Oracle Maximum
Availability Architecture

# Oracle Global Data Services (GDS) MAA Best Practices

Load Balancing & Service Failovers for Replicated Databases

ORACLE WHITE PAPER | DECEMBER 2016

## Introduction

Many organizations maintain one or more replicas of their production databases in local and/or geographically disparate data centers to meet various business requirements such as high availability, disaster recovery, content localization and caching, scalability, optimal performance for local clients or compliance with local laws. Oracle Active Data Guard and Oracle GoldenGate are the strategic replication technologies native to Oracle Database used to synchronize one or more replicated copies for such purposes.

Achieving high performance and high availability by distributing workload across multiple database replicas, however, presents challenges that extend beyond the capabilities of the replication technology. Workload must be intelligently load balanced to effectively utilize all resources and to achieve the best performance. Outages need to be handled intelligently so that applications remain available and deliver the best possible quality of service should a replica go off-line. In an ideal world, load balancing and high availability using a pool of replicated databases occur seamlessly in an optimal fashion using real-time information available in the Oracle Database. This ideal is now achievable with the arrival of Oracle Global Data Services (GDS), available with Oracle Database 12c.

GDS extends the concept of Services, which is a mechanism used for workload management in Oracle Real Application Clusters (RAC), to a globally replicated configuration that includes any combination of Oracle RAC, single instance database, Oracle Active Data Guard and Oracle GoldenGate. This allows Services to be deployed anywhere within this globally replicated configuration, supporting load balancing, high availability, region affinity, and so on. GDS is built on time tested technological building blocks such as Services (Dynamic Workload Management), Oracle Active Data Guard/Oracle Golden Gate replication and Oracle Net Listener.

Even though the feature is called Global Data Services, the databases that constitute the GDS configuration can be globally distributed or located within the same data center. Clients can securely connect by simply specifying a Service name, without needing to know where the physical data center assets providing that Service are located, enabling a highly flexible deployment for an enterprise data cloud. With GDS, a set of replicated databases appear to the applications as single data source.

A GDS configuration contains multiple Global Service Managers (GSM) per region. The GSMs are “Global Listeners” which understand real-time load characteristics and the user-defined

Service placement policies on the replicated databases. These GSMs are instrumental in performing inter-database Service failovers and load balancing of GDS. GDS is a shared infrastructure that can govern multiple sets of replicated databases.

This MAA white paper describes the configuration and operational practices for GDS, and is intended for Enterprise Architects, Database Architects, Database Administrators, Technology Managers, Solution Architects, Application Architects and those who are influential in the overall design of database architecture.

## GDS Concepts

Database Services (Services) are logical abstractions for managing workloads in an Oracle Database. Each Service represents a workload with common attributes, Service-level thresholds, and priorities. The grouping can be based on attributes of work that might include the application function to be used. For example, the Oracle E-Business suite defines a Service for each application module, such as general ledger, accounts receivable, order entry, and so on.

Services are built into the Oracle Database and provide a single system image for workloads. Services enable administrators to configure a workload, administer it, enable and disable it, and measure the workload as a single entity. Clients connect using a database Service name.

In Real Application Clusters, a Service can span one or more instances and facilitate workload balancing based on real-time transaction performance. This provides high availability, rolling changes by workload, and full location transparency.

For replicated environments, GDS introduces the concept of a Global Service. Global Services are provided across a set of databases containing replicated data that belongs to a certain administrative domain known as a *GDS Pool*. Examples of a GDS Pool can be a SALES pool or HR pool etc. A set of databases in a GDS configuration and the database clients are said to be in the same *GDS Region*, if they share the network proximity. Examples of a GDS Regions are Asia region or Europe region etc.

Global Services extend traditional Database Services with an additional attributes such as Global Service Placement, Replication Lag (Active Data Guard only) and Region Affinity. All the attributes of traditional database Services are supported by Global Services.

Global Service Placement:  When a Global Service is created, GDS allows the preferred and available databases for that Service to be specified.  The available databases support a Global Service if the preferred database fails. In addition, GDS allows configuring a Service to run on all the replicas of a given GDS pool.

Replication Lag: Clients can be routed to the Active Data Guard standbys that are not lagging behind by the tolerance limit established with the lag attribute of a global Service.

Region Affinity: A global Service allows customers to set preferences to which region (e.g. Asia, Europe) their given applications should connect.

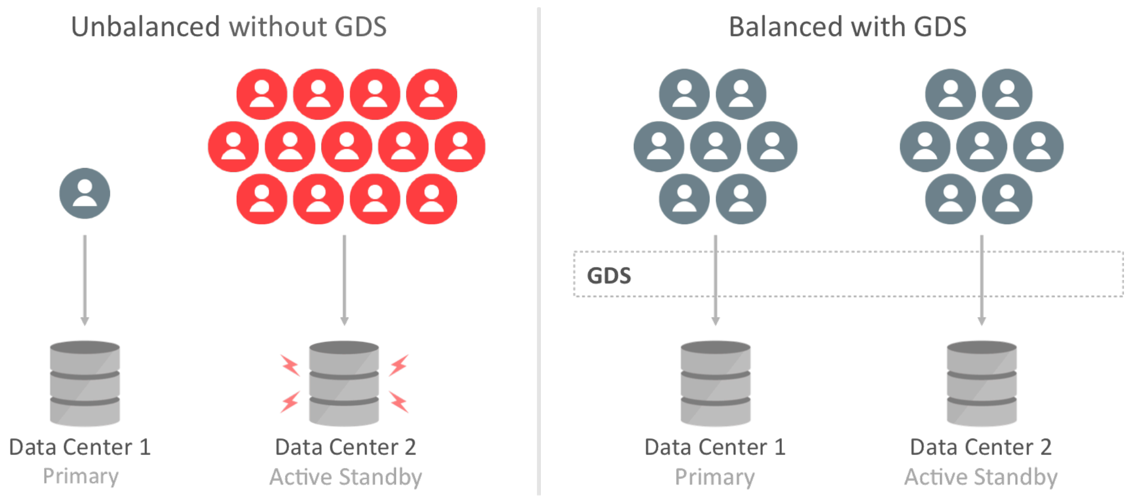

Figure 1: Workload Balancing with GDS

## Key Capabilities of Global Data Services

GDS technology provides the following salient capabilities:

Region-based Workload Routing: With GDS, customers can choose to configure client connections to be always routed among a set of replicated databases in a local region. This capability allows customers maximize their application performance (avoiding the network latency overhead accessing databases in remote regions).

Connect-Time Load Balancing: GSMs use the load statistics from all databases in the GDS pool, inter-region network latency, and the configured connect-time load balancing goal to route the incoming connections to the best database in a GDS pool.

Run-Time Load Balancing: GDS also enables run-time load balancing across replicated databases by publishing real-time load balancing advisory for connection pool based clients (for example – OCI, JDBC, ODP.NET, WebLogic etc.,). The connection pool based clients subscribe to this load balancing advisory and route database requests in real-time across already established connections. With the run-time connection load balancing feature of GDS, application client work requests are dynamically routed to the database that offers the best performance. In addition, GDS also supports the ability to dynamically re-distribute connections when the database performance changes.

Inter-database Service Failover: If a database running a global Service crashes, GDS taking into account the Service placement attributes, automatically performs an inter-database Service failover to another available database in the pool. GDS sends Fast Application Notification (FAN) events so that the client connection pools can reconnect to the new database where the global Service has been newly started.

Replication Lag-based Workload Routing: Active Data Guard standbys may lag behind the primary database. A global Service allows customers to choose the lag tolerance that is acceptable for a given application. GDS routes requests to a standby whose lag is below the limit. If the lag exceeds the lag limit, the Service is relocated to another available standby database that lags below the threshold. New requests are routed to a standby database that satisfies the lag limit. If there is no available database, then the global Service is shutdown. Once the lag is resolved or comes within the limit, GDS automatically brings up the Service.

Role-based Global Services: Upon a database role transition via Data Guard Broker, GDS can automatically relocate the global Service to the new Primary and the new Standby if the role assigned to the service matches the role of the database.

Centralized Workload Management for Replicas: GDS allows easier configuration and management of the resources of the replicated databases that are located anywhere with a single unified framework.

## Benefits of Global Data Services

GDS allows deploying and centrally managing the database Services (across a set of replicated databases) that are fault-tolerant. The GDS framework provides workload balancing across these databases. More specifically, GDS is an Oracle-integrated solution that renders the following benefits:

### Higher Availability and Global Scalability

Supports seamless inter-database Service failover among replicated databases located in any data centers, thus yielding higher application availability.

GDS provides application scalability on demand by allowing dynamic addition of databases. It allows replicated databases to be added to the GDS infrastructure dynamically and transparently to obtain additional resource capability to scale the application workloads. GDS allows this with no change to the application configuration or client connectivity.

### Better Performance and Elasticity

With integrated load balancing across multiple databases, GDS addresses inter-region resource fragmentation. Under-utilized resources in one region can now be put to work to shoulder the workload of over-utilized resources of another region and thus achieves optimal resource utilization.

In a GDS Pool, containing replicated databases running on database servers of different processor generations and different resources (CPU, memory, I/O), GDS sends work requests to less powerful databases, but not until more powerful databases are overloaded and thus strives in equalizing the response time.

### Improved Manageability

GDS configuration can be administered either by the GDSCTL command-line interface or Oracle Enterprise Manager Cloud Control 12c graphical user interface. With centralized administration of global resources, geographically dispersed replicated databases, whether regional or global, can now be effectively utilized within the unified framework of GDS.

Centralized workload management of replicated databases, inter-database Service failover and run-time load balancing are unique features of GDS. GDS enables a truly elastic and agile enterprise and extends the benefits of private data cloud.

## Application Workload Suitability for GDS

GDS is best for application workloads that are replication-aware, i.e. designed to work in replicated environments. Applications that are suitable for GDS adoption possess any of the following characteristics:

- Can separate their work into Read-Only, Read-Mostly and Read-Write Services to take advantage of Oracle Active Data Guard or Oracle GoldenGate replicas. GDS does not distinguish between Read-Only, Read-Mostly and Read-Write transactions. Neither does it know which databases/partitions are Read-Only, Read-Mostly. The application connectivity has to be updated to separate Read-Only, Read-Mostly Services from Read-Write Services and the GDS administrator can configure the Global Services on appropriate databases. For example, a reporting application that can function using a direct connection to a Read-Only Service at an Active Data Guard standby database.
- Are aware of, and avoid or resolve multi-master update conflicts to take advantage of Oracle GoldenGate replicas. For example, an Internet Directory application with built-in conflict resolution that enables Read-Write workload to be distributed across multiple Oracle Databases that are each open Read-Write and synchronized with each other using Oracle GoldenGate multi-master replication.
- Can tolerate some amount of replication lag. For example, a Web-based package tracking application that allows customers to track the status of their shipments using a Read-Only replica where the replica does not lag the source transactional system by more than 10 Seconds.

## GDS in Oracle Maximum Availability Architecture

The Oracle Maximum Availability Architecture (MAA) is Oracle's best practices blueprint for the integrated suite of Oracle's advanced High Availability (HA) technologies. Enterprises that leverage MAA in their IT infrastructure find they can quickly and efficiently deploy applications that meet their business requirements for high availability.

In the pre-12c MAA architecture, in order to optimally distribute workload across multiple synchronized copies, customers employed either 3rd party load balancers or have opted for custom-written connection managers. The 3rd party solutions come with significant integration costs and the DIY home-grown solutions require high initial cost and time to value as well as an overhead of maintenance and support effort.

Starting with Oracle Database 12*c*, with the advent of Global Data Services, Oracle customers can now unify replicated databases' resources with a single framework avoiding the need for home-grown or 3rd party integration for load balancing. Customers can minimize the vendor integration touch points in their overall High Availability/Disaster Recovery stack.

Global Data Services is a strategic new MAA component available with Oracle Database 12*c*. GDS is well integrated with Oracle ecosystem that provides workload routing, load balancing and Service failover across replicated databases located both within and across datacenters. In a nutshell, GDS is a database load balancer for replicated databases, in addition to providing high availability via the inter-database Service failover capability.

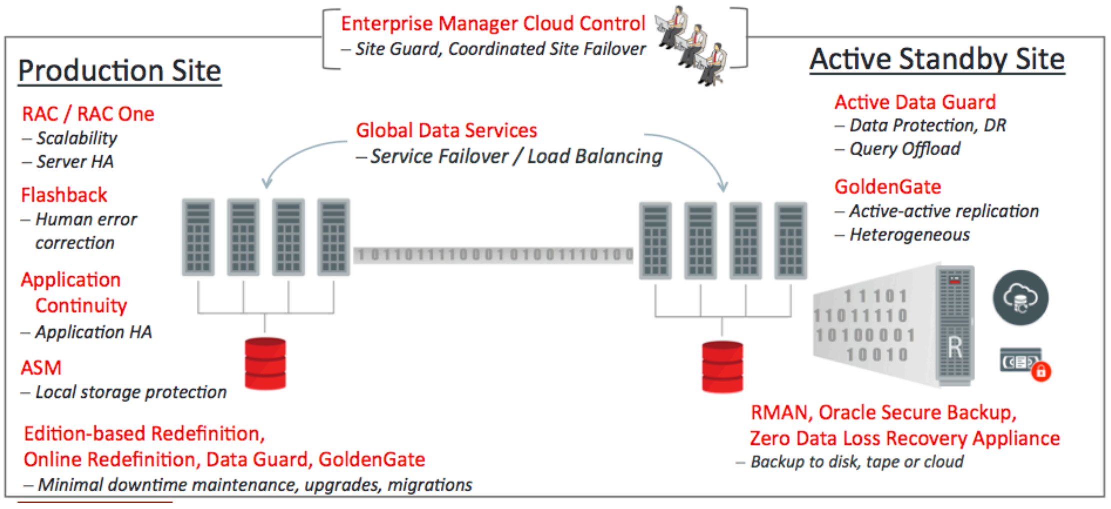
Enterprise Manager Cloud Control
– Site Guard, Coordinated Site Failover
Production Site
RAC / RAC One
– Scalability
– Server HA
Flashback
– Human error correction
Application Continuity
– Application HA
ASM
– Local storage protection
Edition-based Redefinition, Online Redefinition, Data Guard, GoldenGate
– Minimal downtime maintenance, upgrades, migrations
Global Data Services
– Service Failover / Load Balancing
Active Standby Site
Active Data Guard
– Data Protection, DR
– Query Offload
GoldenGate
– Active-active replication
– Heterogeneous
RMAN, Oracle Secure Backup, Zero Data Loss Recovery Appliance
– Backup to disk, tape or cloud

Figure 2: Oracle Maximum Availability Architecture

Global Data Services enables administrators to automatically and transparently manage client workloads across replicated databases that offer common services. A database service is a named representation of one or more database instances. Services enable you to group database workloads and route a particular work request to an appropriate instance. A global service is a service provided by multiple databases synchronized through data replication.

Global Data Services provides dynamic load balancing, failover, and centralized service management for a set of replicated databases that offer common services. The set of databases can include Oracle RAC and non-cluster Oracle databases interrelated through Oracle Data Guard, databases consolidated under Oracle Multitenant, Oracle GoldenGate, or any other replication technology.

For detailed info on GDS, please refer to the Global Data Services White Paper in http://oracle.com/goto/gds.

## Partial versus Full Site Outage

Complete-site failure results in both the application and database tiers being unavailable. To maintain availability users must be redirected to a secondary site that hosts a redundant application tier and a synchronized copy of the production database. MAA best practice is to maintain a running application tier at the standby site to avoid startup time and use Data Guard to maintain the synchronized copy of the production database. Upon site failure, a WAN traffic manager is used to execute a DNS failover (either manually or automatically) to redirect all users to the application tier at standby site while a failover transitions the standby database to the primary production role.

A partial-site failure is either the application stack or the database stack encountered an outage. Application stack failure can be mitigated by making clients connect to the DR site application stack. However, when the primary database becomes unavailable but the application tier that was connected to the primary database remains intact, all that is required to maintain availability is to redirect the application tier to the new primary database after a Data Guard failover has completed. In this use-case the standby database is located within a distance from the surviving application tier where it can deliver acceptable performance using a remote connection after a database failover has occurred.

GDS enables service management and load balancing between replicated databases **within** a region. This is to say when you have a database down but the application tier still functioning when GDS global service managers (GSMs) can route connections to the best available replica based upon load balancing policies and service availability. By contrast, an out-of-region failover requires users to be directed to a remote application tier local to the new production database (serviced by a different set of in-region GSMs).

While this paper focuses on GDS configuration for failover within a region, refer to the appendix for the high level flow of failover out of region.

## GDS Configuration

The following section covers the sample configuration of GDS. Note that the following example utilizes an Admin Managed RAC database. Administrator-managed deployment that you configure database services to run on specific instances belonging to a certain database using the preferred and available designation. Policy-managed deployment is based on server pools, where database services run within a server pool as singleton or uniform across all of the servers in the server pool. Databases are deployed in one or more server pools and the size of the server pools determine the number of database instances in the deployment. For detailed info on GDS, please refer to the GDS documentation. (https://docs.oracle.com/database/121/GSMUG/toc.htm)

1. Create and prepare a GDS Catalog Database

   GDS uses a catalog database to store meta-data relating to the layout and status of the GDS configuration. For the maximum availability, Oracle recommends that the GDS catalog database to be deployed independently and use Oracle high availability features such as Oracle Real Application Clusters (Oracle RAC) and Oracle Data Guard to protect the Global Data Services catalog against outages.

2. Once the catalog database has been created, create the gsm_admin user and assign that user the gsmadmin_role. Note that by default the password for both gsm_admin, gsmuser, and gsmcatuser expire after 180 days.

```
SQL> create user gsm_admin identified by welcome1;
User created.
SQL> grant gsmadmin_role to gsm_admin;
Grant succeeded.
SQL> exit
```

3. Copy the Oracle Net alias that can be used to access the catalog database and place it in the tnsnames.ora file in the GSM home:

```
GDSCAT =
(DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = slcc32-scan5)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = gdscat)
    )
  )
```

4. With the environment configured for the GSM home, use gdsctl to connect to and create the GDS catalog:

```
GDSCTL>create catalog -database gdscat -user gsm_admin
"gsm_admin" password:
Catalog is created
```

5. Connect to the catalog database and unlock the gsmcatuser and set the password:

```
SQL> alter user gsmcatuser account unlock;
User altered.
SQL> alter user gsmcatuser identified by welcome1;
User altered.
```

6. With the environment configured for the GSM home, use gdsctl to connect to, create, and start the GSM listeners.  As a best practice GSM listeners should reside on their own hardware separate from that hosting the Oracle Databases in the GDS configuration.  The resource requirements for hardware needed to run GSM listeners is lightweight and can easily be accommodated using virtual machines.

```
GDSCTL>add gsm -gsm gsm1 -listener 1522 -catalog gdscat
"gsmcatuser" password:
Create credential oracle.security.client.connect_string1
GSM successfully added
GDSCTL>start gsm -gsm gsm1
GSM is started successfully
GDSCTL>status
Alias                     GSM1
Version                   12.1.0.2.0
Start Date                13-APR-2016 09:40:59
Trace Level               off
Listener Log File
/u01/app/oracle/diag/gsm/slcc32adm05/gsm1/alert/log.xml
Listener Trace File
/u01/app/oracle/diag/gsm/slcc32adm05/gsm1/trace/ora_64863_1397397499304
32.trc
Endpoint summary
(ADDRESS=(HOST=slcc32adm05.us.oracle.com)(PORT=1522)(PROTOCOL=tcp))
GSMOCI Version            0.1.11
Mastership                Y
Connected to GDS catalog  Y
Process Id                64883
Number of reconnections   0
Pending tasks.     Total  0
Tasks in  process. Total  0
Regional Mastership       TRUE
Total messages published  0
Time Zone                 -04:00
Orphaned Buddy Regions:
     None
GDS region                regionora
```

7. With the environment configured for the GSM home, use gdsctl to create a default GDS pool and default region:

```
GDSCTL>add gdspool -gdspool sales
GDSCTL>add region -region slc
GDSCTL>add region -region sca
```

8. To provide global services, a database must be added to a Global Data Services pool. Before adding a database to a pool, the database administrator should unlock the `GSMUSER` account and give the password to the Global Data Services pool administrator, as shown in the following example:

```
SQL> alter user gsmuser account unlock;
User altered.
SQL> alter user gsmuser identified by welcome1;
User altered.
```

9. To be part of a Global Data Services pool, a database must use a server parameter file (SPFILE). An Oracle RAC database should also have SCAN set up.  To add a database, the user should connect to the Global Data Services catalog using the Global Data Services pool or Global Data Services administrator credentials, for example:

```
GDSCTL>add database -connect mts -region sca -gdspool sales
Catalog connection is established
"gsmuser" password:
DB Unique Name: mts
```

10. The valid node checking for registration (VNCR) feature provides the ability to configure and dynamically update a set of IP addresses, host names, or subnets from which registration requests are allowed by the global service manager. Database instance registration with a global service manager succeeds only when the request originates from a valid node.  If a host on which a database resides contains multiple network interfaces the auto configuration could register the wrong set of IP addresses which will lead to the database registration being rejected.  In addition, if a firewall exists between the GSMs and the databases and the ports are not opened the registration will also fail.  From the GSM alert log you will see entries similar to the following:

```
Listener(VNCR option 1) rejected Registration request from destination
10.245.19.151
Listener(VNCR option 1) rejected Registration request from destination
10.245.19.150
```

11. You will also find that the database object exists in the GDS catalog but some or all instances associated with certain hosts are missing.  For example

```
GDSCTL>databases
Database: "mts" Registered: Y State: Ok ONS: Y. Role: PRIMARY
Instances: 1 Region: slc
   Registered instances:
     sales%1
```

12. To correct the rejected registration and get all database instances properly discovered run the “`add invitednode`” using the rejected IP address listed in the GSM alert log.  Additionally, if there is a firewall

between the GSMs and the database then once the ports have been opened and verified using tnsping issue the invitenode command as show below:

```
GDSCTL>add invitednode 10.245.19.150

GDSCTL>databases
Database: "mts" Registered: Y State: Ok ONS: Y. Role: PRIMARY
Instances: 2 Region: slc
   Registered instances:
     sales%1
     sales%2
```

13. The `gdsctl add service` command is used to create a service on the Global Data Services pool databases. A simple example of the command is as follows:

```
GDSCTL>add service -service sales_sb -preferred_all -gdspool sales -
notification TRUE
```

14. If this is a RAC database being added with multiple instances then you must modify the service to add the database instances

```
GDSCTL>modify service -gdspool sales -service sales_sb -database mts -
add_instances -preferred mts1,mts2

GDSCTL>modify service -gdspool sales -service sales_sb -database stm -
add_instances -preferred stm1,stm2

GDSCTL>start service -service sales_sb -gdspool sales
```

15. Verify that the global service is running:

```
GDSCTL>services
Service "sales_sb.sales.oradbcloud" has 2 instance(s). Affinity:
ANYWHERE
   Instance "sales%1", name: "mts1", db: "mts", region: "slc", status:
ready.
   Instance "sales%2", name: "mts2", db: "mts", region: "slc", status:
ready.
```

## Configuration Best Practices

MAA GDS requires that

- Each client will communicate via an Oracle integrated connection pool such as UCP, OCI, ODP.NET The connection pools will be notified on the service failovers and load balancing advisory via Fast Application Notification events .
- **Three GSMs per region**. Global Service Managers (GSMs) enable routing of connections among replicated databases. A GSM is a stateless, light weight and intelligent listener that can repopulate its meta data from the GDS catalog. Each GSM should reside in separate hardware. As a best practice three create 3 GSM's per regions so that in the event one GSM goes down we have two remaining GSM's providing redundancy.

- **GDS Catalog Database protected by Data Guard.** The GDS catalog is small (< 100 GBs) repository that hosts the metadata of the GDS configuration, Regions, GSMs, Global services, databases etc.. MAA recommends setting up a local Data Guard standby database configured with Maximum Availability database protection mode and Data Guard Fast-Start failover and a remote physical standby database. All GDS catalog standby databases should use Active Data Guard for best data protection and reside on separate hardware and storage.

### GDS Services and FAN ONS

FAN uses the Oracle Notification Service (ONS) for event propagation to all clients from Oracle Database 12c onwards and for JDBC, Tuxedo and listener clients before 12c. ONS is installed as part of Oracle Global Data services, Oracle Grid Infrastructure on a cluster, in an Oracle Data Guard installation, and when Oracle WebLogic is installed. ONS is responsible for propagating FAN events to all other ONS daemons it is registered with. No steps are needed to configure or enable FAN at the database server side with one small exception: OCI FAN and ODP FAN require thatnotification is set to TRUE for the service by gdsctl. With FAN auto-configuration at the client, ONS jar files must be on the CLASSPATH or in the ORACLE_HOME dependent on your client.

#### General Steps for Configuring FCF Clients

Follow these steps before progressing to driver specific instructions:

Use a Dynamic Database Service Using FAN requires that the application connects to the database using a dynamic global database service. This is a service created using gdsctl.

Do not connect using the database service or PDB service – these are for administration only and are not supported for FAN. The TNSnames entry or URL must use the service name syntax and follow best practice by specifying a dynamic database service name. Refer to the examples later in this section.

Use the Oracle Notification Service when you use FAN with JDBC thin or Oracle Database 12c Release 1 (12.1.0.1) OCI or ODP.Net clients, FAN is received over ONS. Accordingly, in Oracle Database 12c ONS FAN auto-configuration is introduced such that FCF clients discover the server-side ONS networks and self configure. FAN is automatically enabled when ONS libraries or jars are present.

In Oracle Database 12c, it is still necessary to enable FAN on most FCF clients. FAN auto-configuration removes the need to list the Global Service Managers that an FCF client needs.

Listing server hosts is incompatible with location transparency and causes issues with updating clients when the server configuration changes. Clients already use a TNS address string or URL to locate the GSM listeners.

FAN auto-configuration uses the TNS addresses to locate the GSM listeners and then asks each server database for the ONS server-side addresses. When there is more than one GSM, for example, FAN auto-configuration contacts each and obtains an ONS configuration for each one.

When using Oracle Database 12c, the ONS network is discovered from the URL. An ONS node group is automatically obtained for each address list when LOAD_BALANCE is off across the address lists.

By default the FCF client maintains three hosts for redundancy in each node group in the ONS configuration.. Each node group corresponds to each GDS data center. For example, if there is a primary database and several Oracle Data Guard standbys there is by default, 3 ONS connections maintained at each node group. The node groups are discovered when using FAN auto-configuration, or if you are using version older than Oracle Database 12c, then use ons.configuration. With node_groups defined by FAN auto-configuration, load_balance=false (the default), more ONS end points are not required. If you want to increase the number of end points you can do this by increasing maxconnections. This applies to each node group. Increasing to 4 in this example, maintains four ONS connections at each node. Increasing this value consumes more sockets.

```
oracle.ons.maxconnections=4 ONS
```

If the client is to connect to multiple clusters, and receive FAN events from both, for example in the RAC with Data Guard situation, then multiple ONS node groups are needed. FAN auto-configuration creates these node groups using the URL or TNS names for 12c client and 12c database. If not using auto-ons, specify the node groups in the ig or oraaccess.xml configuration files.

### Client Side Configuration

As a best practice Oracle recommends that all client versions be 12.1 or greater. Client versions before 12c have the following limitations:

The pre-12c OCI and ODP.NET clients do not support global services.

The pre-12c JDBC client supports global services, but you must manually configure it to subscribe to the ONS servers co-located with the global service managers in the client's local and buddy regions. It is a best practice to subscribe to three ONS servers in each of the client's local and buddy regions.

The pre-12c Thin JDBC client can only be used with a GDS configuration that has a single region, unless the region parameter is specified in the connect string.

As a best practice there should be multiple GSM's to provide for high availability. Clients should be configured for multiple connection endpoints where these endpoints are global service managers rather than local, remote, or single client access name (SCAN) listeners. For OCI / ODP .Net clients use the following TNS names structure:

```
(DESCRIPTION=(CONNECT_TIMEOUT=90)(RETRY_COUNT=30)(RETRY_DELAY=3)
(TRANSPORT_CONNECT_TIMEOUT=3)
  (ADDRESS_LIST =
   (LOAD_BALANCE=on)
   (ADDRESS=(PROTOCOL=TCP)(HOST=GSM1)(PORT=1522))
   (ADDRESS=(PROTOCOL=TCP)(HOST=GSM2)(PORT=1522))
   (ADDRESS=(PROTOCOL=TCP)(HOST=GSM3)(PORT=1522)))
  (ADDRESS_LIST=
   (LOAD_BALANCE=on)
   (ADDRESS=(PROTOCOL=TCP)(HOST=GSM2)(PORT=1522)))
 (CONNECT_DATA=(SERVICE_NAME=sales)))
```

For UCP Connection Pool / JDBC thin clients use the following URL structure through to 12.1.0.2:

```
jdbc:oracle:thin = (CONNECT_TIMEOUT=90)(RETRY_COUNT=30)(RETRY_DELAY=3)
  (ADDRESS_LIST=
   (LOAD_BALANCE=on)
  (ADDRESS=(PROTOCOL=TCP)(HOST=GSM1)(PORT=1522))
  (ADDRESS=(PROTOCOL=TCP)(HOST=GSM2)(PORT=1522))
  (ADDRESS=(PROTOCOL=TCP)(HOST=GSM1)(PORT=1522)))
  (ADDRESS_LIST=
   (LOAD_BALANCE=on)
```

```
     (ADDRESS=(PROTOCOL=TCP)(HOST=GSM2)(PORT=1522)))
   (CONNECT_DATA=(SERVICE_NAME=sales)))
```

Note that after Oracle Database 12c Release 1 (12.1.0.2), JDBC and OCI align and you should use the OCI version for all connection descriptions.

Always use dynamic global database services created by gdsctl to connect to the database. Do not use the database service or PDB service – these are for administration only, not for application usage and do not provide FAN and many other features because they are available at mount.

For JDBC, use the current client driver (Oracle Database 12c Release 1) with current or older RDBMS Use one DESCRIPTION in the TNS names entry or URL – using more causes long delays connecting when RETRY_COUNT and RETRY_DELAY are used. Set CONNECT_TIMEOUT=90 or higher to prevent logon storms for OCI and ODP clients.

Use a lower setting for JDBC clients, CONNECT_TIMEOUT=4, as a temporary measure until TRANSPORT_CONNECT_TIMEOUT is available. Do not also set JDBC property oracle.net.ns.SQLnetDef.

TCP_CONNTIMEOUT_STR as it overrides CONNECT_TIMEOUT Set LOAD_BALANCE=ON per address to expand SCAN names starting with Oracle Database 11g Release 2 (11.2.0.3) for OCI and Oracle Database 12c Release 1 (12.1.0.2) for JDBC Do not use Easy*Connect syntax (EZConnect) – it has no High Availability capabilities.

## Application Level Configuration

### How to Configure FAN for 12c Java Clients Using Universal Connection Pool

The best way to take advantage of FCF with the Oracle Database JDBC thin driver is to use either the Universal Connection Pool (UCP) or WebLogic Server Active GridLink. Setting the pool property FastConnectionFailoverEnabled on the Universal Connection Pool enables Fast Connection Failover (FCF). Active GridLink always has FCF enabled by default. Third party application servers including IBM WebSphere and Apache Tomcat support UCP as a connection pool replacement. For more information on embedding UCP with other web servers refer to these white papers:

Design and Deploy WebSphere Applications for Planned, Unplanned Database Downtimes and Runtime Load Balancing with UCP (http://www.oracle.com/technetwork/database/application-development/planned-unplannedrlb-ucp-websphere-2409214.pdf) and Design and deploy Tomcat Applications for Planned, Unplanned Database Downtimes and Runtime Load Balancing with UCP (http://www.oracle.com/technetwork/database/application-development/planned-unplanned-rlb-ucptomcat-2265175.pdf).

Follow these configuration steps to enable Fast Connection Failover:

1. The connection URL must use the service name syntax and follow best practice by specifying a dynamic database service name and the JDBC URL structure (above and also below). All other URL format do not provide high availability. The URL may use JDBC thin or JDBC OCI.

2. If wallet authentication has not previously been established or the cluster is running a version earlier than Oracle Grid Infrastructure 12.1.0.2 then remote ONS configuration is needed. This can be done through the pool property setONSConfiguration which can be set through a property file as shown in the following example: The property file specified must contain an oracle.ons.nodes property and optionally, properties for oracle.ons.walletfile and oracle.ons.walletpassword. An example of an ons.properties file is shown here:

```
PoolDataSource pds = PoolDataSourceFactory.getPoolDataSource();

pds.setConnectionPoolName("FCFSamplePool");
pds.setFastConnectionFailoverEnabled(true);
pds.setONSConfiguration("propertiesfile=/usr/ons/ons.properties");
pds.setConnectionFactoryClassName("oracle.jdbc.pool.OracleDataSource");
pds.setURL("jdbc:oracle:thin@((CONNECT_TIMEOUT=4)(RETRY_COUNT=30)(RETRY
_DELAY=3) "+ " (ADDRESS_LIST = "+ " (LOAD_BALANCE=on) "+ " ( ADDRESS =
(PROTOCOL = TCP)(HOST=GSM1)(PORT=1522))) "+ " (ADDRESS_LIST = "+ "
(LOAD_BALANCE=on) "+ "( ADDRESS = (PROTOCOL =
TCP)(HOST=GSM2)(PORT=1522)))"+
"(CONNECT_DATA=(SERVICE_NAME=service_name)))");
```

3. Ensure the pool property `setFastConnectionFailoverEnabled=true` is set.
4. The CLASSPATH must contain ons.jar, ucp.jar, and the jdbc driver jar file, for example ojdbc7.jar.
5. The following is not required and is an optimization to force re-ordering of the SCAN IP addresses returned from DNS for a SCAN address:
6. If you are using JDBC thin with Oracle Database 12c Application Continuity can be configured to failover the connections after FAN is received.
7. If the database is earlier than 12c or if the configuration needs different ONS end points to those autoconfigured, the ONS end points can be enabled as per the following example: Or in the situation with multiple clusters and when using autoons – autoons would gerate node lists similar to the following − `oracle.ons.maxconnections` is set to 3 by default for EVERY active nodelist, so there is no need to explicitly set this. This example will result in the ONS client trying to maintain 6 total connections

## How to Configure FAN for OCI Clients

OCI clients embed FAN at the driver level so that all clients can use them regardless of the pooling solution. Starting with Oracle database and client 12c, OCI clients use ONS for the FAN transport. Both the server and the client must use version 12c. If either the client or server is using version 11.2 or earlier, then the FAN transport used is advanced queues.

### Configuration for SQL*Plus and PHP

1. Set notification for the service
2. For PHP clients only, add oci8.events=On to php.ini: Important If oraccess.xml is present with events=-false or events not specified, this disables the usage of FAN. To maintain FAN with SQL*Plus and PHP when oraccess.xml is in use, set events=-true
3. On the client side and using a 12c client and Oracle Database 12c database, enable FAN in oraaccess.xml.

### Configuration for 12c OCI Clients when using 12c Database Server

1. Tell OCI where to find ONS Listeners Starting with 12c, the client install comes with ONS linked into the client library. Using ons auto-config, the ONS end points are discovered from the TNS address. This automatic method is the recommended approach. Like ODP.Net, manual ONS configuration is also supported using oraaccess.xml.

2. Enable FAN high availability events for the OCI connections To enable FAN requires editing the OCI file oraaccess.xml specify the global parameter events. This file is located in $ORACLE_HOME/network/admin. See the following whitepaper for additional information: http://www.oracle.com/technetwork/database/options/clustering/overview/fastapplicationnotification12c-2538999.pdf
3. Tell OCI where to find ONS Listeners Starting with 12c, the client install comes with ONS linked into the client library. Using ons auto-config, the ONS end points are discovered from the TNS address. This automatic method is the recommended approach. Like ODP.Net, manual ONS configuration is also supported using oraaccess.xml.
4. Enable FAN at the server for all OCI clients Continuing with Oracle Database 12c, it is still necessary to enable FAN at the database server for all OCI clients (including SQL*Plus). Which steps differ if you want to custom configure ONS end points with Oracle Database 12c If you want to custom configure ONS, use syntax similar to the following example. This is not the recommended method.

## Controlling Logon Storms

Small connection pools are strongly recommended but when you have many connections controlling logon storms can be done via tuning. Oracle recommends the following tuning on servers that hosts Global Service Managers:

1. Increase the Listen backlog at the OS level. To have the new value take affect without rebooting the server perform the following as root:

```
echo 8000 > /proc/sys/net/core/somaxconn
```

2. In order to persist the value across reboots, also add to /etc/sysctl.conf

```
net.core.somaxconn=6000
```

3. Increase Queuesize for the GSM listener. Update sqlnet.ora in Oracle home that the listeners are running from to increase the queuesize parameter:

```
TCP.QUEUESIZE=6000
```

## Graceful Application Switchover Procedure

At the heart of this procedure is the use of database services to properly manage workloads during the planned outage. Services must be properly created and the application must obtain its connections from a service. Additional details on how services should be created and used will be provided later in this document.

These steps assume the use of a FAN-aware connection pool such as Oracle Universal Connection Pool (UCP) to gracefully drain connections without application interruption from a service that is stopped. Your applications may use other connection types that don't support FAN-aware connection pools or have long-running transactions. Ideally, these applications will be disconnected before the maintenance window. In the procedure below, we will disconnect some sessions when their transaction ends in a timely manner or ultimately when the instance is shut down for maintenance.

The recommended and validated procedure is detailed below; please note that certain applications may have their own specific procedures to follow.

### Understanding Your Application's Use of Connections

Understanding how your application obtains and releases its connections is important to determining if the application is able to gracefully switchover to other instances in the cluster.

Find the following information about your application:

- Workload at the time of the planned outage (OLTP/short transactions, or batch/long transactions)?
    - Short transactions using a connection pool such as UCP or ODP.NET can be quiesced rapidly
    - Long transactions need additional time to quiesce or must have batch jobs stopped or queued an appropriate time in advance
- Type of connection obtained thin Java, OCI, ODP with C# or ODP with OCI)?
- UCP, ICC, ODP.NET, and OCI session pools use Fast Application Notification (FAN) to drain the pool quickly; other connections require waiting until the connection is closed (or termination if application allows)
- Amount of time to wait for connection pool to quiesce before stopping the service?
    - Useful to know so proper amount of time is given before disconnection is performed
- Application able to handle disconnection after transaction completes (applies to batch workloads)?
    - If the application can't handle being disconnected gracefully, then it must be stopped in advance of the planned maintenance or Application Continuity might be an option to avoid interruption

#### Services and Application Configuration Best Practices

You must have properly configured services and application attributes to successfully perform a graceful switchover.  Please see document 1617163.1 for a matrix of validated drivers and applications clients.

Note: You must test your configuration to ensure it is set up and performs switchover properly before relying on it for a production system.

### Using GDS with Active Data Guard

#### Prerequisites

You must have the following in place for this procedure to work properly:

- Active Data Guard configuration that utilizes Oracle 12.1.0.2  databases (plus GDS fixes maintained in 23075893).
- Global Data Services (GDS) configuration running on 12.1.0.2.
- GDS service that has been created so that it can be ran on all Active Data Guard databases in the configuration.  For example:

```
GDSCTL>add service -service sales_sb -preferred_all -gdspool sales -role
physical_standby -notification TRUE
GDSCTL>modify service -gdspool sales -service sales_sb -database mts -
add_instances -preferred mts1,mts2
GDSCTL>modify service -gdspool sales -service sales_sb -database stm -
```

```
add_instances -preferred stm1,stm2
GDSCTL>start service -service sales_sb -gdspool sales
```

**Procedure for Moving Sessions in a Rolling Fashion for ADG Reader Farm**

1. Check current status of the services and related instances to ensure services can be moved successfully. Note that the service should only be available on the source standby database at this point

```
GDSCTL>services
Service "sales_sb.sales.oradbcloud" has 2 instance(s). Affinity:
ANYWHERE
   Instance "sales%1", name: "mts1", db: "mts", region: "slc",
status: ready.
   Instance "sales%2", name: "mts2", db: "mts", region: "slc",
status: ready.
```

2. Stop services normally (not using FORCE option) on the source database where connections are to be removed

- This step will quiesce the FAN-aware connection pools using FAN.
- New connections are directed to other instances offering that service and idle sessions are disconnected from the pool using the services
- Existing connections are allowed to continue until their work complete and they are returned to the connection pool

```
GDSCTL>stop service -service sales_sb -database mts -gdspool sales
```

Allow an agreed time for the sessions to disconnect and relocate then continue with the next steps.

Note: If this is a scenario where you are performing a rolling upgrade of an Active Data Guard reader farm and the services are not running on other ADG reader nodes you can perform the service stop on this databases prior to performing the stop service described in this step.

3. Disconnect long-running sessions after the current query completes

Preferably long-running queries have been scheduled to stopped or queued prior to the window when connections are to be moved. This step handles long-running sessions that are still running and now needs to be stopped (e.g. killed) abruptly.

Log on to the instance that you intend to shut down

1. Check V$SESSION to see if any sessions are still connected to the service

```
SQL> SELECT service_name, count(1) FROM v$session GROUP BY service_name
ORDER BY 2;
```

2. Run the DBMS_SERVICE.DISCONNECT_SESSION package for the service you stopped earlier, for example:

```
SQL> exec
dbms_service.disconnect_session('oltp_work',DBMS_SERVICE.POST_TRANSACTI
ON);
```

3. Check V$SESSION again to ensure sessions have logged off from the service

```
SQL> SELECT service_name, count(1) FROM v$session GROUP BY service_name
ORDER BY 2;
```

***4***. Start GDS service on the target database and allow sessions to connect

```
GDSCTL>start service -service sales_sb -database stm -gdspool sales
```

- Log on to the target database
- Check V$SESSION to see sessions connected to the service

```
SQL> SELECT service_name, count(1) FROM v$session GROUP BY service_name
ORDER BY 2;
```

## Using GDS with GoldenGate

For this GoldenGate role transition example, there are 2 databases: GG replica1 and GG replica2. The GoldenGate is setup with uni-directional replication with extract running initially on GG replica1 and replicat running initially on GG replica2. The generic steps are still applicable for bi-directional GG replica or downstream mining GG replicas.

### Prerequisites

You must have the following in place for this procedure to work properly:

- GoldenGate (GG) configuration that utilizes Oracle 12.1.0.2 databases (plus GDS fxes maintained in 23075893).
- GoldenGate processes should not connect to the source or target database using the GDS service name, but a dedicated TNS alias. Using the GDS service will get the database connections terminated prematurely causing possible data loss.
- A heartbeat table has been implemented in the GoldenGate source and target databases to keep track of replication latency, and to also ensure synchronization of the Replicat applied SCN. Use GoldenGate 12.2 or later due to the automatic heartbeat table feature. Refer to the Oracle GoldenGate Administration Guide for details enabling the automatic heartbeat table: http://docs.oracle.com/goldengate/c1221/gg-winux/GWUAD/wu_monitoring.htm#GWUAD1213.
- Global Data Services (GDS) configuration running on 12.1.0.2.
- GDS service that has been created so that it can be ran on all databases in the GG configuration. For example:

```
GDSCTL>add service -service sales_sb -preferred_all -gdspool sales
GDSCTL>modify service -gdspool sales -service sales_sb -database mts -
```

```
add_instances -preferred mts1,mts2
GDSCTL>modify service -gdspool sales -service sales_sb -database stm -
add_instances -preferred stm1,stm2
GDSCTL>start service -service sales_sb -gdspool sales
```

**Procedure**

1. Check current status of the services and related instances to ensure services can be moved successfully. Note that the service should only be available on the source database at this point

```
GDSCTL>services
Service "sales_sb.sales.oradbcloud" has 2 instance(s). Affinity:
ANYWHERE
   Instance "sales%1", name: "mts1", db: "mts", region: "slc", status:
ready.
   Instance "sales%2", name: "mts2", db: "mts", region: "slc", status:
ready.
```

2. Stop services normally (not using FORCE option) on the source database where connections are to be removed.

- This step will quiesce the FAN-aware connection pools using FAN.
- New connections are directed to other instances offering that service and idle sessions are disconnected from the pool using the services
- Existing connections are allowed to continue until their work complete and they are returned to the connection pool

```
GDSCTL>stop service -service sales_sb -database mts -gdspool sales -force
```

Allow an agreed time for the sessions to disconnect and relocate then continue with the next steps. The time to allow for sessions to drain depends on the workload and user transactions for your application.

3. Disconnect long-running sessions after the current transaction completes

Preferably long-running batch jobs have been scheduled to stopped or queued prior to the maintenance window. This step handles long-running sessions that are still running and now needs to be stopped (e.g. killed) abruptly. Check with the application developers if these long running batch jobs are idempotent and recoverable prior disconnecting a long-running sessions.

- Log on to the instance that you intend to shut down
- Check V$SESSION to see if any sessions are still connected to the service

```
SQL> SELECT service_name, count(1) FROM v$session GROUP BY service_name
ORDER BY 2;
```

- Run the DBMS_SERVICE.DISCONNECT_SESSION package for the service you stopped earlier, for example:

```
SQL> exec
dbms_service.disconnect_session('oltp_work',DBMS_SERVICE.POST_TRANSACTION);
```

- Check V$SESSION again to ensure sessions have logged off from the service

  ```
  SQL> SELECT service_name, count(1) FROM v$session GROUP BY service_name
  ORDER BY 2;
  ```

***4***. Once all sessions associated with the GDS service have been disconnected verify that all data from the GoldenGate source databases have been replicated to the target database

Record the current database SCN from the GoldenGate SOURCE database:

```
SQL> SELECT current_scn FROM v$database;
```

- On the GoldenGate TARGET database, continue to monitor the Replicat applied SCN using the following query:

  ```
  SQL> SELECT lwm_position FROM v$gg_apply_coordinator;
  ```

- Once the target LWM_POSITION SCN is greater than the CURRENT_SCN recorded in step #1, it is safe to assume that all transactions have been replicated from the source to the target database and the users can now be switched over to the GoldenGate target database.

The above steps allow for a graceful switchover. However, if this is a failover event where the source database is not available you can use the following steps to estimate the data loss.

When using the automatic heartbeat table, introduced in GoldenGate 12.2, use the following query to determine the replication latency:

```
SQL> col Lag(secs) format 999.9
SQL> col "Seconds since heartbeat" format 999.9
SQL> col "GG Path" format a32
SQL> col TARGET format a12
SQL> col SOURCE format a12
SQL> set lines 140
SQL> select remote_database "SOURCE", local_database "TARGET", incoming_path
"GG Path", incoming_lag "Lag(secs)", incoming_heartbeat_age "Seconds since
heartbeat" from gg_lag;

SOURCE       TARGET       GG Path                          Lag(secs) Seconds
since heartbeat
------------ ------------ -------------------------------- --------- -------
----------------
       MTS          GDST          EXT_1A ==> DPUMP_1A ==> REP_1A        7.3
9.0
```

The above example shows there is a possible 7.3 seconds of data loss between the source and target databases.

5. Start GDS service on the target database and allow sessions to connect. Note that if the application workload can accept a certain level of data lag then it is possible to perform this step much earlier before step two listed above.

```
GDSCTL>start service -service sales_sb -database stm -gdspool sales
```

- Log on to the target database
- Check V$SESSION to see sessions connected to the service

  ```
  SQL> SELECT service_name, count(1) FROM v$session GROUP BY service_name
  ORDER BY 2;
  ```

For more information, see MAA resources on OTN (*oracle.com/goto/*maa).  For GDS specific information, go to oracle.com/goto/gds site.

## Conclusion

Global Data Services (GDS) is a holistic automated workload management feature of Oracle Database 12*c* that provides workload routing, load balancing and inter-database Service failover, replication lag based routing, role based global Services and centralized workload management for a set of replicated databases that are globally distributed or located within the same data center. Customers can now use GDS to achieve these benefits without the need to integrate with multiple point solutions or home grown products. Oracle Database 12c GDS provides better hardware/software utilization, better performance, scalability, and availability for application workloads running on replicated databases.

## Appendix:

GDS Failover across regions

1. The administrator has failed over or switched over the production database to the secondary site. This is automatic if you are using Data Guard fast-start failover.
2. The administrator starts the middle-tier application servers on the secondary site, if they are not already running.
3. The wide-area traffic manager selection of the secondary site can be automatic in the case of an entire site failure. The wide-area traffic manager at the secondary site returns the virtual IP address of a load balancer at the secondary site and clients are directed automatically on the subsequent reconnect. In this scenario, the site failover is accomplished by an automatic domain name system (DNS) failover.
4. Alternatively, a DNS administrator can manually change the wide-area traffic manager selection to the secondary site for the entire site or for specific applications. The following is an example of a manual DNS failover:
    - Change the DNS to point to the secondary site load balancer:
    - The master (primary) DNS server is updated with the new zone information, and the change is announced with DNS NOTIFY.
    - The slave DNS servers are notified of the zone update with a DNS NOTIFY announcement, and the slave DNS servers pull the new zone information.
    - Clear affected records from caching DNS servers.
        - A caching DNS server is used primarily for performance and fast response. The caching server obtains information from an authoritative DNS server in response to a host query and then saves (caches) the data locally. On a second or subsequent request for the same data, the caching DNS server responds with its locally stored data (the cache) until the time-to-live (TTL) value of the response expires. At this time, the server refreshes the data from the zone master. If the DNS record is changed on the primary DNS server, then the caching DNS server does not pick up the change for cached records until TTL expires. Flushing the cache forces the caching DNS server to go to an authoritative DNS server again for the updated DNS information.
    - Flush the cache if the DNS server being used supports such a capability. The following is the flush capability of common DNS BIND versions:
        - BIND 9.3.0: The command `rndc flushname` name flushes individual entries from the cache.
        - BIND 9.2.0 and 9.2.1: The entire cache can be flushed with the command `rndc flush`.
        - BIND 8 and BIND 9 up to 9.1.3: Restarting the named server clears the cache.
5. Refresh local DNS service caching.

- Some operating systems might cache DNS information locally in the local name service cache. If so, this cache must also be cleared so that DNS updates are recognized quickly.

6. The secondary site load balancer directs traffic to the secondary site middle-tier application server.