https://${SCSDOMAIN}.storage.oraclecloud.com/v1/Storage-${SCSDOMAIN}/jcsbContainer
curl -v -X PUT \
  -H "X-Auth-Token: ${SCSAUTH}" \
  https://${SCSDOMAIN}.storage.oraclecloud.com/v1/Storage-${SCSDOMAIN}/infrabContainer
curl -v -X PUT \
  -H "X-Auth-Token: ${SCSAUTH}" \
  https://${SCSDOMAIN}.storage.oraclecloud.com/v1/Storage-${SCSDOMAIN}/appdbbContainer
```

¹⁵ http://docs.oracle.com/cloud/latest/dbscs_dbaas/index.html

¹⁶ http://docs.oracle.com/cloud/latest/dbscs_dbaas/CSDBI/GUID-1A380E9C-6DE2-4042-8A31-B43A0081B194.htm#CSDBI3312

1.3. Create an Oracle Database Service Instance for the Application

The following parameters were supplied when creating the DR DBCS instance from the Database Cloud Service – Service Console after clicking the “Create Service” button:

TABLE 8 CASE STUDY DBCS PARAMETERS

Page	Category	Option	Value
Subscription Type		Service Level	Oracle Database Cloud Service
		Billing Frequency	Monthly
Release		Software Release	12.1.0.2
Edition		Software Edition	Enterprise Edition – Extreme Performance
Service Details	Service Configuration	Service Name	appdbb
		Shape	OC4 - 2 OCPU, 15 GB RAM
		Timezone	(UTC) Coordinated Universal Time(UTC)
		SSH Public Key	key.pub
	Database Configuration	Usable Database Storage (GB)	25
		Administration Password	*****
		DB Name	APPDB
		PDB Name	MEDREC
		Character Set	AL32UTF8
		National Character Set	AL16UTF16
		Database Clustering with RAC	Unchecked
		Standby Database with Data Guard	Unchecked
		Enable Oracle Goldengate	Unchecked
		Include “Demos” PDB	Unchecked
	Backup and Recovery Configuration	Backup Destination	Both Cloud Storage and Local Storage
		Cloud Storage Container	Storage-domainb/appdbbContainer
		Cloud Storage User Name	joe@acme.com
		Cloud Storage Password	Acme1234#

1.4. Synchronize Database Patch Levels

The following command was executed as user oracle on both database hosts (appdba and dbb) and the results were compared and found to be the same.

```
$_ORACLE_HOME/OPatch/opatch lspatches
```

The Oracle Home for the primary database must be the same Oracle patchset and have the same patches as the standby database. Compare the output of ‘\$_ORACLE_HOME/OPatch/opatch lspatches’ between the two sites and apply those that are missing on each side.

1.5. Delete or Repurpose the Default Database

The default database created in the application database instance on Site B was deleted using the following command:

```
dbca -silent -deleteDatabase -sourceDB APPDB -sysDBAUserName sys -sysDBAPassword  
Acme1234#  
  
cd /u03/app/oracle/fast_recovery_area  
rm -rf APPDB
```

A default database instance is automatically created when you create a Oracle Database Cloud Service. This database cannot be used as a Data Guard standby database. It can be deleted or reused for a different purpose.

1.6. Configure Network Security

The following network configuration was created from the *Oracle Compute Cloud Service – Service Console – Network Tab* on Site A to give access to Site B:

TABLE 9 SITE 'A' NETWORK CONFIGURATION

Type	Name	Option	Value
Security IP List	appdbb	IP List	< appdbb's public IP address>
Security Rule	appdbb2appdba	Status	Enabled
		Security Application	appdba/db_1/ora_dblistener
		Source Security IP List	appdbb (created above)
		Destination Security List	appdba/db_1/ora_db

The following network configuration was performed from the *Oracle Compute Cloud Service – Service Console – Network Tab* on Site B to give access to Site A:

TABLE 10 SITE 'B' NETWORK CONFIGURATION

Type	Name	Option	Value
Security IP List	appdba	IP List	< appdba's public IP address>
Security Rule	appdba2appdbb	Status	Enabled
		Security Application	appdbb/db_1/ora_dblistener
		Source Security IP List	appdba (created above)
		Destination Security List	appdbb/db_1/ora_db

To function correctly, Data Guard requires Oracle Net communication between the database servers on the primary and standby site. The application servers and possibly other servers may also need access to the database through Oracle Net. All other servers should not be allowed access.

2. Install Grid Infrastructure and Configure Oracle Restart

2.1. Install Grid Infrastructure Software

The Oracle Grid Infrastructure software was installed on the Site B's database host (appdba) in the same way as Site A – see section: 1.1. Install Grid Infrastructure Software.

2.2. Configure the Database Listener in Oracle Restart

As user oracle on Site B's application database host (appdbb):

```
lsnrctl stop listener
cp $ORACLE_HOME/network/admin/listener.ora \
  /u01/app/12.1.0/grid/network/admin/listener.ora
srvctl add listener -listener listener -oraclehome /u01/app/12.1.0/grid
srvctl start listener
```

3. Configure the Operating System and Database

3.1 Prepare the Source File

The scripts used to configure and duplicate the database, and configure Data Guard broker, share a bash source file. The file `~/script.env` was created in the user oracle on Site A's database host as follows:

```
export passwd='<sys password of the app database>'
export DB_NAME='appdb'
export A_DBNM='<Site A's database unique name; in this case study it is 'APPDB'>'
export A_PUB_IP='<Site A's database public IP address>'
export A_PRIV_HOSTNAME='<Site A's database private HOSTNAME >'
export A_PORT='1521'
export A_DB_DOMAIN='<Site A's DB_DOMAIN>'
export B_DBNM=' Site B's database unique name; in this case study it is 'APPDBB''
export B_PUB_IP='<Site B's database public IP address>'
export B_PRIV_HOSTNAME='<Site B's database private HOSTNAME>'
export B_PORT='1521'
export B_DB_DOMAIN='<Site A's DB_DOMAIN>'
export A_FILE_LOC='/u02/app/oracle/oradata'
export A_RECOV_LOC='/u03/app/oracle/fast_recovery_area'
export A_REDO_LOC='/u04/app/oracle/redo'
export B_FILE_LOC='/u02/app/oracle/oradata'
export B_RECOV_LOC='/u03/app/oracle/fast_recovery_area'
export B_REDO_LOC='/u04/app/oracle/redo'
export B_BASE='/u01/app/oracle'
```

The file was copied to user oracle on Site B's database host.

3.2. Set TCP Socket Maximum Sizes

The following commands were run as user root on Site A's database host (appdba):

```
sysctl -w net.core.rmem_max=10485760
sysctl -w net.core.wmem_max=10485760
```

The /etc/sysctl.conf file was also edited to reflect the changes so that they would be preserved during an instance reboot.

As documented in the Oracle Public Cloud documentation, you can gain root access to a provisioned cloud instance by connecting to the opc user and then using the sudo command:

```
sudo -s
```

3.3. Configure Oracle Net Encryption

The wallets were copied from appdba to appdbb.

On appdba as user oracle:

```
cd /u01/app/oracle/admin
tar -czf /tmp/APPDB.tgz APPDB
```

The /tmp/APPDB.tgz file was copied from appdba to appdbb.

On appdbb as user oracle:

```
cd /u01/app/oracle/admin
tar -xzf /tmp/APPDB.tgz
```

The following parameters were added to the \$ORACLE_HOME/network/admin/sqlnet.ora file on Site A's and Site B's application database hosts:

```
SQLNET.ENCRYPTION_CLIENT = requested
SQLNET.ENCRYPTION_TYPES_CLIENT = (AES256, AES192, AES128)
```

For the detailed procedure to create and disperse wallets, refer to Appendix F – Creating and Dispersing Wallets.

In the case of default Database created in the DBCS instance, the wallet creation was done as part of the DBCS instance creation.

3.4. Configure TNS Entries for Redo Transport

The following entries were configured in the tnsnames.ora file in the database hosts of both sites (adddba and appdbb). Executed as user oracle on both database hosts:

```
. ~/script.env
cat >> $ORACLE_HOME/network/admin/tnsnames.ora <<EOF
${A_DBNM} =
  (DESCRIPTION =
    (SDU=65536)
    (RECV_BUF_SIZE=10485760)
```

```

        (SEND_BUF_SIZE=10485760)
        (ADDRESS = (PROTOCOL = TCP)(HOST = ${A_PUB_IP})(PORT = 1521))
        (CONNECT_DATA =
          (SERVER = DEDICATED)
          (SERVICE_NAME = ${A_DBNM}.${A_DB_DOMAIN})
        )
      )
    )

    ${B_DBNM} =
      (DESCRIPTION =
        (SDU=65536)
        (RECV_BUF_SIZE=10485760)
        (SEND_BUF_SIZE=10485760)
        (ADDRESS = (PROTOCOL = TCP)(HOST = ${B_PUB_IP})(PORT = 1521))
        (CONNECT_DATA =
          (SERVER = DEDICATED)
          (SERVICE_NAME = ${B_DBNM}.${B_DB_DOMAIN})
        )
      )
    )
  )
EOF

```

The tsnames.ora file was checked and the duplicate APPDB alias was removed.

Entries for each database are needed in both primary and standby tsnames.ora files for proper redo transport.

NOTE: The primary database may already have a TNS entry in the Site-A DBCS instance tsnames.ora with a server name for the HOST. In this case simply change the server name in that entry to use the IP address for the host instead.

NOTE: IP addresses are used since there is no DNS to resolve server names to IP addresses.

3.5. Configure Static Listeners

The following was executed as user oracle on Site B's database host (appdbb):

```

. ~/script.env
cat >> /u01/app/12.1.0/grid/network/admin/listener.ora <<EOF
SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (GLOBAL_DBNAME = ${B_DBNM}.${B_DB_DOMAIN})
      (ORACLE_HOME = /u01/app/oracle/product/12.1.0/dbhome_1)
      (SID_NAME = ${DB_NAME})
    )
  )
EOF

```

The following was executed as the oracle user on Site B's database host (appdbb):

```
/u01/app/12.1.0/grid/bin/lsnrctl reload
```

A static listener is needed for initial instantiation of a standby database. The static listener enables remote connection to an instance while the database is down in order to start a given instance. The following steps will aid in configuring the static listener on the cloud VM. See MOS 1387859.1 for additional details.

Static "_DGMGRL" entries are no longer needed as of Oracle Database 12.1.0.2 in Oracle Data Guard Broker configurations that are managed by Oracle Restart, RAC One Node or RAC as the Broker will use the clusterware to restart an instance.

For 11.2 configurations, a static listener is also required for Data Guard Broker.

Site A:

```
SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (GLOBAL_DBNAME = <${A_DBNM}_DGMGRL)
      (ORACLE_HOME = /u01/app/oracle/product/12.1.0/dbhome_1)
      (SID_NAME = ${DB_NAME})
    )
  )
```

Site B:

```
SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (GLOBAL_DBNAME = <${B_DBNM})
      (ORACLE_HOME = /u01/app/oracle/product/12.1.0/dbhome_1)
      (SID_NAME = ${DB_NAME})
    )
    (SID_DESC =
      (GLOBAL_DBNAME = <${B_DBNM}_DGMGRL)
      (ORACLE_HOME = /u01/app/oracle/product/12.1.0/dbhome_1)
      (SID_NAME = ${DB_NAME})
    )
  )
```

4. Instantiate the Standby Database

Database instances created prior to July 2016 require manual configuration of matching mtu settings at both Site A and Site B by executing the following commands as user root:

```
ip link set eth0 mtu 1500
echo "ip link set eth0 mtu 1500" >> /etc/rc.local
```

Do not perform this step if your database instance was created in July 2016 or later.

4.1. Create the Auxiliary Database Password and Parameter Files

As user oracle on Site B's database host (appdbb):

```
. ~/script.env
cd $ORACLE_HOME/dbs
$ORACLE_HOME/bin/orapwd file='$ORACLE_HOME/dbs/orapw${DB_NAME}' password=${passwd}
force=y
cat > /tmp/aux.pfile << EOF
db_name=${DB_NAME}
db_unique_name=${B_DBNM}
db_domain=${B_DB_DOMAIN}
sga_target=800M
EOF
```

The sga size in this pfile does not need to match the primary; a new spfile will be generated during the instantiation process with the correct values. This auxiliary pfile is only used to start the auxiliary instance.

4.2. Start the Auxiliary Instance

The auxiliary instance was started as user oracle on site B's database host with the following command:

```
. ~/script.env
export ORACLE_SID=${DB_NAME}
sqlplus "/ as sysdba" <<EOF
startup nomount pfile='/tmp/aux.pfile'
EOF
```

The auxiliary instance is a temporary instance (just memory structures) that enables communication between the primary and standby sites. Think of it as a place-holder until the primary is able to copy its files making the auxiliary instance a database.

4.3. Create Standby Database Audit and File Locations

The following command was executed as user oracle on site B's database host:

```
. ~/script.env
mkdir -p ${B_BASE}/admin/${B_DBNM}/adump
mkdir ${B_FILE_LOC}/${B_DBNM}
mkdir ${B_FILE_LOC}/${B_DBNM}/pdbseed
mkdir ${B_FILE_LOC}/${B_DBNM}/MEDREC
mkdir ${B_RECOV_LOC}/${B_DBNM}
mkdir ${B_REDO_LOC}/${B_DBNM}
```

4.4. Create the Standby Database

The standby database was instantiated by duplicating from the primary database.

The following script was run as user oracle from site B's database host to create the RMAN duplicate script:

```
. ~/script.env
cat >/tmp/${B_DBNM}.rman <<EOF
connect target sys/${passwd}@${A_DBNM}
connect auxiliary sys/${passwd}@${B_DBNM}
run {
allocate channel prmy1 type disk;
allocate auxiliary channel stby1 type disk;
allocate auxiliary channel stby2 type disk;
allocate auxiliary channel stby3 type disk;
allocate auxiliary channel stby4 type disk;
duplicate target database for standby from active database
spfile
PARAMETER_VALUE_CONVERT= '${A_DBNM}', '${B_DBNM}'
set db_unique_name='${B_DBNM}'
set control_files='${B_FILE_LOC}/${B_DBNM}/control01.ctl',
  '${B_RECOV_LOC}/${B_DBNM}/control02.ctl'
set audit_file_dest='${B_BASE}/admin/${B_DBNM}/adump'
set local_listener='(ADDRESS=(PROTOCOL=tcp)(HOST=${B_PRIV_HOSTNAME})(PORT=${B_PORT}))'
set fal_server='${A_DBNM}'
set log_file_name_convert='${A_REDO_LOC}', '${B_REDO_LOC}/${B_DBNM}',
  '${A_REDO_LOC}/${A_DBNM}', '${B_REDO_LOC}/${B_DBNM}'
set db_file_name_convert='${A_FILE_LOC}/${A_DBNM}',
  '${B_FILE_LOC}/${B_DBNM}'
set db_create_file_dest='${B_FILE_LOC}'
set db_create_online_log_dest_1='${B_REDO_LOC}'
set db_recovery_file_dest='${B_RECOV_LOC}'
set db_recovery_file_dest_size='40G'
set diagnostic_dest='${B_BASE}'
set db_domain='${B_DB_DOMAIN}'
set STANDBY_FILE_MANAGEMENT='AUTO'
section size 10M
dorecover
;
}
EOF
```

*NOTE: If files are spread across multiple mount points additional entries will be required for db_file_name_convert. For each different path to the data files a '**<primary path>/\${PRIMARY_DBNM}**', '**\${secondary path}/\${SECONDARY_DBNM}**' pair is required. Retrieve the data file paths from the primary database with 'select name from v\$datafile;'*

*NOTE: If log files are spread across multiple mount points additional entries will be required for log_file_name_convert. For each different path to the log files a '**<primary path>/\${PRIMARY_DBNM}**', '**\${secondary path}/\${SECONDARY_DBNM}**' pair is required. Retrieve the log file paths from the primary database with 'select member from v\$logfile;'*

The duplicate script was executed as user oracle from site B's database host (appdbb):

```
. ~/script.env
rman << EOF
@/tmp/${B_DBNM}.rman
EOF
```

During the duplication process the following error was observed and can be ignored:

```
ORA-38757: Database must be mounted and not open to FLASHBACK.
```

The appropriate file name conversion parameters were set on the primary database by executing the following as oracle from site A's database host (appdba):

```
. ~/script.env
sqlplus / as sysdba <<EOF
alter system set log_file_name_convert='${B_REDO_LOC}/${B_DBNM}', '${A_REDO_LOC}',
  '${B_REDO_LOC}/${B_DBNM}', '${A_REDO_LOC}/${A_DBNM}'
scope=spfile sid='*';
alter system set db_file_name_convert='${B_FILE_LOC}/${B_DBNM}',
  '${A_FILE_LOC}/${A_DBNM}' scope=spfile sid='*';
EOF
```

4.5. Configure the Database in Oracle Restart

As user oracle on the Site A's database host (appdba):

```
. ~/script.env
srvctl add database -db ${B_DBNM} -dbname ${DB_NAME} \
  -oraclehome $ORACLE_HOME -startoption open \
  -instance APPDB
```

4.6. Enable Flashback on New Standby Database

The broker was configured by executing the following as oracle from site A's database host (appdba):

```
. ~/script.env
sqlplus -s sys/${passwd}@${B_DBNM} as sysdba <<EOF
alter database flashback on;
EOF
```


4.7. Configure Data Guard Broker

The broker was configured by executing the following as oracle from site A's database host (appdba):

```
. ~/script.env
sqlplus -s sys/${passwd}@${A_DBNM} as sysdba <<EOF
  alter system set dg_broker_start=FALSE;
  alter system set dg_broker_config_file1='${A_FILE_LOC}/${A_DBNM}/dr1.dat';
  alter system set dg_broker_config_file2='${A_RECOV_LOC}/${A_DBNM}/dr2.dat';
  alter system set dg_broker_start=TRUE;
EOF

sqlplus -s sys/${passwd}@${B_DBNM} as sysdba <<EOF
  alter system set dg_broker_start=FALSE;
  alter system set dg_broker_config_file1='${B_FILE_LOC}/${B_DBNM}/dr1.dat';
  alter system set dg_broker_config_file2='${B_RECOV_LOC}/${B_DBNM}/dr2.dat';
  alter system set dg_broker_start=TRUE;
EOF

sleep 30

create configuration 'DGconfig' as primary database is ${A_DBNM}
dgmgrl sys/${passwd}@${A_DBNM} <<EOF
create configuration 'DGconfig' as primary database is ${A_DBNM}
  connect identifier is ${A_DBNM};
add database ${B_DBNM} as connect identifier is ${B_DBNM};
rem !!The following 2 commands must be uncommented for 11g databases!!
rem edit database ${A_DBNM} set property RedoRoutes='(LOCAL:${B_DBNM} ASYNC)';
rem edit database ${B_DBNM} set property RedoRoutes='(LOCAL:${A_DBNM} ASYNC)';
EDIT CONFIGURATION SET PROTECTION MODE AS MaxPerformance;
enable configuration;
exit
EOF
```

The steps followed in the case study assumed that the primary database was not part of an existing Data Guard broker configuration. If there is an existing broker configuration for the database the administrator will need to remove it or know how to add the new standby database to the configuration.

A return value other than 'NOCONFIG' for the following query implies an existing broker configuration.

```
select decode(count(1),0,'NOCONFIG') from v$DG_BROKER_CONFIG;
```

4.8. Perform Data Guard Health Check

See Appendix B – Data Guard Health Checks.

4.9. Enable Run Time Monitoring

See Appendix C – Data Guard Run-Time Monitoring Queries.

5. Install and Test the Application

5.1. Convert Standby to Snapshot Standby

The following commands were executed from the Data Guard broker command line interface (dgmgrl) as user system logging in on the primary application database (APPDB):

```
convert database APPDB to snapshot standby;
```

5.2. Create Security Rule between JCS Instance and Application DBCS Instance at Site-B

A security rule was created between the JCS instance (jcsb) and APPDB DBCS DB Listener (security application)

It is recommended that the application to database traffic at each site is routed through the internal IP addresses using internal hostnames, instead of using the public IP addresses of DBCS and JCS instance. Refer Oracle Cloud Compute Documentation for more details for configuring security rules in Oracle Public Cloud

5.3. Install, Configure and Test the Application on New Site

The following parameters were supplied when creating Site B's JCS Infrastructure DBCS instance from the Database Cloud Service – Service Console after clicking the “Create Service” button:

TABLE 11 SITE 'B' DBCS PARAMETERS

Page	Category	Option	Value
Subscription Type		Service Level	Oracle Database Cloud Service
		Billing Frequency	Monthly
Release		Software Release	12.1.0.2
Edition		Software Edition	Standard Edition
Service Details	Service Configuration	Service Name	infrab
		Shape	OC3 - 1 OCPU, 7.5 GB RAM
		Timezone	(UTC) Coordinated Universal Time(UTC)
		SSH Public Key	key.pub
	Database Configuration	Usable Database Storage (GB)	25
		Administration Password	Acme1234#
		DB Name	INFRAB
		PDB Name	INFRA
		Character Set	AL32UTF8
		National Character Set	AL16UTF16
		Database Clustering with RAC	Unchecked
		Standby Database with Data Guard	Unchecked
		Enable Oracle Goldengate	Unchecked
		Include “Demos” PDB	Unchecked
	Backup and Recovery Configuration	Backup Destination	Both Cloud Storage and Local Storage
		Cloud Storage Container	Storage-domainb/infrabContainer
		Cloud Storage User Name	joe@acme.com
		Cloud Storage Password	Acme1234#

The following parameters were supplied when creating Site B's JCS instance from the Java Cloud Service – Service Console after clicking the “Create Service” button:

TABLE 12 SITE ‘B’ JCS PARAMETERS

Page	Category	Option	Value
		Subscription Type	Service Level
			Oracle Java Cloud Service
			Billing Frequency
			Monthly
		Software Release	12.2.1.0
		Software Edition	Enterprise Edition
Service Details	Service Configuration	Service Name	jesb
		Domain Partitions	0
		Shape	OC4 - 2 OCPU, 25 GB RAM
		SSH Public Key	key.pub
	WebLogic	Username	weblogic
		Password	Acme1234#
		Deploy Sample Application	Unchecked
	Database Configuration	Name	infraa
		PDB Name	INFRA
		Administrator User Name	SYS
		Password	Acme1234#
	Backup and Recovery Configuration	Cloud Storage Container	Storage-domainb/jcsbContainer
		Cloud Storage User Name	joe@acme.com
		Cloud Storage Password	Acme1234#

In the interests of brevity, the detailed steps that were followed to install and test the MEDREC application are omitted.

If the applications to be deployed on the JCS instance have JMS and JTA requirements, it is recommended to externalize all those runtime artifacts into the database so that those data are protected from any disaster recovery event using Oracle Data Guard DR configuration.

For more information about database persistent storage, see [Configuring WLS JMS with a Database Persistent Store](#) and [Configuring Database Stores to Persist Transaction Logs](#).

5.4. Shutdown the Application on New Site

In the interests of brevity, the detailed steps that were followed to shutdown the MEDREC application are omitted.

5.5. Convert Snapshot Standby to Physical Standby

The following commands were executed from the Data Guard broker command line interface (dgmgrl) as user system logging in on the primary application database (APPDB):

```
convert database APPDB to physical standby;
```

Procedure: Site Test

These steps may be applied to site A or site B, whichever is in standby mode.

As an example from our case study, we show a site test performed on site B.

1. Convert Standby to Snapshot Standby

The following commands were executed from the Data Guard broker command line interface (dgmgrl) as user system logging in on the primary application database (APPDB):

```
convert database APPDB to snapshot standby;
```

2. Start and Test the Application

In the interests of brevity, the detailed steps that were following to start and test the MEDREC application are omitted.

Procedure: Site Test to Standby

These steps may be applied to site A or site B, whichever is currently on Site Test mode.

As an example from our case study, we show the procedure performed on site B.

1. Shutdown Application Under Test

In the interests of brevity, the detailed steps that were following to shutdown the MEDREC application are omitted.

2. Convert Standby to Physical Standby

The following commands were executed from the Data Guard broker command line interface (dgmgrl) as user system logging in on the primary application database (APPDB):

```
convert database APPDB to physical standby;
```

Procedure: Switchover

The switchover procedure can be used to switchover from site A to site B or from site B to site A. In this case, we show an example from our case study of a switchover that was performed from site A to site B.

As a starting point, we assume that the application is up and running at site A and the database server at site A is available.

The following steps were performed:

1. Shutdown the Application

In the interests of brevity, the detailed steps that were following to shutdown the MEDREC application are omitted.

2. Perform Data Guard Switchover

The following commands were executed from the Data Guard broker command line interface (dgmgrl) as user system logging in on the secondary application database (APPDB):

```
switchover to APPDB;
```

At any time you can manually execute a Data Guard switchover (planned event) or failover (unplanned event). Customers may also choose to automate Data Guard failover by configuring Fast-Start failover. Switchover reverses the roles of the databases in a Data Guard configuration – the standby becomes primary and the original primary becomes a standby database. Refer to [Oracle MAA Best Practices](#)¹⁷ for additional information on Data Guard role transitions.

3. Startup the Application on New Primary Site

In the interests of brevity, the detailed steps that were following to startup the MEDREC application are omitted.

Procedure: Failover

The failover procedure can be used to failover from site A to site B or from site B to site A. In this case, we show an example of a failover that was performed from site A to site B in our case study.

As a starting point, we assume that there has been a major failure at site A and the database server at site A is unavailable and as a result a switchover will is not possible.

The following steps were performed:

1. Perform Data Guard Failover

The following commands were executed from the Data Guard broker command line interface (dgmgrl) as user system logging in on the secondary application database (APPDBB):

```
failover to APPDBB;
```

2. Startup the Application on New Primary Site

In the interests of brevity, the detailed steps that were following to startup the MEDREC application are omitted.

Procedure: Reinstate Standby

The reinstate procedure is used to reinstate a former primary database as a standby database after failover has been performed. The same procedure can be used on site A or site B. In this case, we show an example from our case study of a reinstate on site A after a failover to site B.

1. Startup Database Mount

The following commands were executed as user sys on site A's application database (appa):

```
shutdown abort  
startup mount
```

2. Reinstate the Standby Database

The following commands were executed from the Data Guard broker command line interface (dgmgrl) as user system logging in on site A's application database (APPDB):

```
reinstate database APPDB;
```

¹⁷ <http://www.oracle.com/technetwork/database/availability/maa-roletransitionbp-2621582.pdf>

Appendix B – Data Guard Health Checks

After the standby is instantiated, a health check should be performed to ensure the Data Guard databases (primary and standby) are compliant with Oracle MAA best practices. It is also advisable to perform the health check on a monthly basis as well as before and after database maintenance. There are several methods for checking the health of a Data Guard configuration:

Oracle MAA Scorecard

Oracle provides several automated health check tools that can be downloaded from My Oracle Support specific for the type of hardware platform:

- » [ORAchk](#) applicable to generic platform (suitable for Database Cloud Service)¹⁸
- » [exachk](#) applicable to Exadata Database Machine (suitable for Exadata Cloud Service)¹⁹

Each of the automated checks include an Oracle MAA Scorecard that reports on a number of key Data Guard configuration best practices in addition to many other checks.

Oracle strongly recommends the use of these automated tools for comprehensive health check of not only the Data Guard configuration but the system as a whole. The health checks are regularly updated with current information. Be sure to download the latest version of the health checks applicable to your platform.

Data Guard Specific Queries (Applicable from Oracle Database 11g onward)

A set of Data Guard specific queries are provided in Appendix C – Data Guard Run-Time Monitoring Queries along with sample output that can be used to validate the health of the Data Guard configuration.

Data Guard VALIDATE DATABASE (Applicable from Oracle Database 12c onward)

The Data Guard Broker VALIDATE DATABASE command is highly recommended for the most comprehensive Data Guard specific health check. VALIDATE DATABASE performs an extensive configuration check and validates the configuration's readiness for switchover or failover.

Example:

```
DGMGR> validate database APPDBB;
```

```
Database Role:   Physical standby database  
Primary Database: pri
```

```
Ready for Switchover: Yes  
Ready for Failover:  Yes (Primary Running)
```

See the [Data Guard broker documentation](#) for more information on the extensive checks performed by the VALIDATE DATABASE command²⁰.

¹⁸ <https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=1268927.2>

¹⁹ <https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=1070954.1>

Appendix C – Data Guard Run-Time Monitoring Queries

As of the release of this document monitoring the standby database in a Data Guard configuration must be done with queries and scripts. There are a few basic recommendations for monitoring which include:

- 1) Validate redo transport is functioning and there is no transport lag- This means redo is being shipped to the standby in a timely manner.
- 2) Validate redo apply is functioning and there is no apply lag- This means redo is being applied in a timely manner as it is received.

Each of these things can be monitored via Data Guard broker or SQL*Plus queries.

Monitoring with Data Guard Broker

Monitoring with Data Guard should focus on Database Status of the show database <standby db_unique_name> command. A Database Status of SUCCESS implies there are no issues.

```
DGMGRL> show database APPDBB;  
DGMGRL>  
Database - appdbb
```

```
Role:                PHYSICAL STANDBY  
Intended State:      APPLY-ON  
Transport Lag:       0 seconds (computed 2 seconds ago)  
Apply Lag:           0 seconds (computed 2 seconds ago)  
Average Apply Rate: 1.00 KByte/s  
Real Time Query:     ON  
Instance(s):  
  APPDB
```

```
Database Status:  
SUCCESS
```

When there are issues with transport or apply functionality the status will read WARNING or ERROR.

Setting the ApplyLagThreshold and TransportLagThreshold properties on the standby database enables monitoring lag in the same method (Database Status). Each of these properties is expressed in seconds. By setting these properties to a non-zero value, when the value is exceeded the Database Status will be changed to WARNING.

Monitoring with SQL*Plus

To Validate Redo Transport and Redo Apply

The basics of Data Guard are that redo is transported from the primary and then applied on the standby. The queries below can be used to validate these functions are working properly.

PRIMARY DATABASE QUERY: Current state of Data Guard Transport:

To validate that the primary database is shipping redo to the standby execute the following query on the primary database. A STATUS of VALID is expected. ERROR will be populated with additional information if the STATUS

20 <http://docs.oracle.com/database/121/DGBKR/dbresource.htm#DGBKR3855>

does not equal VALID. gv\$archive_dest_status should be used in case of a RAC primary database in order to validate each primary instance.

```
SQL> select sysdate,status,error from v$archive_dest_status where type='PHYSICAL' ;
```

SYSDATE	STATUS	ERROR
13-mar-2015 10:42:10	VALID	

STANDBY DATABASE QUERY: Current State of the Standby Apply

A standby can be receiving redo without applying it. Therefore, the query above is just once piece of the validation. To validate that redo apply is running on the standby database execute the following query. A RECOVERY_MODE of 'MANAGED REAL TIME APPLY' implies that redo is applied as it is received. GAP_STATUS indicates whether there is a gap in redo transport from the primary.

```
SQL> select sysdate,database_mode,recovery_mode, gap_status from v$archive_dest_status where type='PHYSICAL' ;
```

SYSDATE	DATABASE_MODE	RECOVERY_MODE	GAP_STATUS
13-mar-2015 10:43:44	OPEN_READ-ONLY	MANAGED REAL TIME APPLY	NO GAP

Data Guard Performance (Standby Lag)

Redo that has been successfully written but not applied to the standby is called an apply lag. An apply lag indicates there are insufficient resources at the standby, often due to I/O or CPU contention. Apply lag does not indicate potential data loss exposure since the redo is present at the standby it can be applied prior to failover. However, when there is a transport lag, meaning redo is not being received and written to the standby redo logs in a timely fashion, there is a potential to lose the unwritten data should the primary be lost.

Network latency is often the cause of transport lag but can also be the result of insufficient resources at the standby.

The following queries will help identify the presence and size of either lag.

STANDBY DATABASE QUERY: Monitor for Apply and Transport Time Lags

Run the following query to identify an apply or transport lag. A VALUE of '+00 00:00:00' indicates no lag.

```
select name,value,time_computed,datum_time from v$dataguard_stats where name like '%lag%';
```

NAME	VALUE	TIME_COMPUTED	DATUM_TIME
transport lag	+00 00:00:00	06/18/2015 09:26:29	06/18/2015 09:26:27
apply lag	+00 00:00:00	06/18/2015 09:26:29	06/18/2015 09:26:27

STANDBY DATABASE QUERY: Monitor the Size of a Transport Lag.

The following query can be used to determine how far behind in blocks the standby is from the primary when a transport lag is present.

```
SQL> select t.thread#,t.LAST_REDO_SEQUENCE#,t.LAST_REDO_BLOCK, s.thread#,
```



```
s.sequence#, s.block# from v$thread t, gv$managed_standby s where
s.process='LNS' and t.LAST_REDO_SEQUENCE#=s.sequence#;
```

THREAD#	LAST_REDO_SEQUENCE#	LAST_REDO_BLOCK	THREAD#	SEQUENCE#	BLOCK#
1	453	482	1	453	482
2	406	786	2	406	786

STANDBY DATABASE QUERY: Monitor Standby Apply Process

```
SQL> select sysdate,process,status,thread#,sequence#,block# from
gv$managed_standby where status!='IDLE';
```

SYSDATE	PROCESS	STATUS	THREAD#	SEQUENCE#	BLOCK#
13-mar-2015 10:58:31	ARCH	CLOSING	2	403	1
13-mar-2015 10:58:31	ARCH	CONNECTED	0	0	0
13-mar-2015 10:58:31	ARCH	CLOSING	1	449	1
13-mar-2015 10:58:31	ARCH	CLOSING	2	402	4096
13-mar-2015 10:58:31	MRP0	APPLYING_LOG	2	404	1614

STANDBY DATABASE QUERY: Monitor Recovery (advanced monitoring)

The following query can be used to gather valuable information about the apply process such as apply rates and apply lag. gv\$recovery_progress keeps a history of each recovery invocation for the life of the instance(until instance is restarted), this query select only metrics for the most recent invocation.

```
SQL> select start_time, item, units, sofar from gv$recovery_progress where
START_TIME in (select max(START_TIME) from v$recovery_progress);
```

START_TIM	ITEM	UNITS	SO FAR
18-JUN-15	Active Apply Rate	KB/sec	23
18-JUN-15	Average Apply Rate	KB/sec	1
18-JUN-15	Maximum Apply Rate	KB/sec	24
18-JUN-15	Redo Applied	Megabytes	0
18-JUN-15	Last Applied Redo	SCN+Time	0
18-JUN-15	Active Time	Seconds	42
18-JUN-15	Elapsed Time	Seconds	576
18-JUN-15	Standby Apply Lag	Seconds	1

Appendix D – Health Check Queries

As noted in the section [Data Guard Health Check](#) a periodic check of best practices should be executed on the Data Guard configuration to ensure best practices are followed. It is strongly recommended that the automated tools outlined in the 'Data Guard Health Check' section be used as they are updated regularly. However, the following queries can be run in absence of the tools. Also refer to Monitoring a Data Guard Configuration (Doc ID 2064281.1)

Standby Redo Logs(SRL) Health Check

There are a few different best practices for SRLs

1. Number of SRL groups should equal the number of Online Redo Log(ORL) groups +1 for each thread. Run the following queries on the primary and standby database.

```
SQL> select thread#,count(group#)from v$log group by thread#;
```

```
  THREAD#  COUNT(GROUP#)
-----  -
          1              4
```

```
SQL> select thread#,count(group#)from v$standby_log group by thread#;
```

```
  THREAD#  COUNT(GROUP#)
-----  -
          1              5
```

2. All log files (ORL & SRL) should be of the same size. Run the following queries, which should return the same single value.

```
SQL> select distinct bytes from v$log;
```

```
  BYTES
-----
4294967296
```

```
SQL> select distinct bytes from v$standby_log;
```

```
  BYTES
-----
4294967296
```

3. SRL groups should have only one member. Run the following query to verify that the count column is one for all groups.

```
SQL> select group#,count(member) from v$logfile where type='STANDBY'
group by group#;
```

```
  GROUP#  COUNT(MEMBER)
-----  -
          5              1
          6              1
```

7	1
8	1
9	1

Flashback Database and Forced Logging

It is recommended that flashback database be enabled for easy reinstatement from failover operations. Run the following query to verify, the result should be 'YES'.

```
SQL> select flashback_on from v$database;

FLASHBACK_ON
-----
YES
```

It is recommended that force logging be enabled to prevent the loss of data when nologging queries are run on the primary database. Verify with the following query.

```
SQL> select force_logging from v$database;

FORCE_LOGGING
-----
YES
```

Additional parameters

Check that the following parameters are set accordingly. DB_BLOCK_CHECKING can be left at FALSE if it is deemed too much of a performance impact on recovery.

```
SQL> show parameter checking

NAME                                TYPE          VALUE
-----
db_block_checking                    string        MEDIUM

SQL> show parameter checksum

NAME                                TYPE          VALUE
-----
db_block_checksum                    string        FULL

SQL> show parameter lost_write

NAME                                TYPE          VALUE
-----
db_lost_write_protect                string        typical

SQL> show parameter disk_asynch_io

NAME                                TYPE          VALUE
-----
disk_asynch_io                       boolean       TRUE
```

LOG_ARCHIVE_MAX_PROCESSES

The setting for LOG_ARCHIVE_MAX_PROCESSES should be equal to the number of remote destinations (likely 1) Plus the number of threads/instances (greater than 1 for RAC only). The following queries can be used to help determine the current and proper setting.

```
SQL> show parameter log_archive_max_processes
```

NAME	TYPE	VALUE
log_archive_max_processes	integer	2

```
SQL> select ((select count(1) from v$archive_dest where TARGET='STANDBY') +  
(select count(distinct thread#) from v$log)) PROPER_LOG_ARCHIVE_MAX_PROC from  
dual;
```

```
PROPER_LOG_ARCHIVE_MAX_PROC  
-----  
2
```

Appendix E – Configuring Client Failover

Automating client failover, the process by which clients are reconnected to the active primary database after a failure, includes relocating database services to the new primary database as part of a Data Guard failover, notifying clients that a failure has occurred in order to break them out of TCP timeout, and redirecting clients to the new primary database. Configuration details are thoroughly covered in the papers MAA Best Practices for Client Failover for Oracle [Database 11g](#)²¹ and for Oracle [Database 12c](#)²². Please consult these papers and configure your environment appropriately.

Appendix F – Creating and Dispersing Wallets

Creating and dispersing wallets in 12c

1. Create encryption wallet

Set the wallet location in the sqlnet.ora on all nodes of primary and standby.

```
ENCRYPTION_WALLET_LOCATION =  
  (SOURCE = (METHOD = FILE)  
  (METHOD_DATA =  
  (DIRECTORY = /u01/app/oracle/admin/TDE/$ORACLE_SID)  
  )  
  )
```

21 <http://www.oracle.com/technetwork/database/features/availability/maa-wp-11gr2-client-failover-173305.pdf>

22 <http://www.oracle.com/technetwork/database/features/availability/client-failover-2280805.pdf>

NOTE: Using ORACLE_SID in the directory path ensures that all databases do not share the wallet. If there is just one database on the system the ORACLE_SID is not necessary.

2. Create the corresponding directory on all nodes with the proper ORACLE_SID.

```
$mkdir -p /u01/app/oracle/admin/TDE/$ORACLE_SID
```

3. Initiate a new SQL*Plus session. This causes the changes to sqlnet.ora to be picked up.

4. Create the password-based keystore

```
SQL>ADMINISTER KEY MANAGEMENT CREATE KEYSTORE
'/u01/app/oracle/admin/TDE/<ORACLE_SID>' IDENTIFIED BY "AbCdEfGh!";
```

NOTE: Ensure the password string in double quotation marks ("").

5. Open the wallet

```
SQL>ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "AbCdEfGh!";
```

6. Set the Encryption Key

```
SQL>ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY "AbCdEfGh!" WITH BACKUP
USING 'TDE';
```

7. Create Auto-login wallet

An Auto-login wallet removes the requirement of manually opening the wallet when the database is started.

```
SQL>ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE
'/u01/app/oracle/admin/TDE/$ORACLE_SID' IDENTIFIED BY "AbCdEfGh!";
```

8. Copy the files generated in the keystore directory to all nodes of the primary and standby.

Copy files to each node:

```
$scp /u01/app/oracle/admin/TDE/$ORACLE_SID/*
oracle@<host>:/u01/app/oracle/admin/TDE/<SID_NAME>/
```

9. Ensure the wallet is open on all nodes

```
SQL> select * from gv$encryption_wallet;
INST_ID WRL_TYPE WRL_PARAMETER
STATUS
```

```
-----
-----
OPEN          1 file          /u01/app/oracle/admin/TDE/primary1
```

Creating and dispersing wallets in 11gR2

1. Create encryption wallet

Set the wallet location in the sqlnet.ora on all nodes of primary and standby.

```
ENCRYPTION_WALLET_LOCATION =
  (SOURCE = (METHOD = FILE)
    (METHOD_DATA =
      (DIRECTORY = /u01/app/oracle/admin/TDE/$ORACLE_SID)
    )
  )
```

NOTE: Using ORACLE_SID in the directory path ensures that all databases do not share the wallet. If there is just one database on the system the ORACLE_SID is not necessary.

2. Create the corresponding directory on all nodes with the proper ORACLE_SID.

```
$mkdir -p /u01/app/oracle/admin/TDE/$ORACLE_SID
```

3. Initiate a new SQL*Plus session. This causes the changes to sqlnet.ora to be picked up.

4. Set the Master Encryption Key

```
SQL>ALTER SYSTEM SET ENCRYPTION KEY IDENTIFIED BY "AbCdeFgh!";
```

NOTE: Ensure the password string in double quotation marks ("").

5. Create Auto-login wallet

An Auto-login wallet removes the requirement of manually opening the wallet when the database is started.

```
$ orapki wallet create -wallet /u01/app/oracle/admin/TDE/$ORACLE_SID -  
auto_login
```

6. Copy the files generated in the keystore directory to all nodes of the primary and standby.

Copy files to each node:

```
$ scp /u01/app/oracle/admin/TDE/$ORACLE_SID/*  
oracle@host>:/u01/app/oracle/admin/TDE/<SID_NAME>/
```

7. Ensure the wallet is open on all nodes

```
SQL> select * from gv$encryption_wallet;  
INST_ID WRL_TYPE WRL_PARAMETER  
STATUS  
-----  
-----  
1 file /u01/app/oracle/admin/TDE/primary1  
OPEN
```

Appendix G – Standby Instantiation From Database Backup Cloud Service

This section provides the steps required to create a database from the Oracle Database Backup Cloud Service. You need to know DBID of the source database and the CONTROLFILE must be backed up with AUTOBACKUP.

1. Install the Cloud Backup Tool to the Cloud VM.

The library and wallet must be created on the OPC VM in the same way it was done on the primary database machine. This step downloads the library (libopc.so), creates an Oracle wallet and opc<SID>.ora configuration file.

Example:

```
$ mkdir -p /u01/app/oracle/OPC/wallet  
$ mkdir -p /u01/app/oracle/OPC/lib  
$ export ORACLE_SID=stby  
$ export ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1
```

```
$ java -jar opc_install.jar -serviceName <defined service name> -
identityDomain <backup service domain> -opcId '<user@company.com>' -opcPass
'<OPC password>' -walletDir /u01/app/oracle/OPC/wallet -libDir
/u01/app/oracle/OPC/lib
Oracle Database Cloud Backup Module Install Tool, build 2015-05-12
Oracle Database Cloud Backup Module credentials are valid.
Oracle Database Cloud Backup Module wallet created in directory
/u01/app/oracle/OPC/wallet.
Oracle Database Cloud Backup Module initialization file
/u01/app/oracle/product/12.1.0/dbhome_1/dbs/opcstby.ora created.
Downloading Oracle Database Cloud Backup Module Software Library from file
opc_linux64.zip.
Downloaded 23169388 bytes in 20 seconds.
Download complete.
```

2. Retrieve the DBID from the primary database.

```
SQL> select dbid from v$database;

        DBID
-----
812240971
```

3. On the OPC VM, start the instance, set the DBID, password and restore the spfile from the backup and create a pfile from the restored spfile.

```
$ rman target /

Recovery Manager: Release 12.1.0.2.0 - Production on Fri Jul 24 12:25:24 2015
Copyright (c) 1982, 2014, Oracle and/or its affiliates. All rights reserved.

connected to target database (not started)

RMAN> startup nomount force
startup failed: ORA-01078: failure in processing system parameters
LRM-00109: could not open parameter file
'/u01/app/oracle/product/12.1.0/dbhome_1/dbs/initstby.ora'

starting Oracle instance without parameter file for retrieval of spfile
Oracle instance started

Total System Global Area      1073741824 bytes

Fixed Size                     2932632 bytes
Variable Size                  281018472 bytes
Database Buffers               784334848 bytes
Redo Buffers                    5455872 bytes

RMAN> set dbid 812240971;
executing command: SET DBID

RMAN> set decryption identified by <password used when taking backup>;
```

```
executing command: SET decryption
```

```
RMAN> run {  
allocate channel dev1 device type sbt  
parms='SBT_LIBRARY=/u01/app/oracle/OPC/lib/libopc.so';  
restore spfile TO '/tmp/spfile.ora' from autobackup;  
}
```

```
allocated channel: dev1  
channel dev1: SID=12 device type=SBT_TAPE  
channel dev1: Oracle Database Backup Service Library VER=3.15.1.16
```

```
Starting restore at 24-JUL-15
```

```
channel dev1: looking for AUTOBACKUP on day: 20150724  
channel dev1: looking for AUTOBACKUP on day: 20150723  
channel dev1: AUTOBACKUP found: c-812240971-20150723-00  
channel dev1: restoring spfile from AUTOBACKUP c-812240971-20150723-00  
channel dev1: SPFILE restore from AUTOBACKUP complete  
Finished restore at 24-JUL-15  
released channel: dev1
```

```
RMAN> create pfile='/tmp/pfile' from spfile='/tmp/spfile.ora';  
Statement processed
```

```
RMAN> exit
```

```
Recovery Manager complete.
```

4. Edit the pfile, minimally changing the following:

- Remove all double underscore '<primary>.__*' parameters
- Change control_files using /u02 and /u03 mount points

```
*.control_files='/u02/app/oracle/oradata/<db_unique_name>/control01.ctl','  
/u03/app/oracle/fast_recovery_area/<db_unique_name>/control02.ctl'
```

- Change or set the following as listed

```
*.db_create_file_dest='/u02/app/oracle/oradata'  
*.db_create_online_log_dest_1='/u02/app/oracle/oradata'  
*.db_create_online_log_dest_2='/u04/app/oracle/redo'  
*.db_recovery_file_dest='/u03/app/oracle/fast_recovery_area'  
*.dg_broker_start=FALSE  
*.log_archive_dest_1='location=USE_DB_RECOVERY_FILE_DEST', 'valid_for=(ALL_LOG  
FILES, ALL_ROLES)'  
*.diagnostic_dest='/u01/app/oracle'
```

- Remove the following if they exist:

```
*.dg_broker_config_file1  
*.dg_broker_config_file2  
*.db_domain  
*.fal_server  
*.log_archive_config
```


- Change the following according to your environment.

```
*.db_unique_name='<standby_db_unique_name>'
*.local_listener='LISTENER_STBY'
*.audit_file_dest='/u01/app/oracle/admin/<standby_db_unique_name>/adump'
*.db_file_name_convert='< datafile location >', '/u02/app/oracle/oradata'
*.log_file_name_convert='< logfile location 1 >', '/u04/app/oracle/redo',
'< logfile location 2 >', '/u02/app/oracle/oradata'
```

NOTE: If files are spread across multiple mount points additional entries will be required for db_file_name_convert. For each different path to the data files a '<primary path>/\${PRIMARY_DBNM}', '\${secondary path}/\${SECONDARY_DBNM}' pair is required. Retrieve the data file paths from the primary database with 'select name from v\$datafile;'

NOTE: If log files are spread across multiple mount points additional entries will be required for log_file_name_convert. For each different path to the log files '<primary path>/\${PRIMARY_DBNM}', '\${secondary path}/\${SECONDARY_DBNM}' pair is required. Retrieve the log file paths from the primary database with 'select member from v\$logfile;'

5. Create directories for adump, control files and datafiles

```
$mkdir -p /u01/app/oracle/admin/<standby_db_unique_name>/adump
$mkdir -p /u02/app/oracle/oradata/<db_unique_name>/
$mkdir -p /u03/app/oracle/fast_recovery_area/<db_unique_name>
```

6. Create a new spfile from the edited pfile and restart the database.

```
$ rman target /

Recovery Manager: Release 12.1.0.2.0 - Production on Fri Jun 3 14:13:16
2016
Copyright (c) 1982, 2014, Oracle and/or its affiliates. All rights
reserved.

connected to target database: APPDB (DBID=812240971, not open)

RMAN> create spfile from pfile='/tmp/pfile';

Statement processed

RMAN> startup nomount force;

Oracle instance started

Total System Global Area      838860800 bytes
Fixed Size                     2929936 bytes
Variable Size                  608176880 bytes
Database Buffers               222298112 bytes
Redo Buffers                    5455872 bytes
```

7. Restore a standby control file from the backup of the primary

```
RMAN> set dbid 812240971;
executing command: SET DBID

RMAN> set decryption identified by welcome1;
executing command: SET decryption

RMAN> run {
allocate channel dev1 device type sbt
parms='SBT_LIBRARY=/u01/app/oracle/OPC/lib/libopc.so';
restore standby controlfile from autobackup;
alter database mount;
}

allocated channel: dev1
channel dev1: SID=12 device type=SBT_TAPE
channel dev1: Oracle Database Backup Service Library VER=3.15.1.16

Starting restore at 03-JUN-16
channel dev1: looking for AUTOBACKUP on day: 20150724
channel dev1: looking for AUTOBACKUP on day: 20150723
channel dev1: AUTOBACKUP found: c-812240971-20150723-00
channel dev1: restoring control file from AUTOBACKUP c-812240971-20150723-00
channel dev1: control file restore from AUTOBACKUP complete
output file name=/u02/app/oracle/oradata/stby/control01.ctl
output file name=/u03/app/oracle/fast_recovery_area/stby/control02.ctl
Finished restore at 03-JUN-16

Statement processed
released channel: dev1

RMAN> exit
```

8. Reconnect to RMAN with the new control file and restore the database from the backup service.

```
$ rman target /
Recovery Manager: Release 12.1.0.2.0 - Production on Fri Jun 3 14:13:16
2016
Copyright (c) 1982, 2014, Oracle and/or its affiliates. All rights reserved.

connected to target database: PRI (DBID=812240971, not open)

RMAN> CONFIGURE CHANNEL DEVICE TYPE 'SBT_TAPE' PARMS
'SBT_LIBRARY=/u01/app/oracle/OPC/lib/libopc.so,
ENV=(OPC_PFILE=/u01/app/oracle/product/12.1.0/dbhome_1/dbs/opcstby.ora)';

old RMAN configuration parameters:
CONFIGURE CHANNEL DEVICE TYPE 'SBT_TAPE' PARMS
'SBT_LIBRARY=/home/oracle/OPC/lib/libopc.so,
ENV=(OPC_PFILE=/home/oracle/app/oracle/product/12.1.0/dbhome_1/dbs/opcpri.ora
)';
new RMAN configuration parameters:
```

```
CONFIGURE CHANNEL DEVICE TYPE 'SBT_TAPE' PARMS
'SBT_LIBRARY=/u01/app/oracle/OPC/lib/libopc.so,
ENV=(OPC_PFILE=/u01/app/oracle/product/12.1.0/dbhome_1/db/opcstby.ora)';
new RMAN configuration parameters are successfully stored
```

```
RMAN> set decryption identified by <password used for backup>;
```

```
executing command: SET decryption
using target database control file instead of recovery catalog
```

```
RMAN> run {
set newname for database to '/u02/app/oracle/oradata/stby/datafile/%b';
restore database;
switch datafile all;
}
2> 3> 4> 5>
executing command: SET NEWNAME
```

```
Starting restore at 03-JUN-16
using channel ORA_SBT_TAPE_1
using channel ORA_DISK_1
```

```
channel ORA_SBT_TAPE_1: starting datafile backup set restore
channel ORA_SBT_TAPE_1: specifying datafile(s) to restore from backup set
channel ORA_SBT_TAPE_1: restoring datafile 00001 to
/u02/app/oracle/oradata/stby/datafile/system01.dbf
channel ORA_SBT_TAPE_1: restoring datafile 00003 to
/u02/app/oracle/oradata/stby/datafile/sysaux01.dbf
channel ORA_SBT_TAPE_1: restoring datafile 00004 to
/u02/app/oracle/oradata/stby/datafile/undotbs01.dbf
channel ORA_SBT_TAPE_1: restoring datafile 00006 to
/u02/app/oracle/oradata/stby/datafile/users01.dbf
channel ORA_SBT_TAPE_1: reading from backup piece 5dqcoqd8_1_1
channel ORA_SBT_TAPE_1: piece handle=5dqcoqd8_1_1 tag=TAG20150723T104704
channel ORA_SBT_TAPE_1: restored backup piece 1
channel ORA_SBT_TAPE_1: restore complete, elapsed time: 00:02:55
Finished restore at 03-JUN-16
```

```
datafile 1 switched to datafile copy
input datafile copy RECID=5 STAMP=885912874 file
name=/u02/app/oracle/oradata/stby/datafile/system01.dbf
datafile 3 switched to datafile copy
input datafile copy RECID=6 STAMP=885912875 file
name=/u02/app/oracle/oradata/stby/datafile/sysaux01.dbf
datafile 4 switched to datafile copy
input datafile copy RECID=7 STAMP=885912876 file
name=/u02/app/oracle/oradata/stby/datafile/undotbs01.dbf
datafile 6 switched to datafile copy
input datafile copy RECID=8 STAMP=885912877 file
name=/u02/app/oracle/oradata/stby/datafile/users01.dbf
```

```
RMAN> exit
```

```
Recovery Manager complete.
```

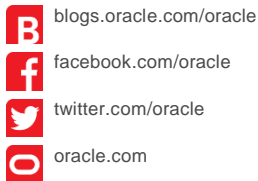
9. Complete the Data Guard broker configuration, see section 4.7. Configure Data Guard Broker as an example.



Oracle Corporation, World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065, USA

Worldwide Inquiries
Phone: +1.650.506.7000
Fax: +1.650.506.7200

CONNECT WITH US



Integrated Cloud Applications & Platform Services

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. This document is provided *for* information purposes only, and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document, and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group. 0615

