

Continuous Availability

Best Practices for Applications Using
Autonomous Database - Dedicated

White Paper / September 2, 2020

| | |
|--|----|
| Introduction | 3 |
| Overview..... | 4 |
| Application Configuration Checklist | 4 |
| Connect Using Services | 5 |
| Configure TNS/URL for High Availability | 5 |
| Use Fast Application Notification | 6 |
| Use Recommended Application Practices to Allow Draining | 8 |
| Option 1 Enabling Transparent Application Continuity..... | 8 |
| Option 2 Enabling Application Continuity..... | 10 |
| Understanding the Protection Level When Using TAC or AC | 11 |
| Configuring your Clients | 12 |
| Conclusion..... | 14 |
| Appendix: Enabling Service Attributes for Failover | 15 |
| Appendix: Enabling Service Attributes for Runtime Load Balancing..... | 16 |
| Appendix: Protection Levels Reported By PDB, Service and History | 17 |
| Appendix: Configuring Clients for FAN including Optional Wallets | 20 |
| Appendix: Additional Technical Resources from the Oracle Library | 23 |

INTRODUCTION

Applications achieve continuous service when planned maintenance, unplanned outages, and load imbalances of the database tier are hidden. A combination of application best practice, simple configuration, and the Oracle Autonomous Database will ensure that your applications are continuously available.

If the recommended application configuration solution cannot be used, alternative application configurations can be found in the white paper [Continuous Availability Application Checklist for Continuous Service for MAA Solutions](#)¹.

¹ <https://www.oracle.com/technetwork/database/options/clustering/applicationcontinuity/adb-continuousavailability-5169724.pdf>

OVERVIEW

The best approach for hiding planned maintenance activities from your applications is to transparently drain work from each database workload location prior to the maintenance window for that workload location. Oracle's connection pools and mid-tiers, including the WebLogic Server, Universal Connection Pool (UCP), OCI Session pool and ODP.NET Unmanaged Provider are Fast Application Notification (FAN²) aware and therefore are notified before database services are scheduled to move to allow graceful draining of work before maintenance. FAN notification automatically triggers closing idle connections, opening new connections in the new service location, and allows a configurable time for active work to complete in the soon-to-be-shutdown service location. The major third-party JDBC mid-tiers, such as IBM WebSphere, allow for the same behavior when configured with UCP³. For JDBC-based applications that cannot use UCP, Oracle provides solutions using Oracle Drivers and connection tests.

In order to hide unplanned outages resulting from a component or communication failure Oracle provides:

Notification - FAN is the first step to hiding outages. FAN notifies clients and breaks them out of their current network wait when an outage occurs. This avoids stalling applications for long network waits. Importantly, FAN also invokes rebalancing of sessions when services are available again.

Recovery – After the client is notified, Application Continuity (AC) or Transparent Application Continuity (TAC), re-establish a connection to a new workload location (which may be to the same or another instance in the Real Application Clusters (RAC) case, or a standby site in the Data Guard case) and replays in-flight (uncommitted) work when possible. By replaying in-flight work on the new location, the application can usually continue executing without knowing that any failure happened.

AC or TAC also executes during planned maintenance: for those sessions that do not drain (complete their current database operation) during the allocated drain interval.

APPLICATION CONFIGURATION CHECKLIST

The following checklist prepares your application for using the Oracle Autonomous Database

- Connect Using Oracle Services
- Configure Connection String/URL for High Availability
- Use Fast Application Notification (FAN)
- Use recommended application practices that allow draining
- Enable Application Continuity or Transparent Application Continuity

² Descriptions of Fast Application Notification, Application Continuity, and Transparent Application Continuity are available from the papers included in the Appendix: Additional Technical Resources, at the end of this paper

³ How to configure Oracle WebLogic Server, IBM WebSphere, IBM Liberty, Apache Tomcat and Redhat WildFly (JBoss) using UCP to hide maintenance windows and unplanned outages is detailed in papers listed in the Appendix: Additional Technical Resources

CONNECT USING SERVICES

Services provide transparency for the underlying ATP-D infrastructure. FAN, connection data, Transparent Application Continuity (TAC), Application Continuity (AC), switchover, consumer groups and many other features and operations are predicated on the use of services. The services you use also define the primary or standby role in the underlying Data Guard environment.

Oracle's Autonomous Database Transaction Processing Dedicated (ATP-D) offers five preconfigured services to choose from. All provide FAN and draining. `TPURGENT` and `TP` have TAC enabled by default in the ATP-D environment. An API is available to change the TAC or AC settings on all preconfigured services (see Appendix).

Pre-configured Services offered by the Oracle Autonomous Database

| SERVICE NAME | DESCRIPTION | DRAINING | FAN | TAC |
|--------------|--|----------|-----|-----|
| TPURGENT | OLTP Highest Priority | Yes | Yes | Yes |
| TP | OLTP General Priority (Use as main service) | Yes | Yes | Yes |
| HIGH | Reporting or Batch (Highest Priority) | Yes | Yes | |
| MEDIUM | Reporting or Batch (Medium Priority) | Yes | Yes | |
| LOW | Reporting or Batch (Lowest Priority) | Yes | Yes | |

To help in choosing the service for batch work:

HIGH: Queries run with a Degree of Parallelism equal to `CPU_COUNT`. There is a limit of three concurrent queries after which statement queuing occurs.

MEDIUM: Queries run with a Degree of Parallelism of four. The maximum number of queries that can run simultaneously is $(CPU_COUNT * 1.25)$.

LOW: Queries run serially. Queueing starts when concurrent queries exceed $(2 * CPU_COUNT)$.

CONFIGURE CONNECTION STRING/URL FOR HIGH AVAILABILITY

Oracle recommends the Connection String/URL configuration shown below when connecting to the Oracle Autonomous Database. Connect strings embedded in the Oracle-supplied wallet are configured in this fashion. Do not use Easy Connect Naming on the client because EZCONNECT has no high-availability capabilities.

Note that the standby-scan specified below refers to the SCAN address available on the standby site specified in the Active Data Guard configuration. The driver attempts to connect to the primary site first, and if the service is not available, then attempts to connect to the service at the standby. Once the connection to the service is made, at whichever site, the Oracle driver version 12.2 and later remembers the TNS address list that offers that service and gives this site priority.

Use this Connection String for ALL Oracle drivers version 12.2 or higher:

```
Alias (or URL) =
(DESCRIPTION =
(CONNECT_TIMEOUT= 90) (RETRY_COUNT=50) (RETRY_DELAY=3) (TRANSPORT_CONNECT_TIMEOUT=3)
  (ADDRESS_LIST =
    (LOAD_BALANCE=on)
    (ADDRESS = (PROTOCOL = TCP) (HOST=primary-scan) (PORT=1521)))
  (ADDRESS_LIST =
    (LOAD_BALANCE=on)
    (ADDRESS = (PROTOCOL = TCP) (HOST=standby-scan) (PORT=1521)))
(CONNECT_DATA=(SERVICE_NAME = ATP-D SERVICE)))
```

Use the following for JDBC connections using Oracle driver version 12.1 or earlier

```
Alias (or URL) =
(DESCRIPTION =
(CONNECT_TIMEOUT= 15) (RETRY_COUNT=50) (RETRY_DELAY=3)
(ADDRESS_LIST =
  (LOAD_BALANCE=on)
  (ADDRESS = (PROTOCOL = TCP) (HOST=primary-scan) (PORT=1521)))
(ADDRESS_LIST =
  (LOAD_BALANCE=on)
  (ADDRESS = (PROTOCOL = TCP) (HOST=standby-scan) (PORT=1521)))
(CONNECT_DATA=(SERVICE_NAME = ATP-D SERVICE)))
```

USE FAST APPLICATION NOTIFICATION

FAN provides immediate notification to an application in the event of an outage or resumption of service. Without FAN, applications can hang on TCP/IP timeout following hardware and network failures, and omit to rebalance when resources resume. All Oracle pools and all Oracle application servers use FAN. Third-party JAVA application servers can use UCP to enable FAN.

No application changes are required to use FAN. These are configuration changes only.

For continuous service during planned maintenance, use FAN with:

- Oracle pools or
- UCP with third-party JDBC application servers or
- The latest Oracle client drivers

For continuous service during unplanned outages, use FAN with

- Application Continuity or
- Transparent Application Continuity

FAN Coverage

FAN events are integrated with:

- Oracle Fusion Middleware and Oracle WebLogic Server
- Oracle Data Guard Broker
- Oracle JDBC Universal Connection Pool or Driver for both JDBC thin and Oracle Call Interface (OCI) interfaces
- ODP.NET Connection Pool for Unmanaged and Managed Providers
- Oracle Tuxedo
- SQL*Plus
- Oracle Database drivers for languages such as Python, Node.js and PHP
- Global Data Services
- Third party JDBC application servers using Oracle JDBC Universal Connection Pool
- Listeners

To enable FAN in the client:

Use the TNS alias or URL shown in the preceding discussion. This connection string is used to auto-configure the Oracle Notification Service (ONS) subscription at the client for FAN-event receipt when using an Oracle Database 12c or later client driver. For older drivers, refer to the FAN white paper in the Appendix for configuration details. ONS provides a secure communication path between the database tier and the client-tier allowing the client to be notified of service availability (components stopping or starting) as well as runtime load balancing advice for better work placement during normal operation.

Depending on the client, enable FAN in the application configuration properties as follows

Universal Connection Pool or JDBC thin driver (starting 12.2)

Set the property `FastConnectionFailoverEnabled`

WebLogic Active GridLink for Oracle RAC

FAN and Fast Connection Failover are enabled by default

Oracle WebLogic Server, IBM WebSphere, IBM Liberty, Apache Tomcat, Red Hat WildFly (JBoss), JDBC Applications

Use Universal Connection Pool as a connection pool replacement

ODP.Net clients (Managed and Unmanaged Providers)

Set `"HA events = true;pooling=true"` in the connect string if using ODP.Net 12.1 or earlier

Oracle Call Interface (OCI) clients and OCI-based drivers

Oracle Call Interface (OCI) clients without native settings can use an `oraacces.xml` file and set `events` to `true`

Python, Node.js and PHP have native options. In Python and Node.js you can set an `events` mode when creating a connection pool. In PHP, edit `php.ini` add the entry `oci8.events=on`

SQL*Plus enables FAN by default

In the ATP-D environment ONS offers an optional TLS (wallet-based) authentication. Depending upon the type of application (JDBC or Oracle Call Interface), the wallet configuration must follow particular rules, as described in the Appendix.

USE RECOMMENDED APPLICATION PRACTICES TO ALLOW DRAINING

Best practice for application usage is to check out connections for the time that they are needed, and then check them back in to the pool when the current action is complete. This is important to achieve good performance, for the rebalancing of work at runtime, and during maintenance windows for draining the work. Refer to the statistics section *Understanding your protection level when using TAC or AC* for an indication of how well your application follows this practice.

Oracle recommends using a FAN-aware Oracle connection pool for hiding planned maintenance. There is no impact to users when your application uses an Oracle Pool with FAN and returns connections to the pool between requests. You do not need to make any application changes to use FAN.

When an Oracle connection pool receives the FAN event for planned downtime, it marks all connections at the instance to be drained. Immediately, checked-in connections are closed so that they are not re-used. As in-use connections are returned to the pool they are closed. This allows all connections to be closed gracefully over time.

If you are using a third-party, Java-based application server, then the most effective method to achieve draining and failover is to replace the pooled data source with UCP. Many application servers support this approach, including Oracle WebLogic Server, IBM WebSphere, IBM Liberty, Apache Tomcat, Red Hat WildFly (JBoss), Spring, Hibernate, and others. White papers from Oracle and other providers, such as IBM, describe how to use UCP with these application servers. Using UCP as the data source allows UCP features such as Fast Connection Failover, Runtime Load Balancing, Application Continuity and Transparent Application Continuity to be used with full certification.

OPTION 1 ENABLING TRANSPARENT APPLICATION CONTINUITY

TAC transparently tracks and records session and transactional state so that a database session can be recovered following recoverable outages. TAC is enabled when you select the appropriate service for the Autonomous Database.

Transparent Application Continuity Coverage

Transparent Application Continuity for Oracle Autonomous Database supports the following clients:

It is strongly recommended to use the latest client drivers. Oracle Database 19c client drivers and later provide full support for TAC.

- Oracle JDBC Replay Driver 18c or later. This is a JDBC driver feature provided with Oracle Database 18c for Application Continuity
- Oracle Universal Connection Pool (UCP) 18c or later with Oracle JDBC Replay Driver 18c or later.
- Oracle WebLogic Server Active GridLink, or third-party JDBC application servers using UCP with Oracle JDBC Replay Driver 18c or later
- Java connection pools or standalone Java applications using Oracle JDBC Replay Driver 18c or later
- Oracle Call Interface Session Pool 19c or later
- SQL*Plus 19c (19.3) or later
- ODP.NET pooled, Unmanaged Driver 18c or later (“Pooling=true” default in 12.2 and later)

- Oracle Call Interface based applications using 19c OCI driver or later

Steps for using Transparent Application Continuity

REFER TO APPENDIX: ENABLING SERVICE ATTRIBUTES FOR FAILOVER

USE A SUPPORTED CLIENT (SEE COVERAGE ABOVE)

RETURN CONNECTIONS TO THE CONNECTION POOL

You do not need to make any changes to your application for identifying request boundaries if the application uses connections:

- from the Oracle connection pools or
- from the Oracle JDBC Replay Driver 18c or later or
- from the Oracle Call Interface based applications using 19c or later

When using connection pools, the application should return the connection to the pool on completion of each request. Oracle recommends that an application checks out a connection only for the time it needs it. Holding a connection when not in use is not good practice. Transparent Application Continuity with the listed drivers also detects where boundaries can be added and makes its own boundaries.

USE `FAILOVER_RESTORE`

Enabling Transparent Application Continuity automatically restores preset session states. If you find that you need preset session states in addition to the standard set, then you can register a callback⁴ or UCP label to restore these states. If you find that you need complex session states, such as temporary tables or `SYS_CONTEXT`, restored, then use Application Continuity that offers this flexibility.

ENABLE MUTABLE USE IN THE APPLICATION

Mutable functions are functions that can return a new value each time they are executed. Support for keeping the original results is provided for `SYSDATE`, `SYSTIMESTAMP`, `SYS_GUID`, and `sequence.NEXTVAL`. Application Continuity 19c and later automatically `KEEP`'s mutables for SQL, so no action is required. If you need mutables for PL/SQL, then the dba must issue the `GRANT KEEP` privilege. When `KEEP` privilege is granted, replay applies the original function result at replay.

For example:

```
SQL> GRANT [KEEP DATE TIME | KEEP SYSGUID] ... TO USER
```

```
SQL> GRANT KEEP SEQUENCE mySequence TO myUser ON sequence.object
```

SIDE EFFECTS ARE DISABLED

⁴ Refer to the whitepaper *Continuous Availability Application Checklist for Continuous Service for MAA Solutions* (<https://www.oracle.com/technetwork/database/options/clustering/applicationcontinuity/adb-continuousavailability-5169724.pdf>) for information on how to register a callback

A side effect is an external action such as sending mail, transferring files or using TCP. Transparent Application Continuity detects side effects and does not replay them. If you want the side effects replayed, then use Application Continuity that allows this extra flexibility.

OPTION 2 ENABLING APPLICATION CONTINUITY

Application Continuity is customizable, allowing you to choose to replay side effects or to add complex callbacks at failover that Transparent Application Continuity does not allow. Use Application Continuity if you are using Oracle 12c drivers (JDBC-thin or Oracle Call Interface), or you want to customize with side effects or callbacks, or have an application that uses state such as session duration temp tables and does not clean up across requests.

Application Continuity Coverage

Application Continuity for Oracle Database 19c supports the following clients:

- Oracle JDBC Replay Driver 12c or later. This is a JDBC driver feature provided with Oracle Database 12c for Application Continuity
- Oracle Universal Connection Pool (UCP) 12c or later with Oracle JDBC Replay Driver 12c or later
- Oracle WebLogic Server Active GridLink and third-party JDBC application servers using UCP with Oracle JDBC Replay Driver 12c or later
- Java connection pools or standalone Java applications using Oracle JDBC Replay Driver 12c or later with Request Boundaries or Pooled Data Source
- Applications and language drivers using Oracle Call Interface Session Pool 12c Release 2 or later
- SQL*Plus 19.3 or later
- ODP.NET pooled, Unmanaged Driver 12c Release 2 or later (`"Pooling=true";"Application Continuity=true"` default in 12.2 and later)

Steps for using Application Continuity

REFER TO APPENDIX: ENABLING SERVICE ATTRIBUTES FOR FAILOVER

USE A SUPPORTED CLIENT (SEE COVERAGE)

RETURN CONNECTIONS TO THE POOL

You do not need to make any changes to your application for identifying request boundaries if the application is using an Oracle connection pool or a third-party JDBC pool that supports request boundaries. It is best practice to use an Oracle pool and return the connections to that pool between requests. Oracle recommends that an application checks out a connection only for the time it needs it. Holding a connection when not in use is not good practice.

USE `FAILOVER_RESTORE`

Most common states are restored automatically with `FAILOVER_RESTORE=LEVEL1`. If your application presets session states in addition to the standard set, then you must register a callback, or UCP label, to restore these states. Use:

`FAILOVER_RESTORE=LEVEL1` is set on the service and

- Connection Initialization Callback for Java or the (older) TAF Callback for Oracle Call Interface or
- Universal Connection Pool or WebLogic Server Connection Labelling

ENABLE MUTABLE USE IN THE APPLICATION (SEE ENABLING TRANSPARENT APPLICATION CONTINUITY SECTION ON MUTABLES)

DECIDE IF YOU WANT TO REPLAY SIDE EFFECTS

A side effect is an external action such as sending mail, transferring files or using TCP. With Application Continuity, side effects are replayed unless the application specifies otherwise. If a request has an external action that should not be replayed, then that request can use a connection that does not have Application Continuity enabled, or replay can be disabled for that request using the `disableReplay()` API for Java or `OCIRequestDisableReplay()` for Oracle Call Interface. If you do not wish to replay all side effects, use Transparent Application Continuity.

UNDERSTANDING THE PROTECTION LEVEL WHEN USING TAC OR AC

Use the statistics for request boundaries and protection level to monitor the level of coverage to know whether your application returns connections to the pool and how well your application is protected.

Application Continuity collects statistics from the system, the session, and the service, enabling you to monitor your protection levels. The statistics are available in `V$SYSSTAT`, `V$SESSTAT`, also in `V$SERVICE_STATS` starting in later 19 versions. These statistics are saved in the Automatic Workload Repository (AWR) and are available in AWR reports and in ASH views.

The output is similar to the following:

| Statistic | Total | per Second | per Trans |
|----------------------------------|-----------|------------|-----------|
| ----- | ----- | ----- | ----- |
| cumulative begin requests | 1,500,000 | 14,192.9 | 2.4 |
| cumulative end requests | 1,500,000 | 14,192.9 | 2.4 |
| cumulative user calls in request | 6,672,566 | 63,135.2 | 10.8 |
| cumulative user calls protected | 6,672,566 | 63,135.2 | 10.8 |

TIP: To report the protection level by PDB or using historic data, see the Appendix for example SQL to use.

TAC or AC are enabled and protecting your application when

- Cumulative user calls in request = cumulative user calls protected
- And the above numbers are not equal to zero

This protection level is measured inside the database. The client may need to use an `ORDER BY` clause in queries and preset the initial session state if this contains state not covered by `FAILOVER_RESTORE` to achieve this level of protection.

The example above shows an increasing number of Begin and End requests. The number itself will depend on how frequently your application checks out and checks into the connection pool, or the request boundaries that the database can discover when using TAC. The rate of increase of these values will depend on the rate your requests are being submitted. You can compute the percentage of user calls being protected using:

$$\text{Percentage of Protected Calls} = \text{cumulative user calls protected} / \text{cumulative user calls in request} * 100$$

It is possible that the percentage of protected calls is less than 100%. You may be using JDBC concrete classes, side effects are disabled, unrecoverable state may be being used, or the application may choose to disable Application Continuity for certain requests. If your application is not 100% protected, the `ORAchk`⁵ component `acchk` can be used, at your own site, to show where

⁵ For information about downloading, configuring and running `ORAchk` utility, refer to My Oracle Support note [1268927.2](https://support.oracle.com/ep6/faces/aces.xhtml?_afPfm=1268927.2)

in your application coverage is below 100%. Your management can decide whether to follow the advisor or take no action by evaluating the impact.

CONFIGURING YOUR CLIENTS

JDBC-thin Driver Checklist

1. Ensure all recommended patches are applied at the client. Refer to the MOS note *Client Validation Matrix for Application Continuity (Doc ID 2511448.1)*
2. Use JDBC Statement Cache for Coverage and Performance

For best coverage and performance, use the JDBC driver statement cache in place of an application server statement cache. This allows the driver to know that statements are closed and memory to be freed at the end of requests.

To use the JDBC statement cache, use the connection property `oracle.jdbc.implicitStatementCacheSize` (`OracleConnection.CONNECTION_PROPERTY_IMPLICIT_STATEMENT_CACHE_SIZE`). The value for the cache size matches your number of `open_cursors`. For example:

```
oracle.jdbc.implicitStatementCacheSize=nnn where nnn is typically between 10 and 100 and is equal to the number of open cursors your application maintains.
```

3. Tune the Garbage Collector

For many applications the default Garbage Collector tuning is sufficient. For applications that return and keep large amounts of data you can use higher values, such as 2G or larger. For example:

```
java -Xms3072m -Xmx3072m
```

It is recommended to set the memory allocation for the initial Java heap size (`ms`) and maximum heap size (`mx`) to the same value. This prevents using system resources on growing and shrinking the memory heap.

4. JDBC Concrete Classes

For JDBC applications, Oracle does not support deprecated `oracle.sql` concrete classes BLOB, CLOB, BFILE, OPAQUE, ARRAY, STRUCT or ORADATA. (See MOS note [1364193.1](#) *New JDBC Interfaces*). Use `ORAchk -acchk` on the client to know if an application passes⁶. The list of restricted concrete classes for JDBC Replay Driver is reduced to the following starting with Oracle JDBC-thin driver version 18c and later: `oracle.sql.OPAQUE`, `oracle.sql.STRUCT`, `oracle.sql.ANYDATA`

5. Configure Fast Connection Failover (FCF)

For client drivers 12c and later

- Use the recommended Connection String or URL for auto-configuration of ONS
- Check that `ons.jar` (plus optional WALLET jars, `osdt_cert.jar`, `osdt_core.jar`, `oraclepki.jar`) are on the CLASSPATH
- Set the pool or driver property `fastConnectionFailoverEnabled=true`

⁶ For more information on `ORAchk -acchk` refer to blog [Using Orachk to Clean Up Concrete Classes for Application Continuity](#)

- For third party JDBC pools, UCP is recommended
- Open port 6200 for ONS (6200 is the default port, a different port may have been chosen)

For client drivers prior to 12c use the addresses provided:

Set `oracle.ons.nodes =XXX01:6200, XXX02:6200, XXX03:6200`

Oracle Call Interface (OCI) Driver Checklist

1. Ensure all recommended patches are applied at the client. Refer to the MOS Note *Client Validation Matrix for Application Continuity (Doc ID 251148.1)*
2. Replace `OCIStmtPrepare` with `OCIStmtPrepare2`. `OCIStmtPrepare()` has been deprecated since 12.2. All applications should use `OCIStmtPrepare2()`. TAC and AC allow `OCIStmtPrepare` but do not replay this statement.

<https://docs.oracle.com/en/database/oracle/oracle-database/19/lnoci/deprecated-oci-functions.html#GUID-FD74B639-8B97-4A5A-BC3E-269CE59345CA>

3. To use FAN for OCI-based applications, do the following:
 - ATP-D presets `aq_ha_notifications` on the services
 - Use the recommended Connection String for auto-configuration of ONS
 - Set `auto_config`, `events`, and `wallet_location` (optional) in `oraaccess.xml` (See Appendix)
 - Link the application with the O/S client thread library
 - Open port 6200 for ONS (6200 is the default port, a different port may have been chosen)

For client drivers prior to 12c copy the addresses provided in `oraaccess.xml`.

ODP.NET Unmanaged Provider Driver Checklist

1. Ensure all recommended patches are applied at the client. Refer to the MOS Note *Client Validation Matrix for Application Continuity (Doc ID 251148.1)*
2. To use FAN for Oracle Call Interface based applications, do the following:
 - ATP-D presets `aq_ha_notifications` on the services
 - Use Recommended Connection String for auto-ons
 - Set `onsConfig` and `wallet_location` (optional) in `oraaccess.xml` (See Appendix)
 - Open port 6200 for ONS (6200 is the default port, a different port may have been chosen)
 - Set FAN, in the connection string -

```
"user id=oracle; password=oracle; data source=HA; pooling=true; HA events=true;"
```

- (optional) Set Runtime Load Balancing, also in the connection string -

```
"user id=oracle; password=oracle; data source=HA; pooling=true; HA events=true; load balancing=true;"
```

CONCLUSION

The Oracle Autonomous Database is configured and managed for high availability on your behalf. No additional configuration or management is required by you.

There are a few simple steps to achieving Continuous Availability for your applications:

- Select the ATP-D service that is appropriate for your SLA's
- Configure Fast Application Notification (FAN)
- Use the recommended connection string for your applications
- Use application best practices to optimize for draining
- Use Transparent Application Continuity or Application Continuity for continuous service

By following these five simple steps, planned maintenance activities will no longer require outages and unplanned events will rarely result in failed transactions and interruptions to service.

APPENDIX: ENABLING SERVICE ATTRIBUTES FOR FAILOVER

Transparent Application Continuity is the default for the preconfigured services `TPURGENT` and `TP`, nothing needs to be done. The `DEFAULT` value for `FAILOVER_RESTORE` is `AUTO` when using `TPURGENT` and `TP`.

You can change the failover type offered on your service by using the generic package `DBMS_APP_CONT_ADMIN`. Use this API to enable Application Continuity, Transparent Application Continuity or Transparent Application Failover (TAF), or to disable failover altogether. New sessions will use the new failover type.

To use these procedures you must have been granted the role `PDBADMIN`.

To enable Transparent Application Continuity for your service:

```
execute DBMS_APP_CONT_ADMIN.ENABLE_TAC(`HIGH`);
```

To enable Application Continuity for your service:

```
execute DBMS_APP_CONT_ADMIN.ENABLE_AC(`TPURGENT`);
```

To enable TAF SELECT for your service::

```
execute DBMS_APP_CONT_ADMIN.ENABLE_TAF(`LOW`);
```

To enable TAF BASIC for you service:

```
execute DBMS_APP_CONT_ADMIN.ENABLE_TAF(`MEDIUM`, `SESSION`);
```

To disable failover for your service:

```
execute DBMS_APP_CONT_ADMIN.DISABLE_FAILOVER(`HIGH`);
```

If you wish to use TAF without modifying services, use the older client-side configuration for TAF in your connection string:

```
(FAILOVER_MODE=(TYPE=select) (METHOD=basic) (OVERRIDE=TRUE))
```

APPENDIX: ENABLING SERVICE ATTRIBUTES FOR RUNTIME LOAD BALANCING

Use Runtime Load Balancing to load balance OLTP and batch applications. It is only suitable for services that run on more than one instance at a time. `NONE` is the default setting. To use these procedures you must have been granted the `PDBADMIN` role.

For OLTP-style applications, enable runtime load balancing where the service response time is used to direct work:

```
execute DBMS_APP_CONT_ADMIN.SET_LOAD_BALANCING_GOAL(`TPURGENT`, `SERVICE_TIME`);
```

For batch-style applications enable runtime load balancing where service throughput is used to direct work

```
execute DBMS_APP_CONT_ADMIN.SET_LOAD_BALANCING_GOAL(`HIGH`, `THROUGHPUT`);
```

To disable load balancing at runtime:

```
execute DBMS_APP_CONT_ADMIN.SET_LOAD_BALANCING_GOAL(`HIGH`, `NONE`);
```


APPENDIX: PROTECTION LEVELS REPORTED BY PDB, SERVICE AND HISTORY

To report protection by PDB, use the following example:

```
set lines 85
col Service_name format a30 trunc heading "Service"
break on con_id skip1
col Total_requests format 999,999,9999 heading "Requests"
col Total_calls format 9,999,9999 heading "Calls in requests"
col Total_protected format 9,999,9999 heading "Calls Protected"
col Protected format 999.9 heading "Protected %"

select con_id, total_requests,
total_calls,total_protected,total_protected*100/NULLIF(total_calls,0) as Protected
from(
select * from
(select  s.con_id, s.name, s.value
  FROM   GV$CON_SYSSTAT s, GV$STATNAME n
 WHERE  s.inst_id   = n.inst_id
 AND    s.statistic# = n.statistic#
 AND    s.value     != 0 )
pivot(
  sum(value)
  for name in ('cumulative begin requests' as total_requests, 'cumulative end requests' as
Total_end_requests, 'cumulative user calls in requests' as Total_calls, 'cumulative user
calls protected by Application Continuity' as total_protected)
))
order by con_id;
```

To report protection by service, use the following example:

```
set pagesize 60
set lines 120
col Service_name format a30 trunc heading "Service"
break on con_id skip1
col Total_requests format 999,999,9999 heading "Requests"
col Total_calls format 9,999,9999 heading "Calls in requests"
col Total_protected format 9,999,9999 heading "Calls Protected"
col Protected format 999.9 heading "Protected %"

select con_id, service_name, total_requests,
total_calls, total_protected, total_protected*100/NULLIF(total_calls,0) as Protected
from(
select * from
(select a.con_id, a.service_name, c.name, b.value
FROM gv$session a, gv$sesstat b, gv$statname c
WHERE a.sid = b.sid
AND a.inst_id = b.inst_id
AND b.value != 0
AND b.statistic# = c.statistic#
AND b.inst_id = c.inst_id
AND a.service_name not in ('SYS$USERS', 'SYS$BACKGROUND'))
pivot(
sum(value)
for name in ('cumulative begin requests' as total_requests, 'cumulative end requests' as
Total_end_requests, 'cumulative user calls in requests' as Total_calls, 'cumulative user
calls protected by Application Continuity' as total_protected) ))
order by con_id, service_name;
```

To report protection history over last three days, use the following example:

```

set lines 85
col Service_name format a30 trunc heading"Service"
break on con_id skip1
col Total_requests format 999,999,9999 heading "Requests"
col Total_calls format 9,999,9999 heading "Calls in requests"
col Total_protected format 9,999,9999 heading "Calls Protected"
col Protected format 999.9 heading "Protected %"

set lines 85
col Service_name format a30 trunc heading"Service"
break on con_id skip1
col Total_requests format 999,999,9999 heading "Requests"
col Total_calls format 9,999,9999 heading "Calls in requests"
col Total_protected format 9,999,9999 heading "Calls Protected"
col Protected format 999.9 heading "Protected %"

select  a.instance_number,begin_interval_time, total_requests, total_calls, total_protected,
total_protected*100/NULLIF(total_calls,0) as Protected
from(
select * from
(select  a.snap_id, a.instance_number,a.stat_name, a.value
FROM    dba_hist_sysstat a
WHERE   a.value      != 0 )
pivot(
sum(value)
for stat_name in ('cumulative begin requests' as total_requests, 'cumulative end requests'
as Total_end_requests, 'cumulative user calls in requests' as Total_calls, 'cumulative user
calls protected by Application Continuity' as total_protected)
)) a,
dba_hist_snapshot b
where a.snap_id=b.snap_id
and a.instance_number=b.instance_number
and begin_interval_time>systemtimestamp - interval '3' day
order by a.snap_id,a.instance_number;

```

APPENDIX: CONFIGURING CLIENTS FOR FAN INCLUDING OPTIONAL WALLETS

Wallet-based authentication is an option for FAN when using ATP-D. Use the same wallet as for the TNS connection.

For JDBC applications:

1. Ensure the following jar files are present in the application's CLASSPATH
(ons.jar, osdt_cert.jar, osdt_core.jar, oraclepki.jar)
2. Specify the wallet for FAN in one of the following ways:
 - o To use auto-configured ONS with wallets, set the following Java system properties:
"-Doracle.ons.walletfile=/replace this with host path/onswallet"
"-Doracle.ons.walletpassword=myONSWalletPassword"

Note that these cannot be set on a per-pool or per-connection basis

- o To explicitly set ONS do one of the following:
 - Set explicitly using an UCP XML Configuration file. For example:

```
<!--?xml version="1.0" encoding="UTF-8"? -->
<ucp-properties>
  <connection-pool
    connection-pool-name="UCP_pool1"
    user="dbuser"
    password="dbuserpasswd"
    connection-factory-class-name="oracle.jdbc.pool.OracleDataSource"
    initial-pool-size="10"
    min-pool-size="5"
    max-pool-size="15"
    validate-connection-on-borrow="true"
    connection-wait-timeout="900"
    max-connections-per-service="50"
    sql-for-validate-connection="select 1 from dual"
    url="jdbc:oracle:thin:@(DESCRIPTION =(CONNECT_TIMEOUT= 120) (RETRY_COUNT=20)
(RETRY_DELAY=3) (TRANSPORT_CONNECT_TIMEOUT=3) (ADDRESS_LIST =(LOAD_BALANCE=on) (ADDRESS
= (PROTOCOL = TCP) (HOST=primary-scan) (PORT=1521))) (ADDRESS_LIST
=(LOAD_BALANCE=on) (ADDRESS = (PROTOCOL = TCP) (HOST=standby-
scan) (PORT=1521))) (CONNECT_DATA=(SERVICE_NAME = MY-SERVICE)))"
    fastConnectionFailoverEnabled="true"
    onsConfiguration="nodes=primary-scanhost:6200,secondary-
scanhost:6200\nwalletfile=/replace_with_host_path/onswallet\nwalletpassword=myWalletP
assword">
  </connection-pool>
</ucp-properties>
```

- Set programmatically from within UCP, using a call to `setONSConfiguration()`, for example:

```
pds.setONSConfiguration("nodes=primary-scanhost:6200,secondary-  
scanhost:6200\nwalletfile=/replace_this_with_host_path/onswallet\nwalletpassword=myWa  
lletPassword");
```

For Oracle Call Interface (OCI) Applications Using Oracle driver version 12.2 or more recent

Add the following to the `<default_parameters>` section of the `oraaccess.xml` file:

```
<default_parameters>  
  (Other settings may be present in this section)  
  <events>  
    true  
  </events>  
  <ons>  
    <auto_config>true</auto_config>  
    <wallet_location>/my_path/onswallet</wallet_location>  
  </ons>  
</default_parameters>
```

The `<wallet_location>` path should be the name of the directory containing the wallet.

Other parameters may be set in the `ons` section of `oraaccess.xml`, including `<hosts>`, `<max_connections>`, and `<subscription_wait_timeout>`.

Drivers that support native event setting controls may omit the `<events>` section and use the driver setting instead.

By default, application connections to the database will succeed even if ONS fails. This allows the application to continue running. To validate your initial ONS configuration, you can force an Oracle error to be thrown when an application attempts to connect to the database but ONS isn't working. Add a section to `oraaccess.xml` at the same level as `<ons>`. This should be used for diagnostic purposes only:

```
<fan>  
  <subscription_failure_action>  
    error  
  </subscription_failure_action>  
</fan>
```

Place the `oraaccess.xml` file in the same directory as the `tnsnames.ora` and `sqlnet.ora` network files. For example, when using Oracle Instant Client these files might be in the default directory `network/admin`. Alternatively, all network configuration files can be put in another accessible directory. Then set the environment variable `TNS_ADMIN` to that directory name.

ODP.Net Managed provider

Use the `application.config` file to specify ONS configuration and wallet location. For example:

```
<oracle.manageddataaccess.client>
  <version number="*">
    <onsConfig mode="remote">
      <settings>
        <setting name="Protocol" value="TCPS" />
        <setting name="WALLET_LOCATION" value="C:\myPath\ONS_SSLWallet" />
      </settings>
      <ons database="atp01db">
        <add name="nodeList" value="racNode1:6205,racNode2:6205,racNode3:6205" />
      </ons>
    </onsConfig>
  </version>
</oracle.manageddataaccess.client>
```

APPENDIX: ADDITIONAL TECHNICAL RESOURCES FROM THE ORACLE LIBRARY

FAST APPLICATION NOTIFICATION

<http://www.oracle.com/technetwork/database/options/clustering/applicationcontinuity/learnmore/fastapplicationnotification12c-2538999.pdf>

EMBEDDING UCP WITH JAVA APPLICATION SERVERS

WLS UCP Datasource, <https://blogs.oracle.com/weblogicserver/wls-ucp-datasource>

Design and Deploy WebSphere, IBM Liberty Applications for Planned, Unplanned Database Downtimes and Runtime Load Balancing with UCP (<http://www.oracle.com/technetwork/database/application-development/planned-unplanned-rlb-ucp-WebSphere, IBM Liberty-2409214.pdf>)

Reactive programming in microservices with MicroProfile on Open Liberty 19.0.0.4
(<https://openliberty.io/blog/2019/04/26/reactive-microservices-microprofile-19004.html#oracle>)

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-ucp-tomcat-2265175.pdf>).

Using Universal Connection Pool with WildFly (JBoss) AS ([https://blogs.oracle.com/dev2dev/using-universal-connection-pooling-ucp-with-WildFly \(JBoss\)-as](https://blogs.oracle.com/dev2dev/using-universal-connection-pooling-ucp-with-WildFly (JBoss)-as))

APPLICATION CONTINUITY

Application Continuity for Oracle 12c
(<http://www.oracle.com/technetwork/database/options/clustering/applicationcontinuity/overview/application-continuity-wp-12c-1966213.pdf>)

Ensuring Application Continuity (<https://docs.oracle.com/en/database/oracle/oracle-database/18/racad/ensuring-application-continuity.html#GUID-C1EF6BDA-5F90-448F-A1E2-DC15AD5CFE75>)

TRANSPARENT APPLICATION CONTINUITY

<https://docs.oracle.com/en/database/oracle/oracle-database/18/adfns/high-availability.html#GUID-96599425-9BDA-483C-9BA2-4A4D13013A37>

TRANSACTION GUARD

Transaction Guard (<http://www.oracle.com/technetwork/database/database-cloud/private/transaction-guard-wp-12c-1966209.pdf>)

ORACLE CORPORATION

Worldwide Headquarters

500 Oracle Parkway, Redwood Shores, CA 94065 USA

Worldwide Inquiries

TELE + 1.650.506.7000 + 1.800.ORACLE1

FAX + 1.650.506.7200

oracle.com

CONNECT WITH US

Call +1.800.ORACLE1 or visit oracle.com. Outside North America, find your local office at oracle.com/contact.

 blogs.oracle.com/oracle

 facebook.com/oracle

 twitter.com/oracle

Integrated Cloud Applications & Platform Services

Copyright © 2020, 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. 0920

White Paper Title Continuous Availability: Best Practices for Applications Using Autonomous Database - Dedicated
May 2019

Authors: Troy Anthony and Carol Colrain

Contributors: Lawrence To, Ian Cookson, Hector Pujol, Hairong Qin

 Oracle is committed to developing practices and products that help protect the environment

ORACLE®