ORACLE

Planned Database Maintenance of Oracle Autonomous Database with JD Edwards EnterpriseOne

Configuration and usage of drain time for planned database maintenance of Oracle Autonomous Database with JD Edwards EnterpriseOne

July, 2020 | Version 1.03

PURPOSE STATEMENT

The purpose of this paper is to provide guidelines for planning Oracle database maintenance using Oracle Autonomous Database (ADB) with JD Edwards EnterpriseOne. Usage of the Oracle ADB drain time is discussed in length in this document and provides a foundation for using the automated maintenance features of Oracle ADB to allow near zero downtime for the JD Edwards application during the Oracle patching window.

DISCLAIMER

TABLE OF CONTENTS

LIST OF IMAGES

LIST OF TABLES

EXECUTIVE SUMMARY

In today's landscape, business requirements are always evolving. IT decision makers are looking for products that improve their employees' efficiency throughout the life cycle of enterprise resource planning (ERP) administration.

Database maintenance is one of the key activities of ERP environment administration. Maintenance for the database can require application downtime so that database patches and upgrades can be performed. Application downtime is required to maintain data integrity and to ensure that no active EnterpriseOne database transactions occur during the planned maintenance. Without downtime, issues such as integrity, data corruption, and partial completion of EnterpriseOne application database transactions can occur. However, the issue of downtime is not unique to the EnterpriseOne software. This document will provide a strategy to overcome downtime during database maintenance.

Oracle Autonomous Database (ADB) is the world's first autonomous data management system in the cloud to deliver automated patching, upgrades, and tuning. As part of these processes, all routine database maintenance tasks are performed without any human intervention. Oracle JD Edwards EnterpriseOne customers can benefit from the self-driving, self-securing, and self-repairing capabilities of the Oracle ADB. When an Oracle ADB patch becomes available, you simply specify a start time for it to be applied.

This white paper describes the Oracle ADB ideal service drain time for the Oracle ADB connections to JD Edwards EnterpriseOne to achieve an architecture for database patching with near zero downtime. This white paper outlines the steps involved in the patching process and illustrates the update of a two-node Oracle ADB configured with Oracle Real Application Cluster (RAC) in an EnterpriseOne architecture. The EnterpriseOne architecture used for the testing described in this white paper is Release 9.2 Update 4 of EnterpriseOne Applications with Tools Release 9.2.4.3. The testing and implementation of the contents of this document was performed on the Oracle Cloud Infrastructure.

Application continuity service drain time is the time taken for active connections on one node of an Oracle ADB RAC cluster to migrate to another node in that cluster. This movement of database connections is what facilitates the EnterpriseOne application to function continuously during a planned database maintenance patching window. This document covers drain time in some depth and provides guidelines to determine the ideal drain time settings on the Oracle ADB.

OVERVIEW OF ORACLE ADB PATCHING ON ENTERPRISEONE

The application continuity service drain time configuration has been available since Oracle 12c but has been limited to a default setting of 5 minutes. In Oracle 19c ADB, the application continuity service drain time has been extended from 5 minutes to 8 hours. The extended drain time enables JD Edwards EnterpriseOne customers to apply a database patching strategy for their applications with nearly zero downtime.

Oracle ADB patching is performed by a rolling patching mechanism. In this patching methodology, the Oracle RAC nodes are brought down and patched one after another ensuring that active nodes are always available to fulfill database requests from the EnterpriseOne application.

A summary of the recommendations of this document are given below and are explained in more detail in the later sections of the document:

Recommendation 1: Choose an Oracle ADB maintenance patching window in a time frame of low EnterpriseOne activity to minimize the impact on business continuity.

Recommendation 2: When you set the ideal drain time, take into consideration the 20–30 minutes of Oracle ADB preparation time for the database patch, so critical EnterpriseOne applications may complete prior to the Oracle patching window. Adjust the ideal drain time to avoid an EnterpriseOne process running in the business continuity exposure window whenever possible.

Recommendation 3: Set the ideal drain time to be as long as the longest running EnterpriseOne batch process that is likely to be initiated at approximately the same time as the scheduled Oracle ADB patching or upgrade event. EnterpriseOne interactive user processes are much shorter in duration than batch processes and so batch processing time will most likely be considered to determine the ideal drain time.

Recommendation 4: During the Oracle patching window, the Oracle ADB RAC node availability is diminished. It is recommended that customers decrease their load of EnterpriseOne during this time period as to avoid any resource contention in this temporary diminished capacity.

Recommendation 5: Initiate an EnterpriseOne heartbeat script as part of the Oracle ADB patching process.

Recommendation 6: Increasing the EnterpriseOne Server JDENetTimeout value temporarily will enable EnterpriseOne batch processes to remain in a waiting state and provide enough time for a new EnterpriseOne kernel process to be started. The default JDENetTimeout value is 60,000 milliseconds (ms) and should not be increased to a value higher than 180,000 ms to avoid issues with long-running business functions.

This document covers the steps to enable the Oracle ADB to automatically apply patches or updates while ensuring continuous EnterpriseOne business process flows without the need for manual restart of either the EnterpriseOne services on the Enterprise Server or the restart of any of the web-based services supplied by the EnterpriseOne HTML Server.

The testing was limited to EnterpriseOne interactive user and batch processes and did not cover all the possible conditions that might be configured in an EnterpriseOne architecture. Therefore, it is suggested to perform testing in a customer environment before the patching is implemented in a full production environment.

The following sections of this document describe the aspects of the Oracle patching process that is followed for EnterpriseOne software.

1. EnterpriseOne Kernel Processes and Oracle ADB Connections
2. Setting the Oracle ADB Patching Schedule
3. Understanding Oracle ADB Drain Time
4. Limiting the Business Continuity Exposure Window

The following appendixes in this document describe the technical details of the Oracle database patching process with EnterpriseOne.

1. Testing Use Cases
2. Business Continuity Script

3. Setting and Validating Oracle Drain Timeout
4. SQL Script to Determine Oracle RAC Connections

ENTERPRISEONE KERNEL PROCESSES AND ORACLE ADB CONNECTIONS

The EnterpriseOne application establishes two types of interface connections for all the interactive and batch processes connecting to the Oracle ADB.

1. Oracle Call Interface database connection from the Enterprise Server, and more specifically from the EnterpriseOne kernel processes running on that server to the Oracle ADB
2. Java Database Connection (JDBC) interface that provides connectivity between the Oracle ADB and the EnterpriseOne HTML Server

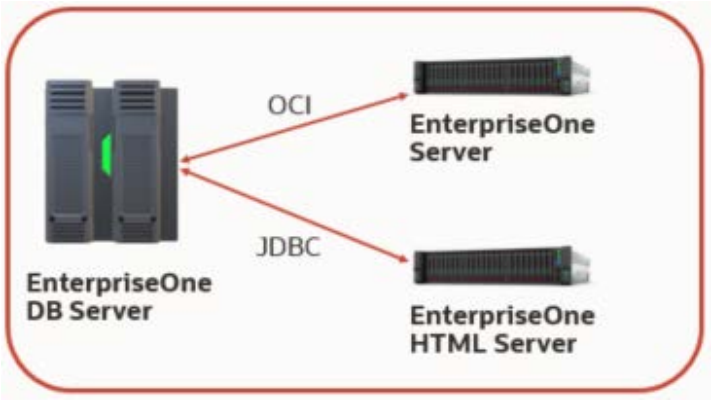

Image Caption 1. EnterpriseOne Database Connections

Although the EnterpriseOne HTML Server can make direct database requests to the Oracle ADB through JDBC, it may also initiate requests to the Enterprise Server to process additional EnterpriseOne logic. These requests are taken up by the Enterprise Server kernel processes. The additional logic processing on the Enterprise Server of the request that originated on the EnterpriseOne HTML Server may further require information from the Oracle ADB. Such database requests are made through the Oracle Call Interface.

Static and Dynamic Database Connections

EnterpriseOne uses connection pools to create database connections to the Oracle ADB. JD Edwards EnterpriseOne establishes connections to the Oracle ADB database through the Oracle Call Interface and the JDBC interface at process initiation and maintains those connections to the Oracle ADB for the entire EnterpriseOne process life cycle.

In case of any interruption to the database connection or Oracle connection sessions therein, the connection or session must be recovered or restarted. Understanding how static and dynamic EnterpriseOne database connections work and their impact on the JD Edwards system is important in the implementation of a proper database patching process.

The table below outlines the two types of connections that the EnterpriseOne kernel processes to the Oracle ADB establish on the Enterprise Server through the Oracle Call Interface.

EnterpriseOne Kernel Process Definitions

EnterpriseOne Kernel Process	Kernel Designation	Connection Type	Function
UBE	DEF2	Dynamic	Processes JD Edwards EnterpriseOne batch process requests
Security	DEF4	Static	Processes security server requests and provides user list information to the Server Manager Console
Call Object	DEF6	Dynamic	Processes XML specification access requests

EnterpriseOne Kernel Process	Kernel Designation	Connection Type	Function
			Note: A specification request is a complete description of an EnterpriseOne object needed to process that request.
SAW	DEF10	Static	Communicates runtime information to the Server Manager Console
Package Build	DEF11	Dynamic	Controls the compilation and distribution of a new logic code to the Enterprise Server
UBE Subsystem	DEF12	Static	Processes UBE subsystem jobs
WorkFlow	DEF13	Static	Controls the internal communication of process messages to the user
Queue	DEF14	Static	Manages the UBE batch job submission
Metadata	DEF30	Static	Processes XML specification access requests Note: A specification request is a complete description of an EnterpriseOne object needed to process that request.
Management	DEF32	Static	Similar to the SAW kernel, this kernel communicates information about the running processes to the Server Manager Console.

Table Caption 1. EnterpriseOne Kernel Processes

The JDBC interface of the EnterpriseOne HTML Server is a single-process connection to the Oracle ADB database; on the other hand, multiple connections are made through the Enterprise Server kernel processes using the Oracle Call Interface. If an EnterpriseOne HTML database connection is lost due to the expiration of patching drain time, then a new connection is automatically made to other active nodes in the Oracle RAC cluster.

The EnterpriseOne HTML Server depends on communication to the Enterprise Server Security kernel for authentication and other security process requests to function properly; thus loss of database connections on the Enterprise Server Security and Metadata kernel processes has a subsequent downstream effect on the EnterpriseOne HTML Server performance.

The four main EnterpriseOne kernel processes involved in interactive and batch processing are the Call Object, Security, UBE, and Metadata processes. The table above includes these four main processes and other kernel processes which are included in the larger discussion around loss of database connectivity later in this document.

Static EnterpriseOne Connections

A static EnterpriseOne connection does not drain its database connection over to the active Oracle RAC node in the Oracle ADB patching process. The Security, UBE, and Metadata kernel processes on the EnterpriseOne Server are static EnterpriseOne connections. Two major conditions can occur with such Oracle ADB connections.

1. **Temporary Loss of Database Connection.** If the database connection is lost temporarily, the kernel process must retry the database request to re-establish the database connection. For connections that are only temporarily lost, EnterpriseOne kernels will try multiple times for up to 75 seconds to re-issue the SQL request to the Oracle ADB through that same Oracle ADB session.
2. **Failed Database Connection.** If multiple attempts to submit the database request fail after 75 seconds, the current connection and session to the Oracle database is terminated and a new EnterpriseOne process is started. The new EnterpriseOne process establishes a new connection to an active Oracle DB node that is resolving database requests. For the failed EnterpriseOne process, the state information of any EnterpriseOne active transactions is lost and must be recovered or transferred to the new EnterpriseOne process.

Dynamic EnterpriseOne Connections

A dynamic EnterpriseOne connection can move idle connections to other active Oracle RAC nodes in the Oracle ADB patching process. The EnterpriseOne kernel process Call Object Kernel (COK) is as fault-tolerant as a static EnterpriseOne connection, and has high availability properties. These properties enable a kernel process to determine that a database connection must be terminated after a number of failed attempts in the current database session. Upon such an event, the kernel process is terminated for any current-session database connections and a new database connection is established to an active Oracle ADB node for fulfilling application database requests.

An understanding of static and dynamic EnterpriseOne database connections provides the foundation for discussing the Oracle ADB database patching process and how to work with these processes so that business continuity can be maintained and unnecessary recycling of EnterpriseOne services can be avoided.

For details on EnterpriseOne and its network and kernel processes, refer to:

JD Edwards EnterpriseOne Administrative Guide

A foundational understanding of EnterpriseOne kernel processes is critical to understand the importance of the business continuity strategy and the configuration of drain time presented in the later sections of this document.

Note: All EnterpriseOne processes that connect from the Oracle Call Interface or the JDBC interface have a limit of 75 seconds to re-establish lost connections.

SETTING THE ORACLE ADB PATCHING SCHEDULE

Oracle ADB allows manual configuration of patching schedules for planned database maintenance at the container database level. Although there is no downtime or impact on processes during the patching window, it is recommended that customers avoid scheduling their patching windows for the production environment during periods of heavy workload on their JD Edwards implementation, such as during month-end, quarter-end, or year-end. Ideally, patching windows should be scheduled either during non-business hours or on weekends. Such timing helps customers mitigate risks that might occur due to failures in the patching process. For additional details, refer to the section “Create and Manage Autonomous Container Databases” of this Oracle ADB document: Create and Manage Autonomous Container Database.

UNDERSTANDING ORACLE ADB DRAIN TIME

This section provides an understanding of how static and dynamic database connections impact the JD Edwards transactions and processes during an Oracle ADB patching event and how to configure the Oracle ADB application continuity service drain time.

Steps to Evaluate Ideal Drain Time

Application continuity service drain time is the time taken for active database connections to be moved from one Oracle RAC database node to another. Drain time provides not only a timeframe for an EnterpriseOne process to complete, but also allows for connections to be re-established on another Oracle RAC node prior to the database patching .

When drain time is initiated, no new database connections are allowed to the Oracle RAC node that is being drained in preparation for the patching event. All new database connection requests from the EnterpriseOne application are redirected to the remaining active nodes in Oracle RAC. When the drain time expires and all the active connections are drained, any remaining connections or Oracle ADB sessions are terminated and the node is made ready for the Oracle ADB planned maintenance.

The appendix “Business Continuity Script” contains a set of commands that returns the mapping between the EnterpriseOne kernel processes and relates them to the Oracle RAC node to which they are connected. By observing the results of these commands throughout the Oracle patching process, the EnterpriseOne user can verify if connections to the Oracle ADB database are being drained from one Oracle RAC node to another in preparation for the scheduled Oracle patching event. The commands in the appendix are not required to be initiated for the Oracle patching process to complete successfully, they are only provided for a customer to be able to observe in which Oracle RAC nodes EnterpriseOne connections are established.

Oracle ADB Patching Process

The steps involved in the patching of a two-node Oracle ADB RAC environment is described in this section. The image below illustrates the details of the steps involved in the Oracle patching methodology.

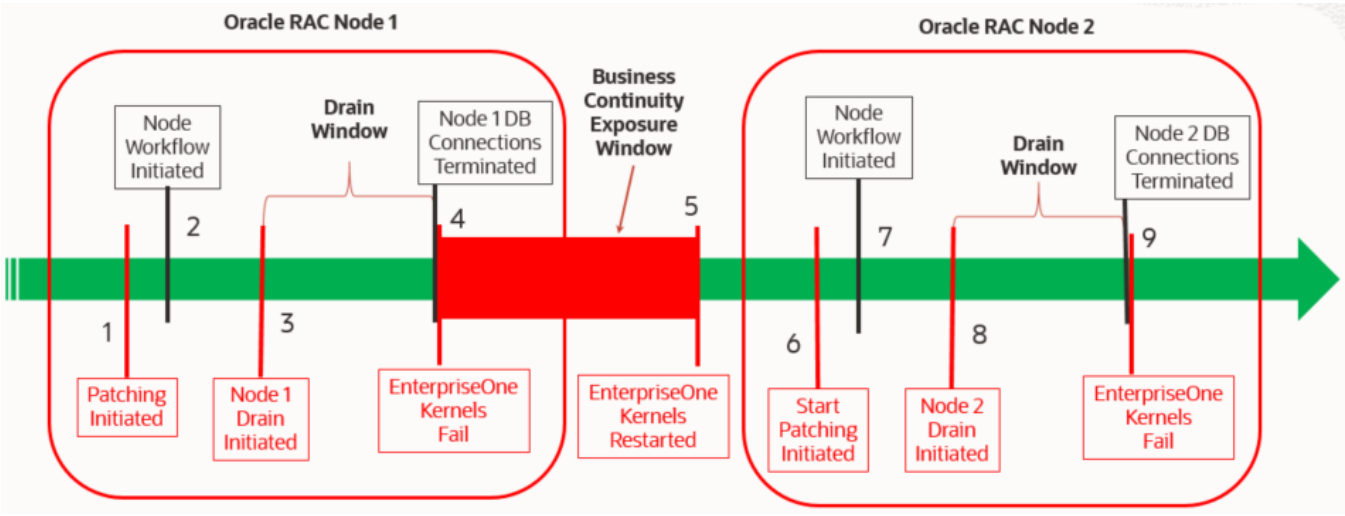

Image Caption 2. Oracle Database Patching Process for JD Edwards EnterpriseOne

The above image shows the nine stages of the Oracle database patching process on a two-node Oracle RAC configuration. In steps 1 through 5, Oracle RAC Node 1 is upgraded and in steps 5 through 9 Oracle RAC Node 2 is upgraded.

Oracle ADB RAC Node 1 Patching

STEP 1: Patching Initiated

This event is initiated from a scheduled planned maintenance window configured through the Oracle Cloud Infrastructure interface as illustrated in the below image.

Image Caption 3. Oracle ADB Cloud Database Maintenance Window Configuration

As shown in the above image, an Oracle ADB patch can be scheduled using the Autonomous Container Database application. This image shows that an Oracle ADB patch for a "Q2 2020 Release update" is available and it is scheduled to be applied on "Sun Jun 7, 2020 at 20:00:00 UTC". These parameters can be edited and changed and also can be configured to be applied quarterly. This schedule is important to note as it will be useful in any pre-Oracle patching steps for EnterpriseOne.

STEP 2: Node Workflow Initiated

The Oracle node workflow is started immediately after the patching start time event occurs. In this step, the Oracle ADB RAC prepares the patch to be applied to the first Oracle ADB node.

Because the Oracle ADB controls node sequencing for patching, the customer has no control over this automated process. For the purpose of this discussion, the database patch was applied to Oracle RAC Node 1 before Oracle RAC Node 2. The the

time may vary, it was observed in testing that 20–30 minutes was required to move from the patching start time (step 1) to the workflow initiation process controlled by the Oracle ADB (step 2)

Recommendation 1: Choose an Oracle ADB maintenance patching window in a period of low EnterpriseOne activity to minimize the impact on business continuity.

Recommendation 2: When setting the ideal drain time, take into consideration the 20–30 minutes of Oracle ADB preparation time for the database patch, so critical EnterpriseOne applications may complete prior to the Oracle patching window. Adjust the ideal drain time to avoid an EnterpriseOne process running in the business continuity exposure window whenever possible.

During this time, EnterpriseOne functions normally, and database requests are serviced through both Oracle RAC Nodes 1 and 2.

STEP 3: Oracle Node 1 Drain Process Is Initiated

After the Oracle ADB completes preparing the database patch to be applied, the Oracle ADB enters the drain process. The instructions on how to set and validate the drain time that will be used in the Oracle ADB patching are provided in the appendix "Setting and Validating Drain Timeout." As previously mentioned, four events can occur when the drain process is initiated:

1. After the drain process has been initiated, no new connections are allowed onto the Oracle RAC node being patched. In this example, no new connections are allowed on Oracle RAC Node 1.
2. Existing EnterpriseOne database requests are fulfilled with the already established connections and sessions to Oracle RAC Node 1 for the entire drain time.
3. When an Oracle ADB database connection is idle, the Oracle ADB database attempts to drain that connection over to other active Oracle RAC node. In this example, Oracle RAC Node 1 connections and sessions are moved to Oracle RAC Node 2 whenever database connections are idle. Any new database requests from those "drained" sessions are subsequently serviced by Oracle RAC Node 2.
4. At the end of the drain time, all Oracle ADB connections, regardless of their state are terminated because this is the absolute time period defined to apply Oracle patching and updates. In this example, all EnterpriseOne connections and sessions are terminated on Oracle RAC Node 1 at the end of the drain time.

This drain time enables an EnterpriseOne user to ensure that all of the critical processes finish prior to the dropping of the Oracle ADB connections. In this example, any critical EnterpriseOne business processes running on Oracle Node 1 should be scheduled to complete in this time period.

Recommendation 3: Set the ideal drain time to be as long as the longest running EnterpriseOne batch process that is likely to be initiated at approximately the same time as the scheduled Oracle ADB patching or upgrade event. EnterpriseOne interactive user processes are much shorter in duration than batch processes and so batch processing time will most likely be considered to determine the ideal drain time.

STEP 4: Oracle Node 1 Database Connections Are Terminated

At the end of the drain time, it is critical to maintain business continuity for the EnterpriseOne kernels, EnterpriseOne HTML processes, and any third-party process integrated into JD Edwards EnterpriseOne that require connections to the database.

This period marks the transition of an active Oracle RAC node that is servicing database requests into a database patching and update mode. The Oracle RAC node in this mode terminates all database connections and sessions before applying the Oracle database patch.

For example, in the patching of Oracle RAC Node 1, all database connections and sessions that existed on the node to be patched are dropped and terminated. As a result of these terminated connections, EnterpriseOne kernel processes will either terminate and be spawned as another process, or attempt to connect to the other active Oracle RAC nodes.

The specific ramifications to the EnterpriseOne application for the Enterprise Server and EnterpriseOne HTML Server are provided in the table below.

EnterpriseOne Kernel Connection Loss

EnterpriseOne Kernel Process	Kernel Designation	EnterpriseOne Connection Loss Effects
UBE	DEF2	This process is dynamic and will re-establish a connection to an active Oracle RAC node.
Security	DEF4	Usually, this process will re-establish connections to active Oracle RAC nodes. However, it is possible that the Security kernel might stop functioning. It is recommended that you initiate the EnterpriseOne heartbeat script to avoid these conditions.
Call Object	DEF6	This process is dynamic and re-establishes a connection to an active Oracle RAC node.
SAW	DEF10	This process terminates when it loses connections to the Oracle ADB database in this step. A new SAW kernel spawns when a new SAW kernel request is made. The user list information in the Server Manager might be out of synchronization temporarily by the time the SAW kernel automatically restarts.
Package Build	DEF11	This process terminates when it loses connections to the Oracle ADB database in this step. A new Package Build kernel spawns when a new package kernel request is made.
UBE Subsystem	DEF12	Usually, this process re-establishes connections to active Oracle RAC nodes.
Workflow	DEF13	This process terminates when its connection is lost to the Oracle ADB database. A new Workflow kernel spawns when a new workflow kernel request is made.
Queue	DEF14	Usually, these processes re-establish connections to active Oracle RAC nodes. However, the Queue kernel might stop functioning. It is recommended to initiate the EnterpriseOne heartbeat script to avoid the conditions that may cause the processes to fail.
Metadata	DEF30	Usually, these processes re-establish connections to active Oracle RAC nodes. However, the Metadata kernel might stop functioning. It is recommended to initiate the EnterpriseOne heartbeat script to avoid these conditions that may cause the processes to fail.
Management	DEF32	Similar to the SAW kernel, the Server Manager information might be out of synchronization while the Metadata kernel process restarts.
EnterprriseOne HTML Server - JDBC	Not Applicable	The EnterpriseOne HTML Server database connections in this step are re-established with the Oracle ADB server if the 75-second timeout is not exceeded. If the timeout is exceeded, the EnterpriseOne HTML Server application request must be resubmitted.

Table Caption 2. Effects of Oracle ADB Connection Loss on EnterpriseOne Kernel Processes

To maintain business continuity and to ensure that EnterpriseOne processes continue to function throughout the Oracle patching process, it is suggested that you implement the EnterpriseOne heartbeat script. The EnterpriseOne heartbeat script helps to mitigate the effects of the 75-second database request timeout and help in restoring the Enterprise Server kernel and the EnterpriseOne HTML Server connections to active Oracle RAC node machines.

STEP 5: EnterpriseOne Kernel Processes Restart

In this step, the EnterpriseOne kernel processes restart and new database connections are established to the remaining active Oracle ADB node running in the Oracle RAC. When the EnterpriseOne kernel process is reinitialized, the EnterpriseOne architecture and processes start functioning but the capability is diminished due to limited node capacity.

In the example of a two-node Oracle ADB RAC configuration, when one of the Oracle RAC nodes is being patched, all EnterpriseOne requests must be handled by the remaining active Oracle RAC node. During the Oracle ADB patching process, it is recommended EnterpriseOne load be lessened as to ensure business continuity, as diminished node capacity is available during this period.

Recommendation 4: During the Oracle patching window, the Oracle ADB RAC node availability is diminished. It is recommended that customers decrease their load of EnterpriseOne during this time period as to avoid any resource contention in this temporary diminished capacity.

EnterpriseOne business continuity is also critical during this step of the Oracle ADB patching process. To ensure that EnterpriseOne processes can span the business continuity exposure window, a separate process needs to be running in the background on the Enterprise Server to avoid timeouts and ensure that EnterpriseOne kernel processes are maintained.

In this document, a heartbeat type script is presented which has been successfully tested to both limit the exposure window and ensure that EnterpriseOne kernel processes are maintained.

Recommendation 5: Limit the business continuity exposure window by initiating an EnterpriseOne heartbeat script as part of the Oracle ADB patching process.

Oracle ADB RAC Node 2 Patching

The steps for patching Oracle ADB RAC Node 2 are similar to the steps 6 through 9 for patching Oracle RAC Node 1.

LIMITING THE BUSINESS CONTINUITY EXPOSURE WINDOW

This section describes how to limit the business continuity exposure window and develop a plan for applying the Oracle ADB patch in the EnterpriseOne architecture with a near zero downtime for the EnterpriseOne functionality.

Automated EnterpriseOne Heartbeat Script

The image below shows the EnterpriseOne heartbeat script being run during the patching process to decrease the business continuity exposure window from 10–15 minutes to 1–2 minutes.

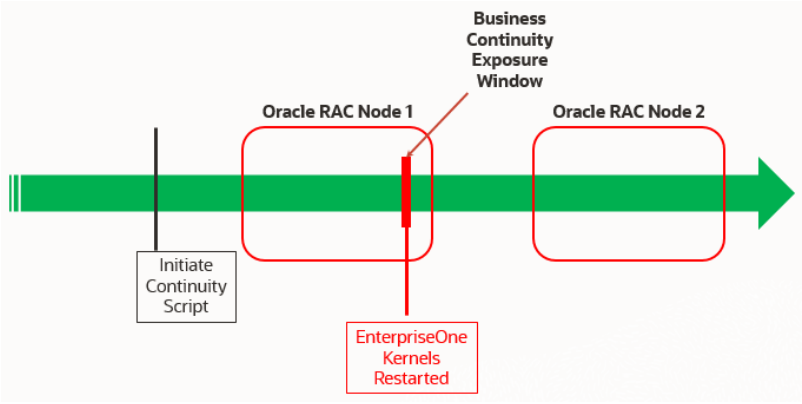

Image Caption 4. Continuity Script for Limiting the EnterpriseOne Business Continuity Exposure Window

This document provides an example of an EnterpriseOne heartbeat script designed to provide continuous database requests to EnterpriseOne. The initiation of such requests is required prior to the Oracle patching window so that the requests can be processed effectively. The EnterpriseOne heartbeat script provides a continuous method of generating

traffic to the EnterpriseOne kernel processes by using the out-of-the-box EnterpriseOne functions of “porttest” and “runube”.

The “porttest” EnterpriseOne command generates traffic to the EnterpriseOne Security and Metadata kernels. If the command fails, then those processes are terminated and a new process is started. The “runube” EnterpriseOne command facilitates the continual testing of the EnterpriseOne UBE process and its restart if necessary. Finally, the script also detects if an Oracle failover event has occurred by inspecting the EnterpriseOne server log messages. If an Oracle failover is detected, both the Security and Metadata kernel processes are restarted.

In testing, if the script is configured to run every 30 seconds, the business continuity exposure window is reduced from up to 15 minutes to a maximum of 2 minutes. The script also ensured that the EnterpriseOne kernel process became functional before it encountered a timeout or became inactive due to a hung process.

The EnterpriseOne heartbeat script must be started prior to the Oracle ADB patching window and should continue to be run throughout the database patching process. It is initiated on the EnterpriseOne Server as the jde920 operating system user. The script produces minimal overhead and only executes if a loss of EnterpriseOne kernel processes is detected or if an Oracle database connection fails.

Image Caption 5. EnterpriseOne Heartbeat Script for Oracle Database Patching

CONCLUSION

Oracle Autonomous Database with its zero-downtime automatic patching capabilities helps Oracle JD Edwards customers stay current on database patches without disrupting their system availability. Oracle Autonomous Database enables JD Edwards customers to set their maintenance window and define the application continuity service drain time to ensure system and transaction continuity.

Oracle JD Edwards customers can configure the application continuity service drain time to last up to 8 hours, enabling long-running transactions to complete and ensure data integrity before the node is brought down for maintenance. Longer drain time enables Oracle JD Edwards customers to carry out planned maintenance activities on their Oracle Autonomous Database without disrupting their ongoing transactions or operations.

By following the best practices described in this document, Oracle JD Edwards customers can achieve planned database maintenance with zero downtime, which provides increased system availability while eliminating human effort. By eliminating the overhead and challenge associated with traditional planned database maintenance, the Oracle Autonomous Database helps JD Edwards customers achieve better efficiencies and greater innovation in the core areas of their businesses.

APPENDIX A: TESTING USE CASES

This appendix describes the EnterpriseOne architecture and use cases used in testing this solution.

Overview of the EnterpriseOne Architecture

For the purpose of this discussion, a simple three-tier architecture was constructed. An illustration of this architecture is provided below.

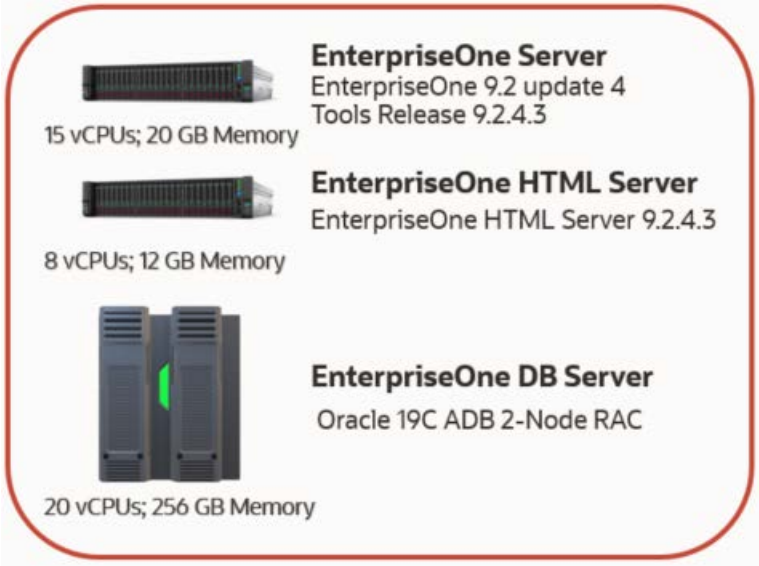

Image Caption 6. EnterpriseOne Testing Architecture

All of the EnterpriseOne environment components in the above image are implemented in Oracle Cloud Infrastructure. This document covers the testing of interactive and batch processes, thus an EnterpriseOne Enterprise Server (logic only), EnterpriseOne HTML Server, and a Database Server using Oracle ADB RAC are the only EnterpriseOne components that were necessary for testing.

Oracle ADB was configured as a two-node Oracle RAC implementation. If EnterpriseOne is implemented in the Oracle Cloud Infrastructure on a single Oracle RAC node, then the patching mechanism is the same as the mechanism followed for a multi-node RAC configuration. In the case of a single-node Oracle RAC configuration, an internally spawned temporary second Oracle RAC node is automatically allocated. This second Oracle RAC node facilitates the relocation of services and provides the mechanism for Oracle database patches and updates. When the database patching process completes, the system returns to the original single Oracle RAC node.

The image below provides an illustration of the four use cases that were evaluated in this testing of continuous EnterpriseOne availability during an Oracle patching process.

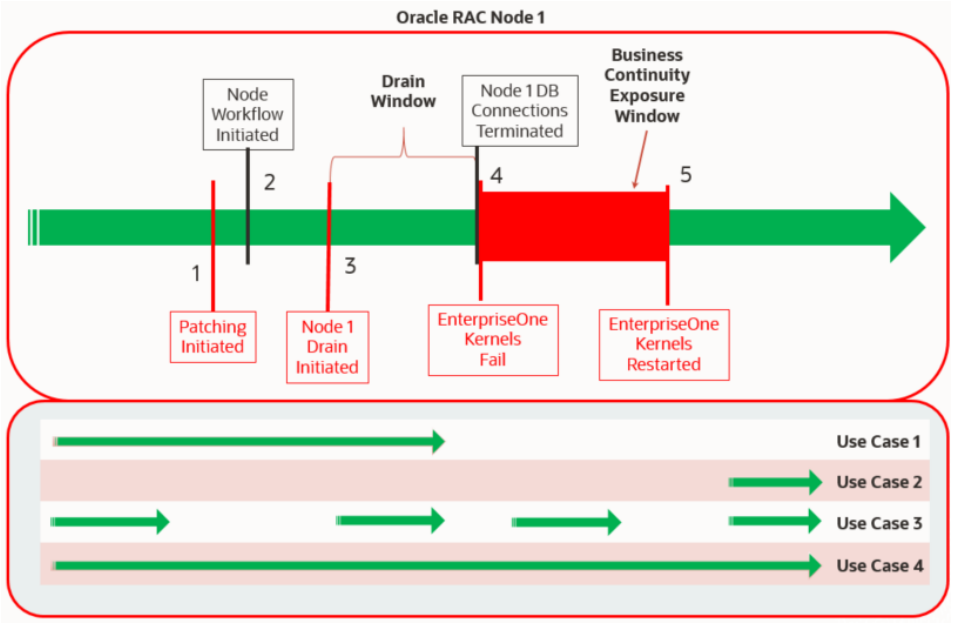

Image Caption 7. EnterpriseOne Patching Process and Use Cases

There are four major categories of use cases based on the combinations of time of initiation of the transaction, amount of overlap of execution time with the database drain time, and the time of completion of the transaction. These four categories of use cases provide adequate coverage of possible scenarios to identify the ideal drain time that would ensure that transactions are not impacted.

These four use case scenarios were used in testing the Oracle ADB patching process. For both EnterpriseOne interactive and batch processes, they were initiated at different times during the Oracle ADB patching process to provide a thorough understanding of the EnterpriseOne functionality and business continuity during the Oracle patching process. The green arrows in the above image depict the different times in which the batch and interactive processes must maintain continuity for the database patching process to complete successfully.

EnterpriseOne batch and interactive tests were performed separately along with Oracle ADB patching events to isolate any issues that can be directly attributed to the use cases used for testing. Each use case was separately performed for interactive and batch processes.

Use Case 1

In this scenario, the EnterpriseOne batch or interactive transaction is started before the Oracle patching window and completes within the drain timeout. For both interactive and batch process testing for Use Case 1, if the process is started prior to the Oracle ADB patching window and completes before the drain time expires, then all EnterpriseOne application processes complete successfully.

The evaluation of drain time in Use Case 1 leads to the first recommendation to set the ideal drain time to be long enough to complete all interactive and batch processes. For an EnterpriseOne application, a batch process usually takes the longest to complete.

An evaluation of EnterpriseOne batch processes and their typical runtimes will help in identifying the batch processes that will be initiated during a scheduled Oracle ADB patching or update event. Based on this evaluation, the drain time can be set long enough to ensure that the patching process does not interfere with the completion of the longest batch process. This configuration can be done prior to the patching window.

For example, if the longest EnterpriseOne batch application is 2 hours, and this batch process is likely to be initiated just before the scheduled Oracle ADB patching event, then setting the drain time longer than 2 hours ensures that this batch process completes successfully without any impact due to patching.

The default drain time is 5 minutes, which is normally sufficient to complete any interactive user process. Therefore as already stated, the limiting condition for setting the ideal drain time is the longest running EnterpriseOne batch process.

Use Case 2

In this scenario, an EnterpriseOne transaction is initiated during the patching of Oracle RAC nodes.

To determine where the database connections reside and to which nodes in the Oracle ADB they are connected after the patching of Oracle RAC Node 1, refer to the SQL script provided in the appendix "SQL Script to Determine ORACLE RAC Connections."

Use Case 3

In this scenario, an EnterpriseOne process is initiated during the Oracle ADB patching window and more specifically during the drain time. If an EnterpriseOne process is started during the drain time, any new database connections initiated are on an active Oracle ADB RAC node that is not being patched and not on the Oracle RAC node being patched. Thus, all tested batch and interactive processes succeeded while testing this use case.

Use Case 4

In this scenario, the EnterpriseOne transaction is started before the Oracle patching window and completes after the patching window. The EnterpriseOne process that fits this scenario is a long-running batch process. If the drain time is not set to account for this possibility, this scenario is most likely to fail during patching.

More extensive testing is required by customers for their sprecific needs including assessing other EnterpriseOne components that connect to the database as well as third-party integrations into the EnterpriseOne application that may send database requests to the Oracle ADB. The intention of the use cases is to demonstrate the functionality of a batch and an interactive process. These processes represent the two types of connection interfaces through which all the database requests are sent to the Oracle ADB Server.

Note: For testing the use cases, the drain time was set to 1 hour.

Oracle Call Interface and UBE Process Testing

Oracle Call Interface is a comprehensive high-performance native C language interface to the Oracle ADB. The testing process used the simple universal batch engine (UBE) functionality of EnterpriseOne to submit database requests through the Oracle Call Interface to the Oracle ADB.

The UBE process performs many such database request functions such as the Oracle DML statements of select, insert, update, and delete across multiple EnterpriseOne tables in the Oracle ADB. The EnterpriseOne UBE was used because it makes large numbers of database requests throughout its processing and extensively uses the EnterpriseOne UBE, Metadata, and Security kernel processes on the EnterpriseOne Server, making it ideal for this testing. The UBE process chosen for testing is Purchase Order Processing (R42800), one of the more widely used and heavy database request batch applications.

JDBC Interface and Interactive Process Testing

EnterpriseOne users access the EnterpriseOne architecture through their browser that communicates to the EnterpriseOne HTML Server architecture component. The EnterpriseOne HTML Server uses Java to service these requests through an end user graphical interface. This Java process requires direct connectivity for the EnterpriseOne HTML Server to Oracle ADB using the JDBC interface.

Although the interactive requests are initiated on the EnterpriseOne HTML Server in Java, much of the logic processing of interactive users is redirected to the EnterpriseOne Server and performed using the native C-based functionality through primarily the Call Object kernel processes. The database connection requests for the database requests on the EnterpriseOne Server are processed through the Oracle Call Interface. For the remaining database requests on the EnterpriseOne HTML Server, the database connection requests to the Oracle ADB are processed through the JDBC interface.

Two interactive processes—Sales Order Entry (P42101) interactive application and Sales Order Update (P42101)— were chosen for testing. Sales Order Entry (P42101) application is one of the most widely-used interactive processes by the EnterpriseOne customer community. Sales Order Update (P42101) includes the "Find" and "GoToEnd" action functionality. As "GoToEnd" logic processing involves many JDBC database requests to Oracle ADB, P42101 represents a good use case to be tested during the Oracle ADB patching process.

Use Case Scenarios Used for Testing

For testing, the use cases must be initiated in the different steps of the Oracle ADB patching process. Because this document has already provided an overview of the use cases in a previous section, this section presents these use case scenarios and the testing process in more detail.

The below illustration is same as the illustration provided in the section "Overview of the EnterpriseOne Architecture" and is provided here for quick reference.

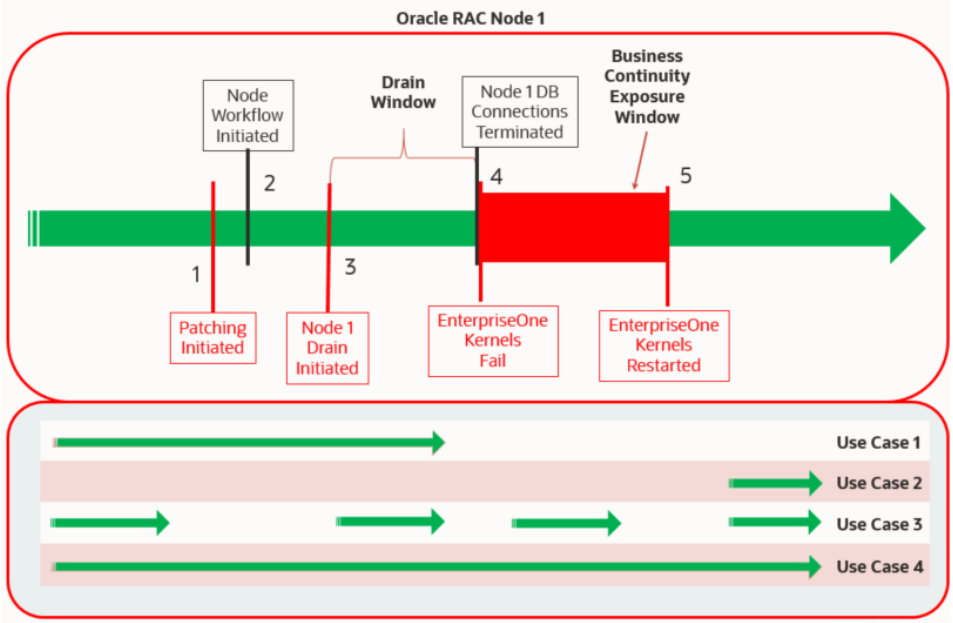

Image Caption 8. EnterpriseOne Patching Process and Use Cases

Testing Details

The details of the interactive and batch processes are provided in the tables below.

Sales Order Update Batch Process Use Case Details

The table below provides testing details for the batch process Sales Order Update (R42800). In the use case details in the following table, a number of steps are referenced that are presented in the figure "EnterpriseOne Patching Process and Use Cases" in the section "Use Case Scenarios Used for Testing."

The batch process only uses Oracle Call Interface database requests and are initiated on the EnterpriseOne Server.

EnterpriseOne Batch Testing

USE CASE	START TIME	END TIME	GOAL OR PURPOSE
1	30 minutes prior to the patching window	Prior to completion of drain time	Test batch processing throughout the drain time window
2	After the business continuity exposure window	After RAC Node 2 patching completed	Observe if batch process is initiated on Node 2 and completed when initiated after the business continuity window
3-1	Before the patching process is initiated	Before patching process is initiated	Perform a baseline test for normal processing of batch process

USE CASE	START TIME	END TIME	GOAL OR PURPOSE
3-2	After the Node 1 drain process is initiated	Before Node 1 connections are terminated	Test batch process during the drain time period
3-3	Within the business continuity exposure window	Within the business continuity exposure window	Test batch process within the business continuity exposure window
3-4	After the EnterpriseOne kernels restarted on the new node	Before patching initiation on Node 2	Test batch process after recovery or restart of kernel processes with new active database connections
3-5	Within the business continuity exposure window	Before patching initiation on Node 2	Test batch process within the business continuity exposure window up to patching initiation
3-6	After patching completion on Nodes 1 and 2	After patching completion on Nodes 1 and 2	Test batch process after completion of patching for both the nodes
4	Prior to the database patching window on Node 1	After patching completion on Nodes 1 and 2	Test batch processing and business continuity throughout the patching process of both the nodes

Table Caption 3. EnterpriseOne Batch Testing Use Cases

Interactive Sales Order Entry and Update Process Use Case Details

The table below provides some testing details for the interactive user processes that are run when a new sales order is entered using the Sales Order Entry application (P42101). It also provides the details of the interactive processes that are run when a query for records is processed by the "Find" and "GotoEnd" functionalities of P42101. In the use case details, a number of steps are referenced that are presented in the figure "EnterpriseOne Patching Process and Use Cases" in the above section.

Most interactive applications normally take a few minutes, and this behavior was simulated during the testing process. An interactive user would perform the following actions:

Sales Order Entry Use Case:

1. Log in to EnterpriseOne.
2. Navigate to Sales Order Application (P42101).
3. Add a new single-line sales order.
   (Note: The entry of sales order lines will call JDBC interface database requests.)
4. Submit the order.
   (Note: The submission of the completed sales order will call Oracle Call Interface database requests.)
5. Close and log out of the application.

Sales Order Find and GotoEnd Use Case:

1. Log in to EnterpriseOne.
2. Navigate to Sales Order Application (P42101).
3. Retrieve a 100 line Sales Order (Find).
4. The first 10 lines of the Sales Order is retrieved.
5. Choose GotoEnd to retrieve the remaining lines.
   (Note: JDBC interface database requests are made when GotoEnd processing is initiated.)

As both of the above processes took less than 5 minutes to complete, the use cases were continuously repeated for the duration of the test. The below table outlines the interactive use case details.

EnterpriseOne Interactive Use Cases

USE CASE	START TIME	END TIME	GOAL OR PURPOSE
1	10 minutes prior to the patching window	Prior to completion of the drain time	Test user interactive processing throughout the drain time window
2	After the business continuity exposure window	After RAC Node 2 patching completion	Observe if an interactive user process would be initiated on Node 2 and completed when initiated after the business continuity window
3-1	Before the patching process is initiated	Before the patching process is initiated	Perform a baseline test for normal processing of an interactive user process
3-2	After Node 1 drain process is initiated	Before Node 1 connections are terminated	Test an interactive user process during the drain time
3-3	Within the business continuity exposure window	Within the business continuity exposure window	Test an interactive user process within the business continuity exposure window
3-4	After the EnterpriseOne kernels restarted on the new node	Before patching initiation on Node 2	Test an interactive user process after recovery or restart of kernel processes with new active database connections
3-5	Within the business continuity exposure Window	Before patching initiation on Node 2	Test an interactive user process within the business continuity exposure window up to patching initiation
3-6	After patching completion on Nodes 1 and 2	After patching completion on Nodes 1 and 2	Test an interactive user process after completion of patching for both nodes
4	Prior to the database patching window on Node 1	After patching completion on Nodes 1 and 2	Test interactive processing and business continuity throughout the complete patching process of both the nodes

Table Caption 4. EnterpriseOne Interactive Use Testing Use Cases

A number of validations were performed to certify the success or failure of the interactive and batch processes that were tested. These validation metrics included a SQL validation of record count changes and inspection of the EnterpriseOne Server and EnterpriseOne HTML log files for errors and any failures to perform any of the Oracle DML operations consistent with the proper functioning of the EnterpriseOne application.

Testing Results

The tables below list the use-case scenarios used for testing batch and interactive user processes during a two-node Oracle ADB patching event where both nodes in the Oracle RAC were updated. For all the use case scenarios, the drain time was set to 30 minutes.

EnterpriseOne Batch Results

The results for the EnterpriseOne batch process of the Sales Order Update application (R42800) are provided in the table below. The table provides information on the use case scenarios tested, processing time of the batch process for each use case scenario, results of the tests, and observations made from the tests.

EnterpriseOne Batch Testing Results

USE CASE	START TIME	END TIME	PROCESSING TIME	RESULT	OBSERVATIONS
1	30 minutes prior to the patching window	Prior to completion of drain time	1 hour 25 minutes	Pass	Positive control test with successful completion of batch
2	After the business continuity exposure window	After RAC Node 2 patching completed	2 hours 5 minutes	Pass	Positive control test with successful completion of batch. EnterpriseOne kernel processes existed to an active Oracle RAC node after the Oracle RAC Node was upgraded.
3-1	Before the patching process is initiated	Before the patching process is initiated	15 minutes	Pass	Positive control test with successful completion of batch, as no Oracle Autonomous patching occurred during the test.
3-2	After the Node 1 drain process is initiated	Before Node 1 connections are terminated	15 minutes	Pass	The batch process started and ended within the drain time, EnterpriseOne had available kernel database connections resulting in successful completion of batch processing.
3-3	Within the business continuity exposure window	Within the business continuity exposure window	15 minutes	Pass**	The batch process was started and it ended within the business continuity exposure window. See the section “EnterpriseOne Batch Additional Observations” for more detail about this test result.
3-4	After EnterpriseOne kernels restarted on the new node	Before patching initiation on Node 2	15 minutes	Pass	Batch processing succeeded because all EnterpriseOne kernel processes had re-established their database connections and sessions to the Oracle ADB. This was an expected result.
3-5	Within the business continuity exposure window	Before patching initiation on Node 2	15 minutes	Pass**	Batch processing testing was started in the business continuity exposure window and completed after EnterpriseOne kernel processes were successfully restarted. This test scenario is covered in more detail in the section “EnterpriseOne Batch Additional Observations.”
3-6	After patching completion	After patching completion on Nodes 1 and 2	15 minutes	Pass	Positive control test with successful completion of the batch process

USE CASE	START TIME	END TIME	PROCESSING TIME	RESULT	OBSERVATIONS
	on Nodes 1 and 2				
4	Prior to the database patching window on Node 1	After patching completion on Nodes 1 and 2	4 hours 35 minutes	Pass, Exposure	This test scenario was run multiple times with the batch process spanning the entire Oracle ADB patching window. Some of the tests succeeded and some failed to receive validation or failed and ended in error. A more detailed analysis is provided in the section "EnterpriseOne Batch Additional Observations."

Table Caption 5. EnterpriseOne Batch Testing Results

EnterpriseOne Batch Additional Observations

The results of three of the EnterpriseOne batch testing scenarios described in the above table require more discussion. Initiating EnterpriseOne as a batch application around the business continuity exposure windows should either be avoided altogether or should be mitigated by adjusting several EnterpriseOne configurations. These specific scenarios are discussed in more detail in this section.

Scenario 3-3 and Scenario 3-5: In these scenarios, the batch process started and ended within the business continuity exposure window and all the EnterpriseOne kernel processes that had connections were just terminated. In these scenarios, the following two environment conditions can exist.

First, the EnterpriseOne connections can exist on multiple Oracle RAC nodes while EnterpriseOne database service connections are still residing on Oracle RAC Node 2. In this scenario, if a batch process was started in this window or during Steps 4-5 or Steps 4-6 of the Oracle ADB database patching, then all batch process steps succeeded.

However, in the second condition, that is when all the database connections exist on the Oracle RAC node being patched, the results of the testing indicated an exposure. This type of exposure must be mitigated to ensure that the EnterpriseOne kernel processes that have to be started anew can acquire a new Oracle ADB connection without any delay to avoid failure of the batch process. If processes are initiated, then this recommendation should be followed to avoid exposure.

Recommendation 6: Increasing the EnterpriseOne Server JDENetTimeout temporarily will assist EnterpriseOne batch processes to be in a waiting state giving enough time to start a new EnterpriseOne kernel process. The default JDENetTimeout is 60,000 ms and should not be increased longer than 180,000 ms to avoid issues with long-running business functions.

Scenario 4: Although a successful run of an EnterpriseOne batch process was observed during testing, it could not be ascertained in this testing that this scenario will always work. The business logic behind an EnterpriseOne batch job processing has many characteristics. These characteristics can include transaction boundaries, pessimistic and optimistic record locking, and retaining the state of the EnterpriseOne batch process.

Pessimistic locking is where a SQL is issued to the database to update the row set; this is commonly done with the SELECT for UPDATE clause. Optimistic locking is where the data is read once, followed by the logic being evaluated to determine if a record needs to change, and then the data is re-read and modified if it did not change since the initial fetch.

The greatest concern is to ensure that the EnterpriseOne batch process retains state information required to continue logic processing and complete the batch process successfully.

EnterpriseOne Interactive User Results

The results for the EnterpriseOne interactive user processes using the Sales Order application (P42101) is provided in the table below. The table provides information on the use case scenarios tested, processing time of the interactive processes, results of the test, and observations made from the test.

As previously mentioned, for the Sales Order interactive applications, approximately 4 minutes were required to complete a single iteration of the use case. In the scenarios where the interactive user processes were to cross a step boundary (as outlined in the figure "Oracle Database Patching Process for JD Edwards EnterpriseOne " of the "Oracle ADB Patching Process" section), then the use case scenario was re-run from the beginning until the test completed.

EnterpriseOne Interactive User Results

USE CASE	START TIME	END TIME	PROCESSING TIME	RESULT	OBSERVATIONS
1	15 minutes prior to the patching window	Prior to the completion of drain time	~4 minutes	Pass	Positive control test with successful completion of interactive user process is observed.
2	After the business continuity exposure window	After RAC Node 2 patching completion	~4 minutes	Pass	Positive control test with successful completion of interactive user process is observed. EnterpriseOne kernel processes existed to an active Oracle RAC node after the Oracle RAC Node upgrade.
3-1	Before the patching process is initiated	Before patching process is initiated	~4 minutes	Pass	Positive control test with successful completion of the interactive user process is observed, as no Oracle Autonomous patching occurred during the test.
3-2	After the Node 1 drain process is initiated	Before Node 1 connections are terminated	~4 minutes	Pass	The interactive user process started and ended within the drain time, EnterpriseOne had available kernel database connections resulting in successful completion of the interactive process.
3-3	Within the business continuity exposure window	Within the business continuity exposure window	~4 minutes	Pass**	The interactive user process was started and it ended in the business continuity exposure window. This particular result is described in more detail in the "EnterpriseOne Interactive User Additional Observations" section.
3-4	After EnterpriseOne kernels restarted on the new node	Before patching initiation on Node 2	~4 minutes	Pass	Interactive user processing succeeded because all EnterpriseOne kernel processes had re-established their database connections and sessions to Oracle ADB. This was an expected result.
3-5	Within the business continuity exposure window	Before patching initiation on Node 2	~4 minutes	Pass**	Interactive user processing testing was started in the business continuity exposure window and completed after the EnterpriseOne kernel processes were successfully restarted. This test scenario is covered in more detail in the "EnterpriseOne

USE CASE	START TIME	END TIME	PROCESSING TIME	RESULT	OBSERVATIONS
					Interactive User Additional Observations" section.
3-6	After patching completion on Nodes 1 and 2	After patching completion on nodes 1 and 2	~4 minutes	Pass	Positive control test with successful completion of interactive user process was observed.
4	Prior to the database patching window on Node 1	After patching completion on Nodes 1 and 2	~4 minutes	Pass and Exposure	This test scenario was run multiple times with the interactive user process spanning the entire Oracle ADB patching window. Some of the tests succeeded and some failed to receive validation or failed and ended in error. A more detailed analysis is provided in the "EnterpriseOne Interactive User Additional Observations" section.

Table Caption 6. EnterpriseOne Interactive User Test Results

EnterpriseOne Interactive User Additional Observations

The results of three of the EnterpriseOne interactive testing scenarios described the above table require more discussion. Although the risk of business continuity disruption is not as high for EnterpriseOne interactive applications as it is for EnterpriseOne batch processes, interactive processes can encounter major issues.

Initiating any EnterpriseOne application around business continuity exposure windows should either be avoided or should be limited by adjusting several EnterpriseOne configurations. These specific scenarios are discussed in more detail in this section.

Scenario 3-3 and Scenario 3-5: In these scenarios, the EnterpriseOne interactive user process started and ended within the business continuity exposure window and all the EnterpriseOne kernel processes that had connections were just terminated. In these scenarios, two environment conditions can exist and are identical to those discussed in the "EnterpriseOne Batch Additional Observations" section.

First, the EnterpriseOne connections can exist on multiple Oracle RAC nodes while EnterpriseOne database service connections are still residing on Oracle RAC Node 2. In this scenario, if an interactive user process was started in this window or during Steps 4–5 or Steps 4–6 of the Oracle ADB database patching, then all interactive user process steps succeeded.

However, in the second condition, that is when all the database connections exist on the Oracle RAC node being patched, the results of the testing indicated an exposure. This type of exposure must be mitigated to ensure that the EnterpriseOne kernel processes that have to be started anew can acquire a new Oracle ADB connection without any delay to avoid failure of the interactive user process. Recommendations 1 through 6 outlined in this document will assist in avoiding this exposure.

Scenario 4: Although a successful run of an EnterpriseOne interactive user process was observed during testing, it could not be ascertained in this testing that this scenario will always work. EnterpriseOne interactive user processes are usually short in duration and do not fail during the Oracle ADB patching window.

Testing Results Summary

The test results and their analysis are important for two reasons: first, they help to identify any issue with the Oracle ADB patching methodology being used for the EnterpriseOne application and whether these issues can be mitigated with adjustments to the EnterpriseOne configuration. Second, the results can demonstrate if EnterpriseOne services can be maintained throughout the Oracle ADB patching process without the need to be recycled. If the services can be maintained, then the Oracle ADB patching event can be planned with minimal adjustments to the EnterpriseOne configuration.

The above test results indicate that planned database maintenance can be achieved with zero downtime by configuring the ideal drain time. The results also indicate when EnterpriseOne batch and interactive processes may have issues. These periods should be either mitigated by running an additional script as well as making temporary changes to the EnterpriseOne configuration to allow for easy startup of the necessary EnterpriseOne kernel processes.

APPENDIX B: BUSINESS CONTINUITY SCRIPT

This appendix contains the full code listing of the business continuity script that is initiated before the scheduled Oracle patching window.

The script is designed to run continuously for 100,000 iterations until it completes or is manually terminated at the end of the Oracle ADB patching process.

```
#!/bin/bash
# Script to launch porttest Continuously, once every minute
# Copyright Oracle 2020 All Rights Reserved
#

e1_passwd=$1
e1_pathcode=$2
log_file="$0_logfile.out"

### Define EnterpriseOne variables used in script
e1_batchjob="R0006P XJDE0001"
e1_batchqueue="QBATCH"
e1_role="*ALL"


## Read in Script arguments
echo "Log File" > ${log_file}
if [[ -n "$e1_passwd" ]]; then
   echo "Password: $1" >> ${log_file}
   if [[ -n "$e1_pathcode" ]]; then
      echo "Pathcode: $2" >> ${log_file}
   else
      echo "Argument Error: Syntax: $0 <E1 Password> <E1 Pathcode>"
      echo "Example: $0 passwd DV920"
      exit 1
   fi
else
   echo "Argument Error: Syntax: $0 <E1 Password> <E1 Pathcode>"
   echo "Example: $0 passwd DV920"
   exit 1
fi

## Verify User ID running script is jde920
if [[ $(whoami) == "jde920" ]]; then
  echo "User is jde920" >> ${log_file}
else
  echo "This script must be run as user: jde920"
  exit 1
fi

## Check jde920 system variables are not empty
if [[ -z $SYSTEM ]]; then
  echo "jde920 SYSTEM variable not defined"
  exit 1
fi
if [[ -z $EVRHOME ]]; then
  echo "jde920 EVRHOME variable not defined"
  exit 1
fi

## Check for the presence of the password file for runube command
PWD_FILE=$SYSTEM/bin32/passwd.txt
```

```
if ! [ -f "$PWD_FILE" ]; then
  echo "File: $SYSTEM/bin32/passwd.txt is needed.  Exiting"
  exit 1
fi

cd $SYSTEM/bin32
### Get initial count of Oracle failures in logs
counter1=`grep JDB9900256 $EVRHOME/log/jde_*.log | wc -l`

## Initiate runube and check for success -- if it fails script cannot be run correctly
$SYSTEM/bin32/runube -f $SYSTEM/bin32/passwd.txt $e1_pathcode *ALL $e1_batchjob $e1_batchqueue "Batch" "Hold" "Delete" &
queuepid=`$SYSTEM/bin32/netwm | grep kdef-14 | awk '{ print $1 }' | awk -F- '{ print $2 }'`
echo "Queue ID: $queuepid"
if [ $? -eq 0 ]
then
   echo "Batch is Successful"
else
   echo "Batch Process Failed - Cannot start script without batch functionality"
   exit 1
fi

## Iterate script 100,000 times until killed or terminated
for x in {1..100000}; do
  echo "Iteration: $x"

  ### EnterpriseOne commands for heartbeat
  ### In case of failure, script will terminate Security, Metadata, and Queue Kernel Processes
  ### It will be assumed at this stage to be non-functional, the next script submission of porttest/runbatch will restart them
  $SYSTEM/bin32/porttest jde $e1_passwd $e1_pathcode 1>/dev/null 2>&1

  if [ $? -eq 0 ]
  then
     echo "PORTTEST is Successful1"
  else
     echo "PORTTEST Failed - Terminating Inoperatble Kernel Processes"
     echo "Oracle Failover Detected"

     ## Get all of the Security (KDEF-4) jobs
     security_jobs=`$SYSTEM/bin32/netwm | grep kdef-4 | awk '{print $1}'`
     security_job_ids=$(echo "$security_jobs" | awk -F'[-]' '{print $2}')

     while read -r line; do
       echo "     Restarting Security Kernel (KDEF-4)"
       kill -9 $line
     done <<< $security_job_ids

     ## Get all of the Metadata (KDEF-30) jobs
     metadata_jobs=`$SYSTEM/bin32/netwm | grep kdef-30 | awk '{print $1}'`
     metadata_job_ids=$(echo "$metadata_jobs" | awk -F'[-]' '{print $2}')

     while read -r line; do
       echo "     Restarting Metadata Kernel (KDEF-30)"
       kill -9 $line
     done <<< $metadata_job_ids

     sleep 30
     ### Restart Queue ID, there is only one Queue ID
     queue_id=`$SYSTEM/bin32/netwm | grep kdef-14 | awk '{ print $1 }' | awk -F- '{ print $2 }'`
     echo "     Restarting Queue Kernel (KDEF-14): $queue_id"
     kill -9 $queue_id
  fi

  ### Check Oracle Failover errors in logs, if present, restart Queue kernel
  ### Successful PORTTEST restarts KDEF4 (Security) and KDEF30 (Metadata) kernels automatically
  counter2=`grep JDB9900256 $EVRHOME/log/jde_*.log | wc -l`

  if [ $counter2 \> $counter1 ]
  then
    echo "Oracle Failover Detected"
    ### Restart QUEUE ID
    queue_id=`$SYSTEM/bin32/netwm | grep kdef-14 | awk '{ print $1 }' | awk -F- '{ print $2 }'`
    echo "     Terminating Queue Kernel (KDEF-14): $queue_id"
    kill -9 $queue_id
```

```
    echo "     Restarting EntperpriseOne QUEUE Kernel"
    $SYSTEM/bin32/runube -f $SYSTEM/bin32/passwd.txt $e1_pathcode *ALL $e1_batchjob $e1_batchqueue "Batch" "Hold" "Delete" &
    sleep 30
    counter1=`grep JDB9900256 $EVRHOME/log/jde_*.log | wc -l`
   fi

   sleep 30
done
```

The above script is a basic template, and should be customized based on the customer's requirements. The most notable changes required for the script are:

1. Creating a $SYSTEM/bin32/passwd.txt file for the runube command to initiate the batch process without requiring a password passed in clear text.
2. Passing the following arguments into the script: <EnterpriseOne jde User Password> and <EnterpriseOne Pathcode>
3. Performing a manual cleanup of the UBE jobs created by the script.
   This is a required manual script because UBE jobs can be implemented in more than one way by customers.

APPENDIX C: SETTING AND VALIDATING DRAIN TIMEOUT

When Oracle ADB planned maintenance starts and the Oracle patch is ready to be applied to the first node in the Oracle RAC, sessions need to be drained from the Oracle RAC node that is to be patched to the active Oracle RAC node that is not to be patched. This movement of sessions allows for continuous service. The time allowed for this draining is called the drain timeout.

The Oracle ADB drain timeout by default is configured for 300 seconds. Starting with Oracle 19c, a stored procedure is added to the Oracle administrative utilities that allows this drain timeout to be increased from the default of 300 seconds (5 minutes) to a significantly higher value, that is up to 8 hours.

To use these procedures, you must be granted the role PDBADMIN and have the OPatch 30463938 applied to the Oracle ADB instance.

Setting the Drain Time

The SQLPlus command that can be run to reset the drain timeout to 30 minutes (1,800 seconds) is provided below: `execute dbms_cont_admin.set_draining( '<PDB instance>', 1800, 'immediate') ;`

For a more detailed description of the drain timeout procedures, refer to:

Continuous Availability, Best Practices for Applications Using ADB

Validating the Drain Time

A simple SQL script is used to check the drain time of the Oracle ADB:

```
set lines 100
col name format a40
col failover_restore format a10
col failover_type format a10
col stop_option format a10

select NAME, FAILOVER_TYPE, FAILOVER_RESTORE, DRAIN_TIMEOUT, STOP_OPTION
from dba_services order by NAME;
```

A sample output of the above SQLPlus Script:

```
NAME                                     FAILOVER_T FAILOVER_R DRAIN_TIMEOUT STOP_OPTIO
---------------------------------------- ---------- ---------- ------------- ----------
ADBINSTANCE
ADBINSTANCE_high.atp.oraclecloud.com         NONE       NONE                300 IMMEDIATE
ADBINSTANCE_low.atp.oraclecloud.com          NONE       NONE                300 IMMEDIATE
ADBINSTANCE_medium.atp.oraclecloud.com       NONE       NONE                300 IMMEDIATE
ADBINSTANCE_ords.adb.oraclecloud.com         NONE       NONE                300 IMMEDIATE
ADBINSTANCE_tp.atp.oraclecloud.com           AUTO       AUTO               1800 IMMEDIATE
ADBOMSTAMCE_tpurgent.atp.oraclecloud.com     AUTO       AUTO                300 IMMEDIATE
```

In the above example, ADBINSTANCE is the name of the Oracle connect string. The Oracle Call Interface and the JDBC interface that the EnterpriseOne processes will use to connect to the database is defined by the tagged label: “ADBINSTANCE_tp.atp.oraclecloud.com”. The above example shows a 30-minute drain time setting.

APPENDIX D: SQL Script to Determine Oracle RAC Connections

This appendix provides a simple SQL script to show the Oracle RAC connections and which EnterpriseOne kernel process is being established to the Oracle ADB.

EnterpriseOne kernel processes need to be identified first to determine which EnterpriseOne KDEF process is running and connecting to the Oracle ADB. On an EnterpriseOne Linux Server implementation, the two EnterpriseOne commands of *netwm* and *jdejobs* provide this information. Alternatively, the information can be viewed from the Server Manager Console.

```
% netwm
Initializing JDEIPC
About to setupWorkManagementStruct
setupWorkManagementStruct ok
getJDENetWorkManagement ok
********************
2 net processes
********************
pid-10605       start-Wed Apr 15 18:34:00 2020  dport-6017      sport-6017      in-0    out-0   total-0
pid-10773       start-Wed Apr 15 18:34:35 2020  dport-6018      sport-6018      in-1    out-0   total-1
********************
17 kernel processes
********************
pid-10606       parent-10605    start-Wed Apr 15 18:34:00 2020  kdef-4  requests-118    outstanding-0
UserCount-0     ActiveThreads-0
pid-10608       parent-10605    start-Wed Apr 15 18:34:01 2020  kdef-4  requests-119    outstanding-0
UserCount-0     ActiveThreads-0
pid-10618       parent-10605    start-Wed Apr 15 18:34:02 2020  kdef-4  requests-123    outstanding-0
UserCount-0     ActiveThreads-0
pid-10621       parent-10605    start-Wed Apr 15 18:34:03 2020  kdef-4  requests-118    outstanding-0
UserCount-0     ActiveThreads-0
pid-10625       parent-10605    start-Wed Apr 15 18:34:04 2020  kdef-4  requests-122    outstanding-0
UserCount-0     ActiveThreads-0
pid-10629       parent-10605    start-Wed Apr 15 18:34:05 2020  kdef-30         requests-124    outstanding-0
UserCount-0     ActiveThreads-0
pid-10634       parent-10605    start-Wed Apr 15 18:34:06 2020  kdef-2  requests-91     outstanding-0
UserCount-0     ActiveThreads-0
pid-10640       parent-10605    start-Wed Apr 15 18:34:07 2020  kdef-6  requests-111    outstanding-0
UserCount-0     ActiveThreads-0
pid-10645       parent-10605    start-Wed Apr 15 18:34:08 2020  kdef-6  requests-105    outstanding-0
UserCount-0     ActiveThreads-0
pid-10658       parent-10605    start-Wed Apr 15 18:34:09 2020  kdef-6  requests-104    outstanding-0
UserCount-0     ActiveThreads-0
pid-10670       parent-10605    start-Wed Apr 15 18:34:10 2020  kdef-6  requests-105    outstanding-0
UserCount-0     ActiveThreads-0
pid-10687       parent-10605    start-Wed Apr 15 18:34:11 2020  kdef-6  requests-105    outstanding-0
UserCount-0     ActiveThreads-0
pid-10700       parent-10605    start-Wed Apr 15 18:34:12 2020  kdef-13         requests-84     outstanding-0
UserCount-0     ActiveThreads-0
pid-10722       parent-10605    start-Wed Apr 15 18:34:13 2020  kdef-32         requests-85     outstanding-0
UserCount-0     ActiveThreads-0
pid-10726       parent-10605    start-Wed Apr 15 18:34:14 2020  kdef-34         requests-85     outstanding-0
UserCount-0     ActiveThreads-0
pid-10776       parent-10773    start-Wed Apr 15 18:34:37 2020  kdef-9  requests-861    outstanding-0
UserCount-0     ActiveThreads-0
pid-10788       parent-10773    start-Wed Apr 15 18:34:41 2020  kdef-14         requests-98     outstanding-0
UserCount-0     ActiveThreads-0

% jdejobs

jde920 ():
 Semaphores: 1   Shmem Segs: 6   Msg.Queues: 36
 Jobs on psrent19c:
10605 pts/4    00:00:00 jdenet_n
10606 pts/4    00:00:01 jdenet_k
10608 pts/4    00:00:01 jdenet_k
10618 pts/4    00:00:01 jdenet_k
10621 pts/4    00:00:01 jdenet_k
10625 pts/4    00:00:01 jdenet_k
10629 pts/4    00:00:02 jdenet_k
10634 pts/4    00:00:00 jdenet_k
10640 pts/4    00:00:00 jdenet_k
10645 pts/4    00:00:00 jdenet_k
10658 pts/4    00:00:00 jdenet_k
10670 pts/4    00:00:00 jdenet_k
10687 pts/4    00:00:00 jdenet_k
10700 pts/4    00:00:01 jdenet_k
10722 pts/4    00:00:03 jdenet_k
10726 pts/4    00:00:00 jdenet_k
10773 pts/4    00:00:00 jdenet_n
10776 pts/4    00:00:01 jdenet_k
10788 pts/4    00:00:00 jdenet_k
```

After running these commands that show the respective Oracle RAC nodes to which the EnterpriseOne kernel connections are attached, the EnterpriseOne kernel processes can be mapped to the Oracle RAC session and processes.

```
SQL> select count(s.machine),s.process,s.machine,i.host_name,s.program,s.inst_id
  2  from gv$session s join gv$instance i on (i.inst_id=s.inst_id)
  3  where username is not null
  4  group by process,machine,host_name,program,s.inst_id order by host_name,process;

COUNT(S.MACHINE) PROCESS    MACHINE              HOST_NAME       PROGRAM                          INST_ID
---------------- ---------- -------------------- --------------- ------------------------------ ----------
               1 10606      psrent19c                            jdenet_k@psrent19c (TNS V1-V3)          2
               1 10608      psrent19c                            jdenet_k@psrent19c (TNS V1-V3)          2
               1 10618      psrent19c                            jdenet_k@psrent19c (TNS V1-V3)          2
               1 10621      psrent19c                            jdenet_k@psrent19c (TNS V1-V3)          1
               1 10625      psrent19c                            jdenet_k@psrent19c (TNS V1-V3)          1
               3 10629      psrent19c                            jdenet_k@psrent19c (TNS V1-V3)          1
```

In the above example, process ID 10606 is an EnterpriseOne KDEF4 process or Security kernel. Process 10606 has a connection to the Oracle ADB on RAC Node 2. Similarly, three database connections exist for process 10629, which is an EnterpriseOne kernel KDEF30 Metadata process.

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 facebook.com/oracle twitter.com/oracle

Planned Database Maintenance of Oracle Autonomous Database with JD Edwards EnterpriseOne[White Paper Title]
July 2020