19c ORACLE Database

# Continuous Availability

## Best Practices for Adopting Transparent Application Continuity

White Paper / December 22 2020

ORACLE

## 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 using the Oracle MAA approach 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 to 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 (FAN[2]) aware and therefore will be notified before database services are scheduled to move to allow graceful draining of work before maintenance. Fan notification will automatically trigger closing idle connections, opening new connections in the new service location and allowing a configurable time for active work to complete in the soon-to-be-shutdown service location. The major third-party mid-tiers, such as IBM WebSphere, allow for the same behavior when configured with UCP[3]. For older 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. It notifies clients and breaks them out of their current network wait when an outage occurs. This avoids stalling applications for long network waits.
- Recovery – After the client is notified, Application Continuity (AC) or Transparent Application Continuity (TAC), reestablish 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) prior to the maintenance window.

## APPLICATION CONFIGURATION CHECKLIST

The following checklist prepares your application for continuous service with the Oracle Database

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

## 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.

As an example, the Oracle Autonomous Database Transaction Processing Dedicated (ATP-D) offers five preconfigured services to choose from. In this example, all services provide FAN and draining. The OLTP services have TAC enabled by default in the ATPD environment. These services should be configured specifically for your workload and draining expectations.

[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 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

**Example Services offered by the Oracle Autonomous Database**

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

## CONFIGURE TNS/URL FOR HIGH AVAILABILITY

Oracle recommends the TNS/URL configuration shown below when connecting to the Oracle 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 TNS for ALL Oracle drivers version 12.2 or higher:

```
Alias (or URL) =
(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 = YOUR SERVICE)))
```

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

```
Alias (or URL) =
(DESCRIPTION =
(CONNECT_TIMEOUT= 15)(RETRY_COUNT=20)(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 = YOUR 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 for both JDBC thin and OCI interfaces
- ODP.NET Connection Pool for Unmanaged and Managed Providers
- SQLPLUS
- PHP
- Global Data Services

To enable FAN in the client:

1. 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.

2. Depending on the client, enable FAN (FCF) in the application configuration properties as follows

    - Universal Connection Pool

- Set the property `FastConnectionFailoverEnabled`

- WebLogic Active GridLink for Oracle RAC
  - FAN and Fast Connection Failover are enabled by default
- IBM WebSphere, IBM Liberty, IBM Liberty, Apache Tomcat, Red Hat WildFly (WildFly (JBoss)), third-party Application Server
  - 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
- OCI clients
  - OCI clients using `oraacces.xml` set `events` to `true` o SQL*Plus enables FAN by default
- PHP
  - In `php.ini` add the entry `oci8.events=on`

3. In secured environments ONS uses either TLS (wallet-based) authentication or a direct connection over the Virtual Cloud Network (VCN). Depending upon the type of application (JDBC or OCI), 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 IBM WebSphere, IBM Liberty, IBM Liberty, Apache Tomcat, Red Hat WildFly (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 must be enabled on the services by the dba.

### TRANSPARENT APPLICATION CONTINUITY COVERAGE

Transparent Application Continuity supports the following clients:

- 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 18c, 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 with Request Boundaries
- OCI Session Pool 18c or later
- SQL*Plus 18c or later
- ODP.NET pooled, Unmanaged Driver 18c or later ("`Pooling=true`" default in 12.2 and later)
- OCI-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 OCI-based applications using 19c or later

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. Transparent Application Continuity 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 callback[4] 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:

```
GRANT [KEEP DATE TIME | KEEP SYSGUID] … TO USER
```

```
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 OCI), 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.

### 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 12c, 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
- OCI Session Pool 12c Release 2 or later
- SQL*Plus 12c Release 2 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 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. Transparent Application Continuity also detects where boundaries can be added and makes its own boundaries.

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 OCI 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 OCI. 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` and, when service statistics are enabled, in `V$SERVICE_STATS`. These statistics are saved in the Automatic Workload Repository (AWR) and are available in AWR reports.

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
```

TAC or AC are enabled and protecting your application when

- Cumulative user calls in request = cumulative user calls protected AND
- Above numbers are not equal to zero

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-in to the connection pool or what request boundaries 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:

Percentage of Protected Calls = cumulative user calls protected / 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 in your application coverage is below 100%. Your management can decide whether to follow the advisor or take no action by evaluating the impact.

## ADDITIONAL CONSIDERATIONS

### 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)*

[5] For information about downloading, configuring and running ORAchk utility, refer to My Oracle Support note 1268927.2

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 passes[6]. 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`

OCI (ORACLE CALL INTERFACE) 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. To use FAN for OCI-based applications, do the following:

    - On the service, the dba must set `AQ_HA_NOTIFICATIONS=true`
    - Link with a thread library. Most applications use thread libraries by default.
    - Initialize the OCI environment in `OCI_EVENTS` mode

## CONCLUSION

The Oracle 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:

- Use a database 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

[6] For more information on ORAchk –acchk refer to blog Using Orachk to Clean Up Concrete Classes for Application Continuity

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

## APPENDIX: CONFIGURING YOUR SERVICES

When using Oracle Database on premise, you can create services on Oracle RAC that use Transparent Application Continuity, or Application Continuity, or TAF. You can use roles to distinguish whether the services are active on Active Data Guard or the primary database. The following examples illustrate this:

Transparent Application Continuity

```
srvctl add service -db mydb -service SILVER -preferred serv1 -available serv2 -
failover_restore AUTO -failoverretry 30 -failoverdelay 10 commit_outcome TRUE -
failovertype AUTO -replay_init_time 1800 retention 86400 -notification TRUE -
drain_timeout 300 -stopoption IMMEDIATE
```

Application Continuity

```
srvctl add service -db mydb -service GOLD -preferred serv1 -available serv2 -
failover_restore LEVEL1 -failoverretry 30 -failoverdelay 10 commit_outcome TRUE -
failovertype TRANSACTION -replay_init_time 1800 -retention
86400 -notification TRUE -drain_timeout 300 -stopoption IMMEDIATE
```

To add with the Data Guard role, here is the TAC example:

```
srvctl add service -db mydb -service GOLD -preferred serv1 -available serv2 -
failover_restore AUTO -failoverretry 30 -failoverdelay 10 commit_outcome TRUE -
failovertype AUTO -replay_init_time 1800 -retention 86400 -notification TRUE   -role
PHYSICAL_STANDBY -drain_timeout 300 stopoption IMMEDIATE
```

## APPENDIX: CONFIGURING WALLET-BASED AUTHENTICATION

**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:

   - 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

   - To explicitly set ONS do one of the following:

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

```
pds.setONSConfiguration("nodes=primary-
scanhost:6200,secondaryscanhost:6200\nwalletfile=/replace_this_with_host_path/onswallet\
nwalletpassword=myWalletPasswo rd");
```

OR 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,secondaryscanhost:6200\nwalletfile=/replace_with_host_path/onswallet\nwall
etpassword=myWalletPassword">
  </connection-pool>
</ucp-properties>
```

FOR OCI APPLICATIONS USING ORACLE DRIVER VERSION 12.2 OR MORE RECENT

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

```
<default_parameters>
      (Other settings may be present in this section)
   <ons>
      <auto_config>true</auto_config>
      <wallet_location>/replace with host path/onswallet</wallet_location>
</ons>
</default_parameters>
```

2. Place the `oraaccess.xml` file in the same directory as the `tnsnames.ora` and `sqlnet.ora` network files

## 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 OTHER APPLICATION SERVERS

*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-rlbucp-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-tomcat2265175.pdf).

*Using Universal Connection Pool with WildFly (JBoss) AS* (https://blogs.oracle.com/dev2dev/using-universal-connectionpooling-ucp-with-WildFly (JBoss)-as)

### APPLICATION CONTINUITY

*Application Continuity for Oracle 12c* (http://www.oracle.com/technetwork/database/options/clustering/applicationcontinuity/overview/application-continuity-wp12c-1966213.pdf)

*Ensuring Application Continuity* (https://docs.oracle.com/en/database/oracle/oracle-database/18/racad/ensuringapplication-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-483C9BA2-4A4D13013A37

### 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



Title: Continuous Availability: Best Practices for Adopting Transparent Application Continuity May, 2019
Author: Carol Colrain, Troy Anthony, Ian Cookson
Contributing Authors: Lawrence To


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