# Zero Downtime Database Upgrade Using Oracle GoldenGate

ORACLE WHITE PAPER | AUGUST 2015

## Table of Contents

.

## Executive Overview

Eliminating database downtime poses a significant challenge for IT organizations that need to upgrade or migrate mission-critical database environments running older versions of Oracle to newer ones, such as Oracle 11*g* and Oracle 12*c*. This is particularly true for applications that must provide continuous or near-continuous operations to users who increasingly expect uninterrupted availability of online services. Any outage of an application or website—even if that outage is scheduled or planned—has an impact on the revenue and reputation of the business.

For databases that host the data store for these mission-critical applications, availability requirements have become stringent. Unfortunately, there are essential events that require application downtime, including modifying hardware or database software, upgrading applications, applying software patches, and migrating to different computing architectures. Because such events are not conceived via system or data failures, they are aptly classified as planned outages. Empirical data from nontrivial applications deployed in very large database environments demonstrates that minimizing downtime to handle planned outages is a complex, time-consuming, error-prone, and costly process.

This white paper explains how organizations can upgrade or migrate from Oracle Database 8i, through Oracle Database 11*g* to Oracle Database 12*c*—without downtime. Using Oracle GoldenGate's real-time data integration and replication capabilities, businesses can perform rolling upgrades, create a clone database to offload instantiation and conversions, keep transactions in sync across the databases, manage partial or phased migrations and upgrades, conduct post upgrade/migration data verification, and implement a reliable failback strategy. Oracle GoldenGate can be used to reduce the downtime for numerous types of migrations and upgrades, including application upgrades, database upgrades, switching storage or hardware infrastructures, and even endian changes; however the focus of this paper is upgrading the database.

## Introduction to Oracle GoldenGate 12*c*

The primary goal of this white paper is to increase awareness of a solution that eliminates database downtime during a planned outage for database upgrades and migrations. As such, resource issues such as disk space and software licensing costs are not discussed—not because such issues are unimportant, but because the need to minimize downtime or completely avoid it outweighs other considerations in demanding, high-availability environments. Due to its scope, this white paper is primarily for advanced database users.

Furthermore, this document is not intended to repeat the upgrade steps, typical warnings, and preparatory details associated with an Oracle Database upgrade. There are many articles and notes that exist on the Oracle Support customer portal, in various books, and on the Web, that outline the operational procedures associated with upgrades and migrations.

The Oracle GoldenGate solution presented here leverages real-time synchronization; clone databases; rolling upgrades; in some cases, transportable and cross-platform transportable tablespaces; and failback. Every effort is made to avoid overhead at the primary database to ensure application availability.

A secondary goal of this white paper is to explain how to minimize the wall clock time required to perform the entire upgrade. Experienced users will realize that wall clock time can be on the order of days or weeks, yet the downtime to the application can still be near zero. The key here, as will be explained, is to offload work processing to additional database copies.

## Concepts and Terminology

To help comprehend the solutions, key concepts and terms are outlined in the following subsections.

### Source and Target

Throughout this document, the production database is referred to as the source and the secondary copy as the target database.

### Oracle GoldenGate

Oracle GoldenGate is a real-time change data capture application that provides guaranteed data capture, routing, transformation, and delivery across heterogeneous business systems. The application uses a low-overhead architecture to capture transactions non-intrusively from a source database by reading online transaction logs, transforming the data when needed, and applying those transactions with guaranteed integrity to a target database in real time.[1] Oracle GoldenGate's processes run continuously—even bi-directionally—and support high-volume, rapidly changing environments, moving tens of thousands of transactions per second with very low impact. The target database is a fully functional database open in read/write mode and is a transactional replica at a logical level, which can be leveraged for multiple applications, including a rolling database upgrade.

1 Noteworthy is that Oracle GoldenGate can be used when the target and source database software come from different vendors. It can also apply changed data in real time to Java Message Service message queues.

## Database Upgrade Options

A database upgrade advances the Oracle Database software release number from one version to another. There are two primary ways to perform an upgrade. There are numerous methods to upgrade a database, and most depend on the original version and final version.  For specific options for upgrading the database please refer to My Oracle Support knowledge document – “Different Upgrade Methods For Upgrading Your Database" (Doc ID 419550.1)

» **In-place upgrade.** An in-place upgrade renders the database inaccessible for business applications while the database software is being upgraded. This procedure entails running an upgrade script, recompiling invalid PL/SQL, and downtime that is usually not acceptable in most mission-critical environments.

» **Rolling upgrade.** The term rolling upgrade refers to upgrading different databases or different instances of the same database, such as in an Oracle Real Application Clusters (Oracle RAC) environment, one at a time, without stopping the database. A rolling upgrade consists of the following high-level steps:

  » The application points to the production database running software version VOLD.
  » A secondary logical database copy is constructed running software version VOLD.
  » The secondary database copy is upgraded to the next database version VNEW.
  » The secondary and primary databases are synchronized.
  » The primary database is shut down.
  » The application is pointed to the secondary database.
  » The primary database is kept in the same version VOLD for failback reasons.

### Database Migration

Moving an Oracle Database across different operating systems is a common requirement in many computing environments. A migration enables the underlying operating system or hardware platform to be changed. In Oracle, on-disk file formats are not homogeneous across platforms. Under Oracle 10*g* and up compatibility, the on-disk structures for platforms that appear in v$TRANSPORTABLE_PLATFORM are identical, but the endian format could differ.

### Standby Database

A *standby database* is a copy of the main production database that is maintained for high availability or disaster tolerance purposes through the use of Oracle Data Guard or Oracle Active Data Guard. The standby database can be physical or logical.

In the physical standby copy, the database is kept in recovery mode. More specifically, the redo logs of the production database are applied to a mounted copy of the production database. There are some hardware and operating system limitations because redo across Oracle platforms is not always compatible.

A logical standby is a copy of the production database that contains the same objects, but doesn’t physically match the structure of the production database copy. It is maintained by instantiating a logical equivalent of the production database and replaying the SQL that modifies the production database at the standby site.

### Transportable Tablespace

Transportable tablespace is a feature introduced in Oracle Database 8*i* that allows nonsystem tablespaces to be moved from one database to another by physically grafting the tablespace datafiles into the control files on the target database, and then importing object metadata into the target database’s dictionary. Transportable tablespace has three main phases:

» Exporting the metadata (dictionary data for the objects)
» Transferring the datafiles from one database to another
» Importing the metadata and datafiles

Transportable tablespace sets can be created from an Oracle Recovery Manager (RMAN) backup to avoid downtime on the primary database.

### Cross-Platform Transportable Tablespace

Cross-platform transportable tablespace is an extension of the transportable tablespace feature that enables tablespaces to be transported across database platforms. This feature can only be used after the database compatibility has been advanced to Oracle Database version 10.0.0.0 or later. The example provided in this white paper uses this feature in conjunction with Oracle GoldenGate to migrate previous Oracle Database versions to Oracle Database 12*c*.

### Oracle Recovery Manager

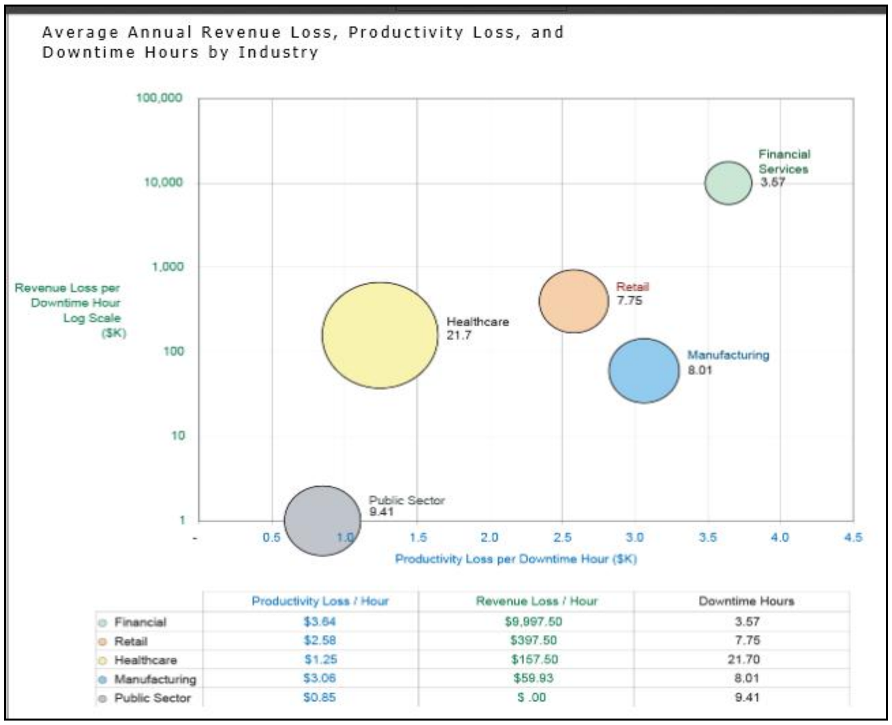
Oracle RMAN is a database tool that manages the process of making backups and managing the process of restoring and recovering from them. It is also used for the conversion of endian systems during a cross-platform CONVERT.

## Upgrade and Migration Challenges

Upgrade and migration projects pose challenges for mitigating business risk and navigating technology issues. Some of the more-common and critical challenges are listed in the subsections that follow. These issues help to highlight why a real-time, low-overhead solution using Oracle GoldenGate is the best way to perform database upgrades and migrations for mission-critical environments.

### Revenue Impact

Put simply, downtime equals lost revenue, productivity, and brand equity. Many businesses, such as retail, travel, and banking, no longer have the extended downtime windows for planning upgrades and migrations as it impacts their bottom line significantly. However, for scalability, performance, and cost-saving reasons, most database environments can benefit by taking advantage of recent technology advancements: low-cost storage, clusterware enhancements, and software improvements, for example. To capture these gains, businesses must find a way to upgrade or migrate while minimizing the high cost of downtime. Figure 1 on the next page highlights IDC's findings on revenue loss and productivity loss per downtime hour in various industries[2] .



| | Productivity Loss / Hour | Revenue Loss / Hour | Downtime Hours |
|---|---|---|---|
| ● Financial | $3.64 | $9,997.50 | 3.57 |
| ● Retail | $2.58 | $397.50 | 7.75 |
| ● Healthcare | $1.25 | $157.50 | 21.70 |
| ● Manufacturing | $3.06 | $59.93 | 8.01 |
| ● Public Sector | $0.85 | $ .00 | 9.41 |

Figure 1. Average revenue loss and productivity loss per hour of downtime for major industries

The Independent Oracle User Group (IOUG) conducted a study with Unisphere in 2012 among Oracle Database users on the impact of downtime. As shown in Figure 2, the study found that over 50 percent of the respondents listed disruption to business and IT operations as the top area of impact. In addition, 37 percent listed reduced

2 IDC White Paper, sponsored by Oracle, Maximize System Performance to Manage the Cost of IT Operations, April 2012 (Update of this chart by IDC is in progress at time of publication)

productivity, 34 percent listed negative impact on IT organization's reputation, and 24 percent listed negative impact to business reputation. While nearly 30 percent noted loss in productivity, 25 percent noted the negative impact on the overall reputation of the business.  You can find the full report here: [Enterprise Data and the Cost of Downtime, 2012 IOUG Database Availability Survey]

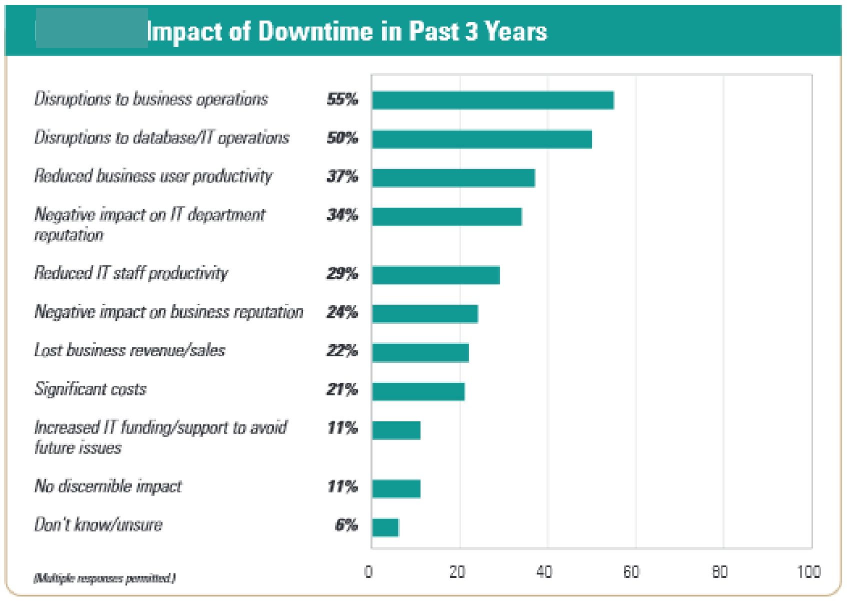


Figure 2. Impact of downtime on business and IT operations

To calculate the impact of downtime in your own organization, you can use a calculator such as this one:  Oracle Database 12*c* cost of downtime calculator.

## Customer Expectations and Loyalty

In this internet era, businesses must continue to support online processing beyond conventional business hours. Users expect continuous access to services, and unfortunately for providers, the cost of switching to the competition is negligible. For example, look at what happened when Virgin Blue had an outage that lasted up to 11 long days and customers couldn't board their scheduled flights during this time. In addition to the negative press, the company lost millions in profits. Virgin Blue's reservations management company, Navitaire, had to compensate Virgin Blue for up to $20 million.3 Even in the public sector, system downtime has major consequences. Several Virginia state agencies experienced data access issues due to an outage related to problems in a storage-area network (SAN). Because of this problem in a single data center as many as seven key agencies had problems accessing applications, shared folders and other data stored on servers in the state's Enterprise Solutions Center in Richmond. The DMV was among the agencies affected by the hardware failure and was not able to process in-person driver's licenses or ID cards at its 74 customer service centers.

3 Source: http://www.evolven.com/blog/downtime-outages-and-failures-understanding-their-true-costs.html#sthash.TVeEXAwV.dpuf

## Interdependencies and Business Reputation

E-commerce has resulted in digital business partnerships where inaccessibility of critical data at one site might also result in loss of business services at multiple sites—via data access from either Web services, direct querying, or preconfigured batch loading. Extended downtime is becoming difficult to plan for because significant coordination is required with business partners. This results in postponement of upgrades and migrations, which has associated indirect costs, such as increased support costs for extended maintenance, running the existing software with complex workarounds for bugs, and not being able to take advantage of new software functionality in later releases.

## Instantiation

The first technical problem that needs to be addressed when performing a rolling upgrade is choosing the method by which the target database is instantiated. In other words, how do you create a first version of the target database?

The complexity of the problem is magnified when the target database is on a different platform, because physical backups cannot simply be restored at the target and converted into a clone.

To handle multi-terabyte databases, procedures such as export/import are tedious and labor intensive. Failures during the instantiation or global data consistency are not adequately addressed with methods that are time consuming, unless some form of isolation is used. Extracting data from the production database over a period of time results in a performance impact on the source database that is likely unacceptable. Therefore, a good instantiation solution must properly address:

- » Global data consistency
- » Efficiency (speed)
- » Performance (low impact on the production database)
- » Manageability

## Incremental Data Movement

Following instantiation, businesses must determine the point that demarcates the last committed transaction picked up by the instantiation process, with the subsequent transactions submitted against the production database and the method by which the subsequent transactions get propagated to the target database.

A good solution that addresses incremental data movement must

- » Ensure global consistency
- » Identify a clean instantiation termination point
- » Allow transactions from that point onward to be propagated to the target database
- » Handle any failures during incremental data propagation (such as loss of network and media failure)

## Performance Impact

Another major concern during the upgrade is understanding what kind of impact is experienced by the application running on the production database. The upgrade steps should not degrade performance on the production database such that the application service levels (service-level agreements) are compromised.

### Staging Areas

Adequate disk resources need to be configured and managed when addressing rolling upgrades or migrations. Especially when a no-downtime migration is required, proper consideration must be given to handling the disk requirements for the clone database.

### Change Management

Changes, other than those made by data manipulation language (DML) updates, need to be addressed during the upgrade or migration process. Examples include datafile additions and PL/SQL package creations. Careful considerations must be made to ensure that the initial loading process handles these database changes.

### Special Handling of Legacy Data and Database Layout

Any data types that are not supported by the solution must be identified and handled in advance of the upgrade or migration. This can include converting LONG and LONG RAW to LOBs or taking advantage of new Oracle options, such as Transparent Tablespace Encryption, EHCC compression in Exadata or even converting to ASM storage.  Refer to the Oracle documentation for the exact details on supported data types for the technologies chosen to perform the zero-downtime migration.

### Verification

A post upgrade or post migration confirmation that the source and target database are synchronized must be conducted. Any out-of-sync conditions at the new source or failback database could pose risks to the business; therefore, they should be identified and resolved. Conducting this verification should be possible even as ongoing transactions are being processed at the source database. Thus, a good solution must address fast and efficient comparison of data across the two databases.

### Failback

Once the upgrade or migration is completed, a failback strategy should be in place to avoid any downtime and data loss, in case the new environment is not stable. Therefore, all transactions that have been processed at the target (the new production database) need to be propagated back to the original production database in a consistent and manageable manner for a successful failback.

## Achieving Zero-Downtime with Oracle GoldenGate

In-place upgrades require application downtime, which is not feasible in mission-critical, high-availability environments. Therefore, the remainder of this document focuses only on rolling upgrades.

Using Oracle GoldenGate in conjunction with Oracle Database features, a rolling upgrade or a rolling migration can be performed without any application downtime—other than the very minimal time required for application switchover, typically less than one minute and in most cases, only seconds. Using its real-time, heterogeneous data movement technology, Oracle GoldenGate imposes negligible database downtime for upgrades or migrations from one version of Oracle to another, including the new multitenant architecture of Oracle 12*c*.

If  Oracle GoldenGate is set up for bidirectional replication between old and new environments and both systems support the application in transaction processing, end users have the option to implement phased migration to the new environment and eliminate application switchover related downtime. The transition to the new database environment can be completely transparent to the end users.

Upgrading to Oracle Database 12*c* using Oracle GoldenGate consists of the following high-level steps:

1. Set up a standby database running the previous database software version using an existing database backup.
2. Upgrade the standby database to Oracle Database 12*c.*
3. Synchronize the standby database with the production database.
4. Test in active/live mode.
5. Switch over the application to the standby database.
6. Upgrade the primary database to Oracle Database 12*c* after comprehensive application testing at standby.

The next section describes how Oracle GoldenGate can be used to conduct zero-downtime upgrades or migrations. “Detailed Steps Using a Platform Migration Example” discusses the steps in greater detail. Because a migration is a more-complicated procedure than an upgrade, the detailed steps use a case study of migrating an Oracle Database 10g on one platform to an Oracle Database 12c on another platform.

## Comparison to Alternative Upgrade Methods

Another method of upgrading a database (Oracle 11g and up) is through the use of a transient logical standby database.  This method can be used to help reduce the downtime during an upgrade while utilizing an existing physical Data Guard system.  In essence, you’ll change the physical standby to a logical standby database, execute the rolling upgrade procedures, and then switch it back to a physical standby once the upgrade is complete. By performing a roll transition the newly upgraded standby database can be opened for read/write activity and become the primary system for normal day to day use.  However, there are some limitations to this option as the two servers must be the same OS and endiannes, and Oracle 11g has some documented limitations for data types in a logical standby environment.

Oracle GoldenGate gives the flexibility to perform upgrades from earlier versions of Oracle, all the way back to Oracle 8*i*.  It can also be used to migrate across endiannes and operating systems, as well as converting of environments from non-ASM to ASM, rebuilding and defragmentation of tables and indexes, and migration into Exadata environments with full EHCC compression of all legacy data. Oracle GoldenGate provides the most flexible solution to migrate or upgrade databases.

## Overview of Oracle GoldenGate Architecture

As shown in Figure 3, Oracle GoldenGate leverages a decoupled architecture comprising independent application modules to capture and replicate data in real time, with low impact on the source database.

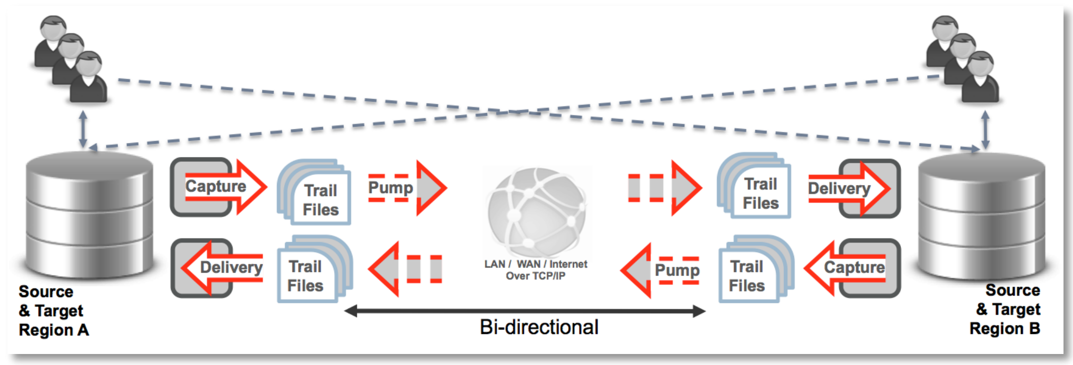


Figure 3. High-level architecture of Oracle GoldenGate, running in a bidirectional configuration between two databases

### Capture

The Oracle GoldenGate Capture module can reside on the source database system and is multithreaded in an Oracle RAC environment. It mines transactions from the Oracle redo log and propagates transactions over the network to an on-disk queue. Only committed transactions are written to the queues. For Oracle Database 11.2.0.3 and above, Oracle GoldenGate offers Integrated Capture, a multi-threaded capture mechanism that improves performance by interacting directly with a database log mining server to receive data changes. Integrated Capture also provides users with the ability to reduce overhead on the source system by offloading the Capture process to an alternate location, such as the target.

#### Initialization

The Capture module can be used either to initialize the target database directly or to start propagating transactions against an existing target database from a given point.

#### Change Capture

DML and, optionally, Data Dictionary Language (DDL) changes are captured from the transaction logs.

### Trail Files

The Oracle GoldenGate Trail Files module can be conceptualized as a persistent ordered set of committed transactions generated by the Oracle GoldenGate Capture process. Trail Files describe the DDL and DML operations (inserts, updates, and deletes), and contain transactional context as captured from the source database.

### Delivery

The Oracle GoldenGate Delivery module is a process that runs on the target system. It reads the captured data from the Trail Files module and applies the captured transactions at the target using dynamic SQL. To maintain synchronization between the source and target databases, Oracle GoldenGate applies data changes to the target tables using native database calls, statement caches, and local database access. Oracle GoldenGate 12*c* now includes an Integrated Delivery for Oracle Database versions 11.2.0.4 and above. The Integrated Delivery leverages Oracle Database parallel apply servers for automatic dependency tracking and parallel aware apply. It enables two times or more higher performance compared to the Classic Delivery process.

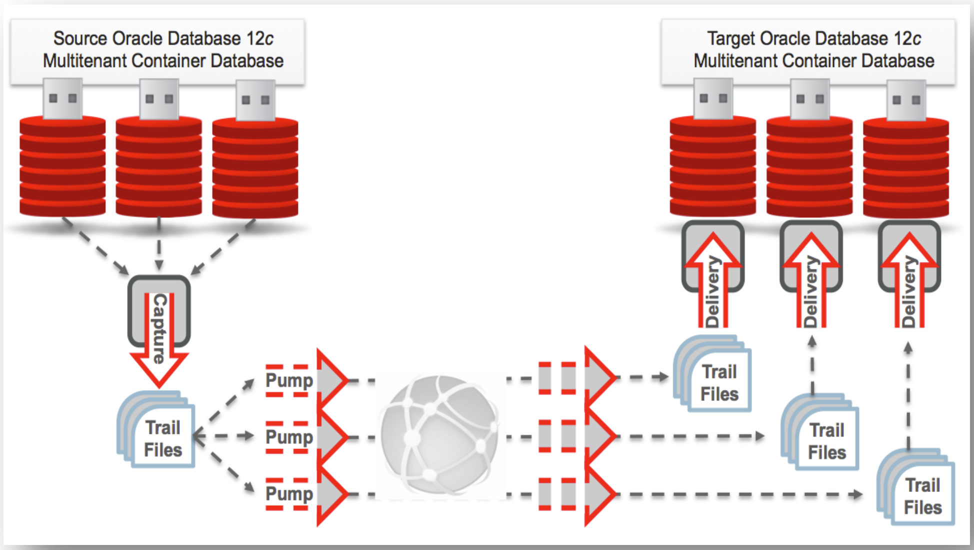
### Oracle GoldenGate Veridata

Oracle GoldenGate Veridata is a standalone, high-speed data comparison and repair solution that identifies, reports and fixes data discrepancies between two heterogeneous databases, without interrupting ongoing business processes. It allows data discrepancies to be isolated for testing and troubleshooting. Oracle GoldenGate Veridata is ideal for conducting data validation after the rolling upgrade, once the source and target are fully operational and running different versions of Oracle Database. It can also help to determine if a failback is needed, in case of any risky data anomalies.

## Using Oracle GoldenGate for Multitenant Databases

In the new Oracle Database 12c multitenant architecture, there are some additional considerations for Oracle GoldenGate. When capturing from any pluggable database (PDB) that is plugged into a container database, you only need to set up a single Oracle GoldenGate Integrated Capture (Extract) process. This is because all of the changes in a PDB are actually written to a single stream of redo data that is managed by the container. Since a multiple PDBs can contain the same owner.tablename, we have included the option to add in the PDB name.  So objects can be uniquely identified across an entire multitenant system.  In the Extract parameter file you can now use 3 part names, like pdb.owner.object.

When delivering data into a PDB, the Delivery (Replicat) process should connect directly to the PDB.  Data in the trails can include operations from multiple PDBs.  So, in the Replicat parameter file, you can do the 3-part to 2-part naming conversion.  Data from multiple PDBs can be applied to a single PDB.


Source Oracle Database 12c
Multitenant Container Database
Target Oracle Database 12c
Multitenant Container Database
Capture
Trail Files
Pump
Pump
Pump
Delivery
Delivery
Delivery
Trail Files
Trail Files
Trail Files

## Detailed Steps Using a Platform Migration Example

This is a case study of moving an Oracle Database 10*g* on IBM AIX to Oracle Database 12*c* on Linux and the detailed implementation steps. These steps can be simplified and adopted to accomplish a rolling upgrade.

Two scenarios are described next: a migration without the addition of a failback solution and the same migration with the addition of a failback.

One thing to keep in mind is that this process, especially in very large databases can take many days, especially for the initial loading of the new database. Keep in mind that even during this entire process, the source database is still up and running and can be fully utilized for normal day to day processing. The only downtime that the application and its users will experience is during the final step when switching the users and the application from the original system to the newly upgraded environment.

### Migration Without Failback

Figure 4 provides an overview of a database migration without using a failback. The numbers shown in the diagram correspond to the detailed steps below.

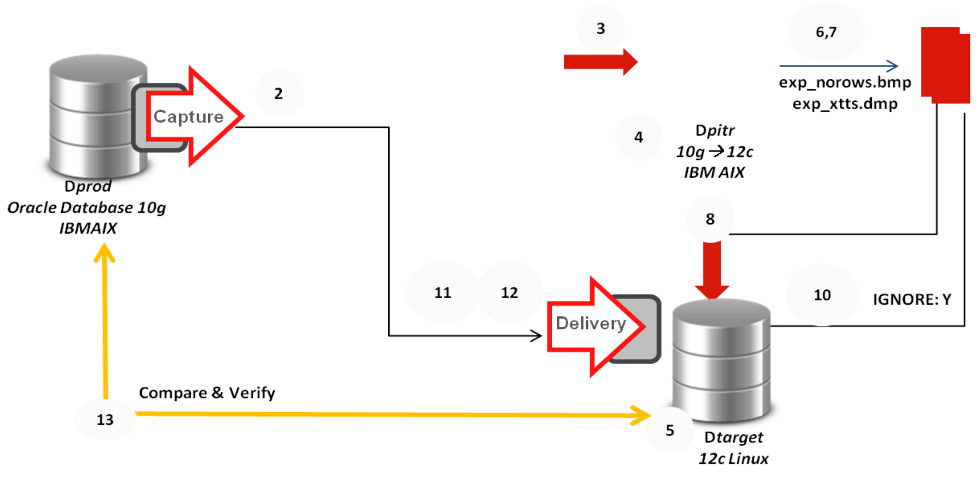


Figure 4. Overview of a cross-platform database migration without failback

1. Although Oracle GoldenGate provides some support for DDL replication, it is best to restrict these activities as much as possible during the migration process (not shown). Any newly created tables will need to also be added into the replication stream and be included in the initial load. New datafiles, tablespaces, etc. will need to be mapped to the new structures on the Dtarget environment. It is best to avoid creating new packages and tablespaces during migration. Performing other types of DDL, however, such as creating new users, permissions, indexes, views and stored procedures usually will not affect the migration process.
2. Start the Oracle GoldenGate Capture process at the production database Dprod. This ensures that any open transactions at the time of the initial load are captured, and any subsequent transactions are correctly moved over to the new database.
3. In a separate staging area, do a point-in-time recovery up to a given SCN, Qscn, of an existing backup of Dprod. Call this database Dpitr. The Capture process in Step 2 must capture all transactions that have a commit time stamp higher than this Qscn. It must have been positioned at or before the beginning of the longest-running transaction when the source database was at Qscn. You can query v$transaction

and v$database at the source database to identify the longest-running transaction and the current SCN at any point in time to identify where Capture should be positioned.

4. (optional) Upgrade Dpitr to Oracle Database 12c on IBM AIX. Advance compatibility to Oracle Database version 10.0.0.0 or later. This step is only necessary if you are going to be using transportable tablespaces or advanced load/unload features that are not available in the original source database version.
5. Set up a vanilla Oracle Database 12c on Linux. Call this database Dtarget. During this step you can configure storage requirements such as ASM, or in the case of Exadata you can configure EHCC compression. With Oracle Database 12c, this new database could also be configured as a multitenant architecture as a pluggable database.

Steps 6-10 perform the initial load of the data from Dpitr to Dtarget. There are a number of ways to perform this initial load. The steps below describe the method using transportable tablespaces. The load can be done using Oracle Data Pump, or Oracle GoldenGate's own utilities. For more details on which initial load utility to use, please refer to Oracle GoldenGate: Initial Load Techniques and References (Doc ID 1311707.1) or Oracle GoldenGate Best Practices: Instantiation from an Oracle Source Database (Doc ID 1276058.1). Some of the methods in that article also remove the need for having a staging environment like Dptir; however some methods may not be available to all Oracle database releases.

6. Take a full export without rows at Dpitr. Call the generated export exp_norows.dmp. This will pick up any objects that are not picked up as part of the transportable tablespace export.
7. Unplug the user tablespaces from Dpitr with the Oracle Database cross-platform transportable tablespace feature, using source-side endian conversion. (Note that the conversion would not be required if the endian systems were the same.) This is the step that avoids any performance degradation and does not require any quiescing at Dprod. This step will create a small export dump file. Call this exp_xtts.dmp.
8. Plug the set of tablespaces from Dpitr into Dtarget using the cross-platform transportable tablespace feature. Use the exp_xtts.dmp file created from Step 7. (Note that the plugged in tablespaces are in read-only mode.)
9. Make the set of user tablespaces in Dtarget Read Write (not shown).
10. Do a NOROWS import with IGNORE=Y option at Dtarget using the exp_norows.dmp dump file.
11. Start the Oracle GoldenGate Delivery process at Dtarget and synchronize up to the changes generated since Qscn.
12. (only needed if transportable tablespaces are being used for the initial load) If any datatypes are not supported by transportable tablespace or Oracle GoldenGate, then do a special export/import of these objects from Dprod to Dtarget.
13. Use Oracle GoldenGate Veridata to verify that the data at Dprod and Dtarget is synchronized.
14. Switch the application from Dprod to Dtarget (not shown).

The above procedure offloads any quiescing, and conversion work to a clone database, and takes advantage of Oracle GoldenGate's incremental real-time data capture and delivery to eliminate the downtime to zero, excluding the application switchover time.

## Migration with Failback

It is often necessary to maintain or keep the old environment around in case there are any unforeseen issues with the new system. Oracle GoldenGate's capability to replicate from any platform to any other platform allows it to be used to provide a failback option. In this scenario Oracle GoldenGate will replicate from the new environment (Dtarget) back to the original system (Dprod) for just this purpose. Figure 5 provides an overview of a database migration with using a failback. The numbers shown in the diagram correspond to the detailed steps below, including the Capture process capturing data from Dtarget and replicating back to Dprod.

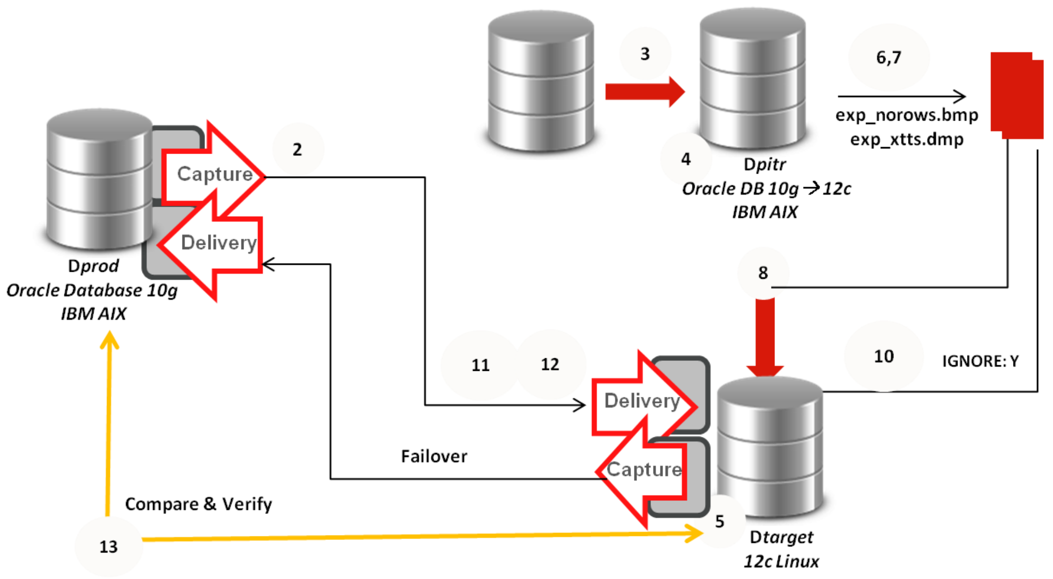


**Figure 5. A cross-platform database migration with failback**

1. Although Oracle GoldenGate provides some support for DDL replication, it is best to restrict these activities as much as possible during the migration process (not shown). Any newly created tables will need to also be added into the replication stream and be included in the initial load. New datafiles, tablespaces, etc. will need to be mapped to the new structures on the Dtarget environment. It is best to avoid creating new packages and tablespaces during migration. Performing other types of DDL, however, such as creating new users, permissions, indexes, views and stored procedures usually will not affect the migration process.
2. Start the Oracle GoldenGate Capture process at the production database Dprod. This ensures that any open transactions at the time of the initial load are captured, and any subsequent transactions are correctly moved over to the new database.
3. In a separate staging area, do a point-in-time recovery up to a given SCN, Qscn, of an existing backup of Dprod. Call this database Dpitr. The Capture process in Step 2 must capture all transactions that have a commit time stamp higher than this Qscn. It must have been positioned at or before the beginning of the longest-running transaction when the source database was at Qscn. You can query gv$transaction and gv$database at the source database to identify the longest-running transaction and the current SCN at any point in time to identify where the Capture process should be positioned.
4. (optional) Upgrade Dpitr to Oracle Database 12*c* on IBM AIX. Advance compatibility to Oracle Database version 10.0.0.0 or later. This step is only necessary if you are going to be using transportable tablespaces or advanced load/unload features that are not available in the original source database version.
5. Set up a vanilla Oracle Database 12*c* database on Linux. Call this database Dtarget. During this step you can configure storage requirements such as ASM, or in the case of Exadata you can configure EHCC compression. With Oracle Database 12c, this new database could also be configured as a multitenant architecture as a pluggable database.

Steps 6-10 perform the initial load of the data from Dpitr to Dtarget. There are a number of ways to perform this initial load. The steps below describe the method using transportable tablespaces. The load can be done using

Oracle Data Pump, or Oracle GoldenGate's own utilities. For more details on which initial load utility to use, please refer to Oracle GoldenGate: Initial Load Techniques and References (Doc ID 1311707.1) or Oracle GoldenGate Best Practices: Instantiation from an Oracle Source Database (Doc ID 1276058.1). Some of the methods in that article also remove the need for having a staging environment like Dptir; however some methods may not be available to all Oracle database releases.

6. Take a full export without rows at Dpitr. Call the generated export exp_norows.dmp. This will pick up any objects that are not picked up as part of the transportable tablespace export.
7. Unplug the user tablespaces from Dpitr with the Oracle Database cross-platform transportable tablespace feature, using source-side endian conversion. (Note that the conversion would not be required if the endian systems were the same.) This is the step that avoids any performance degradation and does not require any quiescing at Dprod. This step will create a small export dump file. Call this exp_xtts.dmp.
8. Plug the set of tablespaces from Dpitr into Dtarget using the cross-platform transportable tablespace feature. Use the exp_xtts.dmp file created from Step 7. (Note that the plugged in tablespaces are in read-only mode.)
9. Make the set of user tablespaces in Dtarget Read Write.
10. Do a NOROWS import with IGNORE=Y option at Dtarget using the exp_norows.dmp dump file.
11. Start the Oracle GoldenGate Delivery process at Dtarget and synchronize up to the changes generated since Qscn.
12. (only needed if using transportable tablespaces for the initial load) If any datatypes are not supported by transportable tablespace or Oracle GoldenGate, then do a special export/import of these objects from Dprod to Dtarget.
13. After Oracle GoldenGate eliminates the lag between Dprod and Dtarget, use Oracle GoldenGate Veridata to verify that the data at Dprod and Dtarget is synchronized.
14. Start the Oracle GoldenGate Capture process on Dtarget.
15. Switch the application from Dprod to Dtarget (not shown).
16. Start the Oracle GoldenGate Delivery process on Dprod.

### Failback Steps

During the real-time bidirectional data movement, Oracle GoldenGate allows two databases to support transaction processing. This makes failback a trivial process, if the new database is not stable:

1. Stop the application at Dtarget (the new primary) running Oracle Database 12c.
2. Once Oracle GoldenGate has applied all transactions from Dtarget to Dprod, switch the application to Dprod.
3. Declare Dprod as the new primary.

### Phased Migration Using Bidirectional Synchronization

In certain environments, it might be possible to migrate a limited number of applications or users over to the new target database while running the rest of the applications against the original source database.  This can be done based on logical application partitioning or geographical partitioning.

For example, if the application supports 100 bank branches, only one branch might be switched over to the new database until the new environment is deemed stable after appropriate testing and validation. Oracle GoldenGate can be run bi-directionally to synchronize the data across the source and the target systems, thus supporting active-active database environments.

## Conclusion

Users expect mission-critical applications to be continuously available. Even planned outages can have a negative impact on user satisfaction. Using Oracle GoldenGate, a rolling upgrade or migration can be completed with zero-database downtime and only very minimal application switchover downtime. Key technical advantages of this solution include the following:

» Rolling upgrades or migrations using two databases
» No instantiation of the primary database by offloading to a clone database
» Conversions offloaded to a clone staging database
» Transaction synchronization across databases
» Data replication and transactional integrity verification using Oracle GoldenGate Veridata
» Database failover using a bidirectional Oracle GoldenGate configuration

Oracle GoldenGate enables the world's largest enterprises to improve the availability, performance, and accessibility of the transactional data that drives mission-critical business processes. Oracle GoldenGate's wide variety of use cases includes real-time data integration for business intelligence, query offloading, zero-downtime upgrades and migrations, disaster recovery, and active-active databases for data distribution, , and high availability.

ORACLE

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

**Worldwide Inquiries**
Phone: +1.650.506.7000
Fax: +1.650.506.7200

CONNECT WITH US

blogs.oracle.com/dataintegration
facebook.com/oracledataintegration
twitter.com/OracleDI
oracle.com/goto/oraclegoldengate

**Hardware and Software, Engineered to Work Together**

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